z/VM
7.4

Performance

SC24-6301-74

Note:

Before you use this information and the product it supports, read the information in “Notices” on page 281.

This edition applies to version 7, release 4 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-06-12

Contents

Figures

Tables

About This Document

This document describes the planning, managing, measuring, and tuning considerations for improving the performance of an IBM z/VM system.

Intended Audience

This information is intended for system programmers and others involved in z/VM performance monitoring and tuning activities. To derive the most benefit from this information, you should be acquainted with z/VM concepts such as paging and minidisk caching. You should also have a working knowledge or familiarity with a z/VM system installation.

Where to Find More Information

For more information about z/VM functions, see the documents listed in the "Bibliography" on page 285.

The z/VM performance website at IBM: VM Performance Resources (https://www.ibm.com/vm/perf/) provides additional z/VM performance information such as the IBM: z/VM Performance Report (https://www.ibm.com/vm/perf/reports/), performance tips, FAQs, and lists of z/VM release-to-release performance changes.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

Summary of Changes for z/VM: Performance

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (**|**) to the left of the change.

SC24-6301-74, z/VM 7.4 (June 2025)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3 and 7.4.

[7.4 VM66824, 7.3 VM66823] z/VM support for the IBM z17 family

With the PTFs for APARs VM66824 (7.4) and VM66823 (7.3), z/VM 7.4 and 7.3 provide support for the IBM z17 family. For more information, see z/VM support for the IBM z17 family in the *z/VM: Migration Guide*.

The following CP monitor records are new:

- Domain 0 Record 25 - MRSYTPOW - Power Consumption Information
- Domain 6 Record 54 - MRIODVSE - Virtual Switch EQDIO Activity

The following CP monitor records are updated:

- Domain 0 Record 20 - MRSYTEPM - Extended Channel Measurement Data
- Domain 1 Record 4 - MRMTRSYS - System Configuration Data
- Domain 1 Record 19 - MRMTRQDC - QDIO Device Configuration
- Domain 4 Record 2 - MRUSELOF - User Logoff Data
- Domain 4 Record 3 - MRUSEACT - User Activity Data
- Domain 6 Record 21 - MRIODVSW - Virtual Switch Activity
- Domain 6 Record 22 - MRIODVSF - Virtual Switch Failover
- Domain 6 Record 25 - MRIODQDA - QDIO Device Activation
- Domain 6 Record 26 - MRIODQDS - QDIO Device Activity
- Domain 6 Record 27 - MRIODQDD - QDIO Device Deactivation
- Domain 6 Record 34 - MRIODBPD - Virtual Switch Bridge Port Deactivation

The layouts of CP monitor records can be found at the following location:

z/VM Data Areas, Control Blocks, and Monitor Records (https://www.vm.ibm.com/pubs/ctlblk.html)

[7.4 VM66826, 7.3 VM66825] z/VM Performance Data Pump enhancements

With the PTFs for APARs VM66826 (7.4) and VM66825 (7.3), z/VM 7.4 and 7.3 provide support for a new z/VM Performance Data Pump dashboard. The dashboard provides a graphical representation of IBM z17 family power consumption metrics from the z/VM monitor stream. The display includes information for the following components:

- CPC
- LPAR
- CPU
- I/O
- Memory

The information can be used to calculate or approximate guest level apportionment.

Information about setting up Data Pump is available. See z/VM Performance Data Pump in *z/VM: Performance*.

Information is available about other services that process the Data Pump output stream, including links to Grafana sample dashboards. See z/VM Performance Data Pump (https://www.vm.ibm.com/related/perfkit/datapump/).

[7.4 PH65378, 7.3 PH65377] z/VM TCP/IP support for EQDIO

With the PTFs for APARs PH65378 (7.4) and PH65377 (7.3), z/VM 7.4 and 7.3 provide a native network device driver for the z/VM TCP/IP stack that uses EQDIO adapters for network transport.

The following topic is updated:

- Appendix F, "TCP/IP Monitor Records," on page 215

  The following TCP/IP monitor records are updated:

  - TCP/IP MIB Record - Type '00'x - Sample Data
  - TCP/IP LCB Record - Type '05'x - Sample Data
  - TCP/IP Link Definition Record - Type '08'x - Configuration Data
  - TCP/IP Link Deletion Record - Type '10'x - Event Data

Miscellaneous updates for June 2025

The following topics are updated:

- "Major factors that affect performance" on page 26
- "CP performance facilities" on page 28
- "Additional features of QDIO performance" on page 47

SC24-6301-74, z/VM 7.4 (March 2025)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.4.

[VM66829] Two-speed Monitor sampling

With the PTF for APAR VM66829, support is added to z/VM 7.4 to collect z/VM Monitor data at two different sampling rates. Particular guests can collect detailed data for a relatively brief period and preserve the data in case a situation arises that requires the data for analysis. Other guests can collect data less frequently and over longer time periods for customary purposes such as real-time monitoring or feeding the z/VM Performance Data Pump.

The following topics are updated with new SUBINTERVAL or SUBINT options:

- "CONNECT Interface for Shared Mode" on page 172
- "The Monitor Control Area" on page 177

SC24-6301-74, z/VM 7.4 (September 2024)

This edition supports the general availability of z/VM 7.4. Note that the publication number suffix (-74) indicates the z/VM release to which this edition applies.

The following table is updated:

- Table 29 on page 232 (in Appendix F, "TCP/IP Monitor Records," on page 215)

SC24-6301-73, z/VM 7.3 (April 2024)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[VM66758, VM66759] SCP identification enhancements

With the PTF for APAR VM66758, the z/VM Control Program (CP) can display identification information that is provided by guest operating systems. The CP display is similar to the HMC Systems Management "Partitions" display.

The identification information is also provided in the following monitor records:

- Updated record: Domain 1 sample configuration record 15 (MRMTRUSR), "Logged on User"
- New record: Domain 4 event record 14 (MRUSESCP), "SCP Identification"

The following CP command is new:

- QUERY SCPID (See QUERY SCPID in z/VM: CP Commands and Utilities Reference.)

With the PTF for APAR VM66759, z/VM's Control Monitor System (CMS) is enhanced. A user can change the CMS system identification information that a guest sends to the hypervisor.

The following CMS command is new:

- SET SCPID (See SET SCPID in z/VM: CMS Commands and Utilities Reference.)

SC24-6301-73, z/VM 7.3 (September 2023)

This edition includes changes to support product changes that are provided or announced after the general availability of z/VM 7.3.

[VM66678, VM66709] Warning Track Interruption Facility

With the PTFs for APARs VM66678 (CP) and VM66709 (Performance Toolkit), z/VM 7.3 exploits a feature of Processor Resource/Systems Manager (PR/SM) called the *warning-track-interruption facility.* z/VM's exploitation of this facility helps improve guest response time and overall performance of workloads that are run on vertical-low or vertical-medium logical processors.

The following CP monitor records are updated:

- Domain 0 Record 2 - MRSYTPRP - Processor Data (Per processor)
- Domain 0 Record 19 - MRSYTSYG - System Data (Global)
- Domain 1 Record 4 - MRMTRSYS - System Configuration Data
- Domain 2 Record 7 - MRSCLSRM - Set SRM Changes
- Domain 4 Record 2 - MRUSELOF - User Logoff Data
- Domain 4 Record 3 - MRUSEACT - User Activity Data

The layouts of CP monitor records can be found at:

z/VM Data Areas, Control Blocks, and Monitor Records (https://www.vm.ibm.com/pubs/ctlblk.html)

SC24-6301-73, z/VM 7.3 (August 2023)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.3.

[VM66687] z/VM Performance Data Pump

With the PTF for APAR VM66687, z/VM 7.3 supports z/VM Performance Data Pump.

z/VM Performance Data Pump (Data Pump) converts machine-readable z/VM monitor and SFS data into a generic text-based data stream. Modern tools can use the data stream to display real-time performance dashboards, aggregate real-time data for long-term usage analysis, or integrate with existing enterprise observability solutions.

Data Pump provides high-quality z/VM performance data to enterprise monitoring tools that are already deployed for application monitoring or capacity planning. Such tools align with skills and experiences of many users and offer integration with other tools and solutions.

Data Pump by itself does not deliver value that a user can readily use. To take advantage of Data Pump the customer must provision, configure, and deploy other services to process the data stream. While instructions for deploying the open source solutions are available, these components are not delivered with the z/VM product. For more information, see z/VM Performance Data Pump (https://www.vm.ibm.com/related/perfkit/datapump/).

z/VM Performance Data Pump is licensed with Performance Toolkit for z/VM but does not support, depend upon, or interact with Performance Toolkit for z/VM in any way.

The following topicsnare new:

- Appendix I, “z/VM Performance Data Pump,” on page 253
- “Setting up the DATAPUMP virtual service machine” on page 253
- “Starting and stopping Data Pump” on page 256
- “The DATAPUMP command” on page 257
- “Configuring Data Pump ” on page 258
- “Data Pump MONITOR plug-in” on page 262
- “Data Pump MONWRITE plug-in” on page 264
- “Data Pump SFS plug-in” on page 266
- “Data Pump INFLUXDB plug-in” on page 268
- “Data Pump SPLUNK plug-in” on page 273
- “Data Pump STATISTICS plug-in” on page 277
- “Data Pump DATADUMP plug-in” on page 278

The following topics are updated:

- “Performance Monitoring” on page 75
- Chapter 9, “Monitoring Performance Using CP Monitor,” on page 79
- Chapter 10, “Monitoring SFS Performance,” on page 97

SC24-6301-73, z/VM 7.3 (May 2023)

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.3.

[VM66424, VM66434, VM66650] Guest Secure IPL Support

With the PTFs for APARs VM66424, VM66434, and VM66650, z/VM 7.3 supports guest secure IPL (load and dump) for both ECKD and SCSI devices. A z/VM user can request that the machine loader validate the signed IPL code by using the security keys that were previously loaded by the customer into the HMC certificate store. The validation ensures that the IPL code is intact, unaltered, and originates from a trusted build-time source.

Support is provided for the following guest operating systems:

- This support provides the ability for a Linux® guest to exploit hardware to validate the code being booted, helping to ensure it is signed by the client or its supplier.

Linux on IBM zSystems™ instances that previously were able to perform secure boot first level on an IBM z15® or IBM LinuxONE III prior to Driver D41C Bundle S73a, or an IBM z16™ or IBM LinuxONE 4 prior to Driver D51C Bundle S18, will no longer be able to use secure boot until appropriate additional support is applied to the Linux image. Details are available about the required service level of Linux to properly IPL securely first or second level after driver D41C Bundle S73a or Driver D51C Bundle S18 has been applied. See 230428 Machine Alert for 8561, 8562, 3931, 3932 (https://www-40.ibm.com/servers/resourcelink/lib03020.nsf/pagesByDocid/272B3DD994A65B538525899F005FA0E6?OpenDocument).

- z/OS® is supported in audit mode only. Full exploitation requires Virtual Flash Memory support, which is not available to a guest. In audit mode, the IPL code is checked but the IPL continues even if the code is not valid.

z/VM and the z/VM stand-alone dump utility do not support performing host IPL via List-Directed IPL (LD-IPL) from ECKD. In addition, Secure IPL of the z/VM host and z/VM stand-alone dump are not supported.

The following CP monitor records are updated:

- Domain 1 Record 4 - MRMTRSYS - System Configuration Data
- Domain 1 Record 15 - MRMTRUSR - Logged on User
- Domain 4 Record 2 - MRUSELOF - User Logoff
- Domain 4 Record 3 - MRUSEACT - User Activity

SC24-6301-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

Eight-member SSI support

This support increases the maximum size of a single system image (SSI) cluster from four members to eight, enabling clients to grow their SSI clusters to allow for increased workloads and providing more flexibility to use live guest relocation (LGR) for nondisruptive upgrades and workload balancing.

The following CP monitor record is updated:

- Domain 1 Record 25 - MRMTRSSI - SSI Configuration

Change in location and size of the MONDCSS and PERFOUT saved segments

The default CP MONITOR MONDCSS saved segment starting virtual address has been changed to 1 GB and the size has been increased to 96 MB. The IBM Performance Toolkit PERFOUT saved segment starting virtual address has been changed to 1120 MB, immediately following the new MONDCSS. The reasons for these changes include:

- Moving the segment starting locations addresses a problem with the Linux kdump configuration, which failed in a virtual machine with 1 GB of virtual memory that had attached one or both current versions of these segments.
- Increasing the size of the MONDCSS segment accommodates the increased volume of CP Monitor data produced and provides room for its future growth.

These changes will not affect existing versions of these segments; they must be re-created for the new defaults to take effect.

The following topics are updated:

- “Creating the Saved Segment” on page 82
- “Enabling MONITOR and MONWRITE: An Example” on page 93

Miscellaneous updates for z/VM 7.3

The following topic is updated:

- Appendix F, “TCP/IP Monitor Records,” on page 215

Part 1. Introduction

The topics in this section introduce you to z/VM performance:

- Chapter 1, “Introduction,” on page 3
- Chapter 2, “Characteristics of a z/VM System,” on page 5.

Chapter 1. Introduction

Whether you have just been introduced to the responsibility of system performance or have been closely monitoring your system for years, achieving optimum system performance is a seemingly elusive goal. Determining the performance bottlenecks is a challenge at the very least. Understanding and detecting the root cause of a performance problem can be even more difficult.

This document uncovers several of the mysteries surrounding this challenge by first giving you an understanding of the z/VM system characteristics and unfolding the performance methodology, planning measures, monitoring facility tools and tuning actions available for your use.

Conversion or Migration Performance Information

This document covers performance information for a z/VM system only. For performance information concerning system migrations or conversions, see *z/VM: Migration Guide*.

Chapter 2. Characteristics of a z/VM System

This section discusses the characteristics of your z/VM system, especially those that pertain to the issue of system performance. Topics discussed are:

- Real processor management
- Virtual processor management
- Scheduling and dispatching
- Hardware timing facilities.

There are basically two types of z/VM processor management: real processor management and virtual processor management. This overview and discussion are specifically related to their effect on system performance.

Real Processor Management

The Control Program (CP) is responsible for allocating the processing power of the available real processors in order to process virtual machine instructions (scheduling and dispatching) as well as handling all real machine interrupts. In this way, a real processor may be one of two types: a master processor or an alternate processor.

Master and Alternate Processors

z/VM classifies real processors according to their use by CP. A real processor may be one of two types: a master processor or an alternate processor. Use the QUERY PROCESSORS command to determine master processor designation.

The master processor is the processor on which CP is IPLed and on which certain CP work is required to run. CP assigns the rest of the processors as alternate until the existing master processor fails, in which case CP chooses one of the remaining processors to be the master. In a uniprocessor system, the sole real processor is, of course, also the master processor.

In a real multiprocessor system, the non-master processor(s) are alternate processors. An alternate processor can be used to run CP work or the work of virtual machines. When the operator IPLs CP on the real machine, CP initially designates the processor on which CP is IPLed as the master processor and all other processors in the processor complex as alternate processors.

Virtual Processor Management

The dispatcher selects virtual machines from the dispatch list for the processors to run. The scheduler function of the Control Program is responsible for calculating the priorities and making the necessary decisions on when to move a particular machine to the dispatch list.

Note: CP's virtual processor management has been improved so that no virtual machine stays in the eligible list more than an instant before being added to the dispatch list. Therefore some functions intended to improve performance by managing the eligible list, such as the QUICKDSP option, are now less meaningful.

Virtual Machine Scheduling and Dispatching

Through its virtual machine scheduling function, CP attempts to keep as many of the logged-on virtual machines as possible operating concurrently. It takes into account the availability of processing time, paging resources, and real storage (as compared to virtual machine requirements for these resources), as well as limits in effect on the number of virtual machines waiting to be given processor control. The availability of, and requirements for, I/O resources are not factored into the scheduler's calculations.

Each virtual machine is permitted to remain in the set of virtual machines that are competing for use of the real processor for an interval of time called an *elapsed time slice.* A virtual machine is permitted to have processor control only for a portion of the elapsed time slice (called the *dispatch time slice*) each time it is dispatched. When a virtual machine has accumulated its dispatch time slice, its priority is readjusted in relation to the other competing virtual machines. When a virtual machine has accumulated its elapsed time slice, it is dropped from the set of competing virtual machines and the scheduler attempts to add as many waiting virtual machines to the competing set as possible.

Therefore, the scheduler controls the level of multiprogramming in effect in the real machine at any given time. Using its virtual machine dispatching function, CP allocates the processing power of the available real processors on a time-sliced basis to the set of virtual machines that are currently permitted to operate concurrently.

The dispatch time slice is computed at CP initialization, but can be queried and changed at any time. While it is difficult to predict changes in system behavior resulting from changes in the dispatch time slice, you may want to make small changes based on the workload characteristics of your system.

To query the current time slice, use the following command:

```
QUERY SRM DSPSLICE
```

To change the current time slice, use the following command:

```
SET SRM DSPSLICE
```

The scheduling and dispatching routines that are implemented in CP are designed to favor:

- **Interactive (conversational) virtual machines over noninteractive (batch-oriented) virtual machines.** Scheduling and dispatching priority is given to interactive virtual machines so that good response time can be provided to users entering commands at displays.
- **Installation-specified virtual machines.** The installation can explicitly favor certain virtual machines over others. z/VM commands can be used to designate quick dispatch virtual machines and assign high absolute or relative shares to selected virtual machines.

When CP is ready to add one or more logged-on virtual machines to the set of virtual machines that is currently competing for use of the real processor, waiting virtual machines that are currently favored are considered for addition before those that are not.

Note: For the purposes of this scheduling discussion, the terms “virtual machine” and “virtual machine definition block” are generally synonymous. However, a multiprocessing virtual machine has multiple virtual machine definition blocks in its virtual configuration to represent multiple virtual processors. The scheduling of virtual multiprocessors is discussed in more detail in “Scheduling Virtual Multiprocessors” on page 13.

Scheduling and Dispatching Lists

In order to schedule and dispatch the set of virtual machines that are logged on at any given time, CP groups virtual machines according to their current execution characteristics and resource requirements. Grouping is accomplished by creating and maintaining three lists of the virtual machine definition blocks of logged-on virtual machines: the *dormant list*, the *eligible list*, and the *dispatch list*.

The dormant list consists of the logged-on virtual machines that are not currently being considered for dispatching because one of the following:

- They have no work to do (are idle).
- They are waiting for the completion of a long event (such as an I/O operation to a tape drive or a response to a console read on the virtual operator's console).

On receipt of work to do or on completion of the long event, virtual machines in the dormant list enter the eligible list, where they may wait to enter the dispatch list.

The eligible list consists of the logged-on virtual machines that are not currently being considered for dispatching because to do so would cause the multiprogramming capacity of one or more real system

resources (such as paging or storage) to be exceeded. The eligible list forms when there is more demand by virtual machines for real system resources than is available. Therefore, the virtual machines in the eligible list are ready to run and are waiting to enter the dispatch list.

The dispatch list consists of the logged-on virtual machines that are currently in the set of virtual machines being considered for dispatching. That is, the virtual machines in the dispatch list are competing for use of the real processor(s) and all other scheduled system resources.

CP keeps a record of the amount of pageable real storage that is currently available (the number of pageable page frames in the dynamic paging area) and the amount of pageable real storage each virtual machine is expected to use based on past executions. Using these measurements and a set of storage commitment limits that can be changed by the SET SRM STORBUF command, the scheduler controls the number of virtual machines that are part of the dispatch list at any given time in an attempt to prevent a thrashing condition.

CP also keeps a record of the total paging capacity of the system in terms of the number of heavily paging virtual machines, called *loading users*, the system can support. CP determines whether each virtual machine is a loading user based on its history. Using these measurements and a set of loading user limits that can be changed by the SET SRM LDUBUF command, the scheduler controls the number of loading users in the dispatch list at any given time in an attempt to avoid overloading the system paging devices.

Finally, CP keeps track of the number of virtual machines in the dispatch list in different transaction classes. Using these numbers and a set of limits that can be changed by the SET SRM DSPBUF command, the scheduler controls the number of users in the dispatch list at any given time. A subset of the dispatch list exists for virtual machines that have exceeded their maximum share limit. This subset is called the limit list. These controls can be used to overcommit or undercommit the use of processing and I/O resources in a general way.

Scheduling Overview

When a virtual machine is logged on, it is placed in the dormant list. It is moved to the eligible list by the scheduler only when it has work to do.

When a virtual machine enters the eligible list, it is assigned a priority for entry to the dispatch list (the eligible priority). This priority is based on the virtual machine's share, its resource requirement, and the contention for resources in the system.

As resources become available, the scheduler moves virtual machines from the eligible list to the dispatch list. When a virtual machine enters the dispatch list, it is assigned a dispatch priority. Periodically, the dispatcher sorts the virtual machines in the dispatch list into smaller lists for each real processor. These smaller lists are called *dispatch vectors*. The dispatch vectors are kept in ascending dispatch priority order and are used by the dispatcher to select virtual machines to run. As virtual machines consume processor time in the dispatch list (during their allotted elapsed time slices), they are examined and reassigned priority as their dispatch time slices end. Because a virtual machine consumes a given amount of processing or storage resource, becomes idle, or is preempted in favor of certain virtual machines in the eligible list, it moves back to the eligible list (if it is still dispatchable) or to the dormant list (if it is not still dispatchable).

The z/VM scheduler controls the cycling of virtual machines through the three lists. The scheduler consists of a set of routines that calculate the dispatch and eligible priorities, select virtual machine definition blocks to be moved from one list to another, and move the virtual machine definition blocks among the lists. Figure 1 on page 8 illustrates the lists maintained by the scheduler and indicates the flow of virtual machines from one list to another.

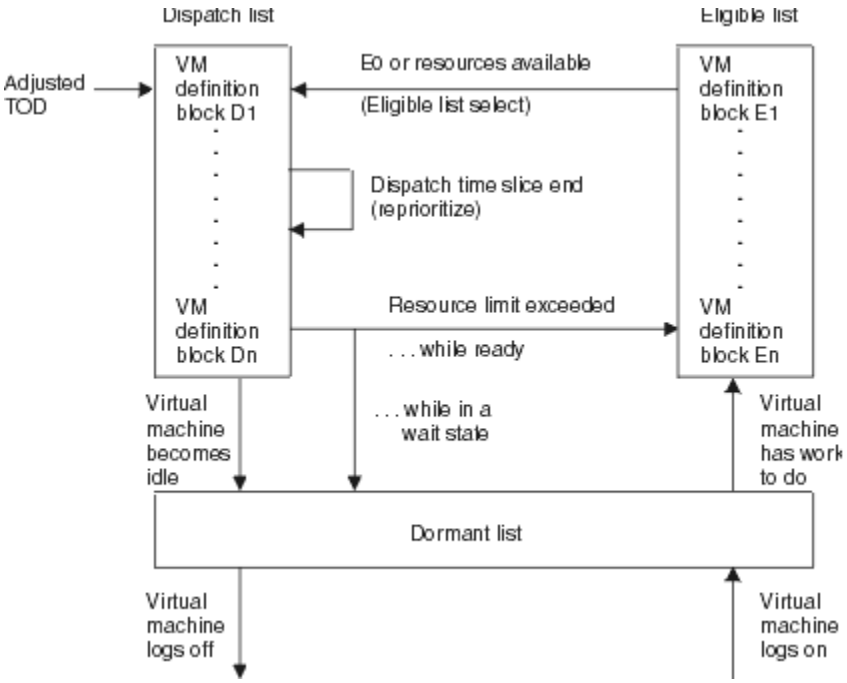


Figure 1. Virtual Machine Scheduling Flow

Dormant List

The virtual machines in the dormant list are in one of the following states:

- **Idle.** For example, a virtual machine awaiting an unsolicited interrupt from a display is idle and is therefore placed in the dormant list.
- **In an enabled wait state.** When a virtual machine loads a wait state PSW (to wait for a timer external interrupt, for example), it is placed in the dormant list.
- **Waiting for the completion of a long event.** For example, if elapsed-time-slice-end occurs for a virtual machine that is waiting in the dispatch list for a page-in operation to be performed, the virtual machine is placed in the dormant list until the page-in is complete.

Eligible List

Note: CP's virtual processor management has been improved so that no virtual machine stays in the eligible list more than an instant before being added to the dispatch list.

The virtual machines in the eligible list belong to one of four transaction classes, based on whether they are to wait in the eligible list at all (E0 virtual machines) or on the expected length of their transactions (E1, E2, and E3 virtual machines).

A transaction is the basic unit of work in z/VM. Typically, a transaction consists of an exchange between a user (for example, when a user presses ENTER) and CP (which processes the user's input and displays a response).

The end of one transaction and the beginning of another can be difficult to distinguish because a virtual machine can enter a wait state for a variety of reasons. Accordingly, the z/VM scheduler defines the start of a transaction as follows: a virtual machine is starting a new transaction if it returns to the dispatchable state after being in a wait state with no outstanding I/O for more than 300 milliseconds. If it returns in less than 300 milliseconds, it is continuing its previous transaction.

The four transaction classes are:

- **E0.** These virtual machines do not wait in the eligible list for resources to become available. E0 virtual machines include quick dispatch virtual machines, virtual machines with “hot-shot” transactions, and “lock-shot” virtual machines.

Lock-shot virtual machines hold certain CP locks. They are placed immediately in the dispatch list and remain until the locks are released. The lock-shot mechanism prevents these CP locks from being held while virtual machines wait for resources in the eligible list.

- **E1.** These virtual machines are expected to have short transactions. They have just started their transactions.
- **E2.** These virtual machines are expected to have transactions of medium length. They are virtual machines who have dropped to the eligible list at elapsed-time-slice-end without having finished their transactions during an E1 stay in the dispatch list.
- **E3.** These virtual machines are executing long-running transactions. They have dropped to the eligible list at elapsed-time-slice-end without having finished their transactions during at least one E2 stay in the dispatch list. Thus, E3 virtual machines have had at least two stays in the dispatch list (an E1 stay and an E2 stay) without having finished their transactions.

The virtual machines in the eligible list are maintained in ascending eligible priority sequence. First-in, first-out queuing is used within the eligible list when eligible priorities are equal.

Eligible priority is calculated for a virtual machine when it is placed in the eligible list. The eligible priority is a deadline priority that represents the time by which the virtual machine should be selected to enter the dispatch list. The relative priorities assigned to virtual machines are designed to:

- Slow down virtual machines that require highly-demanded resources and favor virtual machines requiring less-demanded resources, thus reducing contention for resources in high demand.
- Deliver to virtual machines their shares of available system resources. Virtual machines with larger shares are favored so that they wait no longer in the eligible list than their shares dictate.
- Control the amount and type of service given to virtual machines in each transaction class. E2 and E3 virtual machines wait longer in the eligible list but receive longer times (elapsed time slices) in the dispatch list. This allows for both the efficient use of system resources and the rapid completion of interactive work.

Figure 2 on page 10 shows the use of the eligible list by the scheduler.

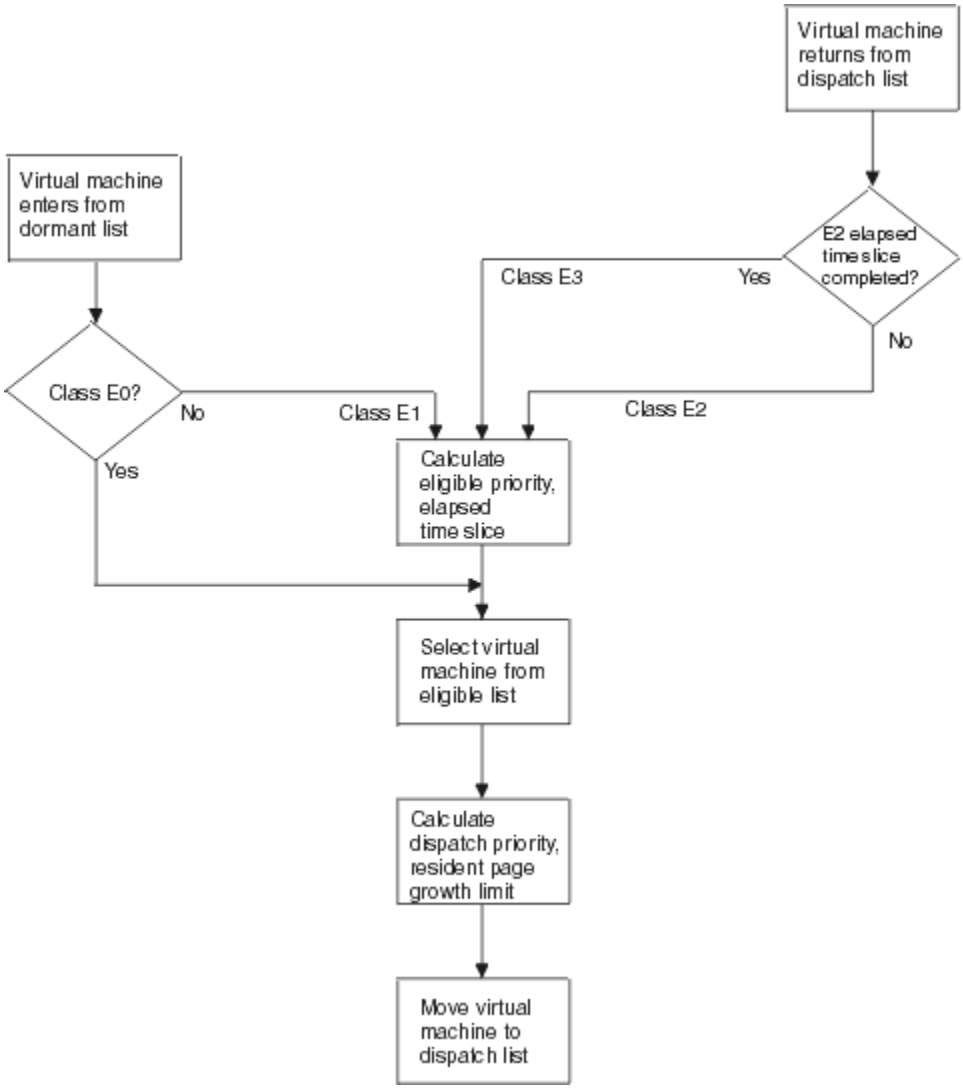


Figure 2. Use of the Eligible List by the z/VM Scheduler

Dispatch List

Note: CP's virtual processor management has been improved so that no virtual machine stays in the eligible list more than an instant before being added to the dispatch list.

The virtual machines in the dispatch list can be in two states: dispatchable or nondispatchable. A dispatchable virtual machine is one that is ready to use a real processor. A nondispatchable virtual machine is one that is not ready to use a real processor because it is waiting for a resource (completion of a page-in operation) or the completion of an activity (CP processing of a Start I/O or Start Subchannel instruction, CP simulation of a privileged instruction, or the termination of an I/O operation).

A virtual machine in the dispatch list is marked nondispatchable when it enters a wait state (is waiting for a paging I/O operation, Start I/O instruction initiation, privileged instruction simulation, or I/O operation). A virtual machine in the dispatch list is marked dispatchable when the operation for which it is waiting completes and it is, therefore, ready to run.

Because virtual machines frequently switch between the dispatchable and nondispatchable states, they are left in the dispatch list when they become nondispatchable to eliminate the frequent processing that would be required to remove them from and return them to the dispatch list.

Virtual machines in the dispatch list retain the transaction class they were assigned while waiting in the eligible list. When E0 virtual machines enter the dispatch list, they are included in the count of Q0 virtual machines displayed by the class E INDICATE LOAD command. Thus, E0 virtual machines are called

Q0 virtual machines while they are in the dispatch list. Similarly, Q1, Q2, and Q3 virtual machines are virtual machines that on entry to the dispatch list belonged to the E1, E2, and E3 transaction classes, respectively.

A maximum limit of system resources may be set for a virtual machine. If this limit exists and the virtual machine has exceeded the limit, the virtual machine is grouped in a subset of the dispatch list. This subset is known as the limit list.

The virtual machines in the dispatch list are sorted periodically into smaller lists called dispatch vectors. There are two types of dispatch vectors: the master-only dispatch vector and the processor local dispatch vector. Each real processor selects work from its own processor local dispatch vector, except the master processor, which first selects work from the master-only dispatch vector, then from its own processor local dispatch vector, and finally from the local dispatch vectors of other processors. A dispatch vector contains virtual machines that are dispatchable on entry to the dispatch list and nondispatchable virtual machines in the dispatch list that become dispatchable. Those dispatchable virtual machines whose work can be performed only on the master processor are placed in the master-only dispatch vector.

The dispatcher inspects the dispatch vector for the processor on which it is currently running when the processor is available to be allocated. Virtual machines are queued in the dispatch vectors in ascending dispatch priority.

The dispatch priority of a virtual machine is a deadline priority that represents the time of day by which the virtual machine should complete its next dispatch time slice under ideal circumstances. Dispatch priority is calculated for a virtual machine when it enters the dispatch list and when dispatch-time-slice-end occurs.

The lower the dispatch priority, the closer a virtual machine is to the beginning of the dispatch vector, and the sooner it will be dispatched. Through biasing, interactive virtual machines and more I/O-oriented, noninteractive virtual machines are queued before the more processor-oriented, noninteractive virtual machines in the dispatch vectors and, hence, have a higher priority for dispatching. Because dispatching priorities are dynamically calculated, the sequence of the virtual machines in the dispatch vectors varies according to the changing operating characteristics of the virtual machines in the dispatch list.

Figure 3 on page 12 shows the use of the dispatch list by the scheduler.

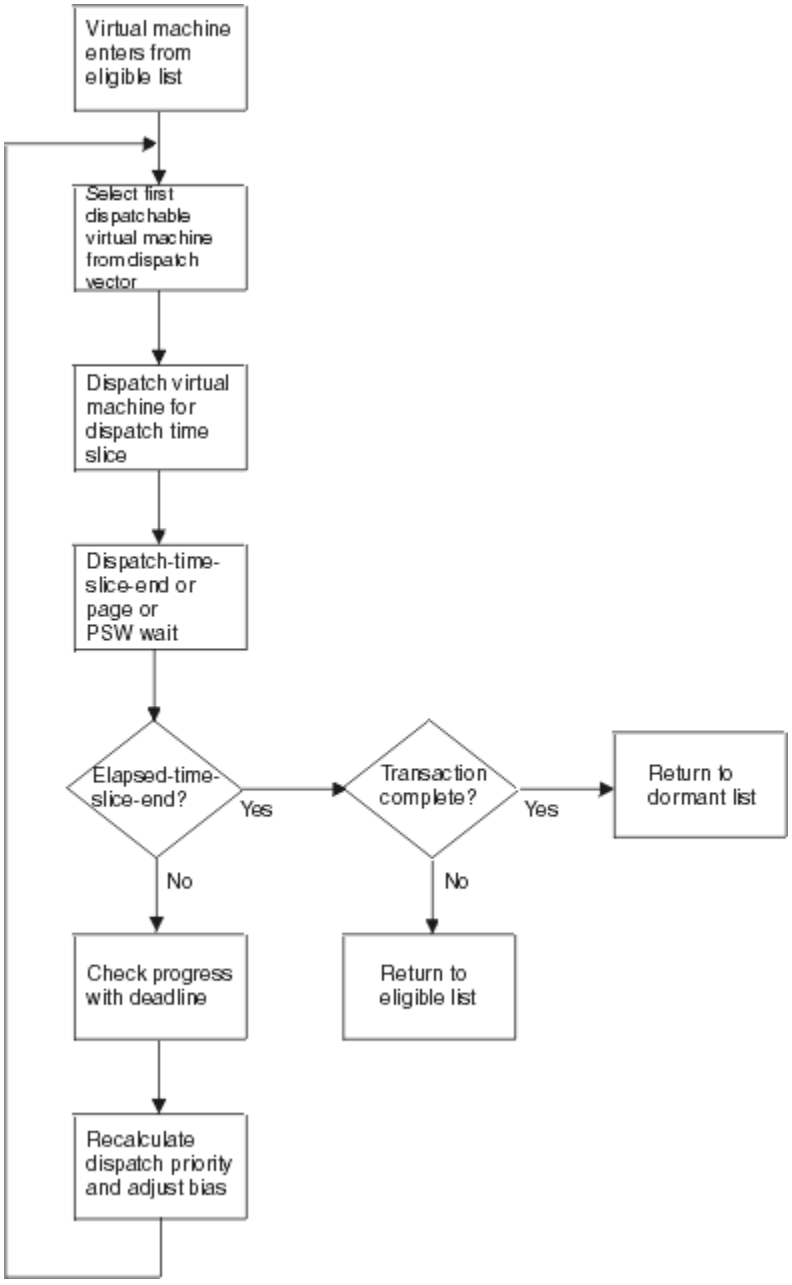


Figure 3. Use of the Dispatch List by the z/VM Scheduler

Summary of Scheduling and Dispatching Lists

The lists that are maintained by the virtual machine scheduling and dispatching routines are summarized in Table 1 on page 12.

Table 1. Lists Used by the Virtual Machine Scheduling and Dispatching Routines

List	Virtual Machines
Dormant List	Virtual machines that are idle, in an enabled wait state, or waiting for the completion of a long event

Table 1. Lists Used by the Virtual Machine Scheduling and Dispatching Routines (continued)

List	Virtual Machines
Eligible List	Virtual machines of the following types, maintained in ascending eligible priority order: **E0** Virtual machines that do not wait in the eligible list (quick dispatch, hot-shot, and lock-shot) **E1** Virtual machines with short transactions that are waiting for a resource **E2** Virtual machines with medium-length transactions that are waiting for a resource **E3** Virtual machines with long transactions that are waiting for a resource
Dispatch List	Dispatchable virtual machines and nondispatchable virtual machines whose waits are expected to be short
Dispatch Vectors	Virtual machines from the dispatch list, sorted by real processor for dispatching and maintained in ascending dispatch priority order

Scheduling Virtual Multiprocessors

In z/VM, a virtual machine can have more than one virtual processor and, therefore, more than one virtual machine definition block. The base virtual machine definition block, which is created when the virtual machine logs on, is in the global cyclic list of all logged-on virtual machines. The additional virtual machine definition blocks, which represent virtual processors other than the base, are chained to the base on the virtual machine's local cyclic list.

Both base and additional virtual machine definition blocks cycle through the three scheduler lists. However, because the base definition block owns the virtual storage and most of the other virtual resources for the virtual machine as a whole, it is tied to its adjunct definition blocks in the following way: in the hierarchy of lists (where the dispatch list is higher than the eligible list, and the eligible list is higher than the dormant list), the base definition block must be in a list higher than or equal to the highest list occupied by one of its adjunct definition blocks.

For example, a virtual machine with two virtual processors has one base and one adjunct definition block, both of which start in the dormant list. Suppose the adjunct definition block becomes dispatchable. Both the base and the adjunct definition blocks are moved up to the eligible list. After a certain wait, they are placed together in the dispatch list and given the same priority. However, as the adjunct definition block consumes resources, it receives different treatment than the base definition block receives (it is assigned different intermediate dispatch priorities). This is the only way in which a base definition block can reach the dispatch list without ever having been dispatchable.

By tying the base definition block to its adjunct definition blocks, the scheduler ensures that the resource requirements of the virtual machine as a whole are considered when scheduling a virtual machine's definition blocks. However, the processor time consumed by an adjunct definition block (a resource whose consumption is measured separately) is factored into the scheduling of that block while it is in the dispatch list.

Entering the Eligible List

Virtual machines enter the eligible list from either the dormant list or the dispatch list. All non-E0 virtual machines that have been dormant for more than 300 milliseconds (the length of a transaction) enter the eligible list from the dormant list as E1 virtual machines. In essence, the scheduler classifies all virtual

machines beginning a new transaction (moving from the dormant to the eligible list) as interactive. They are classified as E1 virtual machines to ensure that any initial delay in the eligible list is relatively short.

Should a virtual machine's transaction last longer than one stay in the dispatch list, the virtual machine becomes progressively less interactive (moves from E1 to E2 status, and then to E3 status, if necessary).

Elapsed Time Slice

When a virtual machine enters the eligible list, it is assigned an elapsed time slice, which is the amount of time the virtual machine is allowed to remain in the dispatch list. The elapsed time slice assigned to a virtual machine depends on its transaction class. During system initialization, CP sets an initial value of 1.2 seconds for the E1 elapsed time slice. During system operation, this initial value is adjusted dynamically. As E1 virtual machines are dispatched, the scheduler keeps track of the number of virtual machines that complete their transactions during their elapsed time slices. As the number of virtual machines that complete their transactions during their E1 elapsed time slices increases, the size of the E1 elapsed time slice decreases. As the number of E2 virtual machines increases (because the number of virtual machines that do not complete their transactions during their E1 elapsed time slices increases), the size of the E1 elapsed time slice increases.

If the number of dispatched or eligible E1 virtual machines drops below a threshold and the number of dispatched or eligible E3 virtual machines rises above another threshold, the E1 elapsed time slice also increases. This allows a system with predominantly large jobs to work more efficiently. The E1 elapsed time slice must be a value between 50 milliseconds and 16 seconds, but it is allowed to reach an equilibrium within this range.

The E0, E2 and E3 elapsed time slices are fixed multiples of the (varying) E1 elapsed time slice; the E0, E2, and E3 multipliers are 6, 6, and 48, respectively. The E3 elapsed time slice is the larger of the following:

- A fixed multiple of the E1 elapsed time slice
- A fixed multiple of the time required to page in the E3 virtual machine's working set.

Eligible Priority

When a virtual machine enters the eligible list, it also receives an eligible priority. The eligible priority is calculated based on the current time-of-day and a delay that represents the amount of time the virtual machine should wait in the eligible list. This delay is the product of the elapsed time slice assigned to the virtual machine (based on its transaction class) and an *eligible list delay factor*.

The eligible list delay factor represents the ratio of the time the virtual machine must wait in the eligible list to the time it is allowed to remain in the dispatch list. The eligible list delay factor is a function of:

- The virtual machine's share of the system
- The amount of each scheduled resource that the virtual machine requires compared with an “average” virtual machine
- The current system load on each scheduled resource
- The current service the virtual machine receives while in the dispatch list (the *dispatch list expansion factor*).

Leaving and Entering the Dispatch List

Virtual machines alternate between the dispatch list and the eligible and dormant lists as follows: a virtual machine in the dispatch list is permitted to remain there for its elapsed time slice, which is assigned when it enters the eligible list. During the elapsed time slice interval, the virtual machine runs for one or more dispatch time slice intervals, according to its dispatch priority. Assuming no event occurs that causes the virtual machine to move to the eligible or dormant list, the virtual machine waits in the dispatch list during periods of its elapsed time slice when it is not running. As soon as the assigned elapsed time slice interval expires, the virtual machine is dropped from the dispatch list and placed in the eligible list in eligible priority sequence.

When dropped from the dispatch list at the end of its elapsed time slice end without completing its transaction, an E1 virtual machine becomes an E2 virtual machine, and an E2 virtual machine becomes an E3 virtual machine. An E3 virtual machine cycles between the dispatch list and the eligible list as an E3 virtual machine until its transaction is complete.

A virtual machine is dropped from the dispatch list and placed in the dormant list if it enters an enabled wait state (becomes idle) before its elapsed time slice ends or if is not dispatchable when its elapsed time slice ends.

Selecting Work for Dispatching (the Dispatcher)

The dispatcher is entered upon completion of processing of any first-level interrupt handler or any other unit of work. The dispatcher first performs an accounting function and, if possible, redispatches the last-dispatched virtual machine. The fast redispatch path is possible except under the following conditions:

- A CP routine has requested preemption on the processor, or a virtual machine with a higher priority has become ready.
- An event has occurred that requires a call to the scheduler.
- The last virtual machine dispatched is no longer ready.

If the fast redispatch path cannot be taken, the dispatcher must “undispatch” the last-dispatched virtual machine. The dispatcher calls the scheduler, which adjusts the virtual machine's scheduling state and either moves the virtual machine to the correct scheduling list or repositions it in the dispatch list and dispatch vector, as required.

When the undispatch function is complete, the dispatcher looks for new work to do for the processor on which it is running. First, the dispatcher looks to see if there is any emergency system work that needs to be done to handle a machine check, malfunction alert, or emergency signal. If there is, the dispatcher passes control to the appropriate routine.

If there is no emergency work to be done, the kind of work the dispatcher selects next depends on the type of processor on which it is running. If the dispatcher is running on the master processor, the dispatcher searches for runnable CP work, which is represented by any I/O request blocks, timer request blocks, CP execution blocks, or save area blocks stacked on the system virtual machine definition block.

To handle CP work, the dispatcher (on the master processor) unstacks the block that represents it, dispatches the system virtual machine definition block, and passes control to the routine specified in the block. Upon completion of CP work, control returns to the dispatcher.

Alternate processors bypass the search for CP work that can be run and do not dispatch the system virtual machine definition block. Only the master processor can select the system virtual machine definition block for dispatching.

On the master processor, when there is no CP work to do, and on all other processors, the dispatcher selects a virtual machine from a dispatch vector. The scheduler maintains the dispatch vectors in dispatch priority order, so the dispatcher tries to select a ready virtual machine from the top of the appropriate dispatch vector. The order in which the dispatch vectors are searched depends on the type of processor on which the dispatcher is running. On the master processor, the dispatcher looks for a virtual machine in the following order:

1. Master-only dispatch vector
2. Master processor's own local dispatch vector
3. Local dispatch vectors of other processors.

On an alternate processor, the dispatcher tries to select a virtual machine first from the alternate processor's own dispatch vector, and then from the local dispatch vectors of other processors.

Because the dispatcher finds a ready virtual machine on a dispatch vector with work the processor can do, it selects the virtual machine for dispatching. If there is system work to perform on behalf of the virtual machine as represented by a stacked I/O request block, timer request block, or CP execution block, the dispatcher unstacks the block and passes control to the CP routine indicated in the block. Upon completion of the work, control returns to the dispatcher.

If there is no CP work to perform for the virtual machine, the virtual machine is given control of the real processor by way of the Start Interpretive Execution (SIE) instruction. The virtual machine runs in interpretive-execution mode until a condition occurs that causes an interrupt or intercept. The dispatcher then regains control.

If after all searching there is no work for a processor to do, it loads an enabled-wait-state PSW. Either an interrupt or some other processor will wake up this processor when there is work to be done.

Selecting Virtual Machines to Be Added to the Dispatch List (the Scheduler)

When the dispatcher calls the scheduler to undispatch a virtual machine, the scheduler adjusts the status of the virtual machine and makes appropriate changes to the lists it maintains. After making the required changes, the scheduler enters a procedure to determine whether any virtual machine can be added to the dispatch list.

The scheduler looks at the first virtual machine in the eligible list and determines whether it can be added to the dispatch list. If it can be added, a dispatch priority and a resident-page growth limit are calculated, and the virtual machine is placed in the dispatch list. The scheduler then attempts to add the next virtual machine in the eligible list using the same procedure. As many virtual machines as can be added to the dispatch list, without exceeding the multiprogramming capacity of one of the real system resources, are moved from the eligible list to the dispatch list.

Virtual machines in the eligible list are selected for entry into the dispatch list as follows:

- E0 virtual machines are selected immediately without regard to their resource requirements.
- An E1, E2, or E3 virtual machine is selected if it meets the following resource requirements:
  - Its projected working set size fits into real storage in accordance with the percentages specified by the SET SRM STORBUF command (or the initial percentages).
  - Adding the virtual machine does not violate limits on the number of loading virtual machines allowed per transaction class specified by the SET SRM LDUBUF command (or the initial limits).
  - Adding the virtual machine does not violate limits on the number of virtual machines allowed in the dispatch list per transaction class specified by the SET SRM DSPBUF command (or the initial limits).

When selecting non-E0 eligible list virtual machines for entry into the dispatch list, the scheduler attempts to ensure that the requirements for storage and paging resources by all of the virtual machines in the dispatch list do not exceed the storage and paging resources available in the system. It also tries to ensure that the number of virtual machines in the dispatch list do not exceed the limits in effect.

First, the scheduler checks to see if the virtual machine's projected working set size fits into the real storage allocated to its transaction class. The total available real storage consists of those pages in the dynamic paging area that are not locked, shared, or reserved.

This storage resource can be allocated to virtual machines of different transaction classes by means of the class A SET SRM STORBUF command. For more information on the SET SRM STORBUF command, see page SET SRM STORBUF .

The *working set size* of a virtual machine is a projection of the number of pages that must occupy real storage in order to process a virtual machine's transaction efficiently (that is, with a minimum number of page faults). The working set size is based on the virtual machine's run history and is calculated each time the virtual machine is dropped from the dispatch list.

To an E1 virtual machine, which has no history for its current transaction, the scheduler assigns an average E1 working set size. An E2 or E3 virtual machine's working set size is the larger of:

- The number of pages that were resident when the virtual machine was added to the dispatch list plus the number of page reads
- The number of pages that are resident when the virtual machine is dropped from the dispatch list.

The virtual machine's working set size is compared to the sum of the working sets of the virtual machines in the dispatch list that belong to the relevant transaction classes. If the virtual machine's projected

working set size fits into the real storage in accordance with the percentages established by the SET SRM STORBUF command, the virtual machine can be selected (provided it passes the paging resource test).

Whenever a virtual machine cannot fit into real storage and is behind schedule, no virtual machine of the same transaction class or below can be added to the dispatch list. Therefore, an E3 virtual machine can block only other E3 virtual machines, an E2 virtual machine can block E2 and E3 virtual machines, and an E1 virtual machine can block E1, E2, and E3 virtual machines from being added to the dispatch list. In the latter case, the scheduler attempts *dispatch list preemption*.

Dispatch list preemption is intended to preempt noninteractive work in the dispatch list in favor of interactive work that is waiting in the eligible list when real storage is so constrained that there is not enough real storage to support an E1 working set. If the E1 virtual machine is behind schedule by more than a certain amount of time, a flag is set to indicate that dispatch list preemption is required. The scheduler then looks for virtual machines to preempt. If no virtual machines can be preempted, the blockage continues until the behind-schedule E1 virtual machine can fit into real storage (as virtual machines leave the dispatch list in the usual course of events). If enough virtual machines can be preempted, they are removed from the dispatch list and the E1 virtual machine is added to the dispatch list, removing the blockage.

When an E2 or E3 virtual machine is behind schedule and blocking other virtual machines, the scheduler does not attempt preemption. Instead, the virtual machine waits until all virtual machines of its transaction class have left the dispatch list normally. The behind-schedule virtual machine is then placed in the dispatch list, even if it does not fit into real storage.

For example, if an E2 virtual machine is behind schedule, the scheduler stops adding E2 and E3 virtual machines to the dispatch list. As E2 and E3 virtual machines leave the dispatch list, they begin to make room for the waiting E2 virtual machine. If, after all E2 and E3 virtual machines have left the dispatch list, the waiting E2 virtual machine still requires more real storage than is available, it is placed in the dispatch list anyway.

In addition to real storage, the scheduler considers the paging multiprogramming level when selecting virtual machines in the eligible list for entry into the dispatch list. This ensures that the number of virtual machines paging heavily (*loading users*) at any one time is consistent with the capacity of the online paging devices.

During system initialization the scheduler calculates the total paging capacity of the system, which is based on the number of logical paths (*exposures*) to devices used for paging. A CP-owned paging volume can have more than one exposure. Based on the number of paging exposures, CP determines the number of loading users the system can support.

The paging resource can be allocated to loading users of different transaction classes by means of the class A SET SRM LDUBUF command. For more information on the SET SRM LDUBUF command, see “Tuning the Paging Subsystem” on page 116.

The scheduler assumes the following virtual machines are loading users:

- They have just logged on
- They read pages in throughout a dispatch time slice
- They had recently referenced pages stolen during a transaction.

When selecting a virtual machine from the eligible list, and at the end of each dispatch time slice, the scheduler determines if the virtual machine will page heavily during the next dispatch time slice. If so, the virtual machine is marked as a loading user.

Counts of the loading virtual machines in the dispatch list by transaction class are also maintained. A loading user cannot be selected if doing so would cause the relevant count to exceed the limits specified by the LDUBUF settings. However, this count can exceed the limit if a virtual machine was not a loading user when it entered the dispatch list, but was marked as such during its stay.

Finally, when neither storage nor paging barriers prevent the entry of a virtual machine to the dispatch list, the scheduler considers the overall multiprogramming level as indicated by the number of virtual machines in the dispatch list.

The class A operator responsible for tuning the system's performance can set limits by transaction class on the number of virtual machines that are to occupy the dispatch list concurrently. By means of the SET SRM DSPBUF command, the class A operator can ensure that the total number of virtual machines running at any one time is consistent with the capacity of the system as a whole and, in particular, with the capacity of the processing and I/O resources, which are otherwise unscheduled. For more information on the SET SRM DSPBUF command, see “Controlling Processor Resources” on page 113.

Counts of the number of virtual machines in the dispatch list by transaction class are maintained. A virtual machine cannot be selected if doing so would cause the relevant count to exceed the limits specified by the DSPBUF settings. The limits apply only to E1, E2, and E3 virtual machines. E0 virtual machines enter the dispatch list immediately without regard to the DSPBUF settings.

Note that if the requirements for real storage and paging resources of the current set of logged-on virtual machines are less than or equal to the real storage and paging capacity of the real machine, and if entry to the dispatch list is not restricted by the DSPBUF settings, all virtual machines that are ready to run will be in the dispatch list and the eligible list will be empty after each processing by the scheduler.

Because an eligible list virtual machine is selected to be moved to the dispatch list, a dispatch priority and a resident-page growth limit are calculated.

Dispatch Priority

The dispatch priority is calculated for a virtual machine each time it enters the dispatch list and whenever the dispatch time slice assigned to a virtual machine expires during its execution. The dispatch priority is calculated based on:

- The current adjusted time-of-day. For the purpose of calculating dispatch priority, CP calculates an adjusted time-of-day by subtracting accumulated system overhead from the real time-of-day. (*System overhead* is CP work that is not associated with a particular virtual machine and consists of such tasks as paging, interrupt handling, scheduling, and dispatching.) The adjusted TOD clock runs at the same rate as the real TOD clock, but it stops when CP is doing overhead work.

  The adjusted time-of-day is used so that the installation can allocate system resources to virtual machines without having to reserve resources for system overhead, which tends to vary widely. When an installation allocates an absolute share of 50% to a virtual machine, the virtual machine receives half of the system resources available after the resources used for system work have been taken out. The adjusted time-of-day, used to calculate the dispatch priority of the virtual machine, reflects this net approach.
- An offset that represents the amount of time it should take the virtual machine to complete its next dispatch time slice. This offset is calculated based on the size of the dispatch time slice, the virtual machine's normalized share of the system, any delay the virtual machine has already experienced in the eligible list, and the number of real processors available to deliver service.

  For more information on virtual machine shares see “Scheduling Share Option” on page 32.

  A feedback mechanism is also used to adjust the dispatch priority given to a virtual machine during its second and subsequent dispatch time slices. The mechanism increases the offset for a virtual machine that finishes its dispatch time slice earlier than expected and decreases the offset for a virtual machine that finishes its dispatch time slice later than expected.
- Any biases (interactive or hot-shot) that apply to the virtual machine. These biases are discussed further in “Biased Scheduling” on page 20.

Lastly, in making scheduling decisions, z/VM uses only a VMDBK's recent dispatch history. What happened long ago is routinely forgotten. In this way the scheduler reacts to what is happening right now in the system, not what happened ages ago. This strategy helps the scheduler to make decisions that are correct for the users' current demands for CPU.

Resident-Page Growth Limit

The scheduler calculates a virtual machine's working set size in order to project the virtual machine's requirement for real storage. When real storage is at a premium, the scheduler needs to be able to

recognize a virtual machine whose working set size is growing, so that the virtual machine does not use more than its fair share of real storage. The scheduler does this by calculating the resident-page growth limit, which the page translation routine compares to the virtual machine's count of resident pages, as follows.

Each time the page translation routine allocates a page frame to a virtual machine's virtual storage page, it updates the virtual machine's count of resident pages, which is then compared to the virtual machine's resident-page growth limit. If the count exceeds the growth limit, the page translation routine calls the scheduler, which recalculates the virtual machine's working set size and checks to see if the virtual machine still fits into available real storage. If the virtual machine no longer fits, it is dropped from the dispatch list.

The resident-page growth limit is calculated when the virtual machine is added to the dispatch list. The limit is the larger of:

- The working set size plus a growth allowance of a small percentage of available real storage
- The virtual machine's resident page count plus a smaller growth allowance.

Dispatching Virtual Machines

When CP is ready to dispatch work on the master processor, it first determines whether there is any system work to do. This determination is made by inspecting the system virtual machine definition block for stacked work represented by I/O request blocks, timer request blocks, CP execution blocks, and save area blocks. If there is system work to do, the block representing it is unstacked, the system virtual machine definition block is dispatched, and control passes to the CP routine specified by the block. A limit on the amount of time the CP routine can run (dispatch time slice) is not established. On all other processors and on the master processor if there is no system work to do, the first ready virtual machine with work that can be done on the processor is dispatched. If there is no work to do, the real processor is placed in an active wait state.

The dispatcher keeps track of:

- The time each processor spends in active wait
- The time each processor spends in interpretive-execution mode running virtual machine work
- The total time each processor spends running virtual machine work, which includes both CP work on behalf of the virtual machine and the time the virtual machine runs in interpretive-execution mode.

Whenever a virtual machine is given real processor control, CP establishes the required states and modes of system operation in the virtual general registers, virtual control registers, and the virtual PSW, as required by the interpretive-execution facility. Because virtual machines always run under control of the interpretive-execution facility, address translation is performed.

Dispatch Time Slice

A virtual machine is assigned a dispatch time slice each time it is dispatched (or a portion thereof when the virtual machine is interrupted and then redispatched after the interrupt is processed). The size of the dispatch time slice is based on the speed of the real processor and represents a fixed number of instructions processed. This value is determined dynamically during system initialization.

The class A operator can change the size of the dispatch time slice with the SET SRM DSPSLICE command.

When a virtual machine enters the set of competing virtual machines and is dispatched for the first time, it is assigned a dispatch time slice. The virtual machine is allowed to run until one of the following events happens:

- The dispatch-time-slice interval expires.
- It voluntarily enters a wait state because no task is ready to dispatch.
- It enters the page-in wait state, Start-I/O-instruction-processing wait state, or I/O-completion wait state.

- It enters CP mode.
- Some other interrupt occurs.

When a virtual machine gives up real processor control, the amount of time it ran is added to the accumulated real processor time used by the virtual machine. If the virtual machine is not the next virtual machine to be dispatched (is not redispatched after the interrupt or intercept), the next time the virtual machine is dispatched, it will be assigned a new dispatch time slice or the unused portion of its previous dispatch time slice, depending on the condition that caused the virtual machine to give up real processor control previously. When the virtual machine is redispatched after an interrupt, it is given the unused portion of its previous dispatch time slice as its execution interval.

Biased Scheduling

The z/VM scheduler uses various bias factors, some of which can be supplied by the installation, in determining virtual machine priority. These bias factors determine the relative importance of the user-assigned priority and operational characteristics of a virtual machine in the calculation of dispatch priority.

Because dispatch priority determines the sequence in which virtual machines are selected to run, selection of appropriate factors can bias the scheduler and dispatcher in favor of virtual machines with certain characteristics at the expense of others.

The commands that are available to help control the way that the scheduler allocates system resources to virtual machines are discussed in detail in “Scheduling Share Option” on page 32.

The biases used in the calculation of dispatch priority for a virtual machine are the *interactive* and *hot-shot* biases.

Interactive Bias

Interactive bias is a method of weighting the service a virtual machine receives so that it receives more service at the beginning of a transaction and less service towards the end of a transaction. Virtual machines with very short transactions receive only the boosted part of the service and, therefore, better response time than they would otherwise receive.

The amount of interactive bias to be assigned to virtual machines can be adjusted by the class A operator by way of the SET SRM IABIAS command.

The bias consists of two elements, an *intensity* (a percentage) and a *duration* (a number of dispatch time slices). The intensity is a percentage of the difference between the virtual machine's usual dispatch priority and that of the virtual machine with the best priority; it indicates the strength of the boost. At the beginning of a new transaction, the scheduler calculates the usual dispatch priority. (The usual dispatch priority includes any paging bias, but excludes any interactive or hot-shot bias). The difference between this priority and that of the virtual machine with the best priority in the dispatch list is then calculated. This difference, called *parity*, is multiplied by the intensity, and the product is subtracted from the virtual machine's priority. (Because the priority is in TOD clock units, a lower value is a better priority and places the virtual machine higher in the dispatch list.) The result is that the scheduler places the virtual machine higher in the dispatch list than it would otherwise have done.

The duration is a count of dispatch time slices; it indicates how long the interactive boost should last before the virtual machine fades back to its usual position in the dispatch list. For example, suppose the interactive bias intensity is set to 60% and the interactive bias duration is set to 3. On entry to the dispatch list, a virtual machine receives an interactive boost of 60%. On the next dispatch time slice, it receives a boost of 40%. On the third dispatch time slice, it receives 20%; and on the fourth, it receives 0%. The virtual machine is then released from interactive bias, and the feedback mechanism immediately starts compensating for the extra service that this virtual machine has received by placing the virtual machine in a lower-than-normal position in the dispatch list for the same number of dispatch time slices. After the specified number of dispatch time slices, the transaction is considered noninteractive and is treated as if it had never received the interactive bias.

Hot-Shot Bias

When the user of a virtual machine interacts with the display (causing an unsolicited interrupt) while a transaction is already in progress, the scheduler marks the virtual machine as a hot-shot virtual machine. The purpose of a hot-shot is to give the virtual machine service fast enough to cause the display to blink immediately, reassuring the user that CP is still operational. Hot-shot service is just long enough to complete trivial, but useful, interactions, such as processing an INDICATE LOAD or DISPLAY PSW command.

The hot-shot virtual machine is given one short dispatch time slice with very good priority. The hot-shot dispatch time slice is shorter than usual to minimize the effects of the boost, but long enough to allow the user of the virtual machine to receive very good response time. The hot-shot bias is calculated in the same way as the interactive bias. However, the intensity of the hot-shot cannot be altered by CP command, and its duration is always 1.

Use of Hardware Timing Facilities

The hardware timing facilities of the real machine are used by CP to support virtual timing and accounting facilities for virtual machines and in the scheduling and dispatching of virtual machines.

The time of day is maintained in the time-of-day (TOD) clock associated with a real processor. When a processor complex has more than one TOD clock, the clocks are synchronized during system initialization. The elapsed time slice, artificial time-of-day, eligible list priority, and dispatch list priority of a virtual processor are calculated using TOD clock values. All time-of-day requests from CP and virtual machines (made by Store Clock instructions) are satisfied using the TOD clock.

z/VM HiperDispatch

The prime objective of z/VM HiperDispatch is to help virtual servers to get good performance from the IBM Z® memory subsystem. z/VM HiperDispatch works toward this objective by managing the logical partition and dispatching virtual CPUs in a way that takes account of the physical machine's organization and especially of its memory caches. z/VM's behavior in this way can help the workload to achieve good performance on the IBM Z hardware.

For a high-level description of z/VM HiperDispatch, see z/VM HiperDispatch in *z/VM: CP Planning and Administration*, which includes an overview of IBM Z CPU and memory hardware, some background information on the PR/SM hypervisor, and descriptions of system administration considerations for HiperDispatch.

For an in-depth technical article that explains the workings of z/VM HiperDispatch, go to the following website: IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

This article discusses IBM Z hardware and firmware, z/VM dispatching heuristics, characteristics or traits that likely make a workload's performance improved by z/VM HiperDispatch, measuring the performance changes that result from using z/VM HiperDispatch, and Performance Toolkit for z/VM updates.

Simultaneous Multithreading (SMT)

With the introduction of simultaneous multithreading (SMT) on the IBM z13®, z/VM can dispatch work on up to two threads of an IFL core. Although IBM SMT support includes IFLs and zIIPs, z/VM supports only IFLs. The primary objective of SMT is to increase work throughput of processor cores by dispatching two threads on one core. When multithreading is enabled and work is dispatched on multiple threads of the same core concurrently, each of the individual threads will be less productive per unit of time due to the need to share core resources. However, on average it is expected that the aggregate work performed by the core is greater than if the tasks needed to take turns using the core without multithreading enabled.

A system that is enabled for multithreading can use the CP SET MULTITHREAD command to change the activated thread count per CPU type to a value between 1 and the maximum without an IPL. The maximum activated thread count is set when multithreading is enabled and hardware does not allow it

to be changed. A system that is enabled for multithreading but has only one activated thread per core performs similarly to a system with multithreading disabled.

For more information about SMT support, see *z/VM: Migration Guide*.

For a performance analysis on how SMT benefited z/VM workloads, see IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

Three Measures of CPU Time When Multithreading Is Enabled

z/VM support for SMT provides three measures of CPU time when multithreading is enabled, because the hardware CPU timer is no longer an indication of core utilization. These three measures of CPU time are described below and are reported by accounting records and monitor records.

Raw Time
: This is a measure of the CPU time each virtual CPU spent dispatched on a thread, and is the CPU timer information provided directly by the hardware. When all cores have only one thread, this is an accurate measure of CPU time used by the task running on the single-threaded core. When multithreading is enabled, and some cores are running with more than one thread, the CPU Timer is no longer a direct indication of physical core consumption, so you might want one of the other times.

MT-1 Equivalent Time
: This is a measure of effective capacity, taking into account the multithreading benefit. The CPU time charged approximates the time that would have been spent if the workload had been run with multithreading disabled; that is, with all core resources available to one thread. The effect is to "discount" the time charged to compensate for the slowdown induced by the activity on other threads in the core.

Prorated Core Time
: This is a measure of core utilization regardless of the multithreading benefit. Time is charged by dividing the time the core was dispatched evenly among the threads dispatched in that interval. Under this method, the total time charged to all guests equals the total time the logical cores of the z/VM partition were dispatched. This method is consistent with cost recovery for core-based software licensing.

Note: When a user is running on a system where multithreading is not installed, not enabled, or enabled and thread level is 1, MT-1 equivalent time and prorated core time consumed will be identical to raw time.

Part 2. Planning for Performance

The topics in this section talk about planning for performance on z/VM and about the performance guidelines for z/VM, SFS, CRR, AVS, and TSAF.

Chapter 3. z/VM Performance Guidelines

This section describes performance factors in a z/VM system that involve problem prevention and tells you about the general performance characteristics of an operating system.

Performance Planning and Administration

The performance characteristics of an operating system are dependent on such factors as hardware, the number of users on the system during peak periods, functions being performed by the system, and how system parameters are set up. You can improve performance to some degree by the choice of hardware and system options. The following general tasks pertain to improving your z/VM system efficiency:

- Plan how you will handle performance monitoring, measurements, improvements, and problems. Become familiar with the CP monitor facility and the facilities you can manipulate to change the performance characteristics of the system as a whole or of selected virtual machines.
- Before you decide which performance options to apply, monitor the system's current performance. This will help you determine which options would most likely give the system a performance gain and where performance bottlenecks are occurring. The CP monitor facility collects such data, which can then be processed to produce statistics to give you an understanding of system operation. Chapter 9, “Monitoring Performance Using CP Monitor,” on page 79 tells you how to use this facility.
- Perform system tuning to do any of the following:
  - Process a larger or more demanding work load without increasing the system configuration
  - Obtain better system response or throughput
  - Reduce processing costs without affecting service to users.

Details on system tuning can be found in Chapter 12, “Tuning Your System,” on page 103.

Performance Considerations

In a z/VM environment, there are two areas of performance to consider: the performance of individual virtual machines and the performance of the total system. CP utilizes both processor and I/O time in order to support a virtual machine environment. The virtual machine itself experiences delays while it is in the eligible list. In view of these two findings, the performance achieved when a specific workload is processed in a virtual machine usually does not equal the performance achieved when the same workload is processed in a real machine in a native environment (assuming the same real machine configuration is used in both instances).

However, in certain instances, the performance achieved in a virtual machine can be better than that achieved in a real machine if a performance facility is used in the virtual machine that was not being used in the real machine. (For example, a guest that uses a z/VM virtual disk in storage could perform better than the same system running on a real machine.)

While it may take longer to process individual workloads in virtual machines, the total system performance achieved in a virtual machine environment may be equal to or better than the performance achieved in a native environment, using the same real machine configuration. This occurs if more work can be completed in a given time period by the use of available real machine resources that are not being used in a native environment.

The percentage of real machine resources currently being used in the native environment is the major factor that affects the total system performance achieved when the same real machine configuration supports a virtual machine environment. Of real machine resources, current processor usage is the most important factor in determining the performance that will be achieved in a virtual machine environment.

Factors that affect the performance achieved in a z/VM environment are discussed, as are steps that can be taken to improve performance. The performance factors presented are those that are unique to a

virtual machine environment and that do not apply to a native environment. These new factors, as well as specific factors that affect the performance of operating systems that are to operate under CP, must be considered when planning for a z/VM installation.

Major factors that affect performance

Certain aspects of the real processor configuration, the size and certain characteristics of the total workload being handled, and the CP performance options and facilities used are the major items that affect performance in a z/VM environment. The interrelationship of the following factors primarily affects total system performance:

- The speed and number of paging devices
- The amount of auxiliary storage made available
- Real storage (memory) size
- Real processor speed
- Characteristics of the workload being processed in each virtual machine
- The number of virtual machines logged on concurrently.

In general, except for options that are designed for a specific operating system (such as additional features of QDIO and Collaborative Memory Management Assist), the CP performance options and facilities that are provided can be used primarily to improve the performance of a small number of individual virtual machines, but often at the expense of other virtual machines. The following topics briefly discuss how the real processor configuration and the characteristics of the workload affect the performance of a z/VM system.

Speed and Number of Paging Devices

Given the availability of a number of different resources, a system can sustain a certain level of paging activity without becoming I/O-bound by paging. Paging activity is affected by:

- The number of paging devices and their speed
- The amount of auxiliary storage allocated on these paging devices
- Command-mode channel programs versus transport-mode channel programs
- The number of HyperPAV aliases available to the Paging Subsystem
- Encryption settings for the paging subsystem.

The level of paging activity achieved is also affected by the amount of contention encountered for the channel paths to which the paging devices are attached and by the contention for the paging devices themselves.

Encrypted Paging improves system security by exploiting hardware on the IBM z14™ to encrypt and decrypt guest pages and VDISK pages which CP writes to disk. Enabling encryption for the paging subsystem increases CPU time spent in CP.

Real Storage (Memory) Size

The amount of paging activity required is affected by the amount of pageable real storage present in the system, among other factors. As more pageable real storage is made available for handling a given workload, less paging activity is required to process that workload. Similarly, more paging activity is required to handle the given workload if less pageable real storage is available.

The number of virtual machines running concurrently affects the amount of pageable real storage available as page frames are taken from the dynamic paging area to satisfy requests for free storage. In addition, use of the locked pages option reduces the total amount of pageable real storage that is available in the system.

The amount of real storage available for paging operations for virtual machines is also affected by the amount of spooling in operation. This includes:

- The number of data transcription operations (input spool file creation and output spool file printing and punching) that are being performed by the real spooling manager, because spool buffers are allocated from the dynamic paging area and locked.
- The number of spool input files that are being read and spool output files that are being written by the virtual spooling manager.

The amount of real storage available for minidisk cache can also affect performance.

If accounting and Environmental Record Editing and Printing (EREP) program records are not retrieved from real storage, they cause more page frames in the dynamic paging area to be allocated as free storage.

Real Processor Speed

An improperly balanced relationship between processor speed and paging device speed can also cause the system to become I/O-bound as a result of paging. As long as there is useful work for the processor to perform while paging operations occur, the system is not kept waiting for paging I/O. However, if the concurrently operating virtual machines are constantly executing instructions faster than the pages they require can be brought into real storage, an excessively high paging rate could develop. Therefore, large-scale processors require both more and, if possible, faster paging devices to handle a specific amount of paging activity than smaller processors.

The z/VM scheduler and dispatching algorithms assume that all processors in the complex have relatively the same processor speed. When running z/VM in a logical partition with LPAR or a similar hardware feature, it is important that all logical processors used by the z/VM system have the same performance characteristics. This is also true when running as a guest of z/VM with multiple virtual processors.

Workload Characteristics

Two characteristics of the workload being processed by the concurrently operating virtual machines affect the total system performance. The first significant characteristic is the way in which virtual storage is used by the programs running in the virtual machines. This affects the paging requirements of the operating virtual machines. The other significant characteristic is the demand made on CP services by these programs. Delays in virtual machine operation can result if the real processor time that CP uses to perform required functions cannot be overlapped with I/O operations in the virtual machine.

Virtual Storage Usage

The total amount of virtual storage a program uses is not nearly as significant a performance factor as is the way in which virtual storage is used. That is, the pattern and frequency of reference to pages in a program have more effect on the number of page faults that occur than does the total size of the program.

For example, assume a case in which a program needs access to 100 KB of virtual storage. If the program can be structured to run as a series of subroutines of four or five pages each, and the pages of each subroutine reference only each other, no more than four or five page frames (16 KB to 20 KB of real storage in a z/VM environment) need to be dynamically available to the program at one time, and paging activity occurs only as the program progresses from one subroutine to the next.

However, assume the program is structured in such a way that during its execution each page of instructions constantly references many different pages of instructions and data for very short durations on a highly random basis. An excessively high paging rate could occur if only four or five page frames were dynamically available to such a program at any time.

Most types of programs naturally have a locality of reference characteristic, so that they can be structured to operate as a series of subroutines. In the simplest case, for example, a program can logically consist of an initialization subroutine, a main subroutine, one or more exception handling subroutines, and a termination subroutine. The total amount of virtual storage referenced in each subroutine usually varies but, generally, the amount is less than the total size of the program. In addition, the pages that are part of (referenced in) a given subroutine can usually be described as active or passive.

An active page is defined as one with a high probability of being referenced multiple times during execution of the subroutine, while a passive page has a low probability of being referenced more than once during execution of the subroutine. A subroutine experiences the least amount of paging activity when its active pages remain in real storage during its execution and its passive pages are paged in when required. A program uses real storage most efficiently when the active instructions and data in each subroutine are contained within the fewest number of pages possible.

The locality of reference characteristic does not apply to certain types of programs. For example, it does not apply to any program that is designed to optimize its performance at run time by using the total amount of storage it has been allocated. This characteristic is usually true of sort/merge programs that initialize themselves to use all the storage made available to them in their partition or region during the sorting passes. The reference pattern for such a sort/merge is random and encompasses all the storage (and, therefore, all the pages) the program is assigned.

CP Overhead

CP uses a certain amount of real processor time to simulate the existence of multiple virtual machines. To the degree that the real-processor-time CP uses to support a given virtual machine cannot be overlapped with I/O operations for that virtual machine, the throughput achieved by an operating system in a virtual machine will be reduced when compared with that achieved in a real machine with the same configuration. The throughput of a virtual machine is also affected by the non-overlapped-processor usage of other virtual machines, as well as that of CP.

A virtual machine specifically makes a demand for a CP service when it attempts to run a privileged instruction that is not handled by the interpretive-execution facility or an assist. The amount of work CP must perform depends on the type of privileged instruction. For example, relatively little processing is required to simulate a Set Storage Key instruction, while a large amount of processing is required to simulate a Start Subchannel instruction. The real processor time CP spends servicing Start Subchannel requests can be one of the most significant causes of reduced performance for virtual machines.

The performance of an individual virtual machine can be improved by reducing the number of privileged instructions that are issued by the virtual machine.

CP performance facilities

CP provides a set of performance facilities that can be used by individual virtual machines specifically to improve their performance. The performance of a virtual machine that uses one or more of these performance facilities usually improves at the expense of reduced performance for other virtual machines, because they must compete for the use of a smaller portion of the real machine resources.

CP provides the following performance facilities (described in more detail in the topics that follow):

- **Processor dedication option.** This facility can be used to reserve a real processor for the sole use of a virtual processor belonging to any virtual machine. This option can be assigned to multiple virtual machines.
- **Virtual machine multiprocessing.** This option, particularly in conjunction with the dedication of real processors to virtual processors, can be used to increase the amount of work that can be done by a virtual machine. This option can be assigned to multiple virtual machines. See “Virtual Machine Multiprocessing” on page 30 for more detailed information.
- **System scheduling control options.** These options can be used to:
  - Change the maximum working set size of a virtual machine allowed in the dispatch list
  - Adjust the size of the dispatch time slice for all virtual machines
  - Allocate more or less storage to virtual machines in different transaction classes
  - Allocate more or less paging resource to heavily-paging virtual machines in different transaction classes
  - Change the maximum number of virtual machines of different transaction classes allowed in the dispatch list

  - Adjust the amount of interactive bias assigned to interactive virtual machines.

  These options affect the performance of all virtual machines. See "System Scheduling Control Options" on page 30 for more detailed information.

- **Scheduling share option.** The scheduling share option can ensure that a virtual machine has priority access to real processor, real storage, and paging resources. This option can be assigned to multiple virtual machines simultaneously. See "Scheduling Share Option" on page 32 for more detailed information.
- **Scheduling maximum share option.** The maximum share option can be used to limit virtual machines from using more than a given amount of resources. See "Scheduling Maximum Share Option" on page 33 for more detailed information.
- **Scheduling maximum share using CPU pools.** CPU pools can be used to set the maximum share of virtual CPU resources for groups of virtual machines. See "Scheduling Maximum Share Using CPU Pools" on page 34 for more detailed information.
- **Quick dispatch option.** The quick dispatch option can be used to ensure that a virtual machine does not wait in the eligible list for resources but is dispatched immediately, whenever it has work to do. See "Quick Dispatch Option" on page 34 for more detailed information.
- **Reserved page frames option.** This facility can be used to maintain a resident set of private pages for a virtual machine or shared pages for an NSS or DCSS. This option can be assigned to more than one virtual machine, NSS, or DCSS at a time. See "Reserved Page Frames Option" on page 35 for more detailed information.
- **Locked pages option.** This facility is provided to cause specific pages of a virtual machine to be permanently locked, so that no paging occurs for these pages. (In this case permanently means until the next IPL.) This facility can be used concurrently by multiple virtual machines. See "Locked Pages Option" on page 36 for more detailed information.
- **Collaborative Memory Management Assist.** The Collaborative Memory Management Assist is a machine feature that allows z/Architecture® guests with the appropriate support to exchange memory usage and status information with z/VM. For more information, see "Collaborative Memory Management Assist" on page 36.
- **Real channel program execution option.** This facility is provided to allow V=V machines to run real channel programs, bypassing CP channel program translation. This facility can be used concurrently by multiple virtual machines. See "Real Channel Program Execution Option" on page 37 for more detailed information.
- **Named saved systems (NSSs).** This facility is provided to reduce the significant amount of CP processing that is required to IPL an operating system in a virtual machine. This facility includes the capability of sharing segments of virtual storage of the NSS among concurrently operating virtual machines on a shared read-only or read-write basis. See "Named Saved Systems (NSSs)" on page 37 for more detailed information.
- **Saved segments.** This facility enables segments of virtual storage that are not part of an NSS to be saved. It enables the virtual storage used by a virtual machine to be dynamically expanded and reduced during system operation without operator intervention and provides for the sharing of reenterable segments by concurrently operating virtual machines. See "Saved Segments" on page 39 for more detailed information.
- **VM/VS handshaking.** This facility can be used to improve the operation of VSE operating systems running in a virtual machine. It includes a facility to reduce the amount of CP processing required to handle BTAM autopolling channel programs. See "VM/VS Handshaking" on page 39 for more detailed information.
- **Interpretive-execution facility.** This processor facility eliminates much of the CP processing that would otherwise be required to simulate most privileged instructions and certain interrupts for a virtual machine by performing these functions by way of hardware. The interpretive-execution facility can be used concurrently by all logged-on virtual machines.
- **Guest wait-state interpretation capability.** This capability allows a virtual processor to remain dispatched even when it enters an enabled wait state.

- **Minidisk caching.** Minidisk caching could provide performance and administrative benefits to z/VM systems. For minidisk caching, CP uses real storage as a cache for virtual I/O data. Accessing the data from electronic storage is much more efficient than accessing DASD. See "Minidisk Cache" on page 40 for more detailed information.
- **VM data spaces.** VM Data Spaces provide increased storage addressability and therefore can move the burden of I/O from an application to the CP paging subsystem. The use of VM data spaces also extends the concept of sharing data. See "VM Data Spaces" on page 43 for more detailed information.
- **File caching option for CP-accessed minidisks.** This option enables CP to cache its frequently used information (such as log message and logo picture files) in storage. See "File Caching Option for CP-Accessed Minidisks" on page 44 for more detailed information.
- **Hot I/O detection.** The hot I/O detection function prevents broken hardware from degrading performance by flooding the system with unsolicited interrupts. See "Hot I/O Detection" on page 45 for more detailed information.
- **Virtual disks in storage.** Virtual disks in storage are FBA minidisks allocated from host real storage instead of on real DASD, which avoids the I/O overhead. See "Virtual Disks in Storage" on page 45 for more detailed information.
- **I/O throttling.** The rate of I/Os issued to a real device can be throttled. This is particularly helpful in environments with shared devices. The rate can be set dynamically or in the system CONFIG file. See "I/O Throttling" on page 46 for more detailed information.
- **Additional features of QDIO.** QDIO virtualization technologies benefit guest operating system I/O and page management. For more information, see "Additional features of QDIO performance" on page 47.
- **Parallel access volumes (PAV) and HyperPAV for guest I/O to minidisks.** CP can take advantage of PAV or HyperPAV technology to increase the performance of guest minidisk I/O. If the DASD subsystem offers alias devices for a real volume that contains guest minidisks, CP can use those alias devices to launch multiple concurrent real I/Os against the real volume, each real I/O corresponding to a virtual I/O a guest has started against a minidisk. In this way, guest I/Os generally experience reduced real-volume queueing and, in turn, decreased response time. For more information, see "Parallel Access Volumes (PAV) and HyperPAV for guest I/O to minidisks" on page 48.
- **HyperPAV for the CP paging subsystem.** CP can take advantage of HyperPAV technology to increase the performance of Paging Subsystem I/O. If the DASD subsystem offers alias devices for a real volume that contains page, spool, directory, or mapped-minidisk pool space, CP can use those alias devices to launch multiple concurrent I/O operations against the volume. This parallelism reduces volume-queuing, and, in turn, decreases response time. For more information, see "HyperPAV for the Paging Subsystem" on page 50.

Virtual Machine Multiprocessing

Defining multiple virtual processors for a virtual machine can increase the amount of work that can be done by the virtual machine. For more information on virtual machine multiprocessing, see *z/VM: General Information*.

System Scheduling Control Options

Scheduling control options can be set for the z/VM system with the SET SRM command. Operands of the SET SRM command provide:

- Control of the intensity and duration of the interactive bias that is to be assigned to interactive virtual machines
- The size of the dispatch time slice
- The percentage of storage (memory) and paging resources that are to be assigned to different transaction classes of virtual machines
- The maximum number of virtual machines of different transaction classes that are to be allowed in the dispatch list concurrently.

Note: CP's virtual processor management has been improved so that no virtual machine stays in the eligible list more than an instant before being added to the dispatch list. Therefore some functions intended to improve performance by managing the eligible list, such as the DSPBUF, LDUBUF, and STORBUF options on the SET SRM command, are now less meaningful.

The class A or E user can display the current settings of these scheduling controls by entering the QUERY SRM command.

The class A SET SRM IABIAS command can be used to change the interactive bias to be assigned to interactive virtual machines. Interactive bias, discussed in Chapter 12, "Tuning Your System," on page 103, is one of the factors that determines virtual machine priority. By assigning a higher interactive bias intensity than the default intensity of 90%, the installation can strengthen the interactive boost given to interactive virtual machines. By assigning a longer interactive bias duration than the default duration of two dispatch time slices, the installation can cause the boost to last longer.

The interactive bias intensity and duration can be increased to provide better service to and therefore improve the response time of interactive virtual machines. Note, however, that increasing the values of interactive bias intensity and duration causes the scheduling shares assigned to virtual machines to become less accurate in determining the amount of system resources a virtual machine is to receive.

The class A SET SRM DSPSLICE command can be used to change the size of the dispatch time slice, whose default value is assigned dynamically during system initialization based on the speed of the real processor. Note that changing the size of the dispatch time slice changes the effect of interactive bias duration as well, because the number of dispatch time slices specified by the duration factor represents a shorter or longer time after the change. The interactive bias duration must be changed by way of the SET SRM IABIAS command if it is to represent the same amount of processing time as before.

The class A SET SRM STORBUF command can be used to allocate the available storage (memory) resource to different transaction classes. This command sets the following parameters for the scheduler:

- If the sum of the working sets of all virtual machines currently in the dispatch list plus the working set of the virtual machine under consideration exceeds this percentage of pageable real storage, the virtual machine is not added to the dispatch list.
- The percentage of pageable real storage the scheduler should consider when determining whether to add an E2 or E3 virtual machine to the dispatch list.
- The percentage of pageable real storage the scheduler should consider when determining whether to add an E3 virtual machine to the dispatch list.

If unchanged by the SET SRM STORBUF command, the STORBUF values after system initialization are 300%, 250%, and 200%, respectively. Recall that Q0, Q1, Q2, and Q3 are the designations for E0, E1, E2, and E3 virtual machines when they are in the dispatch list. The initial STORBUF values can be interpreted as follows:

- The sum of the working sets for all Q0, Q1, Q2, and Q3 virtual machines must be less than or equal to 300% of pageable real storage. While Q0/E0 virtual machines are factored into this equation, they are never delayed in the eligible list.
- The sum of the working sets for all Q2 and Q3 virtual machines must be less than or equal to 250% of pageable real storage.
- The sum of the working sets for all Q3 virtual machines must be less than or equal to 200% of pageable real storage.

Values above 100% for STORBUF are somewhat reflective of the concept of over committing storage (memory) in z/VM. In effect, the initial STORBUF values allow Q0 and Q1 virtual machines to extend over commitment past the level of Q2 and Q3 virtual machines. The same is true for Q2 machines over Q3 machines.

In addition to real storage, the scheduler considers the paging multiprogramming level when selecting virtual machines in the eligible list for entry into the dispatch list. This ensures that the number of virtual machines paging heavily at any time is consistent with the capacity of the online paging devices.

During system initialization the scheduler calculates the total paging capacity of the system and determines the number of *loading users* (virtual machines that page heavily) the system can support.

The paging resource can be allocated to loading users of different transaction classes by means of the class A SET SRM LDUBUF command. This command sets the following parameters for the scheduler:

- The percentage of paging exposures the scheduler should consider when determining whether to add an E1, E2, or E3 loading user to the dispatch list. If the total number of loading users currently in the dispatch list plus the loading user under consideration exceeds this percentage of paging exposures, the virtual machine is not added to the dispatch list.
- The percentage of paging exposures the scheduler should consider when determining whether to add an E2 or E3 loading user to the dispatch list.
- The percentage of paging exposures the scheduler should consider when determining whether to add an E3 virtual machine to the dispatch list.

If unchanged by the SET SRM LDUBUF command, the LDUBUF values after system initialization are 100%, 75%, and 60%, respectively. In effect, the initial LDUBUF values reserve 25% (100%–75%) of the paging exposures for Q0 and Q1 virtual machines and 15% of the paging exposures (75%–60%) for Q2 virtual machines.

The class A SET SRM DSPBUF command can be used to set limits by transaction class on the number of virtual machines that are to occupy the dispatch list concurrently. Thus, it can be used to ensure that the total number of virtual machines running at any one time is consistent with the capacity of the system as a whole, in particular with the capacity of the processing and I/O resources, which are otherwise unscheduled by CP.

The SET SRM DSPBUF command sets the following parameters for the scheduler:

- The number of Q1, Q2, and Q3 virtual machines to be allowed in the dispatch list concurrently. If the total number of virtual machines currently in the dispatch list plus the virtual machine under consideration exceeds this number, the virtual machine is not added to the dispatch list.
- The number of Q2 and Q3 virtual machines to be allowed in the dispatch list concurrently.
- The number of Q3 virtual machines to be allowed in the dispatch list concurrently.

If unchanged by the SET SRM DSPBUF command, all three DSPBUF values after system initialization are 32767. Note that E0 virtual machines enter the dispatch list immediately without regard to the DSPBUF settings.

Finally, the class A SET SRM MAXWSS command can be used to specify an upper limit on how much real storage a user is allowed to occupy. In a heavily used interactive system, some work may be impractical to run because the working set is too large. Running a user with a working set that fills 100%, 90%, 75%, or even 50% of real storage could interfere too much with the interactive workload.

In these cases, the installation should weigh the value of its interactive work against the value of running large users, and determine the largest percentage of real storage that it is willing to give to any single large user. By specifying this percentage on the SET SRM MAXWSS command, you can instruct the system not to let any user (except users with large absolute shares) occupy more than this percentage of real storage, including real storage above the 2 GB line.

Scheduling Share Option

A primary factor in the calculation of both the eligible and dispatch priorities of a virtual machine is the share allocated by the installation to the virtual machine. A virtual machine's share represents the percentage of system resources to which the virtual machine is entitled and can be specified by the installation by way of the SHARE directory control statement or the SET SHARE command.

Two types of shares can be specified:

- **Absolute share.** An absolute share is a percentage of available system resources greater than 0% and less than or equal to 100% that the installation wishes to allocate to this virtual machine. The scheduler reserves 1% of available system resources for virtual machines with relative shares. Therefore, the scheduler reduces a virtual machine's absolute share if the sum of all absolute shares assigned to virtual machines exceeds 99%.

- **Relative share.** A relative share is a value in the range from 1 to 10000, which is meaningful only when compared to the relative shares assigned to other virtual machines. All virtual machines with relative shares compete for the percentage of system resources that remains after system resources have been assigned to all virtual machines with absolute shares. The remaining resources are divided among relative share virtual machines based on their relative values. The default relative share is 100. If necessary, the scheduler reserves 1% of available system resources for virtual machines with relative shares.

Using the shares allocated by the installation to all logged on virtual machines, the scheduler computes each virtual machine's normalized share, which is the resulting percentage of system resources after comparison with all the other logged on users that will be scheduled on the same CPU type. A virtual machine's normalized share is a percentage of system resources in the range of 0% to 100%. The sum of all normalized shares for logged-on virtual machines is 100%.

The following example illustrates the normalized share calculation. Suppose the following shares are specified by the SET SHARE command for a set of virtual machines:

- Virtual machine A has an absolute share of 50%.
- Virtual machines B and C each have absolute shares of 30%.
- Virtual machines D and E each have relative shares of 100.

Although the absolute share virtual machines require 110% of the system resources, only 100% of the system resources are available. The scheduler reserves 1% for the relative share virtual machines, so the available resource percentage is reduced to 99%. Each absolute share virtual machine is scaled down proportionately (99/110, or about 90%). Thus, the virtual machines receive the following normalized shares:

- Virtual machine A receives a normalized share of 45%.
- Virtual machines B and C each receive a normalized share of 27%.
- Virtual machines D and E each receive a normalized share of 0.5%.

Assigning an absolute share to a virtual machine can be used to guarantee the availability of a certain percentage of processing time, storage capacity, and paging capacity to a virtual machine with high requirements for any of these resources. Assigning relative shares to virtual machines can be used to guarantee their relative priority in using system resources. The QUERY SHARE command can be used by the class A or E user to display a virtual machine's assigned share.

Scheduling Maximum Share Option

In addition to the normal share setting discussed above, a maximum share can be set for a virtual machine. The normal share provides a target minimum for percentage of system resources. The maximum share provides a target maximum. Due to the dynamics of the z/VM systems and scheduling, the maximum share target is not always met. The actual usage should be within 5% of the system. The accuracy may be affected by the following:

- Running guest as a virtual MP
- Guest use of DIAGNOSE X'44' or X'9C'
- Running z/VM second level or a LPAR
- Low system utilization

The maximum share can be specified by using the SHARE directory control statement or the SET SHARE command. As with the normal share, the maximum share can be specified as ABSOLUTE or RELATIVE. The limit types that can be specified for the maximum share are:

- NOLIMIT means the user is not limited. This is the default.
- LIMITHARD means the user does not get more than their maximum share.
- LIMITSOFT means the user does not get more than their maximum share, unless no other user can use the available resources.

Scheduling Maximum Share Using CPU Pools

CPU pools can be used to set the maximum share of virtual CPU resources for groups of virtual machines. The DEFINE CPUPOOL command can be used to create a CPU pool with a limit on one type of virtual CPU (virtual CP CPUs or virtual IFL CPUs only). Multiple CPU pools can be defined on the system. One or more virtual machines can be assigned to a CPU pool (a virtual machine can be assigned to one CPU pool at a time) and have their aggregate CPU consumption limited to the pool's capacity.

Two types of resource limits can be set for the group of users in a CPU pool:

- LIMITHARD limits the CPU pool to a specific percentage of the CPUs of the specified CPU type.
- CAPACITY limits the CPU pool to a specific number of CPUs of the specified CPU type.

For more information, see "Using CPU Pools" on page 109.

Quick Dispatch Option

Note: CP's virtual processor management has been improved so that no virtual machine stays in the eligible list more than an instant before being added to the dispatch list. Therefore the quick dispatch option is now less meaningful, although it does get the virtual machine a different size elapsed time slice.

The quick dispatch option is assigned to a virtual machine by the QUICKDSP option on the OPTION directory statement. This option can also be assigned to a logged-on virtual machine by a class A user ID only using the SET QUICKDSP command. This option remains in effect until a class A user ID enters a SET QUICKDSP OFF command. The quick dispatch option can be in effect for multiple virtual machines at the same time.

When the quick dispatch option is assigned to a virtual machine, the quick dispatch virtual machine is added to the dispatch list immediately, whenever it has work to do, without waiting in the eligible list. Events can occur that cause the quick dispatch virtual machine to enter the dormant list. However, as soon as the quick dispatch virtual machine becomes dispatchable again, it enters the dispatch list without waiting in the eligible list.

Because the quick dispatch virtual machines are always in the set of virtual machines being dispatched when not in the dormant list, they are considered for dispatching more frequently than other virtual machines, which may spend time waiting in the eligible list.

When the quick dispatch option is specified for a virtual machine, it is dispatched according to usual dispatching rules (placed in the dispatch vector at its usual priority, for example), but it never waits in the eligible list. The quick dispatch option can be assigned to selected time-sharing virtual machines with critical response time requirements to improve their response time when a large number of virtual machines are operating concurrently. Service virtual machines (in which RSCS and IUCV applications run, for example) and guest operating systems with interactive users (z/VM, for example) are good candidates for the quick dispatch option.

When the reserved pages option is used in conjunction with the quick dispatch option, the virtual machine to which these options are assigned will experience little system overhead to run, further improving response time. Assigning both options can insure fairly stable performance for the virtual machine despite fluctuations in the total system operating environment that could otherwise affect the performance of the virtual machine.

Note that the quick dispatch option is intended for selective use for virtual machines with critical response time requirements. The scheduler always moves a quick dispatch virtual machine immediately into the dispatch list whenever it is ready to run, without regard to the resource requirements of other virtual machines and the current system load. Therefore, indiscriminate use increases response times overall and may disrupt the management of real storage.

The QUERY QUICKDSP command can be used by the class A or E user to determine whether a specified virtual machine is a quick dispatch virtual machine.

Reserved Page Frames Option

The reserved page frames option is set with the CP SET RESERVED command, which can be entered by a user with privilege class A. Reserved page frames can be assigned to multiple logged-on virtual machines and to multiple NSSs and DCSSs (the NSSs and DCSSs do not have to be loaded for the setting to be applied). The command specifies the amount of storage the target entity is entitled to have resident in real storage at all times. You can use the CP QUERY RESERVED command to display the current reserved settings. In effect, this option causes the CP available list replenishment algorithm to skip the virtual machine, NSS, or DCSS whenever it has less resident storage than the reserved setting.

If the total amount of reserved storage is a large percentage of the dynamic paging area, system responsiveness may degrade because CP is limited in the guest storage it can page. When analyzing system capacity, it is suggested you determine a maximum total reserved storage amount and set a RESERVED SYSMAX value on the STORAGE configuration statement or after IPL via the SET RESERVED command.

In addition to the reserved setting, the response to the CP QUERY RESERVED command displays the amount of resident and instantiated storage a virtual machine, NSS, or DCSS has, which may be useful in determining whether the reserved setting is appropriate. When a virtual machine, NSS, or DCSS does not yet have a reserved setting, you may use the INDICATE USER and INDICATE NSS commands to determine the resident and instantiated pages for a virtual machine, NSS, or DCSS.

When a page fault occurs, it is normally handled synchronously. That is, the entire virtual machine waits until that page fault is resolved. (There are two exceptions: (1) Use of the pseudo page fault facility in conjunction with the VM/VS handshaking feature. (2) Use of the PFAULT macro to do asynchronous page fault handling while in access register mode.) If that virtual machine is running on behalf of just one user, this synchronous handling has little significance. However, if that virtual machine is running a multitasking server application, all end users who currently have requests pending to that server are delayed until the page fault is resolved.

Because of this, the reserve page frames option is especially useful for reducing the amount of page faulting that occurs in multitasking server virtual machines. By reserving the working set of each such virtual machine on a z/VM system, paging is, in effect, moved out of those virtual machines where it can delay multiple users and into those virtual machines where it delays just one user.

The VTAM® and VSCS virtual machines are excellent examples of multitasking server virtual machines that can benefit significantly from the use of the reserved page frames option. Minimizing page faults in these particular servers is especially beneficial to end user response time. For each of these servers, consider reserving a number of pages equal to the number of resident pages it uses during peak hours. The INDICATE USER command provides this information.

Other potential uses of the reserved page frames option include:

- It could be assigned to a virtual machine to provide it with generally consistent paging activity and, therefore, performance, because the most active pages of the virtual machine will tend to remain in real storage (as part of the resident working set) regardless of the other activity in the system. This option is suitable for assignment to a batch-oriented production virtual machine, for example, to help ensure that processing is usually completed by a specific time.
- It could be assigned to a response-critical virtual machine whose activity is infrequent or irregular, such as one whose activity is initiated by a remote display. If the reserved page frames option is not assigned to such a virtual machine during periods of inactivity for this virtual machine, its inactive pages are paged out if necessary. Page frame allocation and paging operations are then required in order to reinitiate teleprocessing routines in response to a remote display request. As a result, response time can be longer during start-up periods in a real machine. If the reserved page frames option is in effect, a number of page frames is immediately available, which may still contain some of the required pages.
- It could be assigned for the MONDCSS used to capture monitor data. In an over committed storage environment, keeping MONDCSS pages resident minimizes the risk of incomplete or missing monitor records.

Locked Pages Option

An operator with privilege class A assigned can enter the LOCK command to cause specific pages of one or more virtual machines to be permanently locked in real storage (that is, until the next IPL). The pages indicated in a LOCK command are assigned page frames and paged into these page frames, if they are not currently present in real storage. The frame table entries for these page frames are then removed from the user-owned frame list by the real storage allocation routine, which effectively causes them to be locked until the next IPL.

Note: When using the LOCK command, you must specify whether the guest pages are to be locked into host logical storage or host real storage. Locking pages into host logical storage is intended for debugging purposes. If you are using LOCK for performance reasons, specify the REAL operand to lock the pages into real storage.

Page frames that contain locked pages are not allocated slots in auxiliary storage. The specified pages remain locked until the virtual machine with which they are associated is logged off or until a class A operator unlocks them with the UNLOCK command.

The pages in a shared segment within an NSS or saved segment can be locked. If the SYSTEM CLEAR command is entered for a virtual machine with locked pages, these pages are cleared to zero and unlocked. Further, if a re-IPL of the same or a different operating system is performed in a virtual machine that has locked pages, or virtual storage is redefined for a virtual machine with locked pages, any locked pages are unlocked.

The locked pages option can be used to eliminate paging activity for the most heavily-used pages of virtual machines. Virtual storage page 0 of any virtual machine is the most likely candidate for locking, because it is referenced quite frequently (for example, every time an interrupt occurs for the virtual machine). The virtual storage pages that contain the interrupt-handling routines of an operating system are other good candidates for locking.

Because locked pages cause the amount of real storage that is available for paging to be reduced, excessive use of this facility can cause reduced performance for virtual machines in general, because they must compete for the use of less pageable real storage.

The MAP operand can be specified on a LOCK command to cause a display on the virtual operator's console of the virtual storage addresses of the pages of a virtual machine that are currently locked. The real storage addresses of the page frames in which these pages are locked are also displayed.

The number of page frames locked for a virtual machine can be determined by entering the INDICATE USER command. The total number of page frames locked using the CP LOCK command can be determined by the class A, B, or E user by entering the QUERY FRAMES command.

Collaborative Memory Management Assist

The Collaborative Memory Management Assist is a machine feature that allows z/Architecture guests with the appropriate support to exchange memory usage and status information with z/VM. This sharing of information provides several benefits:

- The guest memory footprint is reduced, allowing for greater memory overcommitment by z/VM.
- z/VM can make more efficient decisions when selecting page frames to reclaim.
- z/VM page-write overhead is eliminated for pages that the guest has designated as unused or volatile.
- The guest can make better decisions when assigning pages.
- Guest page-clearing overhead can be eliminated for pages that z/VM has indicated contain zeros.

The Collaborative Memory Management Assist provides a means for communicating state information about a 4 KB block of storage between a program running in a z/Architecture virtual machine and the z/VM control program. This sharing of state information allows the program and z/VM to make more efficient memory management decisions. The Collaborative Memory Management Assist includes the following features:

- A unique block-usage state and block-content state that are associated with each 4 KB block of main storage (that is, memory) in the virtual configuration.
- The privileged EXTRACT AND SET STORAGE ATTRIBUTES instruction which can be used to extract and optionally set the block-usage and block-content states of a 4 KB block.
- A new reason for recognizing an addressing-exception program interruption.
- The new block-volatility-exception program-interruption condition.

For more information, see Collaborative memory management assist, in *z/VM: CP Programming Services*.

The following CP commands have been added for this support:

- QUERY MEMASSIST
- SET MEMASSIST

For more information, see QUERY MEMASSIST and SET MEMASSIST, in *z/VM: CP Commands and Utilities Reference*.

Real Channel Program Execution Option

z/VM provides a facility by which programs in authorized virtual machines (such as GCS virtual machines) can run real channel programs. This facility enhances performance on dedicated I/O devices, bypassing CP channel program translation. The virtual machine must be authorized to use the facility by the DIAG98 option of the OPTION statement in the virtual machine's user directory entry.

A virtual machine can issue DIAGNOSE X'98' to perform functions that allow it to run a real channel program. These functions are:

- Lock a page of virtual machine storage in real storage
- Unlock a page previously locked by the DIAGNOSE instruction
- Enter a Start Subchannel instruction to run a real channel program.

A real channel program can run anywhere in real storage. Accordingly, when a virtual machine sends a DIAGNOSE X'98' lock function, CP allocates a real page frame from the dynamic paging area and locks the specified virtual storage page into real storage.

Named Saved Systems (NSSs)

The control program portion of an operating system that is IPLed frequently or used by many virtual machines or both can be saved in pageable format on a system data file and given a unique name. When a user wishes to IPL the NSS, the user can specify in the IPL command the name under which the system is saved, instead of a device number. The NSS is then paged into the user's virtual storage from a system data file on a CP-owned spooling volume on a demand basis as required instead of being totally read in from the system residence volume of the operating system during the IPL command procedure.

The significant amount of Start Subchannel instruction simulation processing that is normally required to load an operating system is eliminated when an NSS is loaded. This improves IPL performance. A shared storage capability is also supported for NSSs.

Two steps must be taken in order to generate an NSS. First, a skeleton system data file for the system to be saved must be defined and named using the DEFSYS command. This command specifies the following:

- Name to be assigned to the NSS
- Minimum virtual storage size in which the NSS can run (MINSIZE operand)
- Pages to be saved and the type of virtual machine access permitted for each range of pages
- Whether the NSS is restricted
- Range of general registers that are to contain the IPL parameter string when the NSS receives control from CP.

The VMGROUP operand can also be specified on the DEFSYS command. This operand indicates that virtual machines that IPL the NSS become members of a virtual machine group identified by the name of the NSS. The VMGROUP option defines the group associated with the NSS.

The following types of virtual machine access may be defined for each range of pages in the NSS:

- Exclusive read/write access
- Exclusive read/write access, no data saved
- Exclusive read-only access
- Shared read/write access
- Shared read/write access, no data saved
- Shared read-only access
- Read/write access by a CP service, shared read-only access by a virtual machine, no data saved.

The “no data saved” designation indicates that the page is treated as a new page when the virtual machine first references it. The contents of the page and its storage key are not saved by the SAVESYS command.

A virtual machine can access a restricted NSS only if there is a NAMESAVE statement for the NSS in its z/VM directory entry.

When saved, the NSS occupies z/VM spooling space as a system data file. One slot for each page to be saved plus one slot for saved control information is required for each NSS. The control information that is saved includes:

- Contents of the PSW
- General registers
- Floating-point registers
- Control registers
- Storage keys of the saved pages at the time the system was saved.

The class E user can verify that the NSS was correctly defined by entering the QUERY NSS command with the MAP operand.

Second, the NSS must be IPLed in a virtual machine that has privilege class E assigned. At the point in processing at which the system is to be saved, the user must enter the SAVESYS command and specify the name that was assigned to the system by the DEFSYS command. This causes selected contents of virtual storage and one page of control information to be saved in the previously defined system data file. The NSS can then be IPLed from the system data file.

Users that IPL an NSS have their own copy of any exclusive pages in the NSS paged into their own virtual storage during operation of the virtual machine. Therefore, real storage and auxiliary storage are allocated to those pages of their NSS as usual.

The point in processing at which the SAVESYS command is entered determines the point at which processing is resumed when the NSS is IPLed. (The SAVESYS command essentially performs a checkpoint function.) Therefore, the NSS facility can be used to eliminate the need to respond to initialization commands that are entered during operating system initialization, if the users of such an NSS do not intend to modify previously selected options during the IPL procedure.

A z/OS operating system can be saved, for example, after nucleus initialization program (NIP) processing and, optionally, job entry subsystem (JES) initialization are completed. A VSE operating system can be saved after VSE initialization is completed. The NSS facility can be used to speed up IPLing of various operating systems. This facility enables a user who does not know how to IPL VSE or z/OS to use the following technique:

1. Virtual machine logs on and IPLs the CMS system automatically.
2. A CMS user profile executes a series of setup commands.
3. The user's profile would IPL the guest operating system.

For more information, see *z/VM: Running Guest Operating Systems*.

The shared storage capability of the NSS facility enables reenterable portions of virtual storage that contain programs or data or both to be shared by concurrently operating virtual machines. A virtual storage area within an NSS that is to be shared must begin on a 1 MB segment boundary and be a multiple of a segment (1 MB) in size. Partially defined segments have the rest of their pages treated as "no data saved" using the shared or exclusive attribute given to the rest of the segment. The maximum size of an NSS is 2047 MB (2047 segments). All of the segments within an NSS except the first segment (segment 0) may be shared. Segment 0 must be defined with exclusive access. Every virtual machine needs its own page 0. Therefore, the segment containing page 0 must be exclusive.

z/VM keeps track of the NSSs that are currently loaded for virtual machines and the shared storage they contain. Virtual machines that are sharing storage also share the page tables used for those segments. Additional information about shared segments and details about their management are discussed in *z/VM: Saved Segments Planning and Administration*.

Saved Segments

The saved segment facility enables segments of virtual storage that are not part of an NSS to be saved. It enables the virtual storage used by a virtual machine to be dynamically changed. The ability to share saved segments is an important performance characteristic.

A saved segment can be attached to (loaded in) a virtual machine. If the saved segment is outside of the virtual machine current storage size, the virtual machine's effective addressability is expanded. A 6 MB virtual machine that loads a 1 MB saved segment at the 8 MB line is able to address 7 MB of storage (original 6 MB plus the 1 MB segment).

A saved segment can be used for code or data. Sharing a saved segment among multiple virtual machines can improve performance by minimizing storage requirements. For example, the code for a licensed program product may require 100 pages. There may be fifty users executing this code at the same time. This would require storage and paging resources for 5000 pages if not shared, but only 100 if a shared saved segment were used. Sharing can also be applied to saved segments containing data. The CMS SAVEFD command can be used to create a segment containing file directory information for a CMS formatted minidisk. This segment can be shared among CMS users. See *z/VM: CMS Planning and Administration* for additional information.

VM/VS Handshaking

VM/VS handshaking is a standard facility of CP. It can be used by all supported releases and versions of VSE operating systems.

VM/VS handshaking support provides a communication path between CP and a virtual machine. This enables CP to know that an operating system with VM/VS handshaking support is running under its control. Also, this permits the VSE operating system to know it is running in a virtual machine under the control of a CP with VM/VS handshaking support.

VM/VS handshaking support is designed to improve the performance and simplify the operation of VSE running in virtual machines. The following facilities are provided by VM/VS handshaking support:

- CP spool files written by VSE output writers are closed by VSE, which eliminates the need for the virtual machine's operator to enter CP CLOSE commands for these files.
- Improved page fault handling is provided, which allows a VSE virtual machine to receive control after a page fault occurs (instead of being placed in a wait state) so that it can dispatch the next ready task. If use of VM/VS handshaking support is specified when VSE is generated, the VSE guest activates this facility automatically by entering a CP SET PAGEX command using the DIAGNOSE X'08' interface.
- A nonpaged mode of operation for VSE virtual machines is supported, which results in the issuing of fewer privileged instructions and the elimination of duplicate paging by VSE.
- VSE avoids issuing certain frequently used privileged instructions, which are inefficient in a virtual machine. Also, VSE eliminates procedures that duplicate functions provided by CP.

In addition to these facilities, the BTAM autopoll facility is supported, which reduces the CP processing required to handle BTAM autopoll channel programs in VSE virtual machines.

When the BTAM autopoll facility of CP is not active, whenever a channel program containing a polling sequence is started by a virtual machine, the program-controlled interrupt (PCI) flag is turned on in the corresponding real autopoll channel program by the channel program translation routine in CP. Thus, each time the real autopoll channel program runs, the PCI flag causes an interrupt and CP gains control to determine whether the corresponding virtual CCW string has been changed. Any changes found are made to the real CCW string while it is operating.

When the SET AUTOPOLL ON command is entered, the BTAM autopoll facility is activated. This causes CP not to set the PCI flag in BTAM autopoll channel programs and, therefore, not to check for changes to the virtual CCW string. CP then expects to be notified of any such changes. VSE checks for the changes and issue a DIAGNOSE X'28' to inform CP of each change.

The SET AUTOPOLL OFF command deactivates the BTAM autopoll facility. This is the default setting for the facility when a virtual machine is logged on. If the facility is activated for a virtual machine that does not support the BTAM autopoll interface to z/VM, CP does not detect changes to the BTAM autopoll channel program and results are unpredictable.

The QUERY SET command can be used to determine the PAGEX and AUTOPOLL settings for a virtual machine.

Minidisk Cache

Minidisk cache may provide performance and administrative benefits to z/VM systems. The amount of data that exists is much larger than the amount of data that is frequently used. This tends to be true for systems as well as individual virtual machines. The concept of caching builds off this behavior by keeping the frequently referenced data where it can be efficiently accessed. For minidisk cache, CP uses real storage as a cache for data from virtual I/O. This is the default. Accessing electronic storage is much more efficient than accessing DASD.

The rest of this section is broken down into subsections. The first five provide background information on how minidisk cache works. The last subsection provides recommendations and guidelines for using minidisk cache more effectively.

Requirements

Restrictions for minidisk cache are in the area of the type of I/O and the type of the data or minidisk. The following is a list of restrictions for minidisk cache eligibility:

- Data must be referenced via Diagnose X'18', DiagnoseX'A4', Diagnose X'A8', Diagnose X'250', *BLOCKIO, SSCH, SIO, or SIOF.
- Minidisks must be on a 3380, 3390, or FBA DASD, or hardware that emulates those architectures.
- Dedicated devices are not eligible.
- FBA minidisks must be defined on page boundaries, and the size must be a multiple of 8 pages (64 512-byte blocks).
- Minidisks created with Diagnose X'E4' are not eligible.
- Minidisk caching is not supported for minidisks on shared DASD. There is no support for handshaking between z/VM systems for minidisk cache.
- When you SET SHARED ON for a device, minidisks on that device become ineligible for minidisk cache.
- Minidisks that overlap CP space allocated for page, spool, directory, or temporary disk are not eligible. In the temporary disk case, this refers to a minidisk defined in the directory or with the DEFINE MDISK command. It does not refer to minidisks created with the DEFINE TEMPDISK command.
- A minidisk, including a temporary disk, is not eligible for minidisk cache if it has been defined with greater than 32,767 cylinders. This does not apply to FBA devices. It applies only to ECKD devices.
- I/Os can be aborted from minidisk cache processing for various reasons. These reasons include:

  - The record size is greater than 32,767 bytes.
  - The channel program includes a backwards TIC that does not point to a search command.
  - The channel program includes a format write (X'03') command.
  - The channel program includes a sense CCW (X'04').
  - The channel program includes a read sector CCW (X'22').
- If the guest I/O is to a minidisk located on a volume that is held in an LCU that contains HyperPAV aliases, and if the guest channel program fails to contain a Prefix CCW as its first command, the I/O is ineligible for minidisk cache.

Concepts

Minidisk cache is a data-in-memory technique that attempts to improve performance by decreasing the I/O to DASD required for minidisk I/O. Minidisk cache trades increased use of real storage for decreased DASD I/O. Since paging to DASD increases as the amount of available real storage decreases, you should expect some increase in paging I/O when exploiting minidisk cache. An increase in paging is not necessarily bad. Looking at the total real DASD I/O rate and user state sampling can show whether the system is benefiting from minidisk cache. For example, if minidisk cache reduces the real DASD I/O rate by 300 I/Os per second and paging DASD I/O increases by 50 per second, then there would be a 250 I/Os per second reduction. This is good. Also, looking at user state sampling might indicate users are waiting for virtual I/O much less than before with just a small increase in page wait time. This would also be good.

Minidisk Cache Arbiter

By default, the CP arbiter function determines how much real storage to give to minidisk cache and paging. The goal of the arbiter is to keep the average page life of a minidisk cache page equal to the average page life of a page used for paging. This is not measured directly, but estimated based on steal rates for the two usage types. Several commands are available to influence the arbiter. The amount of real storage used by minidisk cache can be set by the SET MDCACHE STORAGE command. Minimum and maximum values can be set. When the minimum and maximum values are equal, the storage associated with minidisk cache is a fixed amount and the arbiter is effectively turned off. Unequal minimum and maximum values can be used to bound the cache size determined by the arbiter. An additional method of influencing the arbiter is setting a bias with the SET MDCACHE STORAGE BIAS command. If you find the arbiter favoring paging or minidisk cache more than is optimal for your system, you may bias the arbiter for or against minidisk cache.

Fair Share Limit

The z/VM minidisk cache is implemented with a system cache where the storage for the cache is shared with all the users on the system. In order to prevent any one user from negatively impacting other users on the system, CP enforces a fair share limit. This is a limit on the amount of I/O that a user can insert into the cache over a fixed period of time. The limit is a dynamic value based on the amount of storage available and the number of users that want to make inserts to the data. If a user reaches the fair share limit, the I/O still completes but the data is not inserted into the cache. The NOMDCFS operand on the OPTION statement of a user directory entry can be used to override the fair share limit.

Minidisk Cache Enabling

By default all minidisks that meet the requirements for minidisk cache are enabled. There are three levels of control for enabling minidisk cache.

1. System level

   The system level is the highest level of control and is managed by the SET MDCACHE SYSTEM command. Minidisk cache is either on or off at the system level. If it is off, then no minidisk caching is done regardless of other settings at lower levels. Minidisk cache is on at the system level by default.
2. Real device level

If minidisk cache is enabled at the system level, then the SET MDCACHE RDEV command, SET RDEVICE command, or RDEVICE configuration statement can be used to further enable or disable minidisk cache at the real device level. There are three settings at the real device level.

DFLTON
Default On enables minidisk caching for a real device, yet allows the ability to disable minidisk caching for a particular minidisk on that device. All eligible minidisks will be cached on this device except those that have caching off at the minidisk level. This is the default.

DFLTOFF
Default Off disables minidisk caching for a real device, yet allows the ability to enable caching for a particular minidisk on that device. No minidisks will be cached on this device except those that have caching on at the minidisk level.

OFF
OFF disables minidisk caching for a real device. It cannot be overridden at the minidisk level. No minidisks will be cached on this device.

3. Minidisk level

   The last level of control is the minidisk level which is managed by the CP command SET MDCACHE MDISK or MINIOPT statements in the system directory. By default minidisk cache is enabled at the minidisk level. One can explicitly specify ON for minidisks on devices with a DFLTOFF setting or OFF for minidisks on devices with a DFLTON setting.

The ability also exists to flush data from minidisk cache for particular devices and minidisks with the SET MDCACHE RDEV FLUSH or SET MDCACHE MDISK FLUSH commands.

Guidelines

Here are some suggestions that will help you to use minidisk cache more effectively:

1. As discussed above, the use of minidisk cache usually results in an increase in paging and the arbiter assumes the paging configuration is appropriate. There can be significant performance degradation if minidisk cache is used on a system that has not had the paging configuration tuned. You want to make sure that there is sufficient paging space allocated. z/VM likes large paging allocations in order to effectively use block paging. If the block size of page reads as reported by monitor is less than 10, there is probably not enough DASD space allocated for paging. Paging space should also be isolated so that the seldom-ending channel program (start subchannel/resume subchannel) technique is effective. Do not mix paging space with other types of data, including spool space. Balance the paging I/O over multiple volumes where appropriate.
2. When using minidisk cache, always have some real storage defined for minidisk cache.

   In systems where the I/O buffers in virtual storage of the user are not page aligned, there can be a significant performance degradation if you do not have some real storage allocated for minidisk cache.
3. Consider biasing the arbiter against minidisk cache if the system is very rich in storage.

   Setting a value less than one on the BIAS option of the SET MDCACHE will bias against minidisk cache. Measurement results to date suggest that in environments that show no storage constraint, the arbiters sometime use more storage for minidisk cache than is optimal. The more storage constrained a system is, the better the arbiter tends to work. A bias in minidisk cache size can also be achieved by using SET MDCACHE to set a maximum size.
4. When planning what data should be enabled for minidisk cache, it is generally better to start with everything enabled and then selectively disable minidisks or volumes.

   Volumes with data that are physically shared between z/VM systems should be disabled for minidisk cache. There is no handshaking between z/VM systems to ensure that changes to a minidisk are reflected in the minidisk cache of other systems. This also applies to sharing minidisks between first and second level systems.

   Minidisks where the majority of I/Os are write I/Os should be disabled for minidisk cache. Examples of this include the log disks for Shared File System.

Minidisks with data that is only read once should be disabled for minidisk cache. The real benefit of minidisk cache is only seen for data that is referenced multiple times.

Disable minidisk cache for volumes that are mapped to VM data spaces. If data is being accessed by the mapped mdisk feature of VM data spaces, there is no additional benefit from minidisk cache. There can be significant overhead in processing to ensure data integrity when both minidisk cache and mapped mdisks are used for the same minidisk.

Disable minidisk cache for minidisks that are the target of FlashCopy® requests. This includes FlashCopy requests that might be initiated by virtual machines, such as a z/OS guest running the HSM component.

You might want to temporarily disable minidisk cache for backup or scan routines that reference all the data on a disk, but just once. This can be done with the SET MDCACHE INSERT or SET MDCACHE MDISK CP commands.

5. Disable the minidisk cache fair share limit for key users.

   Server machines or guests that do I/O on behalf of many users should not be subject to the minidisk cache fair share limit. Use the NOMDCFS operand on the OPTION statement in the user entry of the system directory to turn off the fair share limit.

6. Remove duplication of minidisks.

   If you duplicated minidisks in the past in order to balance I/O, that may not be necessary with minidisk cache. Duplication can actually decrease performance since it might result in duplicate copies of the data in minidisk cache.

7. Consider use of record level minidisk cache.

   There may be extreme cases of poor locality of reference where the default caching of a track of data is inappropriate. This may be the case for certain database products. If the minidisks are non-FBA CMS minidisks, then using record level minidisk cache may improve performance. The SET MDCACHE command or directory options can be used to enable record level caching. This approach should be used only after you have made sure all other tuning is appropriate.

VM Data Spaces

z/VM uses an extension to the interpretive-execution facility called VM Data Spaces to provide data sharing services to application programs running in different virtual machines. These services are available to applications that run in XC virtual machines.

CP gives a virtual machine a primary address space when the user logs on to the z/VM system. After logging on, the XC virtual machine user may create other address spaces (called data spaces) and, if desired, share them with other logged on users. VM Data Spaces provide increased storage addressability and can move the burden of I/O from an application to the CP paging subsystem. The use of VM Data Spaces also extends the concept of sharing data. This has two chief advantages:

1. It reduces storage requirements. One copy can be shared among many virtual machines instead of a copy per virtual machine.
2. It reduces the need to transfer data by IUCV, APPC/VM, or some other communication vehicle.

CP macros can be used to create and work with data spaces. See *z/VM: CP Programming Services*. The CMS Callable Services Library (CSL) also provides an interface to VM data spaces from high-level languages.

The XC virtual machine mode is used for exploitation of VM Data Spaces. For non-XC mode virtual machines, DIAGNOSE X'248' (Copy to Primary function) can be used to move data from a data space to the virtual machine's primary address space. The Callable Services Library (CSL) provides an interface with high level language support.

Minidisk Mapping extends the concept of applications using storage and letting CP handle the real I/O. This function provides a means of associating minidisk data with VM data spaces. One or more minidisks can be mapped to one or more data spaces. An application references the data simply by referencing

the corresponding storage location in the data space. The real I/O is handle by the CP paging subsystem, which provides efficiencies.

Some initial setup work is required to establish the mapping rules. This is managed by MAPMDISK, a CP macro. Since virtual storage is volatile, management for integrity must be considered. The SAVE function provides a means of forcing or committing the data to the non-volatile DASD where the minidisk resides. Minidisks that are to be referred to exclusively as mapped minidisks should be made ineligible for minidisk cache.

The processing associated with page fault resolution serializes a virtual machine. The Asynchronous Page Fault capability is available to address the impact of server virtual machine page fault serialization on other dependent virtual machines. Asynch Page Fault allows a virtual machine to continue processing on other work (a different task), while CP handles the page fault. The implementation applies only to page faults of data space pages. CP will provide an interrupt when the page fault is complete. At that time the server application can finish the processing the original task associated with the page fault.

The server application requires logic to work in this environment. This includes:

- A structure that lends itself to tasking or work units.
- Selection of asynchronous page fault function on a data space by space basis. This occurs when adding the data space to the access list.
- Using CP macro PFAULT to establish token for hand shaking.
- Support to handle the associated interrupts.

The PAGEX support is based off the same idea. There are two significant differences between the two:

1. PAGEX deals with the primary address space while the Asynchronous Page Fault support is limited to VM Data Spaces.
2. Asynchronous Page Fault was designed with server virtual machines in mind. The hand shaking interface with CP is easy to work with and lends itself nicely to server applications.

Disk mapping can further improve performance through the use of another CP macro, REFPAGE. This macro allows an application to specify its upcoming storage reference patterns. The z/VM paging subsystem uses this information to form blocks of pages that are related to the application's future reference pattern. When the paging subsystem needs to reference a DASD for one page of a block, it loads the entire block into the data space, eliminating the need to access the DASD on subsequent references to other pages.

Spool File Initialization

Previously, enhancements were made in VM to improve performance of system initialization. In particular, these enhancements affected spool file initialization performance, regardless of whether it is a WARM, COLD, or FORCE start. The benefit of the enhancements is proportional to the number of spool volumes and paths to those volumes. Placing temporary disk (TDSK) and spool (SPOL) areas on the same volume will degrade the performance, especially if FEATURES ENABLE CLEAR_TDISK is coded in the system configuration file. This setting specifies that the temporary disk space be cleared to binary zeros at initialization time. The clearing process and spool file initialization get in the way of each other and slow the system initialization process down.

File Caching Option for CP-Accessed Minidisks

Depending on how you configure your system, you may instruct CP to obtain log message and logo information from files that reside on minidisks accessed by CP. You can use the CP_ACCESS statement in the system configuration file or the CPACCESS command to inform CP what minidisks it is to access. Because log message information is displayed every time a user logs on to the system or issues the QUERY LOGMSG command, and a logo is displayed every time a user logs off or a terminal is enabled, it is a good idea to have CP cache the log message and logo files in storage.

To cache these files in storage, you can enter the CPCACHE command or place a file called CPCACHE FILES on the PARM DISK. Inside this file is a list of files you want CP to cache. You can use the CPLISTFILE command to ensure that all high-usage files are, in fact, being cached.

Hot I/O Detection

The hot I/O detection function protects CP from broken hardware that floods the system with unsolicited interrupts. Without this valuable protection, severe performance degradation or a HARD abend condition occurs. When the system detects a hot I/O problem, it attempts to recover. If recovery is not possible, the system removes the device from active I/O configuration. In either case, CP writes one or more messages to the primary system console. These messages include the address of the broken device.

Virtual Disks in Storage

Virtual disks in storage are temporary FBA minidisks allocated from address spaces in host real storage instead of on real DASD. Because the I/O overhead is avoided, virtual disks in storage may be faster to use than other minidisks. However, the improved I/O performance may be a trade off for increased storage requirements. Unlike temporary minidisks (T-disks), virtual disks in storage defined by MDISK statements in the directory can be shared. A shareable virtual disk in storage is created when the first user links to it and destroyed when the last user detaches it or logs off. Virtual reserve/release is supported for virtual disks in storage, which allows data sharing among multiple guests. Virtual disks in storage may be especially useful for VSE guests, providing fast-access storage for label information areas and the cross-system communication file (the "lock file").

Degree of Benefit

Although the use of virtual disks in storage can reduce processor requirements, the primary performance benefit is the elapsed time reduction that results from eliminating file I/Os. For any given interaction or job, the percentage reduction in elapsed time depends on how many I/Os are eliminated and the average DASD response time of those I/Os. Consider the following example:

- elapsed time: 5 seconds
- file I/Os removed by using a virtual disk in storage: 100
- average DASD response time for those I/Os: 20 milliseconds

In this example, the use of virtual disks in storage would tend to result in an elapsed time decrease of about 2 seconds (100*0.020), which is a 40% reduction. This assumes no significant increase in paging for that interaction. This is often the case for those interactions that use virtual disks in storage. Any increase in paging more typically shows up on a system basis and is distributed among all the users.

In addition to these elapsed time benefits, their use can result in a significant increase in system throughput in cases where it removes or reduces an I/O bottleneck. This can occur, for example, when a virtual disk in storage is used for VSE lock file. By removing the I/O bottleneck, the system workload can be increased until some other system resource becomes the limiting factor.

Guidelines

Here are some suggestions that will help you to use virtual disks in storage effectively:

1. Heavily used temporary minidisks make good candidates, especially if the amount of referenced data is small.

   The more I/Os there are to a given minidisk, the larger the benefit side of the cost/benefit tradeoff if the files on that minidisk are moved to a virtual disk in storage. If the amount of referenced data is small, the cost side of the cost/benefit tradeoff is small.
2. Virtual disks in storage are especially beneficial if minidisk caching is not available.

   When minidisk caching is in effect for a given minidisk, it may already be eliminating most of the read I/Os, leaving only the write I/Os to be eliminated by moving that minidisk's files to a virtual disk in storage. When minidisk caching is not in effect, both read and write I/Os are eliminated.

3. Consider the alternatives.

   For a given situation, other data-in-memory techniques such as shared segments, SFS DIRCONTROL directories in data spaces, and minidisk caching may be more applicable.

4. Do not use more virtual disks in storage than necessary.

   Each virtual disk in storage has one or more fixed pages associated with it. Being fixed, these pages add to real storage requirements even when the virtual disk in storage is not being used.

   The fixed storage requirement per virtual disk in storage is 1 page for the segment table (2 consecutive pages if the virtual disk in storage is larger than 1 GB) plus 1 page for each ever-referenced MB in its address space.

5. Do not make a virtual disk in storage larger than necessary.

   This is a consideration if it is to be formatted with a utility that actually writes out every block. This would be the case, for example, when the DSF initialization function is used to format a minidisk to be used by a VSE guest. This is not a consideration when the CMS FORMAT command is used to format the virtual disk in storage because CMS FORMAT only references a few of the data pages.

6. If necessary, increase the amount of paging space.

   The use of virtual disks in storage will tend to increase the system's paging space requirements. Before making them available on the system, first check to make sure there is enough extra paging space available.

7. Detach a virtual disk in storage as soon as possible.

   This primarily applies to private virtual disks in storage. When a private virtual disk in storage is detached (or a shared one is detached by the last user), all page frames associated with that virtual disk in storage are made available for other uses. This can greatly reduce the degree to which virtual disk in storage usage increases real storage requirements.

8. Set system and user limits to help prevent excessive use.

   This is done using the CP SET VDISK command. The system limit helps to protect the overall system against overcommitment of storage through excessive use of virtual disks in storage. The user limit, which only applies to virtual disks in storage that are obtained using the CP DEFINE command, prevents any one user from acquiring an excessive amount of space.

   The built-in default for the system limit is intended to provide protection against fixed storage requirements (due to page and segment tables needed to map the address spaces) from using more than 1/4 of the dynamic paging area. The built-in default for the user limit is 0, which would restrict usage to virtual disks in storage defined by MDSK directory statements. For more information on the built-in defaults, see the SET VDISK discussion in *z/VM: CP Commands and Utilities Reference*.

9. Monitor performance.

   Monitor overall performance and system paging rate before and after virtual disks in storage are made available. If overall performance shows a decrease that appears to be due to increased paging, you may wish to decrease the system limit for virtual disk in storage usage. If, on the other hand, the system limit is frequently being reached and overall performance is good, you may wish to increase the system limit.

I/O Throttling

The I/O throttling function can be used to set a maximum rate at which the system's virtual machines, in aggregate, are permitted to initiate I/Os to a given real device. This limit does not apply to I/Os initiated by CP. CP converts the specified rate into an interval representing the minimum time that must pass after one I/O is started before the next I/O to that device can start. If CP receives an I/O request to a device that has been limited by throttling, that I/O request is delayed, if necessary, until the minimum time interval has completed.

In multi-system configurations which have shared channels, control units, or devices, throttling can be used to help prevent any one system from over utilizing the shared resources. For example, if systems A and B share a control unit and system A is on a much faster processor complex, system A could

monopolize the control unit because it can issue I/Os faster. By throttling devices owned by system A on the shared control unit, system B would not see as much contention at the control unit level.

The same approach can be used within a system. Just replace systems A and B above with users A and B. The one item to note is that throttling occurs at the real device level, not the minidisk or user ID level.

The throttle rate can be set with the CP SET THROTTLE command or with the THROTTLE statement in the CP CONFIG file. The rate can be set for one or more devices at a time. The QUERY THROTTLE command can be used to determine the current throttle values.

I/O Priority Queueing

I/O Priority Queueing provides a method to favor DASD I/O of one virtual machine over another. This priority is used at two levels. When virtual machine I/O is targeted for the same real DASD volume at a rate that creates a queue for that device, the priority value is used in determining where in the queue the I/O should be placed. The priority value is also used by the hardware I/O subsystem when there is contention for channel paths. The higher priority I/Os will get better access to the channels. I/O priority effects are ineffective in cases where I/O resources are unconstrained. For more information, see the SET IOPRIORITY command in *z/VM: CP Commands and Utilities Reference*.

Additional features of QDIO performance

This topic summarizes additional features of the QDIO protocol on IBM z16™ family or earlier servers.

The QDIO architecture, originally introduced with the OSA-Express, was later extended to HiperSockets and the FCP channels. The architecture itself was extended in HiperSockets to include a type of high-performance I/O interruption known as an adapter interruption. The use of adapter interruptions has been extended to the OSA-Express and FCP channels.

In addition to the use of adapter interruptions by the OSA-Express and FCP channels, the server is designed to include a performance assist for the virtualization of adapter interruptions being given to operating systems running as guests of z/VM. This hardware performance assist is available to guests that support QDIO.

This IBM virtualization technology is designed to benefit all guest operating systems in environments where they can process adapter interruptions. This includes all users of HiperSockets, and guest operating systems that add adapter-interruption support for OSA-Express and FCP channels. TCP/IP can benefit from this performance assist for both HiperSockets and OSA-Express.

The following CP functions have been added for this support:

- QUERY QIOASSIST command
- SET QIOASSIST command

The following CP functions have been updated for this support:

- QUERY VIRTUAL FCP command
- QUERY VIRTUAL OSA command

For more information, see *z/VM: CP Commands and Utilities Reference*.

Additional improvements include:

- QDIO enhanced buffer-state management (QEBSM). There are two hardware instructions designed to help eliminate the overhead of hypervisor interception for guest I/O to HiperSockets, FCP, and OSA devices.
- Host page-management assist (HPMA). This assist is an interface to the z/VM storage (memory) management function designed to allow page frames to be assigned, locked, and unlocked without z/VM hypervisor assistance. HPMA primarily benefits environments with QDIO enhanced buffer-state management and Collaborative Memory Management Assist.

QEBSM and HPMA can allow a cooperating guest operating system to initiate QDIO operations directly to the applicable channel, without interception by z/VM, thereby helping to provide additional performance

improvements. These virtualization technologies are available only to first-level guests of z/VM. That is, they are not available to guests of a z/VM system that is itself running as a guest of another z/VM system.

For information about the hardware availability of these technologies and support in z/VM releases, see *z/VM: Migration Guide*.

Parallel Access Volumes (PAV) and HyperPAV for guest I/O to minidisks

If the DASD subsystem supports PAV or HyperPAV, CP can use a real volume's alias devices to run more than one guest I/O operation at a time to the volume. When CP uses alias devices, the guests' I/O operations tend not to interleave or block each other. In turn, the guests' I/O operations tend to experience reduced real-volume I/O queuing and reduced response time. Reduced response time yields guest applications that run more quickly.

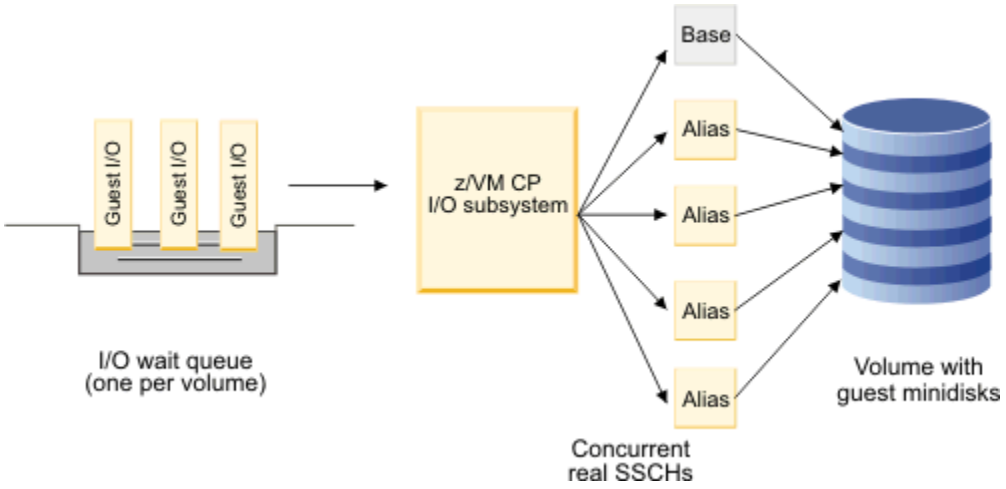


Figure 4. PAV and HyperPAV support

PAV exploitation by CP applies to guest minidisk I/O operations. HyperPAV can be used for guest I/O operations to minidisks and also the CP Paging Subsystem (see "HyperPAV for the Paging Subsystem" on page 50). When CP performs a full-track read operation to populate minidisk cache on a cache miss, that full-track read operation is also eligible for PAV fanout.

To configure guest minidisk I/O operations to run concurrently for a real volume, the first step is to configure the DASD subsystem to offer some alias devices for the real volume. This step is usually a task for the DASD subsystem administrator. The outcome of this step is generally one of two results:

- For PAV, the real disk volume becomes visible to CP as a base device and one or more alias devices. For example, the real volume that customarily appears at device number E000 might now also have alias devices E001, E002, E003, and E004. All five of these devices—the base device and the four alias devices—lead to the same disk volume. An I/O operation with any one of the five devices leads to the same volume and the same data: the same cylinders, the same tracks, and the same records.
- For HyperPAV, the logical subsystem (LSS) in which the real disk volume resides contains one or more HyperPAV alias devices. Each HyperPAV alias device can be used to run an I/O against any real volume configured into the same LSS. For example, the LSS might contain 16 real volumes with base device numbers E000-E00F and also contain six HyperPAV alias devices numbered E020-E025. Any of those alias devices can be used against any base device in the LSS, on a per-I/O basis. CP can use the six aliases as needed to parallelize the I/O operations between the guests and the 16 bases.

No matter whether PAV or HyperPAV is being used, the steps that are needed to get CP to use the aliases to parallelize guest I/O are essentially the same. Previously you would attach only the volume's base device to SYSTEM; now you attach the volume's base device and one or more of its alias devices to SYSTEM. Most installations use the SYSTEM CONFIG file to do such attachment but the CP ATTACH command can also do. Update the DEVICES statement in SYSTEM CONFIG to enable the alias devices, including the base device that is already enabled there. The USER_VOLUME_LIST statements remain unchanged. If you use the CP ATTACH command in a startup script, use the CP ATTACH command to attach the alias devices to SYSTEM at the same point where you attach the base device.

The following example uses PAV. Suppose Performance Toolkit for z/VM indicates that guest I/O operations are queuing frequently at the real volume E000, which is labeled USR001. Suppose further that you decide to try to relieve the constraint by using PAV. First, in the DASD subsystem you define alias device numbers E001, E002, E003, and E004 for base device E000. By using the alias devices, CP can run up to five I/O operations concurrently to the volume. Then, give CP ownership of the alias devices. If you are using SYSTEM CONFIG to set up your DASD, add the alias device numbers to the DEVICES statement. Keep the USER_VOLUME_LIST statement as it was previously:

```
Devices online_at_IPL E000-E004
User_Volume_List USR001
```

When CP processes these statements at IPL time, it determines that volume USR001 has base device E000 and alias devices E001-E004, and it attaches all five devices to SYSTEM.

Modify the example and assume that instead of using the system configuration file you use the CP ATTACH commands in AUTOLOG1's profile to mount user disks. In this case, use the ATTACH command to attach the alias devices at the point in the script where you previously attached only the base volume. In a PROFILE EXEC, the commands would be:

```
'CP ATTACH E000 TO SYSTEM'
'CP ATTACH E001 TO SYSTEM'
'CP ATTACH E002 TO SYSTEM'
'CP ATTACH E003 TO SYSTEM'
'CP ATTACH E004 TO SYSTEM'
```

The steps are only slightly more complex for HyperPAV. First, enable the HyperPAV aliases in the DASD subsystem. If you use SYSTEM CONFIG, adjust the DEVICES statement as before to make sure the HyperPAV alias devices come online. Also, leave the USER_VOLUME_LIST statement unchanged, just as you did for PAV. The one extra step is to code a SYSTEM_ALIAS statement to indicate which HyperPAV alias devices to attach to CP. The result looks like the following:

```
Devices online_at_IPL E000-E00F, /* sixteen bases     */
        online_at_IPL E020-E025  /* six aliases       */

User_Volume_List USR001
User_Volume_List USR002

[… and so on, for the remaining 14 user volumes …]

SYSTEM_ALIAS E020-E025   /* tells CP to use the six HyperPAV aliases */
```

The changes result in devices E000-E00F and E020-E025 all being attached to SYSTEM, ready for CP to use to parallelize user I/O.

If you are using a startup script instead of SYSTEM CONFIG, code the ATTACH commands for the bases and aliases, just as you did for PAV. For example, in a PROFILE EXEC, the commands would be:

```
'CP ATTACH E000-E00F TO SYSTEM'
'CP ATTACH E020-E025 TO SYSTEM'
```

There is more to z/VM's PAV or HyperPAV exploitation. If CP has base and alias devices for a real volume, you can configure a virtual base device and one or more virtual alias devices for a guest's minidisk. This "virtual PAV" facility lets the guest launch multiple concurrent virtual I/O operations with the guest's minidisk. The guest can launch one I/O operation with the virtual base and one I/O operation with each virtual alias. In turn, CP uses the real base and real aliases to run those multiple concurrent virtual I/Os concurrently on the real volume. A guest that is capable of participating in this environment is often called "PAV-aware."

Note:

z/VM treats an NVMe PCIe function as a logical subsystem that contains both base EDEVICEs and HyperPAV alias EDEVICEs. For guest I/O operations, a set of EDEVICEs that are defined on an NVMe PCIe function are a DASD subsystem. Statements about the use of HyperPAV aliases apply to NVMe-EDEVICE HyperPAV aliases.

For more information, see the following topics:

- EDEVICE Statement, Using IBM Parallel Access Volumes, and Using IBM HyperParallel Access Volumes in *z/VM: CP Planning and Administration*
- SET EDEVICE, DEFINE PAVALIAS, and DEFINE HYPERPAVALIAS in *z/VM: CP Commands and Utilities Reference*

HyperPAV for the Paging Subsystem

As with minidisk I/O, HyperPAV alias devices can be used to assist in the execution of CP Paging Subsystem I/O directed at HyperPAV base volumes in the same pool. This support is enabled through the FEATURES ENABLE PAGING_ALIAS system configuration file statement or the CP command SET PAGING ALIAS ON. When enabled, HyperPAV paging I/O operations are optimized by the zVM I/O scheduler to execute the I/O on the HyperPAV base volume (if it is currently idle), or on an available system-attached HyperPAV alias volume in the same pool (if available). Otherwise, the I/O is queued at the base HyperPAV device for subsequent execution.

Sharing HyperPAV Aliases

When some HyperPAV bases are used by the CP Paging Subsystem and some are used for guest I/O to minidisks in the same pool and system-attached HyperPAV aliases exist in that pool, the VM I/O scheduler will use the HyperPAV aliases to assist in the execution of I/O for both types of HyperPAV bases on a first-come, first-served (FCFS) basis. When all system-attached HyperPAV aliases in a pool are currently busy executing I/O and then one becomes available, that FCFS algorithm can be fine-tuned using the SET CU command to give alias share priority to one type over the other, if so desired. Monitor data can be used to determine whether SET CU is needed to change the alias share. For guidance on the use of alias shares, see "HyperPAV Paging Support" on the z/VM performance website at IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

Performance Guidelines in a Virtual Machine Environment

There are some guidelines or hints that can be used to improve the performance of a virtual machine and its applications. The benefit to be achieved by taking any one of the steps outlined must be evaluated on an installation basis, taking the specific configuration and operating environment into account. Some steps, for example, are more practical for large configurations than for small configurations.

The following list contains steps that can be taken to increase performance in a z/VM system. For more detailed information about tuning the z/VM system, see Chapter 12, “Tuning Your System,” on page 103.

- Increase the page fault handling capability of the system.

  This can be accomplished by:

  – Using a direct access device for paging that is faster than the currently used paging device.
  – Allocating more direct access devices for paging to enable more overlap of paging activity
  – Reducing or eliminating contention for the existing paging devices.

  Contention for the paging device can be relieved by using dedicated paging devices, reducing the amount of other I/O activity on the channel path to which the paging device is attached, or dedicating a channel path to paging. Alternatively, the same paging device configuration can be maintained while page fault occurrence is decreased by adding real storage.
- Avoid double spooling operations if the operating system being used does not support blocked spooling output.

  If the unit record data transcription facilities of an operating system are to be used in a virtual machine and spooled output is not blocked, real unit record devices should be dedicated to the virtual machine. This eliminates CP spooling operations. Similarly, if CP spooling facilities are to be used and spooled output is not blocked, spooling by the operating system should be eliminated. For the reasons indicated in the section on spooling in *z/VM: Connectivity*, better performance may be achieved by using CP spooling.
- Use storage and scheduling options.

When multiprogramming is to be used in a virtual machine, the scheduling share and quick dispatch options can be assigned to the virtual machine to ensure the allocation of enough real processor time to the virtual machine for lower-priority tasks to be able to run.

- Use the NSS facility for operating systems that are used in multiple virtual machines or IPLed frequently or both.

  This action eliminates the lengthy CP processing required for a usual IPL. Also, use the shared storage capability, which can reduce the total number of page faults that occur in the system.
- Use the saved segments provided for CMS and install user-written code and licensed programs that are to be used concurrently by multiple virtual machines in saved segments.
- Share the system residence volume of an operating system that is to be used concurrently by two or more virtual machines to avoid having multiple copies of the operating system online.

  In order to share a system residence volume, it should be read-only. Nothing need be done to the CMS system residence volume (S disk) to make it read-only because it is automatically accessed as a read-only disk. To share a VSE or z/OS system residence volume, any read/write data sets must be removed from it.
- Plan the use of the minidisk cache feature.

  Minidisks that are write-mostly or read-once are poor candidates for minidisk cache and should be disabled for the cache via the MINIOPT directory control statement or the SET MDCACHE command.

  Minidisk cache should be turned off for minidisks that are only read and written via MAPMDISK and the corresponding data space. Minidisks which use a combination of data space access/MAPMDISK and virtual I/O (Diagnose and channel program) should be evaluated on an individual basis to determine if minidisk cache should remain enabled.
- Avoid duplicating shared minidisks with minidisk cache

  Prior to the availability of the minidisk cache facility, it was common to duplicate highly shared minidisks across several system volumes to balance I/O. By exploiting minidisk cache, this duplication is no longer necessary. In fact, duplication is actually quite costly to system performance and administration of minidisks. Therefore, when using minidisk cache, do not duplicate minidisks.
- Carefully plan the use of CP-owned volumes.

  Allocate direct access device types for auxiliary storage and spooling space to take advantage of the way in which CP allocates and manages this space. Allocate functions across CP-owned volumes in such a way that arm contention is minimized. It is better, for example, to dedicate one or more volumes to spooling and paging functions than to allocate smaller areas of spool space and auxiliary storage on several volumes.

  Spool space and auxiliary storage should not be allocated on the same volume and, if possible, spool volumes should be on different channel paths from auxiliary storage volumes. Do not place heavily-used data sets or files on disk volumes that contain auxiliary storage and, when possible, do not place disk volumes with heavily-used data sets or files on the same channel paths as paging volumes. If it is necessary to include other data sets on a paging device, the data sets should be used infrequently.
- Carefully plan the allocation of minidisks so that arm contention is minimized, particularly those minidisks that are to be accessed by a VSE or z/OS production virtual machine.
- When running CMS and a heavily I/O-oriented production virtual machine, do not place CMS disks on any channel path that has attached I/O devices that are heavily used by the production virtual machine.
- Prepare guidelines for the selection of an available real I/O device that will minimize device or channel path contention when real I/O device selection for a batch production z/OS virtual machine is to be performed by the operator instead of the operating system.
- Reduce the number of Start Subchannel instructions issued by each virtual machine.

  This can be done by:

  – Using larger I/O buffers. This step is practical primarily for sequential OS data sets and VSE files. It can be most effective when previous real storage limitations prevented the use of larger buffer sizes in general and optimum buffer sizes for disk data sets. In addition to reducing the number of I/O

requests required to process the data set or file, the total I/O time required to process a data set is reduced, as would occur in a native environment.

This technique should be used taking into account the amount of real storage present in the system. If the buffer size of several data sets that are being processed concurrently is increased considerably or made large initially, the amount of real storage that must be temporarily locked also increases considerably and potentially increases the number of active pages. This may adversely affect system performance in systems with a relatively limited amount of real storage available for paging.

- Using any operating system options that reduce the number of Start Subchannel instructions required. Preallocated z/OS work data sets should also be used wherever possible.
- Using virtual storage to contain small temporary sets of data instead of a temporary data set or file on an I/O device. This substitutes CP paging I/O operations for virtual machine-initiated I/O operations to the temporary data set or file. Paging I/O operations require less CP processing to start than usual I/O requests from virtual machines because CCW translation and indirect data address list (IDAL) construction is not required.
- Making routines and data resident in virtual storage so that paging I/O is substituted for virtual machine-initiated I/O operations. For example, increase the size of the link pack area in z/OS. The most frequently used z/OS transient SVC routines should be made resident in the Control Program area, as should frequently used operating system access methods and user-written reenterable routines. ISAM and VSAM index levels should be made resident in virtual storage, as permitted by VSE and z/OS operating systems.
- Increasing the minidisk file cache size. This enables the CMS minidisk file system to read or write more blocks of data per I/O when doing sequential I/O operations. For details on adjusting this cache size, see "Adjusting the Minidisk File Cache Size" on page 54.

- Both the CP LOCATEVM and LOCATE (Storage) commands can consume very large amounts of resources when the given range for the locate is large. The LOCATEVM command is a class G command and therefore permits any end user to greatly impact system performance. To avoid potential problems, use the MODIFY command or statement to change the class of the LOCATEVM command.

In addition, the following steps can be taken to improve the performance of **application programs** in a z/VM environment:

- Use code that does not modify itself. Use of reenterable code can reduce the amount of page-out activity required.
- Structure new application programs to operate efficiently in a paging environment.

  This is done by structuring programs to achieve a reasonable balance between page faults and real storage requirements. The extent to which this can be done can vary widely by installation. The benefits that can be obtained should be evaluated in light of the additional programming effort required.

  In this respect, deciding on the degree to which programs should be structured for efficient operation in a paging environment is similar to deciding how a high-level language should be used. The emphasis can be on the most efficient program execution, which can require more programming effort, or on the most efficient use of programmer time, which can result in less efficient programs. Alternatively, there can be a trade-off between programmer time and program efficiency (only the most frequently or heavily-used programs are optimized, for example).

  Many of the general program structure techniques discussed do not require a considerable amount of additional effort or knowledge on the part of programmers—only that they adopt a particular programming style. All of the suggested techniques can be used by assembler language programmers. Some can be used with certain high-level languages and not with others.

  Two major steps can be taken to structure programs to use real storage more efficiently and to incur the smallest possible number of page faults. The first is to adopt a modular programming style. The second is to take page boundaries into account and to package program code and data into page groups.

  The objective of adopting a modular programming style is to construct a program that consists of a series of subroutines, each of which contains a relatively small number of active pages. The objective of packaging code within pages is to group active code together to avoid crossing page boundaries in

such a way that more real storage than is really necessary is required to contain the active pages of a subroutine.

To cause references to active instructions and data to be localized, the following general rules should be applied to programs:

- A program should consist of a series of sequentially processed subroutines or subprograms. The mainline of the program should contain the most frequently used subroutines in the sequence of most probable use, so that processing proceeds sequentially, with calls being made to the infrequently used subroutines, such as exception and error routines. This structure contrasts with one in which the mainline consists of a series of calls to subroutines. Frequently used subroutines should be located near each other. Infrequently used subroutines that tend to be used at the same time whenever they are processed should be located near each other also.
- The data most frequently used by a subroutine should be defined together so that it is placed within the same page or group of pages instead of being scattered among several pages. If possible, the data should be placed next to the subroutine, so that part or all of the data is contained within a page that contains active subroutine instructions (unless the routine is written so that it is not modified during its execution). This eliminates references to more pages than are actually required to contain the data and tends to keep the pages with frequently referenced data in real storage.
- Data that is to be used by several subroutines of a program (either in series or in parallel by concurrently running subtasks) should be defined together in an area that can be referenced by each subroutine.
- A data field should be initialized as close as possible to the time when it will be used to avoid a page-out and a page-in between initialization and first use of the data field.
- Structures of data, such as arrays, should be defined in virtual storage in the sequence in which they will be referenced, or referenced by the program in the sequence in which a high-level language stores them (by row or by column for arrays, for example).
- Subroutines should be packaged within pages when possible. For example, avoid starting a 2500-byte subroutine in the middle of a 4 KB page so that it crosses a page boundary and requires two page frames instead of one when it is active. Subroutines that are smaller than page size should be packaged together to require the fewest number of pages, with frequently used subroutines placed in the same page when possible. This applies to large groups of data as well.

- Restructure existing application programs to incur as few page faults as possible, to use the least amount of real storage, and to take advantage of the program structure facilities that a virtual storage environment offers.

  This can be accomplished by:

  - Using the techniques described previously
  - Taking planned overlay and dynamic structure programs out of these structures
  - Combining into one logical job step two or more steps of a job that would have been one job step if the required real storage were available.

  The last of these techniques can eliminate redundant I/O time that is currently used to perform such functions as reading the same sequential input data into two or more job steps and writing intermediate results from one job step in one or more sequential data sets for input to the next job step.

- Exploit VM Data Spaces. Use of data spaces can minimize storage and processor requirements.

  Data spaces provide increased virtual storage addressability and sharing capability. By using the increased addressability, an application can buffer more in storage and decrease the processor requirements associated with virtual I/O. The data may still need to be moved in and out of real storage, but the CP paging subsystem can move the data more efficiently than virtual I/O. The use of VM Data Spaces also extends the concept of sharing data. This has two chief advantages:

  1. It reduces storage requirements. One copy can be shared among many virtual machines instead of a copy per virtual machine.
  2. It reduces the need to transfer data by IUCV, APPC/VM, or some other communication vehicle.

In addition, there are other functions associated with data spaces that can improve application performance. These are:

- Minidisk mapping extends the concept of applications using storage and letting CP handle the real I/O. This new function provides a means of associating CP minidisk data with data spaces. One or more minidisks can be mapped to one or more data spaces. An application retrieves the data simply by referencing the corresponding storage location in the data space. The real I/O is handled by the CP paging subsystem, which provides efficiencies over virtual machine I/O. The CP macro MAPMDISK manages this function.
- The asynchronous page fault capability can be used to deal with the problem of serial page faulting. This is a bigger problem in server applications because not only does the server virtual machine wait, but all virtual machines with outstanding requests wait as well. Asynchronous page faults allow a virtual machine to process other work (a different task) while CP handles the page fault. The implementation applies only to page faults of data space pages. CP will provide an interrupt when the page fault is complete. At that time, the server application can finish processing the original task associated with the page fault. The application needs to have a structure that lends itself to tasking and needs to support establishing and handling page fault interrupt logic. The asynchronous page fault function is selected on a data space by data space basis when the data space is added to the access list. The CP macro PFAULT establishes a token for handshaking with CP.
- The page reference pattern function provides a method for the application to give hints to CP as to which pages will be used in the near future. This can be used to help avoid page faults. This function is accomplished with the CP macro REFPAGE, which can be used for the primary address space or data spaces. In all cases, the virtual machine must be running in XC mode.

- Turn off minidisk cache when inappropriate

  Applications, such as programs that scan data or backup data, should turn off minidisk cache insertion to avoid filling the cache with data that will not be referenced again in the near future. This can be done with the SET MDCACHE INSERT command.

Adjusting the Minidisk File Cache Size

When a file is read sequentially, CMS reads as many blocks at a time as will fit into the file cache. When a file is written sequentially, completed blocks are accumulated until they fill the file cache and are then written together.

There is a separate file cache for each sequentially opened file. The minidisk file cache size is specified on a CMS system basis at the time that the CMS nucleus is built. (See Chapter 13, “SFS Tuning,” on page 129, for information on tuning of the SFS file cache size.)

The default CMS minidisk file cache size is 8 KB. You may specify a value from 1 to 96 KB in size when building a CMS nucleus. Note that there are some system restrictions that impact the file cache size used for any given file:

1. A maximum of 24 CMS disk blocks of file data can be cached
2. The cache size will not exceed the size of the file, rounded up to the next multiple of the disk block size. For example, if a file consisted of 60 fixed length records, each 80 bytes long, and the disk has been formatted at 1 KB, its cache size is limited to 5 KB.
3. An integral number of CMS minidisk blocks will be cached. A number that is not an integral multiple of the minidisk block size is rounded up to the next multiple of the minidisk block size.
4. The minidisk file system caches a minimum of 1 CMS minidisk block. A value specified that is less than the disk block size is treated as single CMS disk block.

Table 2. Effective Cache Size		
Disk Block Size	**Maximum Effective Minidisk File Cache**	**Minimum Effective Minidisk File Cache**
4096	96 KB	4 KB
2048	48 KB	2 KB
1024	24 KB	1 KB
512	12 KB	1 KB

The optimal file cache size depends upon how much real storage is available as indicated by the paging rate. The more real storage that is available, the bigger optimum file cache size.

The file cache size is specified on a CMS system basis when the nucleus is built. You can check the coding of the DEFNUC macro in your CMS nucleus generation profile (DMSNGP for ESA/390 CMS, DMSZNGP for z/Architecture CMS) to determine the current setting for the cache buffers. To change the minidisk cache size setting, you need to update the MDBUFSZ parameter in the DEFNUC macro in the DNSNGP ASSEMBLE or DMSZNGP ASSEMBLE file. Then assemble the file and rebuild the CMS nucleus.

Guidelines for Virtual Machine Storage

The size of defined virtual machine storage and location of saved segments can impact performance, depending on workload and virtual machine size. The main issue is storage for the CP control blocks used to manage virtual storage. These control blocks are the page tables, segment tables, and higher level (region) tables used in dynamic address translation (DAT).

Page Tables, Segment Tables, and Higher Level Tables

CP keeps the page tables in page management blocks (PGMBKs). Each 8 KB PGMBK references 1 MB of virtual machine storage. PGMBKs might be pageable; for example, PGMBKs for nonshared guest pages are pageable. CP creates a PGMBK only when the corresponding MB of virtual machine storage is first referenced. They are not created for MB gaps between DCSSs, or between the top of the virtual machine and the first DCSS. Therefore the impact of PGMBKs on real storage depends on how frequently the MBs of storage they reference are used.

Segment tables and region tables are allocated from host real storage and are not pageable:

- To reference the page tables for a primary address space or data space up to 2 GB, 1 - 4 contiguous frames are allocated for the segment table, one frame for each 512 MB of storage.
- For a primary address space larger than 2 GB, multiple segment tables are created, plus one or more region tables to reference the segment tables. Each region table occupies 1 - 4 contiguous frames. If needed, multiple levels of region tables are created.

Also, the CP DEFINE STORAGE CONFIG command allows you to define multiple, noncontiguous storage extents. Segment tables and region tables need to be constructed to the upper bound of the storage extents. PGMBKs continue to be created only for storage that is referenced.

PGMBKs, segment tables and region tables can reside anywhere in host real storage, but CP creates them above 2 GB if possible.

Saved Segments

For shared page ranges within a saved segment that is loaded SHARE, the associated segment table entries will point to the same page tables. However, for a saved segment that is loaded NOSHARE, or for exclusive page ranges within a saved segment, unique page tables are created for each user.

Segment spaces and member segments can be loaded at addresses up to 2047 MB; DCSSs can include addresses up to 512 GB. Because CP dynamically expands the size of a virtual machine to incorporate a saved segment loaded at an address outside the virtual machine, the DAT tables for the virtual machine

also expand. Therefore if many virtual machines load a saved segment defined at a high storage address, it might affect real storage availability.

Increased Paging on CMS Intensive Systems

This topic covers the affects of increased paging and ways to reduce it.

Preventing Oversized Working Sets

CMS function can be provided as modules on the CMS S-disk or other common system disks. This function could be base CMS function or other program products. The NUCXLOAD function is often used to make the appropriate module a nucleus extension to a user's virtual machine when the command is executed the first time. NUCXLOAD loads the module into free storage in the user's virtual machine where it remains for the duration of the user's CMS session or until explicitly dropped. Subsequent calls to the module will not require reading the module from DASD because it is already in free storage.

A side effect is that virtual machines can experience increased working set sizes which will lead to increased loads on the z/VM system's paging subsystem. This increased paging load can be substantial if the nucleus extensions are frequently used. You can check this by obtaining a NUCXMAP from a sampling of your CMS users to see which modules show up frequently in the NUCXMAP output.

Using Logical Segment Support

z/VM logical segment support can provide relief from the increased working set size. By placing modules into a logical segment, all users on the system can use shared storage instead of private storage for those modules.

When modules are placed into a CMS logical segment and made available to the virtual machine by way of the SEGMENT LOAD command, the modules are automatically made nucleus extensions to the CMS nucleus in the virtual machine. The pages occupied by the modules are shared among all users of the logical segment instead of private pages as is the case when the logical segment is not used.

An example of the procedure to place modules into a logical segment follows. A potential location for the logical segment containing the modules is the HELPINST physical segment which contains the logical segments named HELP and CMSINST. The HELPINST physical segment must be installed and saved below the 16 MB virtual line. This makes HELPINST a good choice for adding a logical segment to contain modules that also must reside below the 16 MB line. Installation of the HELPINST physical segment is documented in *z/VM: Installation Guide*. Modules that are not restricted to being loaded below the 16 MB virtual line could be saved in a different segment.

CMS logical segment support enhances CP's capabilities with respect to segments in several ways. One enhancement is that a logical saved segment can contain different types of program objects, such as modules, text files, execs, callable services libraries, language information, and user-defined objects, or minidisk information. Another enhancement is that you can save several different logical segments into a physical saved segment and allow end-users to access only the specific logical saved segments that they need rather than the entire physical saved segment. (For more information, see the section on saved segment planning considerations in *z/VM: Saved Segments Planning and Administration*.)

Creating a Saved Segment That Contains CMS, HELP, and INSTSEG Modules

The following example demonstrates how to add a new logical segment to contain CMS modules into the previously-defined HELPINST segment.

Note: This is only an example of how to create a logical segment for CMS modules. The module names used are only examples. Therefore, you do not need to build this segment for your system. See *z/VM: Installation Guide* for details. The procedure is shown here to illustrate the steps required if you wanted to create a logical segment for your own use.

The steps for adding a new logical segment to the HELPINST physical segment (PSEG) are as follows:

1. Logon to MAINT*vrm*
2. Enter the command

```
VMFSGMAP SEGBLD ESASEGS SEGBLIST
```

   to view the segment map.
3. Locate the line for the HELPINST segment that was predefined by IBM. Move the cursor onto that line and press PF4=Chg_Obj.
4. The display should be modified to add the "PROD(LSEG CMSMODS)" into the BLDPARMS section. Update the OBJDESC description comment also. The display should look like this.

```
                          Change Segment Definition
                                                          Lines 1 to 13 of 13

 OBJNAME....:  HELPINST
 DEFPARMS...:  C00-CFF SR
 SPACE......:
 TYPE.......:  PSEG
 OBJDESC....:  CMSINST AND HELP AND CMSMODS LSEGS
 OBJINFO....:
 GT_16MB....:  NO
 DISKS......:
 SEGREQ.....:
 PRODID.....:  2VMVMA10 CMS
 BLDPARMS...:  PPF(ESA CMS DMSSBINS) PPF(ESA CMS DMSSBHLP) PROD(LSEG
            :  CMSMODS)



 F1=Help     F2=Get Obj   F3=Exit      F4=Add Line  F5=Map        F6=Chk MEM
 F7=Bkwd     F8=Fwd       F9=Retrieve F10=Seginfo   F11=Adj MEM   F12=Cancel
====>
```

   Figure 5. Modifying HELPINST to contain CMSQRY LSEG
5. After adding to the BLDPARMS, press F5 to return to the MAP.
6. From the Segment Map screen, press F5=FILE to file the changes and exit VMFSGMAP.
7. Create the CMSMODS LSEG file

   The CMSMODS LSEG file contains the information for the new logical segment that will contain the CMS disk-resident modules. Create the CMSMODS LSEG file on MAINT*vrm*'s 191-A disk. It needs to be on the A disk because when VMFBLD creates the PSEG file, it includes PROFILE/EPIFILE options for CMSINST and HELP that cause disks other than A, S, Y and the VMSES/E disks to be released. This is alright in this case since we desire to have the modules from the 190-S disk loaded. But the releasing of the disks means that the CMS local modification disk (MAINT*vrm*'s 3C4, by default) will not be accessed. The following modules are saved in this example:

```
   * LOAD THESE MODULES FROM S DISK INTO AN LSEG
   MODULE DMSEX1 * ( SYSTEM )
   MODULE DMSEX2 * ( SYSTEM )
   MODULE DMSEX3 * ( SYSTEM )
   MODULE DMSEX4 * ( SYSTEM )
   MODULE DMSEX5 * ( SYSTEM )
   MODULE DMSEX6 * ( SYSTEM )
   MODULE DMSEX7 * ( SYSTEM )
   MODULE DMSEX8 * ( SYSTEM )
```

   Figure 6. CMSMODS LSEG file
8. Create a local modification to the SYSPROF EXEC using VMSES/E procedures to cause the CMSMODS segment to be used. For information on making a local modification to the source file SYSPROF $EXEC, see the section on installing local service and modifications in *z/VM: Service Guide*.

   Where the SYSPROF EXEC has:

```
'RTNLOAD * (FROM VMLIB SYSTEM GROUP VMLIB'
```

   Insert the following line before the RTNLOAD statement:

```
'SEGMENT LOAD CMSMODS (SYSTEM' /* Get CMS modules */
```

The result should look like this:

```
⋮
'SEGMENT LOAD CMSMODS (SYSTEM' /* Get shared QUERY and SET modules */
'RTNLOAD * (FROM VMLIB SYSTEM GROUP VMLIB'
⋮
```

It is important that the 'SEGMENT LOAD CMSMODS' statement appear before any of the CMS modules are used.

Follow the instructions in *z/VM: Service Guide* to apply your local modification and to get the updated SYSPROF EXEC on the 190 and 490 minidisks.

9. IPL the 190 minidisk and prohibit the SYSPROF EXEC from running and also the CMSINST segment from loading:

   ```
   IPL 190 CLEAR PARM NOSPROF INSTSEG NO
   ```

10. Prohibit execution of the PROFILE EXEC:

    ```
    ACCESS (NOPROF
    ```

11. Reserve the storage where you will be loading the data to be saved:

    ```
    SEGMENT RESERVE HELPINST
    ```

12. Run the PROFILE EXEC:

    ```
    EXEC PROFILE
    ```

13. Execute the VMFBLD EXEC to cause the HELPINST segment to be rebuilt.

    ```
    VMFBLD PPF SEGBLD ESASEGS SEGBLIST HELPINST (ALL
    ```

14. Use VMFVIEW to view the build message log.

    ```
    VMFVIEW BUILD
    ```

    You should see the message:

    ```
    VMFBDS2003W The SYSTEM SEGID D(51D) file has been changed and must
    be moved to the S disk.
    ```

    in the build message log. Copy the SYSTEM SEGID file from the 51D minidisk to both the 190 and 490 and ensure that it has a filemode of **2**.

15. Resave the CMS NSS in order to regain the S-stat.

    Because the SYSTEM SEGID file was altered on the 190 disk, it is necessary to resave the CMS named saved system to regain the shared S-stat. Use the procedures appropriate for your environment (for example: SAMPNSS CMS followed by IPL 190 CLEAR PARM SAVESYS CMS).

    If some CMS users on the system are unable to load this segment because of conflicts with their Page Allocation Table (PAT) or a conflict with another saved segment that they need, then those users will still have private copies of the any individual modules that get dynamically NUCXLOADed because of a QUERY or SET command that they enter.

16. Repeat steps 9 to 15 whenever the modules are serviced.

    In order to have service changes take effect, several of the above steps need to be repeated. The message referred to in step 14 should not appear this time, in which case you may skip step 15.

The only change that the user might see as a result of this process is after implementing this new logical segment, the response to a 'NUCXMAP' command is likely to be longer than end-users typically see. A

typical response to NUCXMAP just after IPLing CMS and after the SYSPROF runs, will be similar to the following. While longer, the response is perfectly valid:

```
Ready;
nucxmap (seginfo
Name      Entry    Userword Origin   Bytes   Amode Segname
DMSEX1    00CCDF80 00000000 00CCDF80 00002AA0  ANY CMSMODS
NAMEFSYS  00E7FB28 00DE22C8 00E7FB28 00000000   31
NAMEFIND  00E7FB28 00DE2930 00E7FB28 00000000   31
DMSEX2    00CCD678 00000000 00CCD678 00000908  ANY CMSMODS
NAMEFUSE  00E7FB28 00DA5B58 00E7FB28 00000000   31
DMSEX3    00CBE1B0 00000000 00CBE1B0 00000440   31 CMSMODS
DMSEX4    00CBE5F0 00000000 00CBE5F0 00001188   31 CMSMODS
DMSEX5    00CBF778 00000000 00CBF778 00002E90  ANY CMSMODS
DMSEX6    00CC2608 00000000 00CC2608 00002510  ANY CMSMODS
DMSEX7    00CC4B18 00000000 00CC4B18 00001338  ANY CMSMRDS
DMSEX8    00CC5E50 00000000 00CC5E50 00000408   31 CMSMODS
PIPMOD    040C1EA0 00000000 040C1EA0 0003A6A0   31 PIPES
Ready;
```

Dynamic SMT to Compare MT-2 to MT-1

Workload performance might vary widely with SMT. Individual processors of a system enabled for SMT with two threads in use per core (MT-2) perform slower than when a single thread is in use (MT-1). However, the benefit of SMT is that generally workloads running MT-2 can achieve higher throughput even though the individual processors are slower. Workload characteristics determine whether a workload benefits from SMT and to what extent. For example, workloads that are unable to spread work over the additional processors see less benefit, if any. The characteristics of work dispatched concurrently on the processors of the same core benefit less, and may be inhibited when there are high levels of competition for core resources.

To determine whether a workload benefits from SMT, compare the external workload throughput of a MT-2 system to a MT-1 system. A system that is enabled for multithreading but has only one activated thread per core performs similarly to a system with multithreading disabled. For this study be sure the z/VM system is enabled for SMT. Use the CP SET MULTITHREAD command to dynamically change the activated threading level, without doing a system IPL.

External throughput rates are an important factor for comparison, but other factors might be important as well. When evaluating total compute cost remember that individual processors are slower when running MT-2, so it is likely that total processor time is higher, even though less total core capacity may be consumed. The SMT Performance Report provides examples of how workloads running MT-1 and MT-2 might be compared.

When the SET MULTITHREAD command changes the activated threading level of cores, there are several implications on Monitor record reporting.

- A new Monitor event record D5 R21 MRPRCSMT is generated at the beginning and end of an SMT configuration change.
- Between the start and end D5 R21 records, additional Monitor event records D5 R1 (MRPRCVON) or D5 R2 (MRPRCVOF) are generated for each processor that is activated or deactivated; respectively.
- The number of per processor sample records generated during each Monitor interval will change as a result of the processor activations and deactivations. Processors that are deactivated are removed from the logical processor configuration.
- Multithreading metrics reported in MRSYTPRP D0 R2 cannot be calculated correctly for intervals that intersect with any portion of the transition for the SMT configuration change. This condition is indicated by metric values equal to SYTPRP_KNODATA_TRANSITION.

Additional considerations on the effects of SMT configuration changes on Monitor reporting are described in the prolog of the D5 R21 MRPRCSMT Monitor record.

Because of the foregoing considerations, caution should be used in interpreting data from Monitor records during the transition. It would be reasonable to treat the periods prior to and after the multithreading

configuration change as completely different configurations during evaluation and completely exclude the period of the transition.

If the results show a workload degradation with the MT-2 environment when compared to the MT-1 environment, then it should be determined whether the workload has a single point of serialization. For example, a workload might become serialized by the number of virtual processors. Adding more virtual processors might remove the bottleneck. For more information on how to determine server bottlenecks and potential resolutions, refer to the Simultaneous Multithreading (SMT) (https://www.ibm.com/vm/perf/reports/zvm/html/1q5smt.html).

Chapter 4. SFS Performance Guidelines

To prevent shared file system (SFS) performance problems you should:

1. Follow the SFS performance guidelines provided in this section.

   This task, also known as *preemptive tuning*, involves reviewing and following a list of performance guidelines when you are defining a new file pool or modifying the structure of an existing one.
2. Plan system capacity.

   Capacity planning involves anticipating future hardware needs so that any necessary upgrades can be made before performance becomes unacceptable. Capacity planning should be done at a system level. Server processing is just one factor you must consider when planning the capacity of your system. For more about capacity planning, see *z/VM: CMS Planning and Administration*.

The remainder of this section contains lists of performance and availability tips that, if followed, will help ensure that your servers are processing efficiently. Many of these guidelines are integrated into the various maintenance procedures described in *z/VM: CMS File Pool Planning, Administration, and Operation*. If you follow the procedures outlined there, you will have already followed the appropriate guidelines. The guidelines are repeated here so that you can easily verify that your file pools are properly tuned.

For a discussion of SFS performance measurement tips, see Chapter 10, “Monitoring SFS Performance,” on page 97. For a discussion of SFS performance tuning, see Chapter 13, “SFS Tuning,” on page 129. If followed, these will help ensure that your servers are processing efficiently.

Multiple File Pools

1. For large systems, determine whether you will require multiple production file pools. See *z/VM: CMS File Pool Planning, Administration, and Operation* for more information.
2. For best performance, the server's executable code should be run in a saved segment. This allows all such servers to be executing the same copy of the code, thus reducing system real storage requirements. The CMSFILES saved segment is automatically loaded during z/VM installation. CMSFILES contains the executable code for SFS and CRR servers (SFS and CRR servers share the same executable code). See the description of the SAVESEGID start-up parameter in *z/VM: CMS File Pool Planning, Administration, and Operation* for more information.

CP Tuning Parameters

1. Using your local operating procedures, update the server's z/VM system directory to add the QUICKDSP operand to the OPTION control statement.

   The QUICKDSP operand ensures that the servers will not have to wait in the eligible list for system resources to become available. For more information on the QUICKDSP operand, see *z/VM: CP Planning and Administration*.
2. Using your local operating procedures, update the server's z/VM system directory to add the SHARE REL 1500 control statement.

   The SHARE REL 1500 control statement places server machines in a more favorable position in the z/VM dispatch queue. For more information on the SHARE control statement, see *z/VM: CP Planning and Administration*.
3. Set up the SFS file pool server's system directory for appropriate use of minidisk caching and control unit caching.

   - NOMDCFS

Specify NOMDCFS on the OPTION directory control statement. This allows the SFS file pool server to use minidisk caching at a rate that exceeds the fair share limit. This is important because the SFS file pool server typically performs file activity on behalf of many end users.

- log minidisks

  Make the SFS log minidisks ineligible for minidisk caching. This can be done by specifying NOMDC on the MINIOPT directory control statements for those minidisks. The SFS logs do not benefit from minidisk caching because the I/O activity to them is almost entirely writes. Caching the SFS logs degrades system performance.

  The same applies to control unit caching when only read caching is available. Specify NOCACHE on the MINIOPT directory control statements.

  Because of the high amount of write activity, control unit caching is quite beneficial when the IBM DASD Fast Write feature is available. In that case, specify CACHE on the MINIOPT directory control statements (or let it default to CACHE).

- control minidisk

  Make the control minidisk ineligible for minidisk caching and control unit caching by specifying NOMDC and NOCACHE on the MINIOPT directory control statement. This minidisk is not a good candidate for caching because SFS already does extensive caching of this minidisk within the SFS file pool virtual machine.

  Note: The size of that cache is controlled by the CTLBUFFERS file pool startup parameter.

- storage group minidisks

  These minidisks are good candidates for minidisk caching and control unit caching, so they should be left eligible (which is the default).

4. Certain CP settings can affect SFS's use of VM Data Spaces. These considerations are covered in "VM Data Spaces" on page 63.

CMS Tuning Parameters

1. Choose the USERS server start-up parameter value carefully. This parameter tells a server how much work it should configure itself to handle. If the USERS start-up parameter is not correctly specified, the server may run inefficiently. See *z/VM: CMS File Pool Planning, Administration, and Operation* for information on how to select a suitable value.
2. The CMS SFS file cache defaults to 20 KB for SFS files. This is a good choice for systems that have moderate paging rates. If your system has an especially high paging rate, you may want to consider setting this to a lower value. If your system has a low paging rate, performance is likely to benefit if you increase the SFS file cache size. To change the SFS file cache size, you need to update the BUFFSIZ parameter in the DEFNUC macro, which is in DMSNGP ASSEMBLE (DMSZNGP ASSEMBLE for z/Architecture CMS). Then assemble the file and rebuild the CMS nucleus. See *z/VM: CMS Planning and Administration* for information about DMSNGP, DMSZNGP, and DEFNUC. See *z/VM: Service Guide* for instructions on rebuilding the CMS nucleus.

DASD Placement

1. Any given DASD volume should only contain minidisks from one file pool. This is to ensure that a DASD failure on any given volume will only affect one file pool. (Consequently, you can disregard this recommendation for cases such as test file pools where recovery is not a significant consideration.)
2. For integrity purposes, each of the two SFS log minidisks should be put on different devices. For SFS file pool server performance purposes, it is best to have each of the two SFS log minidisks also be on separate channels and control units. This maximizes the likelihood that a server can do I/O to the two logs in parallel, thus reducing response time.
3. When each of the two SFS log minidisks are placed on DASD volumes that have no other I/O activity, server log I/O is optimized because there is very little seek activity. Therefore, to minimize seek time, place each of the two SFS log minidisks on a DASD volume that otherwise has low I/O activity. If that

is not practical, place them on DASD volumes where most of the I/O activity is to a small area on that volume, and place the log minidisk adjacent to this area.

Another way to maximize log I/O responsiveness is to place the logs on DASD volumes in IBM DASD subsystems that support the DASD fast write function.

4. The placement of the control minidisk is not critical to server performance because it normally has a relatively low level of I/O activity. It is, however, a good idea to place the control minidisk on the volume (or one of the volumes) that contains storage group 1. The control minidisk and storage group 1 are recovered together. Placing the control minidisk on one of the same volumes that holds storage group 1 reduces the total number of volumes that contain these areas. This reduces the probability that a given DASD failure will be on a volume that contains one or more of these areas.
5. A sizable fraction of all SFS file pool server I/Os are to storage group 1. Therefore, it may be necessary to spread the minidisks of storage group 1 across multiple DASD volumes. This prevents any one volume from becoming an I/O bottleneck. Note that this consideration is for I/O performance, while in item "6" on page 63 it addresses SFS availability.
6. It is best not to spread SFS storage group 1 across more volumes than you really need. The more volumes that storage group 1 resides on, the greater the chance that a DASD failure will be on one of these storage group 1 volumes. Whenever a DASD failure occurs on a storage group 1 volume, the file pool control data must be restored, which requires you to stop multiple user mode operation.
7. An SFS storage group 1 minidisk is a relatively small, I/O-intensive area. Therefore, for any given volume, place the storage group 1 minidisk adjacent to any other small, frequently referenced areas on that volume in order to minimize seek time.
8. Because of I/O load balancing considerations, it may often be necessary to have part of storage group 1 on the same volume as one of the user storage groups. However, it should not generally be necessary to place more than one user storage group on a given volume. It is best to have just one user storage group per volume so that you do not need to restore more than one user storage group if a DASD failure on a given volume occurs.
9. When a storage group spans volumes, a server satisfies storage requests for that storage group by allocating space evenly across those volumes. This tends to spread the I/O demand for that storage group evenly across those volumes. For the overall I/O demand to those volumes to be balanced, you should follow these guidelines:
   - Make sure that any non-SFS space on those volumes is of low usage or uniform usage.
   - Avoid volumes with consistently high usage.
   - Make sure that all volumes within a storage group are of the same device type, or have similar performance characteristics.
   - Give the storage group the same amount of space on each volume.

VM Data Spaces

1. Put read-only files that are to be shared among large numbers of users into directory control directories that are set up to reside in data spaces. See *z/VM: CMS File Pool Planning, Administration, and Operation* for a discussion on how to do this.
2. The processing required to access SFS directories can affect responsiveness during system startup. This effect is normally insignificant, but it can be an important consideration if the directories being accessed contain large numbers of files and if they are accessed by large numbers of users during CMS initialization. In such cases, system startup delays can be minimized by placing such files into directory control directories that are set up to reside in data spaces.
3. Directories that use data spaces should only contain files that are seldom updated.
4. When making updates to a directory that uses data spaces, it is best if you group the updates together and make those changes when relatively few users have that directory accessed. This minimizes the formation of multiple versions of that directory. Each such version requires its own data space.
5. Encourage your CMS users to run in XC mode, as that allows the full benefits of VM data spaces to be realized. CMS users that run in XA or ESA mode should be able to run in XC mode without any

problems. However, neither z/CMS nor guest operating systems like Linux, z/VM, z/OS, and z/VSE® can run in XC mode.

6. For maximum availability, consider placing directories that are to be shared read-only by many users into a separate, "read-only" file pool. This topic is discussed in *z/VM: CMS File Pool Planning, Administration, and Operation*.

Recovery

1. Consider these suggestions if you are interested in minimizing the amount of time required to restore the control data of a file pool:
   - Keep the file pool from growing too large.
   - Do control data backups more frequently. The less file pool change activity that has occurred since the last control data backup, the less time it will take to reapply these changes during control data recovery.
   - Do your control data backups to another file pool. Then, should a control data restore be required, SFS can do double buffering, which reduces restore time.
   - Eliminate formatting time during control data recovery by creating and formatting a spare set of minidisks ahead of time. The sizes of these minidisks need to match those of the control minidisk and storage group 1 minidisks associated with the file pools of interest. These spare minidisks should be placed on a volume or volumes that do not contain any of the minidisks that they are spares for.
   - Specify a very large catalog buffer pool when doing the restore. For example, for a 32 MB virtual machine, try setting CATBUFFERS to 5000.
   - Place the logs and control data on DASD volumes in IBM DASD subsystems that support the DASD fast write function. To determine if the subsystem supports the DASD fast write function, and how to activate this function, see *z/VM: System Operation*.
2. The more storage groups you have, the less time it takes to recover any one of them. Define enough storage groups such that your recovery time requirements are met. Be aware, however, that having more storage groups entails additional administrative overhead.
3. Specifying a large CATBUFFERS value helps the performance of user storage group restores (FILEPOOL RESTORE). If the server is running in multiple user mode, however, you might not want to interrupt user activity by shutting down the server just to temporarily increase CATBUFFERS.

Catalog Reorganization

1. Although not required, it helps performance if you reorganize the file pool catalogs when catalog fragmentation is detected. (See *z/VM: CMS File Pool Planning, Administration, and Operation* for instructions on how to determine this). When the catalogs are properly organized, the I/Os to them are minimized.
2. The time required to reorganize the file pool catalogs can be significantly reduced if you specify a very large catalog buffer pool. Try using a CATBUFFERS start-up parameter value of 5000. After reorganization processing is completed, reset CATBUFFERS to its original value.

SFS Applications

For a list of performance considerations for your programs when using SFS files, see the SFS performance tips section in *z/VM: CMS Application Development Guide*.

Chapter 5. CRR Performance Guidelines

To prevent Coordinated Resource Recovery (CRR) performance problems you should:

1. Follow the CRR performance guidelines provided in this section.

   This task, also known as *preemptive tuning*, involves reviewing and following a list of performance guidelines when you are defining a new CRR server or modifying the structure of an existing one.

2. Plan system capacity.

   Capacity planning involves anticipating future hardware needs so that any necessary upgrades can be made before performance becomes unacceptable. Capacity planning should be done at a system level. Server processing is just one factor you must consider when planning the capacity of your system. For more about capacity planning, see *z/VM: CMS Planning and Administration*.

The remainder of this section contains lists of performance and availability tips that, if followed, will help ensure that your servers are processing efficiently. Many of these guidelines are integrated into the various maintenance procedures described in *z/VM: CMS File Pool Planning, Administration, and Operation*. If you follow the procedures outlined there, you will have already followed the appropriate guidelines. The guidelines are repeated here so that you can easily verify that your CRR servers are properly tuned.

For a discussion of CRR performance measurement tips, see Chapter 11, “Monitoring CRR Performance,” on page 99. For a discussion of CRR performance tuning, see Chapter 14, “CRR Tuning,” on page 143.

CRR Server Machine

1. If the CRR server is not running, users who use SFS (and other resources that participate in CRR) are in a condition called *limp mode*. While in this condition, two-phase commits are not possible. Therefore, applications that use protected conversations or other resources that do not support simple commit logic cannot be run. A user may experience significant SFS performance degradation while in *limp mode*. To avoid *limp mode*, IBM strongly recommends that every such system have the CRR server running. The generation of a CRR server (VMSERVR) and its associated file pool (VMSYSR) are optional z/VM installation tasks.
2. For best performance, the server's executable code should be run in a saved segment (SFS and CRR servers share the same executable code). This allows all such servers to be executing the same copy of the code, thus reducing system real storage requirements. The CMSFILES saved segment is automatically loaded during z/VM installation. CMSFILES contains the executable code for SFS and CRR servers. See the description of the SAVESEGID start-up parameter in *z/VM: CMS File Pool Planning, Administration, and Operation* for more information.

CP Tuning Parameters

1. Using your local operating procedures, update the server's z/VM system directory to add the QUICKDSP operand to the OPTION control statement.

   The QUICKDSP operand ensures that the servers will not have to wait in the eligible list for system resources to become available. For more information on the QUICKDSP operand, see *z/VM: CP Planning and Administration*.

2. Using your local operating procedures, update the server's z/VM system directory to add the SHARE REL 1500 control statement.

   The SHARE REL 1500 control statement places server machines in a more favorable position in the z/VM dispatch queue. For more information on the SHARE control statement, see *z/VM: CP Planning and Administration*.

3. Set up the CRR server's directory entry for appropriate use of minidisk caching and control unit caching.
   - log minidisks

     Be sure to make the CRR log minidisks ineligible for minidisk caching. This can be done by specifying NOMDC on the MINIOPT z/VM system directory control statements for those minidisks. The CRR logs do not benefit from minidisk caching because the I/O activity to them is almost entirely writes. Caching the CRR logs degrades system performance.

     The same applies to control unit caching when only read caching is available. Specify NOCACHE on the MINIOPT directory control statements.

     Because of the high amount of write activity, control unit caching is quite beneficial when the IBM DASD Fast Write feature is available. In that case, specify CACHE on the MINIOPT directory control statements (or let it default to CACHE).
   - other minidisks

     All other CRR server minidisks derive little benefit from caching, so specify NOCACHE and NOMDC on their MINIOPT directory control statements.

CRR Logs

Each CRR server has two CRR logs, in addition to the SFS logs that are also associated with both SFS file pool servers and CRR servers. Consider the following suggestions relating to CRR logs to help improve your system's performance:

1. For integrity purposes, each of the two CRR log minidisks should be put on different devices. For CRR server performance purposes, if you have high CRR log minidisk I/O activity, then the CRR log minidisks should be on separate channels and control units to optimize CRR log I/O processing.

   Note: You should get high CRR log minidisk I/O activity only if you have heavy usage of applications or participating products designed to exploit CRR. Such programs would have to concurrently update multiple protected resources (for example, multiple SFS file pools) or use protected conversations.
2. When each of the two CRR log minidisks are placed on DASD volumes that have no other I/O activity, server log I/O is optimized because there is very little seek activity. Therefore, to minimize seek time, place each of the two CRR log minidisks on a DASD volume that otherwise has low I/O activity. If that is not practical, place them on DASD volumes where most of the I/O activity is to a small area on that volume, and place the log minidisk adjacent to this area.

   Another way to maximize log I/O responsiveness is to place the logs on DASD volumes in IBM DASD subsystems that support the DASD fast write function.
3. The placement of all other CRR server minidisks is not critical to server performance because they normally have a relatively low level of I/O activity.
4. Size of the CRR log minidisks (also the sync point rate and amount of data being logged) affects the rate of CRR checkpoint logging. If the rate of CRR checkpoint logging is high and CRR performance is unacceptable (through analysis of monitor data), then increasing the size of the CRR log minidisks may help. The trade-off is a longer time duration for server startup and some resynchronization processing.

   The rate of checkpoint logging can be determined from the Log Checkpoints Taken field displayed in the output of the QUERY FILEPOOL CRR command. See *z/VM: CMS File Pool Planning, Administration, and Operation* for more information on the QUERY FILEPOOL CRR command. Follow these steps to calculate the rate of CRR checkpoint logging:

   a. Display the status of the CRR server by entering:

      ```
      query filepool crr
      ```

   b. Record the time (call it *T1*) that you issued the query command and record the value in the Log Checkpoints Taken field (call it *C1*).

   c. Again, display the status of the CRR server by entering:

```
query filepool crr
```

d. Again, record the time (call it *T2*) that you issued the query command and record the value in the Log Checkpoints Taken field (call it *C2*).

e. Use this formula to calculate the CRR checkpoint logging rate:

```
CRR CHECKPOINT LOGGING RATE = (C2 - C1) / (T2 - T1)
```

For more information on CRR performance measurement, see Chapter 8, “Monitoring System Performance,” on page 75. For more information on CRR servers and CRR logs, see *z/VM: CMS File Pool Planning, Administration, and Operation*.

Application Programs that use CRR

Application developers that write application programs that take advantage of CRR should consider the following to help improve the application's performance:

- Generally, the overhead for CRR processing increases as the number of protected resources increase. Protected conversations can be more costly than other protected resources. This is because special processing and the fact that protected conversations are always treated as updated resources (write-mode on).
- The number of work units may affect storage requirements and end of command processing.
- The number or rate of commit or rollback (backout) verbs issued (such as SRRCMIT or DMSCOMM, and SRRBACK or DMSROLLB) should be minimized. Use the commit and rollback verbs only when needed to minimize resource consumption.

CRR Participation

An IBM or non-IBM product (or resource) that wants to participate in CRR has to have these interfaces:

- Resource adapter to Synchronization point manager (SPM)
- SPM to resource adapter
- Resource adapter to its participating resource manager
- Participating resource manager to its resource adapter.

For information about the CSL routines used for registering a resource, see *z/VM: CMS Callable Services Reference*.

When registering a resource for CRR participation, you might want to consider the following suggestions for setting your registration values to help improve your system's performance:

- Set the **simple-commit flag** ON when possible. This allows the SPM to process some commit requests as simple commits instead of the more expensive two-phase commits.
- Set the **write-mode flag** ON only when necessary, and set it OFF as soon as possible. This also helps determine when simple commit processing can be used instead of two-phase processing. In addition, the processing for a read resource is less expensive than the processing for a write or update resource.
- Set the **CRR function flags** ON only when necessary. If you will not be participating for a short time, use the change registration (DMSCHREG) CSL routine to set a function OFF instead of unregistering.
- Set multiple values with a single change registration routine call instead of using multiple routine calls.
- Follow the requirements for communication between the resource adapter and the resource manager. If the request ID passed by the SPM is 0, then communication should be synchronous. Otherwise, use asynchronous communication to take advantage of parallelism. The SPM sets the request ID in the more efficient manner depending on the circumstances.

Chapter 6. AVS Performance Guidelines

APPC/VTAM Support (AVS) allows z/VM applications to use APPC/VM and communicate through an SNA LU6.2 network.

The performance of the AVS Virtual machine is affected by several parameters:

- Session pacing count parameters defined in the logon mode table (described in *z/VM: Connectivity*)
- AUTHEXIT parameter on the APPL statement issued for each gateway defined (described in *z/VM: Connectivity*)
- AVS tuning parameters.

The AGWTUN ASSEMBLE file, which is linked to the AVS module, contains AVS tuning parameters. The AVS virtual machine can use the default parameter settings that are provided and from a performance perspective, the default settings are adequate. However for other purposes, you may change the following parameters to meet the needs of your installation:

Pause count
Defines the number of transactions processed by AVS routines before control is relinquished.

Accounting record timer
Defines the number of hours between the creation of accounting records for all currently active conversations.

VTAM-VM request transformation control
Defines the balance of translating requests between APPC/VM and APPC/VTAM protocols and the resources used during the translations.

Problem dump count
Defines the number of problem dumps that could be taken during an AVS session. You can review this dump to diagnose the cause of the problem.

Chapter 15, “AVS Tuning Parameters,” on page 149 has a detailed description of the tuning parameters and the default settings, recommendations, and restrictions.

Chapter 7. TSAF Performance Guidelines

The Transparent Services Access Facility (TSAF) allows z/VM applications to use APPC/VM and to communicate with applications on other z/VM systems using the following:.

- Two APPC/VM paths
- Two or more TSAF virtual machines
- Communications across one or more physical connections.

CP Tuning Parameters

1. Using your local operating procedures, update the TSAF server's z/VM system directory to add the QUICKDSP operand to the OPTION control statement.

   The QUICKDSP operand ensures that the TSAF server will not have to wait in the eligible list for system resources to become available. For more information on the QUICKDSP operand, see *z/VM: CP Planning and Administration*.
2. Using your local operating procedures, update the TSAF server's z/VM system directory to add the SHARE REL 1500 control statement.

   The SHARE REL 1500 control statement places the TSAF server machine in a more favorable position in the z/VM dispatch queue. For more information on the SHARE control statement, see *z/VM: CP Planning and Administration*.

Line Performance Characteristics

TSAF functions that affect all the systems in a TSAF collection include:

- Identifying a new global resource or gateway in the TSAF collection
- Revoking a global resource or gateway from the TSAF collection
- Joining another TSAF collection.

How fast any of these functions complete is directly related to the speed of the slowest line that TSAF is using in the TSAF collection. A channel-to-channel (CTC) link is faster than a LAN link, which is faster than a BSC link. So, using a BSC line can significantly slow down TSAF functions that affect all the systems in a TSAF collection. The speed of an APPC link depends on the type of physical link that is controlled by VTAM. On average, the speed on an APPC link is comparable to a BSC link. TSAF always sends user data on the fastest route in the collection. Therefore, slow lines in the route would only slow down the transmission of user data if these lines had to be used for routing.

You should also consider the transmission error rate associated with each line in the TSAF collection. A BSC line with a fixed-line speed is less reliable and not as available if the number of transmission errors increases. TSAF assumes the error rate to be less than 1 bit in every 500,000 bits sent.

When the error rate is high, TSAF tends to break the communication path and mark the line *down*. The performance of the entire TSAF collection can degrade when TSAF must continually change the status of the line.

APPC Link and VTAM Performance

If your TSAF collection includes systems that are connected by physical VTAM links, you should establish SNA sessions and define explicit routing between the VTAM virtual machines on each system. You establish SNA sessions when you enter the AGW CNOS command (see *z/VM: Connectivity*). Use the VTAM PATH definition statement to define explicit routes between the VTAM virtual machines; see *ACF/VTAM Installation and Resource Definition* for more information.

By establishing explicit VTAM sessions and routes, you enable VTAM to perform the routing between intermediate processors in the TSAF collection when you establish APPC links to those systems. The TSAF virtual machines at the intermediate processors will not be involved in routing these conversations, thus reducing the data traffic through the TSAF virtual machines. In this way, explicit VTAM sessions and routes can enhance the overall performance of the TSAF collection. When you establish explicit VTAM sessions and routes between all systems in the TSAF collection, you create a logically fully-connected TSAF collection.

For example, Figure 7 on page 72 shows a group of three z/VM systems, each with TSAF and VTAM running virtual machines. VMSYS2 is connected to the other remote systems by physical links, labeled A, that are controlled by VTAM. To form a TSAF collection, each system establishes an APPC link to each of the other systems. Each system also defines VTAM sessions and explicit routing to each of the other systems in the group.

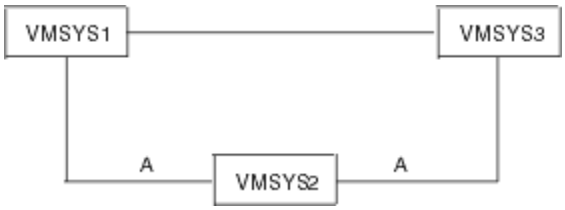


Figure 7. Remote Systems Connected by Two Physical VTAM Controlled Links

As Figure 8 on page 72 shows, the TSAF collection becomes fully-connected because there now is a logical link between VMSYS1 and VMSYS3. VMSYS2 is the physical intermediate system between VMSYS1 and VMSYS3. Because VTAM sessions and explicit routing is defined between each system, data sent from VMSYS1 to VMSYS3 is routed through the VTAM virtual machine on VMSYS2, bypassing the TSAF virtual machine on VMSYS2.

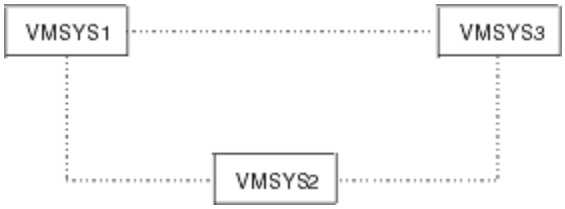


Figure 8. Logically Fully-Connected TSAF Collection

Part 3. Monitoring z/VM Performance

The topics in this section deal with z/VM performance monitoring:

- Chapter 8, “Monitoring System Performance,” on page 75
- Chapter 9, “Monitoring Performance Using CP Monitor,” on page 79
- Chapter 10, “Monitoring SFS Performance,” on page 97
- Chapter 11, “Monitoring CRR Performance,” on page 99.

Chapter 8. Monitoring System Performance

This section describes the concept of performance monitoring and tells you about:

- The types of data that can be monitored
- The commands associated with monitoring
- The saved segment used for monitor data
- The virtual machine used to retrieve monitor data from the monitor saved segment
- Monitor performance considerations.

Performance Monitoring

Performance monitoring is the periodic collection of performance data and serves two major purposes:

1. Early Detection

   An unfavorable trend can be detected early so that corrective action can be taken before it develops into a serious performance problem. Early detection is done by tracking key performance indicators over time and comparing them to established limits of acceptable performance. These indicators are often various measures of response time.

2. Basis for Performance Problem Determination.

   After a problem has been identified, the monitor data serves as the basis for determining the likely cause of the problem. This is done by comparing the current data that reflects the problem to past data collected when performance was adequate.

CP monitor data is the key source of information normally used to monitor the performance of a z/VM system. This is sometimes supplemented with other data. For example, system response times might be sampled by periodically executing benchmark programs and collecting response time information.

Although performance monitoring involves the collection of large amounts of data, only a few key indicators need to be examined regularly. These are for early detection, as explained previously. The remaining monitor data is used only when one or more of those indicators show that there is a performance problem (or a trend towards one).

The INDICATE and MONITOR commands provide a system performance measurement facility for z/VM. These commands provide a method of:

- Obtaining system resource usage data while z/VM is operating so that steps can be taken then or later to improve performance
- Collecting measurement data using a z/VM monitor for later analysis by system analysts.

The INDICATE command can be entered by system resource operators, system programmers, system analysts, and general users (class B, C, E, and G users) to display on the console the use of and contention for system resources. The MONITOR command, which can be entered by class A and E users, starts and stops the recording of one or more classes of events in a user-defined saved segment. Certain events are recorded each time they occur. Others are recorded each time a user-specified interval of time elapses.

In addition, the active wait portion of the dispatcher runs in supervisor key 3. Therefore, on processors with the SAD frame, displaying supervisor key 3 on the system activity display (SAD) frame gives an indication of system idle time.

z/VM also supports the following performance analysis programs:

- Performance Toolkit is a feature of z/VM that is designed to assist operators and system programmers with system console operation in full screen mode, and with performance monitoring on z/VM systems. Performance Toolkit can help system programmers to make more efficient use of system resources,

increase system productivity, and improve end-user satisfaction. For more information, see *z/VM: Performance Toolkit Reference*.

- z/VM Performance Data Pump (Data Pump) converts machine-readable z/VM monitor and SFS data into a generic text-based data stream. Modern tools can use the data stream to display real-time performance dashboards, aggregate real-time data for long-term usage analysis, or integrate with existing enterprise observability solutions.

  Data Pump by itself does not deliver value that a user can readily use. To take advantage of Data Pump the customer must provision, configure, and deploy other services to process the data stream. While instructions for deploying the open source solutions are available, these components are not delivered with the z/VM product. For more information, see z/VM Performance Data Pump (https://www.vm.ibm.com/related/perfkit/datapump/).

  For more information, see Appendix I, "z/VM Performance Data Pump," on page 253.

INDICATE Command

The INDICATE command provides the system analyst with a "snapshot" of system activities. It also provides the general user with a way to determine the execution characteristics of a program with respect to the resources it uses.

For syntax and details about these commands, see *z/VM: CP Commands and Utilities Reference*.

INDICATE USER

General users can enter the INDICATE USER command to obtain the following system resource usage statistics about their own virtual machines. A system analyst can enter the INDICATE USER command to obtain these statistics for any virtual machine. CP displays a set of statistics for each of the virtual machine's virtual processors:

- The user ID and machine mode of the virtual machine (ESA, XA, XC, or Z) and its virtual storage size.
- The device number of the last device or the name of the last NSS IPLed in the virtual machine, and the number of virtual I/O devices in the virtual machine configuration
- The number of pages that were resident in real storage at the time the INDICATE command was entered, the most recent estimate of the virtual machine's working set size, and the current number of pages locked in real, reserved, and instantiated for this user (a copy of a page in multiple places in the paging hierarchy counts as one instantiation).
- The current number of pages allocated on non-preferred and preferred paging volumes for this virtual machine (preferred will always be zero and is kept only for compatibility) and the total number of page-ins and page-outs for the virtual machine since it was logged on
- The processor unit address of the virtual processor to which the response applies; the total connect time, virtual time, and virtual time plus simulation time for the virtual processor; and the total number of non-spooled I/O requests sent since the virtual machine was logged on
- The total number of I/O requests to spooled virtual card readers, printers, and punches since the virtual machine was logged on or accounting was reset for the virtual machine

INDICATE USER EXPANDED

The INDICATE USER command can be issued with the EXPANDED option to get expanded information. As with the normal INDICATE USER command, a general user can issue the expanded version for their own virtual machine and a system analyst can enter the command for any virtual machine. The expanded version differs from the normal INDICATE USER in the following areas:

- The scope of information is broader. For example, storage usage is broken down for private and shared spaces.
- The number of pages resident and locked in host logical storage are also indicated.
- Many of the fields, such as virtual and total time, have more digits and therefore will not wrap as quickly.

- The fields associated with I/O and unit record requests are not reset to zero when an accounting record is written for the given virtual machine.

INDICATE LOAD

The system analyst can enter an INDICATE LOAD command to obtain system information that indicates current contention for system resources. (Values are smoothed except where noted.) For a general user, CP displays a subset of the information. Statistics displayed for the system analyst include:

- The percentage of average processor usage for all processors combined.
- The minidisk caching read and write rates, and the success rate in finding requested data blocks in cache storage.
- The current system paging rate.
- The number of Q0, Q1, Q2, and Q3 virtual machines in the dispatch list.
- The number of E1, E2, and E3 virtual machines in the eligible list.
- The number of virtual machines in the dormant list.
- The expansion factors for Q2 and Q3 virtual machines, which indicate the total delay in response time experienced by these virtual machines because of contention for resources scheduled by the z/VM scheduler.
- The usage percentage of each real processor.

INDICATE QUEUES

The system analyst can enter an INDICATE QUEUES command to obtain the following information about each of the active virtual machines (to determine the virtual machines currently using real storage):

- The transaction class of the virtual machine and whether it is in the dispatch or eligible list
- The current status, which can be one of the following:
  - Running in the real machine
  - In page wait
  - In I/O wait
  - Waiting for the completion of instruction simulation
  - Waiting for the completion of an APPC/VM function
  - In an enabled wait state
  - Idle, but not long enough to be dropped from the dispatch list
  - In a ready state.
- The number of pages belonging to the virtual machine that are currently resident in real storage
- The current estimate of the virtual machine's working set size.
- Additional information about the virtual machine provided by four status indicators
- The virtual machine's priority in the eligible list or the dispatch list
- The processor that the virtual machine is preferred to run on.

INDICATE I/O

The INDICATE I/O command can be entered to determine the current condition of I/O operations in the real machine. This command identifies the virtual machines currently in an I/O wait state and the real I/O device to which the last virtual Start I/O or Start Subchannel operation was mapped.

INDICATE PAGING

The INDICATE PAGING command obtains information about the usage of auxiliary storage. When the ALL operand is specified, CP displays for each logged-on virtual machine the number of pages allocated

on preferred paging devices and nonpreferred paging devices. When the WAIT operand is specified, the same information is displayed for only those virtual machines that are currently in page wait.

INDICATE SPACES

The INDICATE SPACES command obtains information about an address space. General users can obtain information about address spaces they own, while system analysts can obtain information about any address space. For the given space, the number of associated pages in host real storage and DASD are displayed. These page counts are further broken down between private and shared.

INDICATE NSS

The INDICATE NSS command can be used to display information on named saved systems (NSS) and saved segments that are loaded in the system and are in use by at least one user. The INDICATE NSS command can be used for a particular NSS or the ALL option can be used to request information on all NSSs and save segments. For the NSS, the command displays the number of associated pages in host real storage and DASD, the number of instantiated pages, and the rate of page operations between host storage and DASD.

INDICATE MULTITHREAD

The system analyst can enter the INDICATE MULTITHREAD or INDICATE MT command to obtain multithreading status and processor core utilization information if the system is enabled for multithreading. This command obtains core busy, thread density, productivity, utilization, capacity factor, and maximum capacity factor by core type over an interval period of time.

Other Commands

The QUERY AGELIST, QUERY FRAMES, and QUERY SXSPAGES commands provide information about storage usage and status.

QUERY AGELIST

The QUERY AGELIST command displays the status of the global aging list used by the system's frame replenishment algorithm. The written portion of the aging list comprises a reclaimable pool of frames to replenish the system's available frame lists.

For more information on the QUERY AGELIST command, see *z/VM: CP Commands and Utilities Reference*.

QUERY FRAMES

The QUERY FRAMES command displays the status of host real storage. This includes information such as the number of frames that are usable, available for paging, and locked.

QUERY SXSPAGES

The QUERY SXSPAGES command displays the status of pages in the system execution space below 2 GB (the system execution area). This includes information such as the number of pages (backed and unbacked) that are available or in use, the number of free storage reserved pages that are available for use, the number of pages that are locked, and the number of deferred page requests.

Chapter 9. Monitoring Performance Using CP Monitor

The CP Monitor facility collects system performance data. This data can be processed by an external data reduction program to produce statistics to give you an understanding of system operation or help you analyze the use of, and contention for, major system resources. These resources include processors, storage, I/O devices, and the paging subsystem. You can control the amount and nature of the data collected. In general, monitoring is in this order:

1. You use the CP privileged command, MONITOR, to control monitoring, including the type, amount, and nature of data to be collected.
2. An application program running in a CMS virtual machine connects to the CP *MONITOR System Service to establish a data link with CP.
3. The monitor collects performance data during CP operation and stores it, in the form of monitor records, in a saved segment.
4. The application program retrieves monitor records from the saved segment and processes them.

An IBM-supplied program, called MONWRITE, can be used as the application program to establish communication links with CP and retrieve monitor records. MONWRITE not only retrieves monitor records from the saved segment but also stores them on tape or in a CMS file on disk. An application program can then later read the records from the file and perform data reduction on the performance data found in the records.

The performance monitor in the optional Performance Toolkit for z/VM feature collects data from CP control blocks in real storage and also from CP MONITOR records. The Performance Toolkit for z/VM can also display performance data from MONWRITE files on disk or tape. For more information, see *z/VM: Performance Toolkit Reference*.

The z/VM Performance Data Pump (Data Pump) is delivered as an enhancement of the charged Performance Toolkit for z/VM component. Data Pump converts (machine readable) z/VM monitor and SFS data into a generic text-based data stream. Data Pump can also read MONWRITE files. Modern tools can use the data stream to display real-time performance dashboards, aggregate real-time data for long-term usage analysis, or integrate with existing enterprise observability solutions. For more information, see Appendix I, “z/VM Performance Data Pump,” on page 253.

Monitor System Service (*MONITOR)

The monitor system service (*MONITOR) notifies connected virtual machines when records are created by the z/VM monitor.

The monitor collects user-selected sets of statistics about z/VM system operation and stores them in the form of monitor records in a user-defined saved segment. The statistics are grouped into sets called *domains*. These domains, which correspond to areas of system operation for which information of interest is sampled or in which events of interest take place, are:

- System
- Monitor
- Scheduler
- Storage
- User
- Processor
- I/O
- Seek
- Virtual network

- ISFC
- Application data
- Single System Image.

The MONITOR SAMPLE and MONITOR EVENT commands control the collection of system statistics and their storage in monitor records. *MONITOR provides the location of monitor records to a virtual machine. For more information on *MONITOR, see Appendix A, "Monitor System Service (*MONITOR)," on page 171.

Monitor Data

CP Monitor collects data during CP operation and stores ("reports") the data collected in the saved segment in the form of monitor records.

Types of Data Collection

Monitor collects and reports two types of data: event data and sample data.

Event data is collected and reported each time a designated system event occurs. The data reported represents the status of the system at the time the event occurred.

Some events, typically commands that can affect system performance (for example, SET SHARE), occur relatively infrequently but can provide crucial information to a performance analyst. All command related events can be collected separately from those that occur at a much higher rate and typically reflect Control Program functions (for example, add user to Dispatch list), whether or not their respective domains are enabled for event monitoring.

Sample data is collected and reported at the end of each designated time interval. The varieties of sample data are:

Single-sample data, which is collected once during the time interval, and only at the end of that time interval. Some of this data represents the status of the system at the time the data was collected; other data is accumulations of counters, states, or elapsed times collected at the end of each time interval because sampling started.

High-frequency sample data, which is collected at a rate higher than it is reported on. The data is reported along with single-sample data. At each high-frequency sampling time, the data collected is added to its corresponding counters; the data reported is an accumulation of counts or states that had been collected from the time high-frequency sampling started.

With the MONITOR command you can collect either or both types of data. You can also control the time interval for single-sampling and the rate for high-frequency sampling. In addition, you can control the collection of records for events related to CP commands.

As soon as at least one virtual machine has been connected to *MONITOR *and* the MONITOR START command has been entered for the corresponding type of monitoring (in either order), a set of data, called *configuration data*, is immediately collected and reported. The data reported describes the configuration of the system and the monitor profile at the time that monitoring started. There are sample configuration records and event configuration records.

Subsequently, while monitoring is active, every time a new virtual machine connects to *MONITOR, a new set of configuration records is generated to represent the status of the system at the time the connection was made.

Monitor Data Domains

For the purpose of collecting monitor data and generating monitor records, the system is divided into areas called *data domains*:

- The *system domain* contains information on system-wide resource use. This domain contains sample data only.

- The *monitor domain* contains information on your installation's configuration (processors, paging, storage, I/O, and so on) and on the type of monitoring enabled. This domain contains sample and event data.
- The *scheduler domain* contains information on the scheduler queues, the flow of work through the scheduler, and the resource allocation strategies of the scheduler and the dispatcher. This domain contains event data only.
- The *storage domain* contains information on use of real, virtual, and auxiliary storage. This domain contains sample and event data.
- The *user domain* contains information on virtual machines, such as scheduling status, virtual I/O use, and events of logging on and logging off. This domain contains sample and event data.
- The *processor domain* contains data related to work dispatched on a given processor and other data related to processor use. This domain contains sample and event data.
- The *I/O domain* contains information on I/O requests, error recovery, interrupts, and other information for real devices. This domain contains sample and event data.
- The *seek domain* contains information concerning seek operations on DASDs. This domain contains event data only.
- The *virtual network domain* contains information on activity for a virtual network interface card (NIC) connection to a virtual network (guest LAN or virtual switch).
- The *ISFC* domain contains information on the ISFC end points and logical link activity.
- The *application data (APPLDATA) domain* contains application data copied from a virtual machine's storage when this storage has been declared to CP for collecting the data generated by the application program in that virtual machine. The domain contains sample and event data.
- The *Single System Image* domain contains information related to the single system image cluster and shared disk activity.

Note: See *z/VM: CP Programming Services* for information on DIAGNOSE code X'DC' functions through which the virtual machine can declare storage for data collection in the APPLDATA domain.

CP Monitor Commands

The following CP commands are used in the collection and generation of monitor data. For complete information on these commands, see *z/VM: CP Commands and Utilities Reference*. Also provided is a brief summary of the command or utility.

Use the MONITOR command to control monitoring. With the MONITOR command, you can:

- Establish a profile for event data collection. You can select the domains and the elements within them, such as user IDs, device numbers, device types, classes, and volume identifiers.
- Establish a profile for sample data collection. You can select the domains and the elements within them, such as user IDs, device numbers, device types, classes, and volume identifiers.
- Establish the time interval for single-sample data collection and the rate for high-frequency sample data collection.
- Start event and sample data monitoring based on the profiles established.
- Stop event and sample data monitoring.
- Partition the saved segment into sample area and event area. Within each area, you can further partition it into one area for configuration records and the other for data records.
- Establish the maximum time a user has to reply to an IUCV send for configuration data.
- Determine whether events related to CP commands are to be collected.

SET MONDATA Command

Use the SET MONDATA command to establish whether the display device input and output data are to be included in monitor records.

QUERY MONITOR Command

Use the QUERY MONITOR command to view the profiles or settings of CP Monitor. You can view information about event and sample recording, configuration areas, and configuration time limits.

QUERY MONDATA Command

Use the QUERY MONDATA command to determine whether input and output data of the user display devices are to be included in the monitor records.

CP Monitor Utilities

This section briefly describes the monitor utilities. For more complete information on a specific command or utility, see *z/VM: CP Commands and Utilities Reference*.

MONWRITE

Use the MONWRITE utility to retrieve monitor records from the monitor saved segment and store them in a CMS file or on a tape.

MONWSTOP

Use the MONWSTOP utility to stop MONWRITE from writing data.

The Monitor Saved Segment

You must create a saved segment into which CP can place monitor data. The saved segment can be a discontiguous saved segment (DCSS) or a member of a segment space.

Creating the Saved Segment

To create the monitor saved segment:

1. Log on a Class E virtual machine and enter the DEFSEG command to define a saved segment. For example, to define a saved segment as MONDCSS, located in the range of pages from X'40000' to X'45FFF' virtual storage, enter:

   ```
   defseg mondcss 40000-45FFF sc rstd
   ```

2. Enter the SAVESEG command to save the segment. For example, to save the segment defined in the previous DEFSEG command, enter:

   ```
   saveseg mondcss
   ```

3. You can use the SET RESERVED command for the number of pages in your monitor saved segment to make sure the pages remain resident when the system is paging. Reserved settings are not preserved across an IPL; you will need to include the SET RESERVED command for the monitor saved segment in your IPL procedure.

For more information about the DEFSEG, SAVESEG, and SET RESERVED commands, see *z/VM: CP Commands and Utilities Reference*. The following notes pertain to those commands specifically for the monitor saved segment:

- For information about calculating the size of the monitor saved segment, see “Calculating the Space Needed for the Saved Segment” on page 83.
- The name assigned to the saved segment is the name that the application program will specify when it connects to the CP *MONITOR system service.

  You can define and save more than one saved segment, but only one of them can be used by Monitor at a time and that one will be designated by the first application program that connects to *MONITOR.

All other application programs that wish to connect concurrently to *MONITOR must specify that saved segment.

- The location and range of the monitor saved segment should not overlap with existing saved segments or named saved systems (NSSs), especially the CMS-related saved segments and the CMS NSS. To determine where to define the monitor saved segment, enter a QUERY NSS MAP ALL command to display the ranges already in use.
- Although more than one range of page addresses can be specified in the DEFSEG command, only the first one of type SC is used for monitoring. This range must be at least 11 pages.
- SC in the DEFSEG command is a required type code so that CP can write in this segment and the virtual machines can access it.
- RSTD in the DEFSEG command is optional but is recommended to restrict access to the saved segment. Only a virtual machine with a NAMESAVE directory statement specifying this saved segment name can access it.

Calculating the Space Needed for the Saved Segment

To estimate the space that might be required for the monitor saved segment, you need the total of:

- The estimated number of pages that might be required for the sample area of the monitor saved segment
- The estimated number of pages that might be required for the event area of the monitor saved segment.

Saved Segment Space Calculation Notes

- The monitor saved segment must be at least 11 pages (45,056 bytes).
- The total size should satisfy the requirements specified (or defaulted) by the PARTITION and CONFIG options of the MONITOR command. If all options are defaulted, you need at least 8194 pages. See "How Space Is Partitioned in the Saved Segment" on page 87.
- It is recommended that the total number of pages be rounded to the next multiple of a MB.
- The formulas described here are based on scenarios of the worst cases and provide only an estimate of the size of the monitor saved segment that would be needed. The actual size depends on several factors, such as the amount of system activity and what is being monitored.

  If you can anticipate your system or monitor activity, you may adjust the estimates accordingly.
- The formulas described here are based on current size of existing monitor records. The addition of new monitor fields or records could make these estimations inaccurate.

Space Requirements for the Sample Area

The sample area of the monitor saved segment should be large enough to contain the sample data records and the sample configuration records for all domains to be enabled for sample monitoring.

Sample Data Records

To estimate the number of pages needed for the sample data records, add the calculated number of bytes for the following domains that are to be enabled for sample monitoring. Then, divide the total number of bytes by 4096 to get the number of pages and round the result to the next whole page.

Note: At least one page must be reserved for sample data records.

- The system domain:
  - 8000.
  - 1000 times the number of processors.
  - If you are running in a logical partition, add:

  - 80 times the number of logical partitions. Because the partition configuration can change during a session, it is recommended that you include the maximum number of logical partitions allowed on your system.
  - 50 times the number of logical CPUs in the partition. Again, the maximum configuration is recommended. The maximum number of logical CPUs is the maximum number of logical CPUs allowed in a partition times the maximum number of logical partitions allowed on your system.

    Note: The number of physical CPUs on your system is the maximum number of logical CPUs you can have in any logical partition.
  - 120 times the number of channel paths (CHPIDs) defined.

.

- The storage domain:
  - 1400.
  - 130 times the number of named saved systems and saved segments you have defined.
  - 270 times the number of paging and spooling exposures. A CP volume with no exposures counts as 1 exposure.
  - 600 times the number of online processors.
  - 150 times the number of shared address spaces.
  - 80 times the number of virtual disks.
  - 220 times the number of address spaces.
  - 80 times the number of virtual disks.
- The user domain:
  - 2172 times the number of logged-on users to be enabled for sample monitoring in the user domain at the same time.
  - 2172 times the number of virtual CPUs defined by MP users to be enabled for sample monitoring in the user domain at the same time.
- The processor domain:
  - 4000.
  - 1500 times the number of online processors.
  - 900 times the number of PCI crypto cards online.
- The I/O domain:
  - 350 times the maximum number of I/O devices that are to be enabled for sample monitoring in the I/O domain at the same time.
  - 550 times the maximum number of cache devices to be enabled for sample monitoring in the I/O domain at the same time.
  - 750 times the number of virtual switches defined.
  - 330 times the number of SCSI devices online.
  - 360 times the number of Queued Direct I/O (QDIO) devices online.
  - 60 times the number of HyperPAV pools defined.
- The virtual network domain:
  - 350 times the number of virtual network interface cards (NICs) defined.
- The ISFC domain:
  - 120 times the number of ISFC end points defined
  - 300 times the number of ISFC logical links defined.
- The APPLDATA domain:

  - 4096 times the maximum number of buffers to be declared for all the users to be enabled for sample monitoring in the APPLDATA domain at the same time.
- The Single System Image domain:
  - 500.

Sample Configuration Records

To estimate the number of pages needed for sample configuration records, add the number of bytes required for the following items, divide the total by 4096 to get the number of pages, and round the result to the next whole page.

Notes:

1. At least one page must be reserved for sample configuration records.
2. The default number of pages reserved for sample configuration records is 4096. See "Default Sizes for Configuration Records" on page 87. The default can be overridden by the MONITOR SAMPLE CONFIG command.
3. If the size of the Sample Configuration Record area is too small, missing or incomplete monitor records may occur. This usually occurs when a large number of devices are available on your system. To increase the size of the Sample Configuration Record area, use the MONITOR SAMPLE CONFIG command.

- 1000.
- 325 times the number of devices available on your system.
- 60 times the number of paging and spooling areas.
- 50 times the number of online processors.
- 300 times the maximum number of users that may be logged on when configuration records are built.

  Note: Configuration records are first built when (a) at least one virtual machine has connected to *MONITOR or (b) the MONITOR START command has been issued for the corresponding type of monitoring, whichever happens last. New configuration records are built again every time a new virtual machine makes a concurrent connection to *MONITOR.
- The user domain, if enabled:
  - 40.
  - 8 times the maximum number of users to be enabled for sample monitoring in the user domain at the same time.
- APPLDATA domain, if enabled:
  - 40.
  - 8 times the maximum number of users to be enabled for sample monitoring in the APPLDATA domain at the same time.
- I/O domain, if enabled:
  - 40.
  - 2 times the maximum number of devices to be enabled for sample monitoring in the I/O domain at the same time.

Space Requirements for the Event Area

The event area of the monitor saved segment should be large enough to contain the event data records and the event configuration records for all domains to be enabled for event monitoring.

Event Data Records

To estimate the number of pages needed for the event data records, add the calculated number of bytes for the following items.

The speed at which the connected application programs retrieve the event data records from the saved segment can be a factor in the space requirement for the event data records. A page that has been relieved of its records is immediately available for subsequent new event data records. Therefore, the quicker the records are retrieved, the more often the same pages can be reused, and thereby the fewer total pages are required.

The minimum requirement for event data records is 8 pages.

- For each domain to be enabled for event monitoring, add the number of bytes required as follows:

  The scheduler domain:
  12,000 times the maximum number of users to be enabled for event monitoring in the scheduler domain at the same time.

  The storage domain:
  1550.

  The user domain:
  1800 times the maximum number of users to be enabled for event monitoring in the user domain at the same time.

  350 times the number of relocations being monitored.

  The processor domain:
  2000.

  The I/O domain:
  2000.

  The seek domain:
  200 times the maximum number of devices to be enabled for event monitoring in the seek domain at the same time.

  The virtual network domain:
  150.

  The ISFC domain:
  450.

  The APPLDATA domain:
  4096 times the maximum number of buffers to be declared for all the users to be enabled for event monitoring in the APPLDATA domain at the same time.

  The Single System Image domain add:
  100.

- Multiply the total by 20.
- Divide the total by 4096 to get the number of pages and round the result to the next whole page.

Event Configuration Records

To estimate the number of pages needed for event configuration records, add the number of bytes required for the following items, divide the total by 4096 to get the number of pages, and round the result to the next whole page.

At least one page must be reserved for event configuration records.

The default number of pages reserved for event configuration records is 68. See "Default Sizes for Configuration Records" on page 87. The default can be overridden by the MONITOR EVENT CONFIG command.

- 500.
- The scheduler domain:
  - 40.
  - 8 times the maximum number of users to be enabled for event monitoring in the scheduler domain at the same time.
- The user domain:

  - 40.
  - 8 times the maximum number of users to be enabled for event monitoring in the user domain at the same time.
- The I/O domain:
  - 40.
  - 2 times the maximum number of devices to be enabled for event monitoring in the I/O domain at the same time.
- The seek domain:
  - 40.
  - 2 times the maximum number of devices to be enabled for event monitoring in the seek domain at the same time.
- The APPLdata domain:
  - 40.
  - 8 times the maximum number of users to be enabled for event monitoring in the APPLDATA domain at the same time.

Default Sizes for Configuration Records

The number of pages reserved for event configuration records is defaulted at 68; for sample configuration records, the default is 4096.

How Space Is Partitioned in the Saved Segment

When monitoring is started, the monitor saved segment is partitioned according to the monitor profile settings in effect. In general, the saved segment is partitioned into two areas: one for event data, the other for sample data. Within the event area, the space is again divided into two partitions: one for event configuration records, the other for event data records. Likewise, the sample area is partitioned for sample configuration records and sample data records.

The partitioning of the saved segment is set either by default or by the CONFIG or PARTITION options of the MONITOR command.

The default profile for partitioning the monitor saved segment is:

- In the event area: one half of the saved segment
  - Event configuration records: the first 68 pages of the event area
  - Event data records: the rest of the event area (a minimum of 8 pages is required)

  The total default minimum required for the event area: 76 pages
- The sample area: the second half of the saved segment
  - Sample configuration records: the first 4096 pages of the sample area
  - Sample data records: the rest of the sample area (a minimum of one page is required)

  The total default minimum required for the sample area: 4097 pages.

Because the defaulted sample area half requires 4097 pages, the minimum default requirement for the saved segment is thus 8194 pages. This, in turn, results in at least 4029 pages in the event area being available for event data records (4097 pages of event area minus 68 pages reserved for event configuration records).

For better efficiency and use of your saved segment and for efficient performance, you are encouraged to define and partition your saved segment specifically to fit your system requirements or your planned monitor profiles, or both. The defaulted space reserved for configuration records may tend to be a waste of your space (see “Default Sizes for Configuration Records” on page 87). On the other hand, leaving only one page for sample data records may be barely sufficient.

Use the MONITOR START PARTITION command to partition the saved segment between the event area and the sample area.

Note: This command permits you to reserve space only for the event area. You cannot reserve space explicitly for the sample area. Nevertheless, when you have done so for the event area, in effect you have reserved the rest of the saved segment for the sample area.

Use the MONITOR SAMPLE and MONITOR EVENT commands with the CONFIG option to reserve space in the event area for event configuration records and to reserve space in the sample area for sample configuration records.

Use the QUERY MONITOR command to display the monitor profile settings in effect. This command also displays the size and partitioning (in pages) of the saved segment in effect.

See *z/VM: CP Commands and Utilities Reference* for more information on the MONITOR and QUERY MONITOR commands.

What Happens If the Reserved Space Is Not Enough?

The monitor profile for the saved segment, as set or defaulted by the CONFIG or PARTITION options of the MONITOR command, must be compatible with the size of the saved segment provided by the first virtual machine to connect to *MONITOR. Otherwise, the MONITOR START command or the virtual machine's request to make a connection to *MONITOR is rejected.

What happens depends on the sequence of events, according to the following scheme:

- If a virtual machine requests a connection to *MONITOR *before* the MONITOR START command for its corresponding type has been entered:

  Request for connection is rejected if the saved segment does not have at least 11 pages. The 11 pages is the sum of the minimum size for each partition:

  – One for event configuration records
  – Eight for event data records
  – One for sample configuration records
  – One for sample data records
- If MONITOR START is entered *before* any virtual machine has been connected to *MONITOR for the command's corresponding type of monitoring:

  The command is rejected (although CP would not yet have a monitor saved segment to work with) if CP verifies that the requested reserved spaces would exceed a saved segment of the largest possible size. See the DEFSEG command in *z/VM: CP Commands and Utilities Reference* for the maximum number of pages that can be defined for a saved segment of the type SC.
  Rationale for rejecting the command at this point: a saved segment must be successfully defined (that is, within its maximum size) before a virtual machine can use it to connect to *MONITOR. Thus, no matter what its size is, the saved segment will still be too small for the requested reserved spaces, and the connection request would be rejected anyway.
  You should check that the requested partitioning or the sizes of the requested reserved spaces do not total beyond the largest saved segment possible.
- If the first virtual machine requests a connection to *MONITOR *after* the MONITOR START command for its corresponding type has been entered:

  The request for connection is rejected if the saved segment is not large enough for the requested reserved spaces.
  The virtual machine can resolve this problem by providing a larger saved segment. Sometimes, however, this problem can be resolved by changing the requested reserved spaces. In this case you may have to enter MONITOR STOP to stop monitoring, to change the sizes or partitioning.
- If MONITOR START is entered *after* at least one virtual machine has been connected to *MONITOR for the command's corresponding type of monitoring:

The command is rejected if the requested reserved spaces are too large for the saved segment now designated for monitor.

The Virtual Machine to Collect Data Records

To use the CP Monitor facility, a virtual machine must run an application program to retrieve data from the monitor saved segment. The application program must load the saved segment and connect to the *MONITOR system service. IUCV is the data link between the virtual machine and the *MONITOR system service.

*MONITOR Overview

1. The virtual machine makes a connection (IUCV CONNECT) to the *MONITOR system service.

   The first virtual machine to make the connection has the first choice to specify the saved segment. It also has the first choice to select the *mode* of connection, which can be one of the following:

   - *Exclusive Mode.* This virtual machine is the one, and the only one, connected to *MONITOR. Requests for connections from any other virtual machines are rejected.
   - *Shared Mode.* Any other virtual machine may connect to *MONITOR concurrently. Up to 65,535 virtual machines may connect to *MONITOR. (65,535 is the architectural limit. The practical limit depends on available resources.)

     In shared mode, the virtual machine also selects the type of data it wants to process: sample data or event data or both.
2. When records have been placed in the saved segment, *MONITOR notifies (through IUCV SEND commands) all applicable virtual machines that the records are ready to be retrieved. Location and types of data are supplied by *MONITOR.

   In the shared mode, the virtual machine is notified in this manner only for the type of data it has selected when it made the connection to *MONITOR. In exclusive mode, the virtual machine is always notified for either sample data or event data.
3. The virtual machine accesses the saved segment and retrieves the records.
4. The virtual machine notifies *MONITOR (through IUCV REPLY commands) that it is finished with the applicable area of the saved segment. When all applicable virtual machines have done so, this area of the saved segment is available to CP for further recording.

Sample Directory for the Virtual Machine

For a virtual machine to connect to the *MONITOR system service, it must have IUCV entries in its directory entry. The directory entry listed in the following is a sample, but be sure to tailor the directory statements to match your installation's configuration. It is a best practice, not a requirement, that the directory entry be a multiconfiguration virtual machine.

```
IDENTITY MONWRITE pw 4M 8M G
 BUILD ON * USING SUBCONFIG MONWRT-1
 MACHINE ESA
 ACCOUNT XXXXX
 IPL CMS
 OPTION QUICKDSP
 SHARE ABSOLUTE 3%
 IUCV *MONITOR MSGLIMIT 255
 NAMESAVE MONDCSS
 CONSOLE 009 3270 T
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A

SUBCONFIG MONWRT-1
 LINK MAINT 0190 0190 RR * CMS system disk
 LINK MAINT 019D 019D RR * help disk
 LINK MAINT 019E 019E RR * Product code disk
 MDISK 191 3390 2619 300 MO1RES MR readpw     writepw    multipw
```

- When writing to DASD, you might need more DASD space, depending on how much data you collect and how frequently you collect it.
- In general, make sure you define the monitor saved segment at an address higher than the maximum virtual storage size of the virtual machine. This prevents the monitor saved segment from overlaying the segment where CMS loads the application program.
- If you want to enter monitor commands from this virtual machine, you must add class E to the IDENTITY directory statement. You also need class E if you plan to define and save the monitor saved segment from this virtual machine.
- The MSGLIMIT coded on the IUCV directory statement establishes the limit on the number of outstanding messages allowed on the path from *MONITOR to the virtual machine. (The virtual machine cannot send messages to *MONITOR, because this message path is always quiesced.)
- The NAMESAVE statement is necessary if the monitor saved segment has been defined with the RSTD option. (The RSTD option is recommended when you define a saved segment for monitor, because some data may be sensitive.)
- If your installation has issued SET MONDATA ON, remember that the virtual machine has access to the possibly sensitive data produced in the event scheduler domain.
- If the virtual machine does not have an adequate priority, some data may be lost. If this situation happens, try increasing the scheduler share allotted to this virtual machine with the SHARE directory statement or the CP SET SHARE command. For more information on increasing scheduler shares, see “SET SHARE Command” on page 107.

The MONWRITE Program

The MONWRITE program is an IBM-supplied program that can serve as the application program required for retrieving monitor data from the saved segment.

MONWRITE does the following:

- Loads the monitor saved segment into its virtual storage.
- Establishes the appropriate IUCV communication links with the *MONITOR system service. It connects to *MONITOR in *shared* mode. That is, more than one virtual machine running MONWRITE can connect to *MONITOR at one time.
- Accesses the saved segment and retrieves its data.
- Writes the monitor data on a tape or in a CMS file.

An application program can later read the records from the file and perform data reduction on the performance data found in the records.

Using MONWRITE

Use the MONWRITE command to call the MONWRITE program. Through this command, you select the file (disk or tape) to which the monitor data is to be stored.

See *z/VM: CP Commands and Utilities Reference* for information on the MONWRITE and MONWSTOP utilities.

Example 1: To send monitor data to a tape at address 181, enter:

```
monwrite mondcss *monitor tape 181
```

Example 2: To send monitor data from the monitor saved segment to a disk file with a file ID that contains the current date and time, enter:

```
monwrite mondcss *monitor disk
```

Example 3: To send monitor data to the tapes at addresses 181 and 185, enter:

```
monwrite mondcss *monitor tape 181 185
```

Monitor Operations

To use CP Monitor, you must:

1. Create a monitor saved segment.
2. Create a virtual machine with an application program that would:
   a. Load the monitor saved segment into its virtual storage
   b. Connect to the *MONITOR system service
   c. Receive notifications from *MONITOR that data is available
   d. Retrieve data
   e. Notify *MONITOR that it is finished with the data

   Note: The MONWRITE program can be used as the application program in the virtual machine.
3. Establish a monitor profile (unless everything is to be defaulted).

   For more information, see *z/VM: CP Commands and Utilities Reference*.
4. Enter the MONITOR START command.

An Example of Monitoring for Sample Data

Assume an 800-page saved segment has been created and loaded for monitor.

The following is an example of a series of MONITOR commands for sample data monitoring:

```
monitor sample enable processor
monitor sample enable storage
monitor sample enable user userid jordan perkins black worthy dorherty
monitor sample enable i/o device 0120 5140 150 140-145
monitor sample enable i/o type 3380
monitor sample enable i/o class tape
monitor sample enable i/o volume pack1 pack2
monitor sample rate 1 second
monitor sample interval 2 minutes
monitor sample config size 40
monitor sample config limit 3
monitor sample start
```

In the previous example:

- The processor, storage, user, and I/O domains are enabled. The system and monitor domains are always enabled.
- For the user domain, five users have been selected for monitoring.
- For the I/O domain, the following real devices have been selected for monitoring:
  - Devices with the addresses 120, 5140, 150, and the address range from 140 to 145 inclusive
  - All 3380 devices
  - All tape devices
  - DASDs with volume IDs PACK1 and PACK2.
- High-frequency sample data is sampled every second. It is reported, along with the single-sample data, every 2 minutes.
- Single-sample data is reported every 2 minutes.
- Forty 4 KB pages of the sample area of the saved segment are reserved for the sample configuration records.
- Any virtual machine connected to *MONITOR for sample data has 3 minutes to reply to notices of sample configuration data.
- The MONITOR SAMPLE START command starts the collection of data if at least one virtual machine is connected for sample data or as soon as the first virtual machine is connected for sample data.

An Example of Monitoring for Event Data

Assume that monitoring in the previous example continues, using the same 800-page monitor saved segment. Also, event monitoring is to be initiated using the following set of commands.

```
monitor event enable processor
monitor event enable scheduler userid puckett gaetti hrbek viola reardon
monitor event enable user userid jordan perkins black worthy dorherty
monitor event enable i/o device 0120 5140 150 140-145
monitor event enable i/o type 3380
monitor event enable i/o class tape
monitor event enable i/o volume pack1 pack2
monitor event enable seeks volume pack1 pack2
monitor event enable seeks device 0120
monitor event config size 36
monitor event config limit 3
monitor event start block 4 partition 100
```

In the previous example:

- The processor, scheduler, user, I/O, and seek domains are enabled. The monitor domain is always enabled.
- For the scheduler domain, five users have been selected for monitoring.
- For the user domain, five users have been selected for monitoring.
- For the I/O domain, the following real devices have been selected for monitoring:
  - Devices with the addresses 120, 5140, 150, and the address range from 140 to 145 inclusive.
  - All 3380 devices
  - All tape devices
  - DASDs with volume IDs PACK1 and PACK2
- For the seek domain, 3 DASDs: volume PACK1, volume PACK2, and address 0120, are selected for monitoring.
- Thirty-six 4 KB pages of the event area of the saved segment are reserved for the event configuration records.
- Any virtual machine connected to *MONITOR for event data has 3 minutes to reply to notices of event configuration data.
- The MONITOR EVENT START command in this example causes:
  - 100 pages of the saved segment to be partitioned as the event area. As a result of the MONITOR CONFIG SIZE command, the first 36 pages are reserved for the event configuration records.

    The remaining 700 pages of the saved segment constitute the sample area, with 40 pages reserved for sample configuration records.
  - A 4-page block of event records to be accumulated before the virtual machines connected to *MONITOR are notified of the event records.
  - Event data to be collected and reported if at least one virtual machine is connected for event data or as soon as the first virtual machine is connected for event data.
  - The MONITOR SAMPLE START command starts the collection of data

Stopping the Collection of Monitor Records

To stop the collection of both sample and event data, enter:

```
monitor stop
```

To stop the collection of only sample data, enter:

```
monitor sample stop
```

To stop the collection of only event data, enter:

```
monitor event stop
```

To stop the monitor writer from the console of the virtual machine that is running the MONWRITE program, enter:

```
monwstop
```

Note: When you use the MONWRITE program, the MONWSTOP command should always be entered before collection is stopped by a MONITOR command so that the monitor writer can complete the writing of the records in the saved segment. (If the MONITOR command is entered to stop recording while the MONWRITE program is still storing records, null data is written.)

System Shutdown and Hard Abends

Because monitor data is accumulated in the saved segment and requires an application program to collect the data in real time, the loss of data because of a system warm start (IPL) or system hard abend should be minimal. For example, the last interval of data sent may be lost, and for event data, any blocks of data not yet replied to may be lost.

If MONWRITE is being used, the file may be left in an unpredictable state. That is, the last interval of sample data or last block of event data may be incomplete if the shutdown or abend occurred while the MONWRITE program was writing data. In addition, there will be no end-of-data records, and if writing to tape, there will be no tape mark. (All data written before the shutdown/abend will be intact.) For usual system termination, the MONWSTOP utility should be entered before you shut down the system.

Enabling MONITOR and MONWRITE: An Example

The following sample EXEC illustrates how to enable and start the MONITOR and the sample application (MONWRITE) that is used to collect sample and event data. This EXEC, named MONSETUP EXEC, starts the MONITOR, and then stops the MONITOR after MONWRITE finishes. You need to verify that the virtual machine you are using has the required special directory statements:

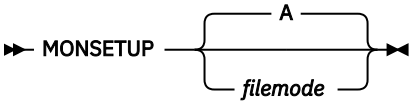
```
IUCV *MONITOR MSGLIMIT 255 NAMESAVE MONDCSS
```

The virtual machine also needs the following special privileges:

Command	Required Class Userid
CP QUERY NSS	E
CP DEFSEG	E
CP SAVESEG	E
CP MONITOR	A or E

The syntax for this EXEC follows:

Syntax for the MONSETUP EXEC




```
/*********************************************************************/

Address Command              /* Restrict command resolution to basic  */
                             /* CMSCALL.                              */

Parse Upper Arg filemode .   /* Retrieve parameters from the command  */
                             /* line.                                 */

If filemode='' Then          /* If no mode specified                  */
  filemode = 'A'             /* default to A.                         */
```

```
'QUERY DISK 'filemode' ( STACK'
Pull header
Pull . . . state .
If state¬='R/W' Then
  Do
    Say 'Filemode('filemode') is not accessed R/W'
    Exit 1
  End

'ESTATE MONWRITE OUTPUT 'filemode
If rc=0 Then
  Do
    Say 'MONWRITE OUTPUT 'filemode' already',
        'exists please remove/rename then retry.'
    Exit 2
  End
/*********************************************************************
 See if we have a DCSS named MONDCSS defined in the system data files
 If not then define and save one
*********************************************************************/

Parse Value Diagrc(8,'QUERY NSS NAME MONDCSS MAP') With rc cc data
If rc¬=0 | cc¬=0 Then
  Do
    Say '"QUERY NSS NAME MONDCSS MAP" returned rc='rc' cc='cc' and'
    Say ' message='Strip(data)
    Exit 3
  End

Parse Var data . ' MONDCSS ' . . begpag endpag type cl .
If cl='S' Then
  Do
    Say '"QUERY NSS NAME MONDCSS MAP" indicates CLASS=S.  Please '
    Say 'use CP SAVESEG to save the DCSS.  Then retry this exec.'
    Exit 4
  End

If Substr(Strip(data),1,15)='FILES:   NO NSS' Then
  Do
    'CP DEFSEG',
      'MONDCSS',             /* Give the segment a name.              */
      '40000-45FFF',         /* Define the page ranges.               */
      'SC',                  /* CP writable pages, shared read-only   */
                             /* access by virtual machine, no data    */,
                             /* saved.                                */,
      'RSTD'                 /* Forces virtual machine to have a      */
                             /* NAMESAVE directory statement when     */
                             /* trying to access this segment.        */
    If rc¬=0 Then
      Do
        Say '"CP DEFSEG MONDCSS 40000-45FFF SC RSTD" returned rc='rc
        Exit 5
      End

    'CP SAVESEG MONDCSS'     /* Now save what was just defined.       */
    If rc¬=0 Then
      Do
        Say '"CP SAVESEG MONDCSS" returned rc='rc
        Exit 6
      End
  End
'CP MONITOR EVENT ENABLE ALL'   /* Enable CP Monitor events          */
If rc¬=0 Then
  Do
    Say '"CP MONITOR EVENT ENABLE ALL" returned rc='rc
    Exit 7
  End

'CP MONITOR SAMPLE ENABLE ALL'  /* Enable CP Monitor sampling        */
If rc¬=0 Then
  Do
    Say '"CP MONITOR SAMPLE ENABLE ALL" returned rc='rc
    Exit 8
  End

'CP MONITOR START'              /* Start CP Monitor now, the rc may   */
                               /* not be equal to zero at this time. */
                               /* Since there are probably no virtual*/
                               /* machines connected to *MONITOR.    */

/*********************************************************************
 Start the sample application MONWRITE to connect to *MONITOR
```

```
 and start the collecting of CP Monitor records to DASD.
*********************************************************************/

Say ' '
Say 'When you are ready to STOP the sample application MONWRITE'
Say 'ENTER the immediate command MONWSTOP.'
Say ' '

'MONWRITE MONDCSS *MONITOR DISK MONWRITE OUTPUT 'filemode
If rc¬=0 Then
  Do
    Say '"MONWRITE MONDCSS *MONITOR DISK MONWRITE OUTPUT 'filemode'’,
        'returned rc='rc
    Exit 9
  End

'CP MONITOR STOP'               /* Stop CP Monitor since we started it*/
If rc¬=0 Then
  Do
    Say '"CP MONITOR STOP" returned rc='rc
    Exit 10
  End

Exit 0
```

Performance Considerations (Monitor)

The performance impact of monitoring depends on what data is being collected, the number of applications connected to monitor, and the data type each application is processing. Each time a set of monitor records is created in the monitor saved segment, a broadcast indicating the location of the newly created records is sent to all applications processing that type of monitor data.

The more domains and their elements (such as users or devices) you enable, the more records monitor collects. If you enable a large number of domains or their elements, system performance may suffer. For this reason, during usual system operation, you should avoid:

- Large amounts of event data collection, especially in the seek and scheduler domains
- Very short time intervals for sample data collections.

Chapter 10. Monitoring SFS Performance

The overall monitoring process for SFS file pool servers does not change. However, you do need to enable the APPLDATA domain with the CP MONITOR EVENT ENABLE command so that SFS file pool server performance data is included with the usual CP monitor data. See the "CP Monitor Commands" on page 81 for information on the CP MONITOR command.

When Performance Toolkit for z/VM reduces the monitor data, the SFS performance data can be integrated into the output reports. For more information about the CP monitor and performance monitoring, see *z/VM: CP Planning and Administration*. For more information about Performance Toolkit for z/VM, see *z/VM: Performance Toolkit Reference*.

The z/VM Performance Data Pump (Data Pump) is delivered as an enhancement of the charged Performance Toolkit for z/VM component. z/VM Performance Data Pump (Data Pump) converts machine-readable z/VM monitor and SFS data into a generic text-based data stream. Modern tools can use the data stream to display real-time performance dashboards, aggregate real-time data for long-term usage analysis, or integrate with existing enterprise observability solutions. For more information, see Appendix I, "z/VM Performance Data Pump," on page 253.

The data that each active server contributes to the CP monitor facility is equivalent to the counts and timings provided by the QUERY FILEPOOL REPORT command. A description of the QUERY FILEPOOL REPORT command is in *z/VM: CMS File Pool Planning, Administration, and Operation*. A description of the records that servers contribute to the monitor data is in Appendix D, "SFS and CRR Server Monitor Records," on page 193.

The performance monitoring data provided by SFS is used when monitoring indicators show a performance problem in this area. Its use in support of performance problem determination is discussed in Chapter 13, "SFS Tuning," on page 129.

Chapter 11. Monitoring CRR Performance

The CRR recovery server also does not change the overall monitoring process. You do need to enable the APPLDATA domain with the CP MONITOR EVENT ENABLE command so that CRR recovery server performance data is included with the usual CP monitor data. See "CP Monitor Commands" on page 81 for information on the CP MONITOR command.

When Performance Toolkit for z/VM reduces the monitor data, the CRR performance data can be integrated into the output reports. For more information about the CP monitor and performance monitoring, see *z/VM: CP Planning and Administration*. For more information about Performance Toolkit for z/VM, see *z/VM: Performance Toolkit Reference*.

The data that the CRR recovery server contributes to the CP monitor facility is equivalent to the counts and timings provided by the QUERY FILEPOOL REPORT command. A description of the QUERY FILEPOOL REPORT command is in *z/VM: CMS File Pool Planning, Administration, and Operation*. A description of the records that servers contribute to the monitor data is in Appendix D, "SFS and CRR Server Monitor Records," on page 193.

The performance monitoring data provided by CRR is used when monitoring indicators show a performance problem in this area. Its use in support of performance problem determination is discussed in Chapter 14, "CRR Tuning," on page 143.

Part 4. Tuning z/VM Performance

The topics in this section discuss how to tune your system for greater performance:

Chapter 12. Tuning Your System

This section describes tuning:

- Why it can be beneficial
- Terms associated with tuning
- How to allocate system resources (processors, storage, and paging and I/O capabilities) to virtual machines
- Change the way the following resources are distributed:
  - Processors and processor time
  - Paging resources
  - Real storage
  - I/O resources.

It also gives examples of performance problems, solutions to these problems, and examples of using the tuning commands described in the previous sections.

Tuning Overview

Why tune a system? In general, performance tuning is undertaken when you want to:

- Process a larger or more demanding work load without increasing the system's configuration
- Improve system response or throughput
- Reduce processing costs without affecting service to your users.

In other words, performance tuning is done when an installation wishes to improve its cost-performance ratio.

What Is the Value of Performance Tuning?

Converting performance from technical to economic terms is difficult. Even so, it is apparent that a tuning project costs money (through hours and processor time). Before you undertake a tuning project, weigh that project's cost against its possible benefits. Some of these benefits are tangible. More efficient use of resources and the ability to add more users to the system are examples. Other benefits, such as greater user satisfaction because of quicker response time, are intangible. All of these benefits must be considered.

Here is a list of guidelines that can make a tuning project useful:

- Remember the law of diminishing returns. Your greatest performance benefits usually come from your initial efforts. Further changes generally produce smaller and smaller benefits and require more and more effort.
- Eliminate the easily removed bottlenecks first. You can take a number of tuning actions that are relatively easy to do. These actions do not require additional resources and take very little time. You should identify and complete these items first.
- Make only one unrelated change at a time. Some tuning changes are related and should be made together (for example, using SET QUICKDSP with SET RESERVED, which is discussed later in this section). However, for unrelated changes, it is best to do them one at a time. If you make many unrelated changes at once, it is difficult or impossible for you to determine the effect of each change on the system. In fact, if some changes improve performance and others degrade performance, they can effectively cancel each other out and lead to erroneous conclusions.
- If you have time limitations, formulate a hypothesis and measure only what is required to verify the hypothesis.

How Much Can a System Be Tuned?

There are limits to how much you can improve the efficiency of a system. Consider how much time and money you should spend on improving system performance, and how much the spending of additional time and money will help the users of the system.

You may be able to run your system without specifying any tuning commands. It is a good idea to bring up your system with the system defaults and then determine whether tuning is necessary.

In some circumstances, there is no benefit to tuning a system. Running with the system defaults may give just as good performance as specifying many tuning commands. In other circumstances, the benefit from tuning may be significant. For example, increasing the number of users by 10 percent while maintaining the same response time is certainly a worthwhile tuning effort.

As your system approaches a performance bottleneck, it is more likely that tuning will be effective. If you are close to hitting a bottleneck and you increase the number of users on the system by 10 percent, the response time is likely to rise by much more than 10 percent. But there is a point beyond which tuning cannot help you. At that point, the only thing to do (other than adding new hardware) is to change your objectives. For example, you may have to favor interactive work over long-running work.

A Step-by-Step Approach to Tuning

Tuning is basically trial and error. However, it is helpful to have a step-by-step approach to any tuning project, as given here:

1. Decide what change to make to your system.
2. Make the change. Observe the system before and after the change. Try to have about the same work load on the system before and after the change. In this way, any change in performance can be attributed to your adjustment. It may be necessary to get several samplings by turning the change on and off a few times. Many short readings are more accurate than one or two long ones.
3. Use monitor data and the INDICATE command to evaluate system performance. Consider setting up a special user ID to run test loads of previously devised transactions. Have this user ID record the response time of each completed transaction.
4. Evaluate your results. Determine whether system performance improved, and why (or why not). If performance does not improve, cancel the change or try the change again with some other tuning control.
5. Go back to step 1 and try again.

Virtual Machine Resource Manager

Dynamic tuning is provided by the Virtual Machine Resource Manager (VMRM). VMRM allows for the dynamic tuning of groups of virtual machines. See Chapter 17, “VMRM Tuning Parameters,” on page 157 for details.

High Frequency User State Sampling

A component of CP Monitor called the *high frequency state sampler* routinely samples each virtual processor to determine whether it is active, and, if it is not active, why it is waiting. CP Monitor collects these samples several times each minute for each virtual processor. At the regular monitor sample interval, CP Monitor generates a record for each virtual processor, describing what this state sampling observed. Performance Toolkit for z/VMprocesses these records and summarizes them by virtual machine. The observed distribution of reported states can help guide system tuning decisions. Some of the key states recorded are the following:

- **Running**. A virtual processor that is in the running state is actually using a real processor.
- **CPU wait**. A virtual processor is in this state when the virtual processor is found waiting to run on a real processor. A high percentage of samples falling into this state indicates a bottleneck in processor resources.

- **I/O wait**. This state indicates that the virtual processor is waiting for completion of an I/O and thus is prevented from running. A high percentage of the samples falling into this state indicates that you should check the effectiveness of minidisk cache, virtual disk in storage, and the I/O hardware configuration.
- **Simulation wait**. A virtual processor is prevented from running and enters simulation wait state when CP is simulating a hardware function such as instructions, interrupts, timer updates, or unique z/VM functions such as IUCV. A high percentage of the samples falling into this state could indicate performance problems in connectivity functions, loss of hardware assists, or other simulation-related bottlenecks.
- **Page wait**. A virtual processor enters page wait state when the virtual processor references a page that is not present in either central or extended storage, and therefore must be brought in from DASD. A high percentage of samples falling into this state indicates the storage and paging subsystem need evaluation. If important virtual machines are routinely experiencing the page wait state, you might want to consider reserving pages for these machines.
- **Console function wait**. Certain functions in CP are serialized by a state called *console function mode*. While a virtual machine is in this state, none of its virtual processors are permitted to run. A high percentage of samples falling into this state could indicate problems in the network, excessive CP command usage, or master processor contention.
- **Test idle**. This is the state in which CP places a virtual processor that has just become idle or is waiting for what is expected to be a short term wait. While in the test idle state, the virtual processor is still in the dispatch list. This state can last up to approximately 300 milliseconds. If no new work arrives for the virtual processor during the test idle period, the virtual processor is dropped from the dispatch list and added to the dormant list. This technique helps avoid much of the overhead of dropping and adding virtual processors to the dispatch list. There are actually two types of test idle states: one where the user has an outstanding communication with another virtual machine (SVM wait) and one where there is not an SVM wait. A large percentage of the samples falling into the test idle state is not necessarily a problem.
- **Dormant**. The user is in the dormant list.
- **I/O active**. A virtual processor is in the I/O active state if the virtual processor is running or runnable and there is also an I/O in progress for the virtual machine. For some virtual machines, it is normal for the state sampler to find a high percentage of samples in I/O active state. For example, guests that use continuously executing channel programs, such as network-related machines, can often be found in this state. If the percentage of samples found in this state increases unexpectedly, it might indicate performance problems in networks or in I/O devices driven by asynchronous techniques.
- **Active page wait**. This is similar to active I/O wait, except instead of an outstanding asynchronous I/O, there is an outstanding page fault that the virtual machine was prepared to handle asynchronously.
- **Limit list wait**. If a user is ready to run, but has exceeded the maximum share setting, the user may be placed on the limit list for a period of time. During that time, the user is in the limit list wait state.

What Is the Scheduler?

To know what adjustments are most likely to improve the performance of your system, it is important that you understand the z/VM scheduler. This section provides you with a high level overview. For a more complete discussion of the scheduler's operation, see Chapter 2, “Characteristics of a z/VM System,” on page 5.

The scheduler is a collection of algorithms that manage the scheduling of virtual machines for real processor time. It controls three lists: the dispatch list, the eligible list, and the dormant list.

Terminology

The scheduler divides transactions (units of work) into short, medium, and long-running classes. These classes are called Q1, Q2, and Q3, respectively, when a user is in the dispatch list. The classes are referred to as E1, E2, and E3 when a user is in the eligible list.

The *dispatch list* contains the virtual machines currently contending for processor time. The closer a virtual machine is to the top of the dispatch list, the more likely it is that the virtual machine will receive processor time.

The *eligible list* contains virtual machines waiting to move into the dispatch list. The eligible list is one physical list separated into four logical lists (or classes), called E0, E1, E2, and E3. The E0 list contains virtual machines about to enter the dispatch list without further waiting. (For more information on the E0 class, see "SET QUICKDSP Command" on page 112.) The E1 list contains virtual machines expected to run short transactions. That is, these virtual machines are expected to require small amounts of service. The E2 list contains virtual machines expected to run medium-length transactions. The E3 list contains virtual machines expected to run long transactions.

A virtual machine is first placed in the dormant list. When that virtual machine has work to do, it is moved into the eligible list. There CP prioritizes the virtual machine for entry into the dispatch list. When in the dispatch list, a virtual machine is qualified to receive real processor time.

Each transaction enters in the Q1 class and progresses through Q2 and Q3 if its processing so requires. Transactions complete in Q3 if they haven't completed earlier.

Controlling the Dispatch List

Controlling the number of users in the dispatch list is one of the most important aspects of tuning a system. The key question is how many users, and which ones, should you allow in the dispatch list at any given time. The number of users you can have in the dispatch list while still getting the maximum performance out of your system is based on the resources (number of processors, number of I/O devices, and so forth) of your system. If you have many resources, you need a large number of users in the dispatch list to be sure that all resources are being fully used. When there are fewer resources, having too many users in the dispatch list is often harmful to system performance. For example, if the working sets of the users in the dispatch list do not fit into the available real storage, thrashing occurs.

The SET commands which control the users in the dispatch list are shown in Table 3 on page 106.

Table 3. CP Commands for Controlling the Dispatch List

CP Command	See
SET SRM IABIAS	"Tuning the Processor Subsystem" on page 112 and "Tuning the I/O Subsystem" on page 121
SET SRM DSPBUF	"Tuning the Processor Subsystem" on page 112 and "Tuning the I/O Subsystem" on page 121
SET SRM STORBUF	"Tuning the Storage Subsystem" on page 118
SET SRM LDUBUF	"Tuning the Paging Subsystem" on page 116
SET SRM DSPSLICE	"Tuning the I/O Subsystem" on page 121
SET QUICKDSP	"Allocating System Resources" on page 106

Allocating System Resources

Allocating system resources is a preliminary step to tuning your system. It is important for you to know, for example, how to give user A twice as much use of a processor as user B.

This section describes CP commands (and directory statements) that you can use to allocate system resources.

Allocating Processors

By default, CP assigns each virtual machine the same share of processor time. CP then allocates processor time to a virtual machine based on its SHARE setting. For more information on how to affect SHARE settings, see "SET SHARE Command" on page 107.

Note that you do not have to change the way CP allocates processor time. If you feel CP's default settings might work for your installation, leave them the way they are. If problems arise (for example, certain virtual machines are not getting the processor time you think they should), use the controls described in the following sections to tune CP's default settings.

SET SHARE Command

The SET SHARE command and the SHARE directory statement allow you to control the percentage of processor time a user receives. With z/VM, a virtual machine receives its proportion of processor time according to its SHARE setting. Both the normal share and the maximum share can be set.

- Normal share—The normal share is also known as the target minimum share. CP attempts to provide at least this amount of processor time to a virtual machine if the virtual machine can use the processor.
- Maximum share—CP attempts to limit a virtual machine from using more than this amount of system processor resources. The limit types that can be specified for the maximum share are:
  - NOLIMIT. A user is not limited. This is the default.
  - LIMITHARD. A user does not get more than the maximum share.
  - LIMITSOFT. A user does not get more than the maximum share, unless no other user can use the available processors.

The operands available with SET SHARE for both normal and maximum share are:

- ABSOLUTE—An ABSOLUTE share allocates to a virtual machine an absolute percentage of all available system processors. For example, if you assign a virtual machine an absolute share of 50%, CP allocates to that virtual machine approximately 50% of all available processors (regardless of the number of other virtual machines running).
- RELATIVE—A RELATIVE share allocates to a virtual machine a portion of the total system processors minus those processors allocated to virtual machines with an ABSOLUTE share. Also, a virtual machine with a RELATIVE share receives access to system processors that is proportional with respect to other virtual machines with RELATIVE shares. For example, if a virtual machine (VM1) has a RELATIVE share of 100, and a second virtual machine (VM2) has a RELATIVE share of 200, VM2 receives twice as much access to system processors as VM1.

  If a virtual machine serves multiple users, its relative share should be set to:

  n × a normal single virtual machine share

  where *n* is the number of users normally expected to be contending concurrently for resources in that virtual machine.
- INITIAL—An INITIAL share means that a virtual machine's access to system processors is reset to that specified in the virtual machine's directory entry.

Setting SHARES

There are three ways to set a virtual machine's SHARE:

- Accept the default value CP assigns
- Use the SHARE directory statement to specify a value in the virtual machine's directory entry
- Use the SET SHARE command to set a value.

This section tells you how to set shares using the SET SHARE command.

If the SHARE directory statement has not been specified, CP initially assigns each virtual machine a relative share of 100 for the normal share and nolimit for the maximum share. To reset a virtual machine's share, use the SET SHARE command. A share remains valid until (a) you reenter the SET SHARE command or (b) the virtual machine is logged off.

Example 1

To assign a normal relative share of 300 to the virtual machine of user USER1, enter:

```
set share user1 relative 300
```

Example 2

To assign a normal absolute share of 60% to the virtual machine of user USER2, enter:

```
set share user2 absolute 60%
```

Example 3

To reset the share of user USER3 to the value specified in USER3's directory entry, enter:

```
set share user3 initial
```

Usage Notes

1. In general, you should assign typical users a relative share rather than an absolute share. Absolute share is more appropriate for guest operating systems. Assigning a relative share to a virtual machine lets you determine the percentage of processor time a virtual machine gets with respect to other virtual machines with relative shares. Assigning an absolute share lets you guarantee the availability of a specific percentage of processor time to a virtual machine.
2. The scheduler reserves at least 1% of available processor time for virtual machines with relative shares; up to 99% is allowed for absolute virtual machines with absolute shares.

   If you assign absolute shares to virtual machines such that the total ABSOLUTE shares exceed 99% of available processor time, the scheduler computes a normalized share value for each virtual machine. The sum of all normalized shares for logged-on virtual machines is 100%. For an example of computing normalized shares, see “Calculating a User's Share of Resources” on page 108.
3. Your installation might need to run background virtual machines. A background virtual machine is one that receives a significant amount of processor time only when more important users are not running. To set up a virtual machine to run as a background virtual machine, assign it a very small share value (for example, SHARE RELATIVE 1). Assign more important virtual machines a high value (for example, SHARE RELATIVE 100). With these settings, the background virtual machines do not interfere with more important virtual machines. However, when the important virtual machines are not running, the background virtual machines get full use of processor time.
4. The SET SHARE command only directly affects paging I/O. If your system has a bottleneck with other I/O, SET SRM DSPBUF (rather than SET SHARE), may possibly ease the problem.
5. For a virtual MP guest, CP distributes the guest's share setting equally over the guest's started virtual CPUs. Stopped virtual CPUs get no apportionment of the guest's share. When the guest operating system starts or stops its virtual CPUs, the CP correspondingly redistributes the guest's share so that the guest's started virtual CPUs, whatever their number, have equal access to the virtual machine's whole CPU entitlement.

Displaying a User's Share Setting

To display a virtual machine's current share setting, enter:

```
query share userid
```

Where *userid* is the user identification of the virtual machine user.

Calculating a User's Share of Resources

The following examples show how to calculate a user's share of available resources.

Example 1

Assume, for example, that your installation typically has four active users. When you enter the QUERY SHARE command for each of the four, you find that they have the following shares:

```
BUKICH  : ABSOLUTE 40%
WADE    : RELATIVE 200
AVELLINI: RELATIVE 100
DOUGLASS: RELATIVE 100
```

- BUKICH has an absolute 40% share and therefore receives approximately 40% of available resources.
- The remaining 60% of shares is split among the three RELATIVE users:
  - WADE receives 200/400 (WADE's relative share divided by the total relative shares) of the remaining 60% of available resources.
  - AVELLINI and DOUGLASS each receive 100/400 of the remaining 60% of available resources.

Example 2

Assume, for this example, that your installation typically has six active users. When you enter the QUERY SHARE command for each of the six, you find that the following shares have been assigned:

```
WALKER  : ABSOLUTE 50%
VANLIER : ABSOLUTE 30%
SLOAN   : ABSOLUTE 30%
MOTTA   : RELATIVE 100
JONES   : RELATIVE 100
SMITH   : RELATIVE 100
```

Note that the total percentage of absolute shares is 110%. However, the actual percentage of resources available is 99%. (One percent is reserved for MOTTA, JONES, and SMITH, the RELATIVE share users.) Each absolute share virtual machine has its share scaled down by 99/110 (approximately 90%). This scaled-down share is called a *normalized* share and refers to the actual percentage of resources that the user receives.

Therefore:

- WALKER receives a normalized share of 45%.
- VANLIER and SLOAN each receive a normalized share of 27%.
- MOTTA, JONES, and SMITH each receive a normalized share of 1/3%.

Using SET SHARE with Other Tuning Controls

SET RESERVED

A virtual machine that has pages reserved with the SET RESERVED command gets to use at least the number of central storage frames specified on the command, for as long as storage contention does not become severe, even when the virtual machine is dormant. Thus, no paging delays are incurred (normally) when the virtual machine wants to use storage.

Using CPU Pools

You can use CPU pools to limit the amount of CPU resources (in terms of real CP or IFL processors) that groups of virtual machines are allowed to consume in aggregate. A CPU pool has a name and an associated CPU resource limit. You can assign one or more virtual machines to a CPU pool and have their combined maximum CPU consumption limited to the pool's capacity.

Note: CPU pools can be defined only when CONSUMPTION limiting (SET SRM LIMITHARD CONSUMPTION) is in effect on the system.

DEFINE CPUPOOL Command

Use the DEFINE CPUPOOL command to add a CPU pool to your system configuration. The command allows you to define the name of the CPU pool, the type of virtual CPU being limited (CP or IFL), and the

amount and type of the limit being set for the pool. Two types of resource limits can be set for the group of users in a CPU pool:

- LIMITHARD
- CAPACITY

After a CPU pool has been created, it remains until the next time z/VM is IPLed or you use the DELETE CPUPOOL command to remove it.

You can use the SET CPUPOOL command to change the CPU limit setting for an already defined CPU pool.

LIMITHARD CPU Limit

The LIMITHARD method limits the CPU pool to a specific percentage of the shared logical CPU resources (1% to 100% of the specified CPU type). The amount of CPU resources available to the CPU pool will change whenever the number of online shared processors changes.

The benefit of this type of limit is that CPUs can be varied online or offline based on system demand, and the amount of CPU resources available to the CPU pool is automatically adjusted. If the last IFL processor is varied offline or the first IFL processor is varied online, the CPU affinity of an IFL-limiting CPU pool will change (toggle between CPU affinity on and CPU affinity suppressed) in addition to the CPU limit change.

Attention: CP always assumes that every logical CPU can run to 100% busy, regardless of the LPAR's entitlement and regardless of the availability of excess power on the CPC. For example, if the system has five shared logical IFL CPUs, CP assumes that the shared IFL CPU resource has a total capacity of 500%. If a CPU pool is defined with a LIMITHARD 50% setting for TYPE IFL, the CPU pool will be allowed to run to 250% busy (that is, consume up to 2.5 CPUs of power). If PR/SM is limiting the LPAR's five logical IFL CPUs to 280% total capacity, CP will still allow the CPU pool to run to 250% busy. In that case, CP is really allowing the CPU pool to consume (250/280) = 89% of the scheduled IFL CPU resource.

CAPACITY CPU Limit

The CAPACITY method limits the CPU pool to an amount of processor power equivalent to a specific number of real CPUs (0.1 to 999 processors of the specified CPU type), although the number of logical CPUs of that type in use by the group can exceed the capacity number. As long as this capacity limit is less than the number of processors of that CPU type on the system, the amount of CPU resources available to the CPU pool will not change if the number of online processors changes, or if the CPU affinity of an IFL-limiting CPU pool toggles between the on and suppressed settings. For example, a CPU pool defined with CAPACITY 0.8 TYPE IFL will retain that limit regardless of the number of IFL processors on the system. If there are no IFL processors online, this CPU pool will be running with CPU affinity suppressed and will be limited to 0.8 CP processors.

The only time a system change can affect the resources given to a CPU pool with a CAPACITY limit is if the CAPACITY value is greater than the number of processors of that CPU type on the system before or after the system change. For example, assume that the limit for CPU pool LINGRP1 is 3.5 IFL processors but there are only 2 IFL processors varied online. In this case, the CAPACITY limit is greater than the amount of shared CPU resources available on the system. Therefore, the CPU pool runs as if there is no CPU limit, and the LINGRP1 users can use all of the IFL processors available (2.0 CPUs) without surpassing their limit. If another IFL processor is varied online, this group can now use more IFL resources (up to the available 3.0 CPUs). If a fourth IFL processor is varied online, this group will be limited to 3.5 IFL CPUs, and that limit will remain if more IFL processors are added to the system.

SCHEDULE Command

Use the SCHEDULE command to assign a user to an existing CPU pool or to remove a user from the CPU pool to which the user is currently assigned. Only a logged on or disconnected user can be assigned to a CPU pool, and a user can be assigned to only one CPU pool at a time. The assignment remains in effect until another SCHEDULE command is issued for that user or the user logs off.

Any individual CPU limits on the user as defined by the SET SHARE command remain in effect. If a user with an individual CPU limit is assigned to a CPU pool, both limits will be checked each time the user's CPU usage is evaluated, and the more restrictive limit will be applied.

In an SSI collection, the CPU pool assignment is retained if the user is relocated to another member of the collection, so a CPU pool with the same name and the same type of CPU being limited must exist on the destination member.

Displaying Information about CPU Pools

Use the QUERY CPUPOOL command to display information about the CPU pools that are defined on the system. You can display information about all CPU pools or a specific CPU pool, or display the name of the CPU pool to which a specific user is assigned.

Example 1

The following example shows the response for the QUERY CPUPOOL ALL command when CPU pools are defined. Each line under the heading contains the pool definition for a CPU pool, which includes the name of the pool, the limit set for the pool, the type of CPU that is limited, and the number of users currently assigned to the pool.

```
CPU pool      Limit     Type  Members
LINUXP2     8.0 CPUs    IFL        0
CPPOOL10     12 %        CP        8
LINUXP3      30 %       IFL       20
LINUXP1     2.5 CPUs    IFL        6
```

In this example there are four CPU pools. CPU pools LINUXP1 and LINUXP2 have CAPACITY limits for IFL processors, CPU pool CPPOOL10 has a LIMITHARD limit for CP processors, and CPU pool LINUXP3 has a LIMITHARD limit for IFL processors.

Example 2

The following example shows the response for the QUERY CPUPOOL POOL LINUXP1 MEMBERS command. The response displays the pool definition for the LINUXP1 CPU pool and lists the members of the pool, five per line.

```
CPU pool      Limit     Type  Members
LINUXP1     2.5 CPUs    IFL        6
The following users are members of CPU pool LINUXP1:
D70LIN12   D79LIN03   D79ADM     D79LIN10   D79LIN07
D79LIN04
```

Example 3

The following example shows the response for the QUERY CPUPOOL USER D79LIN03 command. The response indicates that user D79LIN03 is currently assigned to CPU pool LINUXP1.

```
User D79LIN03 is in CPU pool LINUXP1
```

Setting Up Permanent CPU Pools and Automatic User Assignments

When a z/VM system is shut down, all CPU pools defined on that system are deleted. To reestablish a CPU pool (if desired) after a system IPL, you must define the CPU pool again and assign users to it.

If you want to define a permanent CPU pool that will be created each time the system is IPLed, add the DEFINE CPUPOOL command to the AUTOLOG1 profile or add the command to an exec that AUTOLOG1 runs. This will ensure that the CPU pool is created early in the IPL process before user IDs automatically assigned to the group are logged on.

To automatically assign a user to a CPU pool when the user logs on, use the COMMAND directory statement to place the SCHEDULE command for the assignment in the user's directory entry. If no other SCHEDULE command is issued, the user will remain in that CPU pool until the user logs off. The SCHEDULE command can be placed anywhere the COMMAND directory statement is allowed, including

the directory profile for pooled users, where * should be specified for the user ID to put the logging on user into the CPU pool.

SET QUICKDSP Command

Note: CP's virtual processor management has been improved so that no virtual machine stays in the eligible list more than an instant before being added to the dispatch list. Therefore the quick dispatch option is now less meaningful, although it does get the virtual machine a different size elapsed time slice.

The SET QUICKDSP command and the QUICKDSP operand of the OPTION directory statement allow you to designate virtual machines that will not wait in the eligible list when they have work to do. Instead, a virtual machine with a QUICKDSP setting is assigned an eligible list class of E0 and is added to the dispatch list immediately. The types of virtual machines to which you should consider giving the QUICKDSP designation include:

- Service and other virtual machines with critical response-time requirements
- A production guest that has interactive characteristics
- Virtual machines such as MONWRITE or PERFSVM that collect or interpret performance data
- Virtual machines such as MAINT or OPERATOR, from which you type in tuning commands.

Do not give the QUICKDSP attribute to too many virtual machines, because this can nullify the effectiveness of the SRM controls such as STORBUF, LOADBUF, and DSPBUF.

Setting a User's QUICKDSP Characteristic

To give a user called SNELL the QUICKDSP characteristic, either code the QUICKDSP operand on the OPTION directory statement in SNELL's directory or enter the QUICKDSP command as shown here:

```
set quickdsp snell on
```

To turn the QUICKDSP characteristic off, enter:

```
set quickdsp snell off
```

Displaying a User's QUICKDSP Characteristic

To find out whether a user called MAYNARD has the QUICKDSP attribute, enter:

```
query quickdsp maynard
```

Using QUICKDSP with SET RESERVED

In some cases, you should consider using QUICKDSP together with the SET RESERVED command. SET RESERVED allows a virtual machine to have a specified minimum number of pages resident in real storage. For more information on the SET RESERVED command, see “Controlling Storage Allocation” on page 119.

Tuning the Processor Subsystem

The term *processor subsystem* refers to the resources of the processor (or processors) installed at your location. Processors are generally the most expensive resources in a system. Therefore, if some area of your system must be overcommitted, it can be argued this area should be your processors. In other words, if you must have a system bottleneck somewhere, the best place to have it is in the processor subsystem.

However, overused processors can cause performance problems. To decrease the work load on your processors, you can either install more (or larger) processors or reduce the load. Here are some guidelines for reducing the load on your processors:

- Try to reduce the amount of CCW translation. Blocking minidisks to 4 KB (the CMS maximum) may be a solution. Increase the block size where possible.
- Reduce the SHARE settings of the virtual machines you feel are the least crucial. This lessens the impact of these virtual machines on other users.
- Increase DSPSLICE.
- Use the SET SHARE command to limit those users who consume excessive amount of processor resource.

This section discusses CP commands you can use to tune the processor subsystem. For more information on the syntax of each command, see *z/VM: CP Commands and Utilities Reference*.

Displaying Processor Use

To display the work load on your processors, use the following commands:

- INDICATE LOAD
- INDICATE QUEUES

For a detailed description of these commands and their operands, see *z/VM: CP Commands and Utilities Reference*.

Controlling Processor Resources

You can set the interactive bias with SET SRM IABIAS.

CP sets by default a value, called the interactive bias (IABIAS), that determines how CP responds to interactive transactions.

Note: The term *interactive transaction* is a relative term. There is no specific value that determines when a transaction is interactive and when it is not. In general, the shorter a transaction is, the more interactive it is; the longer a transaction is, the less interactive.

The interactive bias consists of two components:

- An *intensity*—This determines how quickly CP starts processing work for a virtual machine. The intensity value specifies the percentage that this virtual machine moves up the dispatch list from its usual position. CP expresses intensity as a number between 0 and 100 and sets a default of 90. The higher the intensity, the quicker CP starts processing work for a virtual machine.
- A *duration*—This determines the number of time slices the intensity value lasts. However, after the initial time slice, a virtual machine's position in the dispatch list gradually fades to its usual position. The *minor time slice* (or dispatch slice) is the time that a virtual machine stays in the dispatch list before the scheduler repositions that virtual machine.

  For example, if you specify SET SRM IABIAS 90 3, the virtual machine is moved up the dispatch list 90 percent from its usual position for the first time slice (see Figure 9 on page 114). For the second time slice, the virtual machine fades back to a position that is 60 percent higher than normal. For the third time slice, the virtual machine's position falls to 30 percent above normal. For all ensuing time slices, the virtual machine returns to its usual dispatch list position.

  CP expresses duration as a number between 1 and 100 and assigns a default value of 2.

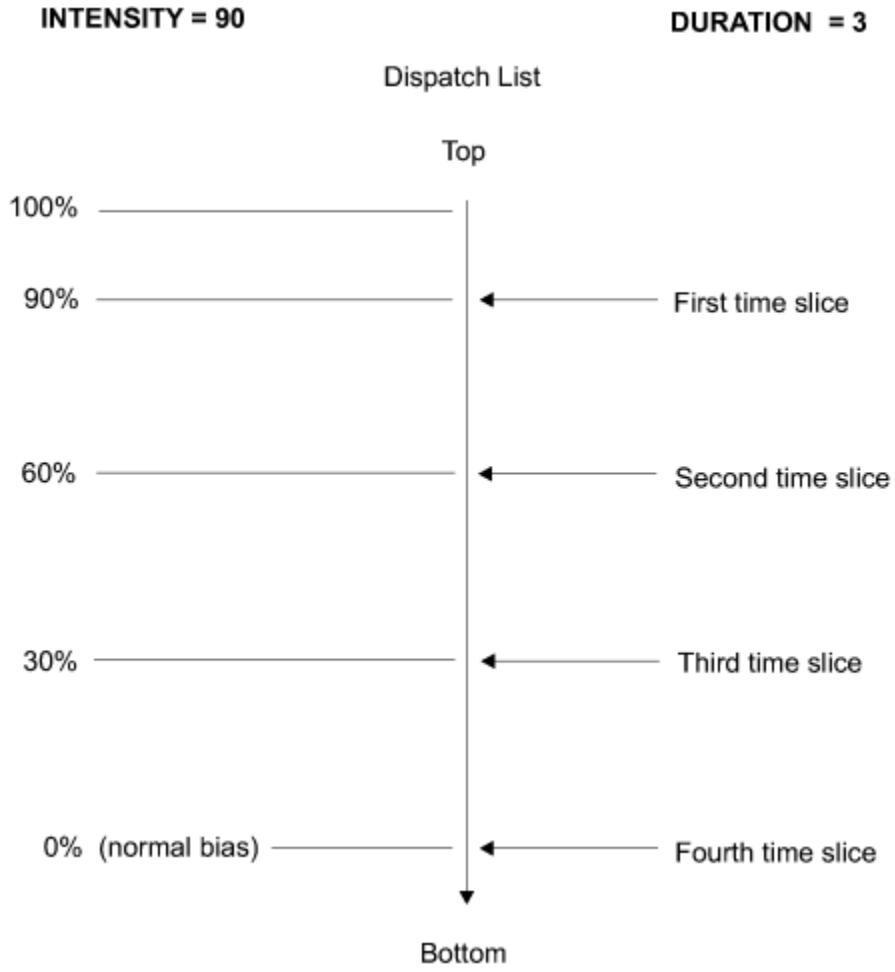


Figure 9. How the SET SRM IABIAS Command Works

Displaying the Current Interactive Bias

To display the current interactive bias, enter:

```
query srm iabias
```

CP responds with a message that says:

```
IABIAS: INTENSITY=%; DURATION=n
```

Changing the Current Interactive Bias

To change the interactive bias CP uses, enter:

```
set srm iabias m n
```

where:

m
is the new intensity value you assign.

n
is the new duration value you assign.

Usage Notes

1. The higher the intensity, the faster CP starts processing work for virtual machines. (The higher the intensity, the less accurate virtual machine scheduler shares become.)

2. The higher the duration, the longer the intensity value lasts. (The longer the intensity value lasts, the less accurate virtual machine scheduler shares become.)
3. In general, it is not useful to set the interactive bias intensity below 50 percent. However, finding an interactive bias that works better than the default value may require a certain amount of experimenting.
4. You may wish to set the interactive bias duration value as low as 1, because some shorter transactions (particularly CMS work) can be completed in a single elapsed or dispatch time slice.

Setting the Dispatch Buffer (SET SRM DSPBUF)

Note: CP's virtual processor management has been improved so that no virtual machine stays in the eligible list more than an instant before being added to the dispatch list. Therefore the DSPBUF option is now less meaningful.

The SET SRM DSPBUF command lets you control the number of users in the dispatch list. It permits you to overcommit or undercommit resources of the processors and I/O devices. (The SET SRM DSPBUF command is a more general tuning control than, for example, SET SRM STORBUF and SET SRM LDUBUF, which control specific resources.)

The three parameters of the SET SRM DSPBUF command determine, by transaction class (E1, E2, and E3), the number of users the system allows in the dispatch list. Recall that users with an eligible list transaction class of E1 are expected to have short-running transactions. Users with a transaction class of E2 are expected to have medium-running transactions. Users with an E3 transaction class are expected to have long-running transactions.

The following is an example of the SET SRM DSPBUF command:

```
set srm dspbuf 35 30 18
```

This command specifies that:

- Thirty-five openings in the dispatch list are available to E1, E2, and E3 users. Of these 35 openings, five are guaranteed to E1 users. This value is determined by subtracting the second value in the command (30) from the first value (35): 35-30 = 5.
- Thirty openings in the dispatch list are available to E2 and E3 users. Of these 30, 12 are guaranteed not to be taken by E3 users (30-18 = 12). However, these 12 may be temporarily occupied by E1 users.
- Eighteen openings are available to E3 users, but these openings may be temporarily occupied by E1 and E2 users.

The default values for DSPBUF are, for practical purposes, infinity. This has the effect of removing DSPBUF as a control mechanism. These default settings are quite satisfactory for most VM systems.

Using SET SRM DSPBUF—An Example

To illustrate how to use the SET SRM DSPBUF command, consider an installation with:

- A 2064 Model 106 (6-way processor)
- 200 DASD
- 1000 logged-on CMS users.

This installation has a total of 206 resources: six processors and 200 DASDs. In general, each CMS user in the dispatch list is either using a processor or performing an I/O operation to exactly one DASD device at any given instant. Thus, a rule of thumb is that each CMS user uses exactly one resource at any time.

If this installation allows exactly 206 users into the dispatch list (one user for each resource), some users will contend for the same resource. For example, more than one CMS user may require access to the same DASD. If four users need the same DASD, three of them will spend some time waiting for this resource. This means that other resources will not be used and the system will not be fully used. Therefore, it is recommended that you allow more users into the dispatch list than the number of available system

resources would indicate. As a starting point for determining the number of users you allow in the dispatch list:

1. Count your total system resources (processors plus DASDs).
2. Add 50 percent.
3. Allocate fractions of your total to each transaction class by using the SET SRM DSPBUF command.

For this hypothetical installation:

- The total number of system resources is 206.
- Adding 50 percent brings this to 309.
- The trial command setting would then be: SET SRM DSPBUF 309 278 247.

Note that you should be counting the number of *effective system resources* (resources that are actually used) rather than just counting all your hardware. In the previous example, all 200 DASDs are counted toward the total number of system resources. However, some DASDs may actually be used much less than others. For example, most of the work may be concentrated on 120 of the 200 DASDs. If this is the case, only 120 of the 200 DASDs are effective system resources.

Setting Upper Limits on the Real Storage Users Occupy (SET SRM MAXWSS)

In a heavily used interactive system, some work may be impractical to run because the working set is too large. For example, if a large virtual machine has a working set that would fill 100% of real storage, that virtual machine cannot be run without causing unacceptable interference with the interactive load. Even a virtual machine with a working set that fills 90%, or 75%, or 50% of storage almost certainly cannot be run on a system that needs most of real storage to support its interactive work load. In these cases, you should weigh the value of your system's interactive work against the value of running large virtual machines. You should also determine the largest percentage of real storage that you can give to a large virtual machine. By specifying this percentage on the SET SRM MAXWSS command, you instruct the system not to let any virtual machine (except virtual machines that have large absolute shares) occupy more than this percentage of storage.

If your system already has a consistent eligible list, specifying a percentage using the SET SRM MAXWSS command is probably all you need to do to prevent virtual machines from occupying more than the desired amount of real storage. The existence of an eligible list is needed for SET SRM MAXWSS to prevent virtual machines from growing beyond the desired maximum. When a virtual machine exceeds its MAXWSS, the scheduler drops it from the dispatch list. Some time later, when it gets back into the dispatch list, it will have lost its pages and will have to load them again. If it once again grows beyond the size specified by MAXWSS, it is quickly dropped again. In this way, large virtual machines are kept out of the dispatch list most of the time. Yet, from time to time, they get another chance to run. Thus, the scheduler can determine whether they are still large. Also, if the interactive work load lessens to the point where the eligible list disappears, the large virtual machines get to run. This is because their delay in the eligible list falls to zero and they will spend all their time in the dispatch list.

Tuning the Paging Subsystem

Paging is often the most influential component of overall response time. In general, a transaction uses a predictable amount of processor time and I/O time, regardless of the amount of system activity. But what can vary greatly is the number of paging operations during that transaction. As the system load increases, page occupancy decreases and the number of page faults increases. This causes an increase in response time.

For guidance on the use of HyperPAV aliases and transport-mode channel programs by the Paging Subsystem, see "HyperPAV Paging Support" on the z/VM performance website at IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

For guidance on AGELIST tuning, see "Configuring Processor Storage" on the z/VM performance website at IBM: VM Performance Resources (https://www.ibm.com/vm/perf/).

To determine whether your paging subsystem needs tuning, check the paging with the INDICATE LOAD and INDICATE PAGING commands (discussed later in this section). If many users are in page wait, the rate can be low because of configuration inadequacies. If the paging rate is low and few users are in a page wait, you do not have a paging problem. If the paging rate is high, you may need to determine whether your installation has enough paging devices. To attempt to overcome a high paging rate, you should consider:

- Adding real storage
- Defining more of your existing DASD as paging devices
- Using the SET SRM STORBUF command to control storage contention
- Using the SET SRM LDUBUF command to minimize queueing time in the paging device queues.

You should define system paging space on devices that are not heavily used for other purposes; do not define paging space on the same device as nonpaging space. Recommendations for setting up your paging configuration are found in *z/VM: CP Planning and Administration*.

This section discusses CP commands you can use to tune the paging subsystem. For more information on the syntax of each command, see *z/VM: CP Commands and Utilities Reference*.

Displaying Paging Information

INDICATE LOAD
Use the INDICATE LOAD command to display the operating load (including paging activity) on your system.

INDICATE PAGING
Use the INDICATE PAGING command to display a list of virtual machines in page wait or to display page residency data for all users of the system.

QUERY ALLOC
Use the QUERY ALLOC command with the PAGE keyword to display the number of pages that are allocated, in use, and available for paging.

Controlling Paging Resources

You can set the loading user buffer with SET SRM LDUBUF.

The SET SRM LDUBUF command partitions the commitment of the system's paging resources. *LDUBUF* stands for *loading user buffer*. A *loading user* is expected to be a heavy user of the system paging resources. Specifically, a loading user is defined as a virtual machine that is expected to have a high paging rate—one that will need to load its working set—while in the dispatch list.

To determine which users are loading users at a given time, enter the INDICATE QUEUES EXP command.

The percentages specified on the SET SRM LDUBUF command determine, by transaction class (E1, E2, or E3), the number of loading users the system allows in the dispatch list. The system allows only a certain number of loading users from each class into the dispatch list at any one time. That number is determined as the system applies the percentages specified with the SET SRM LDUBUF command to the maximum number of loading users the system can handle.

To do this, the system considers the number of paging exposures needed for users in the dispatch list. A *paging exposure* is a logical, addressable paging device. The SET SRM LDUBUF command controls the number of users in the dispatch list so that the system paging devices are kept busy without being overloaded (reducing the total time spent queued on the paging devices and the load on the storage subsystem). It is generally a good idea to overcommit rather heavily on the SET SRM LDUBUF command. The exact settings are generally not too critical.

The following is an example of the SET SRM LDUBUF command:

```
set srm ldubuf 300 200 100
```

This command specifies that:

- Three hundred percent of system paging resources is available to E1, E2, and E3 users. Specifically, the scheduler examines 300 percent of system paging resources when deciding whether an E1, E2, or E3 user can enter the dispatch list. If the total paging exposures being used by virtual machines in the dispatch list, plus the paging exposures being used by the virtual machine that is a candidate for being added to the dispatch list, do not exceed 300 percent of all paging exposures, this user is added to the dispatch list.

  For example, suppose that an installation has 20 addressable paging devices, 15 of which are allocated to users currently in the dispatch list. Also, suppose that for a virtual machine that is a candidate to enter the dispatch list, the scheduler determines that the virtual machine needs a paging device. This virtual machine can enter the dispatch list, because the number of paging devices it needs (one)—along with the number of paging devices that virtual machines already in the dispatch list need (15)—does not exceed 300 percent (60) of the total number of paging devices in the system (20).
- Two hundred percent of system paging resources is available to E2 and E3 users. Specifically, the scheduler examines 200 percent of system paging resources when deciding whether an E2 or E3 user can enter the dispatch list. If the total paging exposures needed by E2 and E3 users in the dispatch list, plus the paging exposures needed by the user that is a candidate for being added to the dispatch list, do not exceed 200 percent of all paging exposures, and the paging exposures needed by this user do not push the Q1 quota over 300%, this user is added to the dispatch list.
- One hundred percent of system paging resources is available to E3 users. Specifically, the scheduler examines 100 percent of system paging resources when deciding whether an E3 user can enter the dispatch list. If the total paging exposures needed by E3 users in the dispatch list, plus the paging exposures needed by the user that is a candidate for being added to the dispatch list, do not exceed 100 percent of all paging exposures, and the paging exposures needed by the user do not push the Q2 quota over 200% nor the Q1 quota over 300%, this user is added to the dispatch list.

Usage Note

Short transactions require more paging than longer transactions. Keep this in mind when you allocate paging resources to users with short-running transactions (the E1 users).

Using SET SRM LDUBUF with Other Tuning Controls

You may need to use SET SRM LDUBUF with SET SRM STORBUF and SET SRM DSPBUF because each of these commands sets limits on the number of users allowed in the dispatch list.

With SET QUICKDSP

A QUICKDSP virtual machine is exempt from any limits you establish with SET SRM LDUBUF.

Tuning the Storage Subsystem

Waiting for storage to become available is a common cause of slow response time. Because of this, the storage subsystem should probably be the first place you look if you are trying to resolve a system bottleneck.

This section discusses CP commands you can use to tune the storage subsystem. For more information on the syntax of each command, see *z/VM: CP Commands and Utilities Reference*.

Displaying Storage Information

QUERY FRAMES

Use the QUERY FRAMES command to display information about real storage frames.

Controlling Storage Allocation

Use the following commands to control storage allocation.

SET RESERVED

Use the SET RESERVED command to allow a virtual machine to have a specified number of pages resident in real storage that are generally exempt from being stolen. SET RESERVED is designed to enhance the efficiency of a particular virtual machine; however, system performance may be degraded if you specify this command for too many virtual machines.

To display the number of reserved page frames for a virtual machine, enter the QUERY RESERVED command. For additional details on the QUERY RESERVED command, see *z/VM: CP Commands and Utilities Reference*.

Usage Notes:

1. Use with server machines.

   When a page fault occurs, normally the fault is handled synchronously; that is, the faulting virtual processor waits until the page fault is resolved. However, there are two exceptions:

   a. Use of the pseudo page fault facility in conjunction with the VM/VS handshaking feature.

   b. Use of the PFAULT macro to do asynchronous page fault handling while in access register mode.

   If that virtual machine is running on behalf of just one user, this synchronous handling has little significance. However, if the virtual machine is serving more than one user, one or more users might be delayed until the page fault is resolved.

   Because of this, the SET RESERVED command is especially useful for reducing the amount of page faulting that occurs in multitasking server virtual machines. By reserving the working set of each such virtual machine on a z/VM system, paging is, in effect, moved out of those virtual machines where it can delay multiple users and into those virtual machines where it delays just one user.

2. Use with virtual machines having stringent response time requirements

   When a virtual machine is dormant, it is likely that CP will steal some of its pages. When it leaves the dormant state, this virtual machine will need its working set to be brought back into storage. This heavy paging requirement will reduce the virtual machine's responsiveness. The solution may be to use SET RESERVED for at least some of the virtual machine's pages and to specify SET QUICKDSP ON to ensure that the virtual machine stays in the dispatch list whenever it is active.

3. Activity pattern considerations

   It is important to consider the virtual machine's activity pattern when deciding whether to use SET RESERVED. Consider the following cases:

   - **The virtual machine has very short dormant periods:** In this case, you should consider using SET RESERVED. Otherwise, CP tries to use the virtual machine's pages while the virtual machine is dormant and then must retrieve these pages when the virtual machine becomes active.
   - **The virtual machine displays long periods of activity followed by long periods of dormancy:** In this case, you should probably not use SET RESERVED. While the virtual machine is dormant, CP can make good use of the virtual machine's pages, provided you do not use SET RESERVED.
   - **The virtual machine displays very short periods of activity followed by long periods of dormancy:** As in the previous case, you should not use SET RESERVED because CP can make good use of the virtual machine pages during the long dormant periods. In fact, QUICKDSP ON may not even be needed because the virtual machine has only short periods of activity. During these active periods, the virtual machine's interactive bias setting takes effect. For more information on interactive bias, see “Controlling Processor Resources” on page 113.

In some cases, you should consider using QUICKDSP together with the SET RESERVED command. For information on the SET QUICKDSP command, see “SET QUICKDSP Command” on page 112.

LOCK

The LOCK command locks selected pages of a virtual machine's virtual storage into real storage. These pages remain in real storage until they are removed with the UNLOCK command. Heavily used pages that otherwise would be constantly paged in and out can be locked in real storage. Generally, it is preferable to reserve a certain number of page frames rather than to lock specific pages into memory.

Note: When using the LOCK command, you must specify whether the guest pages are to be locked into host logical storage or host real storage. Locking pages into host logical storage is intended for debugging purposes. If you are using LOCK for performance reasons, specify the REAL operand to lock the pages into real storage.

SET SRM STORBUF

The SET SRM STORBUF command partitions the commitment of real storage (in terms of pages) based on the transaction class (E1, E2, or E3) of a user. Basically, SET SRM STORBUF lets you under- or overcommit real storage. The SET SRM STORBUF command works in much the same way as the SET SRM LDUBUF command. Note, however, that SET SRM STORBUF partitions the use of real storage, while SET SRM LDUBUF partitions the use of paging resources (in terms of access). The exact settings of the SET SRM STORBUF command are generally more critical than the settings of the SET SRM LDUBUF command and should be given more attention. Incorrect settings can result in thrashing.

The following is an example of the SET SRM STORBUF command (see Figure 10 on page 121):

```
set srm storbuf 100 75 50
```

This command specifies that:

- One hundred percent of storage is available to E1, E2, and E3 users. Specifically, 100% is the percentage of storage the scheduler examines when deciding whether an E1, E2, or E3 user is allowed to enter the dispatch list. If the total working sets of all users in the dispatch list plus the working set of the user that is a candidate for being added to the dispatch list are equal to or greater than 100% of storage (the dynamic paging area), this user is *not* added to the dispatch list.
- Seventy-five percent of storage is available to E2 and E3 users. Specifically, 75% is the percentage of storage the scheduler examines when deciding whether an E2 or E3 user can enter the dispatch list. If the total working sets of all E2 and E3 users in the dispatch list plus the working set of the user that is a candidate for being added to the dispatch list are equal to or greater than 75% of the DPA, this user is *not* added to the dispatch list.
- Fifty percent of storage is available to E3 users. Specifically, 50% is the percentage of storage the scheduler examines when deciding whether an E3 user can enter the dispatch list. If the total working sets of all E3 users in the dispatch list plus the working set of the user that is a candidate for being added to the dispatch list are equal to or greater than 50% of the DPA, this user is *not* added to the dispatch list.

For examples of using SET SRM STORBUF, see “Sample Performance Problems and Solutions” on page 125.

Usage Notes

1. Performance gains might be realized by overcommitting real storage. This treatment can be accomplished by using the SET SRM STORBUF command. When you overcommit storage this way, the virtual machine's working sets are still computed as before, but the apparent real storage available to virtual machines that want to enter the dispatch list appears larger. Therefore, more virtual machines are allowed into the dispatch list, which usually results in higher paging rates; but often the benefit of reducing the eligible list and moving users into the dispatch list offsets this increase. Monitor the paging rate against the response time and command rate, and adjust the STORBUF setting accordingly.
2. To overcommit real storage, specify one or more of the SET SRM STORBUF parameters over 100%.
3. Sometimes it might be desirable to overcommit real storage by a very large amount. For example, assume a 64 MB z/VM system with a 32 MB MVS guest and a 32 MB z/VM guest. The default STORBUF setting only allows Q3 users 95% of the 64 MB. Therefore, one virtual machine is dispatched and runs

fine and the other locks up, and after a few minutes the situation reverses. A possible solution would be to use the following settings for SET SRM STORBUF:

```
set srm storbuf 200% 150% 100%
```

4. There are a few systems for which undercommitting real storage can improve performance. They are characterized by many small users, no large users, and a poor paging subsystem (few DASD). In this case, any paging causes poor performance, therefore very low STORBUF values can be used to hold the users in the eligible list and avoid paging. In these rare instances undercommit storage by specifying all parameters below 100%.

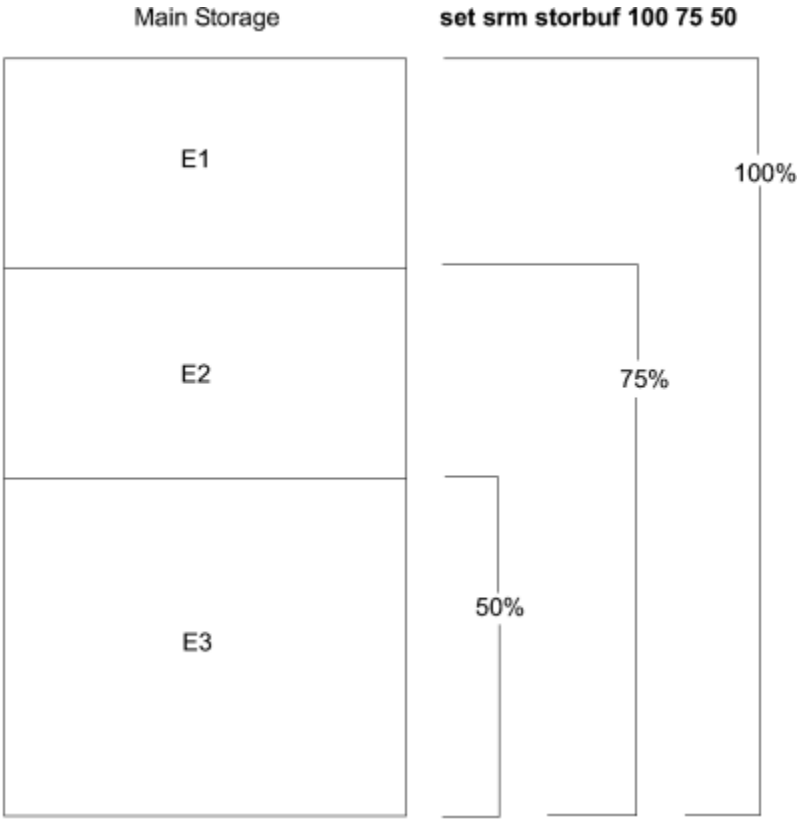


Figure 10. How the SET SRM STORBUF Command Works

Using SET SRM STORBUF with Other Tuning Controls

It may be necessary for you to use SET SRM STORBUF in conjunction with SET SRM LDUBUF and SET SRM DSPBUF, because each of these commands sets limits on the number of users allowed in the dispatch list. Controlling the users in the dispatch list with DSPBUF is a direct control on the number of users and not the resources they consume, therefore STORBUF and LDUBUF are recommended as your first course of action.

Tuning the I/O Subsystem

Because many factors influence I/O performance, use INDICATE QUEUE, INDICATE I/O, or QUERY MDCACHE commands, or use Performance Toolkit for z/VM, to determine which factor is causing a system bottleneck. The following guidelines will help you get the best performance from your system:

- Use minidisk cache to minimize physical I/Os. For more information on minidisk cache, see “Minidisk Cache” on page 40.

  Providing more real storage for minidisk cache usage may improve I/O performance if the cache hit ratio is low because there is not sufficient space to cache highly referenced data. Minidisk cache efficiency can be improved by disabling the cache for minidisks that are poor candidates for this cache. Poor candidates include minidisks that are write-mostly, read-once, or those that are only updated

and accessed via MAPMDISK with data spaces. Use the MINIOPT directory control statement or the SET MDCACHE command to disable minidisk cache for specific minidisks. For applications with I/O characteristics that are not favorable for minidisk cache, the SET MDCACHE INSERT command can be used to temporarily avoid inserting data into cache. An application that scans all the files on a minidisk is an example of a case where disabling the cache might be desirable because the data is read only once.

- Balance the activity on all DASD volumes. To do this, set up your directory so that some DASD definitions use your first DASD volume, some use the second, and so on. After the system is running, check the activity on all volumes to make sure that one DASD does not have 30 active users while another DASD has 30 users who are much less active.
- Use whole volumes for CP-owned paging and spooling areas. A volume should be either all paging or no paging, or all spooling or no spooling.
- TDISK areas are generally very high-use areas and should be separated from other high-use areas. Do not place temporary disks on the same volumes with PAG, DIR, SPOOL, or high-activity minidisks.
- Reduce the number of I/O operations required per transaction. Try increasing the block size of CMS minidisks to 4 KB (the maximum supported by CMS). If the block size is set to 4 KB, you may use up more disk space if the minidisk is only used for files that are less than 4 KB, but the trade-off in performance may be beneficial. If the files are 4 KB or greater, however, you use less space and reduce the number of I/O operations as well. ECKD devices should be formatted without filler records using ICKDSF.
- Try increasing the CMS file cache size as described in "Adjusting the Minidisk File Cache Size" on page 54. This decreases the number of I/O operations required for sequential operations to large CMS files. The system paging rate is increased, but the trade off may be beneficial.
- To help resolve I/O problems that you suspect may be SFS related, see Chapter 13, "SFS Tuning," on page 129.
- Exploit the virtual disks in storage feature. Good candidates for this include the VSE guest lock files, temporary files from compilers, and temporary files used for sorting. Virtual disks in storage can significantly reduce delays because of I/O processing. However, the improved I/O performance may be a trade off for increase storage and paging requirements.

This section discusses CP commands you can use to tune the I/O subsystem. For more information on the syntax of each command, see *z/VM: CP Commands and Utilities Reference*.

Displaying I/O Information (INDICATE I/O)

Use the INDICATE I/O command to display a list of virtual machines that are in I/O wait. Because the response to this command indicates only an instantaneous sample, you should enter the INDICATE I/O command several times before you draw any conclusions.

Using the Subchannel Measurement Block

A subchannel measurement block is a control block that can be associated with a given device. It contains measurement data for the device such as the I/O rate and the timing values for various components of service time. The hardware has responsibility to update this information. From the measurement block information, performance products can compute device service time, I/O rate, and utilization. This is a great source of information for solving I/O related problems.

If a performance product shows the values from the subchannel measurement block as zero, it could be caused by one of the following:

- The subchannel measurement block is owned by a guest operating system. In this case, only the guest sees the most current values.
- The hardware is not properly updating the subchannel measurement block. The timing facility of the measurement block architecture is optional, and some systems may not have implemented it.
- A subchannel measurement block is not associated with the device. This can be set or queried with the SET SCMEASURE and QUERY SCMEASURE CP commands.

The service time for a device consists of three components which are determined by the subchannel measurement block.

- **Connect Time** is the time a device is logically connected to a channel path for transferring data between it and the channel subsystem. Except for some protocol and search time, the greatest portion of connect time is due to transferring data. Therefore, greater connect times usually indicate larger amounts of data being moved.
- **Function Pending Time** is the time accumulated when a complete path to a device cannot be obtained. Aside from some typical protocol time (usually less than 1ms), pending time can result from delays caused by shared resources by another system. These resources can be channel, control unit, or devices. Pending time can also result on some cached controllers for certain internal path busy conditions.
- **Disconnect Time** is the time accumulated when the device is logically disconnected from the channel subsystem while the subchannel is subchannel-active. Disconnect time can result from a seek, latency, rotational positioning sensing delay, or cache controller processing. Seek time is the delay that occurs while the DASD arm moves to the correct cylinder. Latency is the time spent waiting for the track to rotate to the head. Rotational positioning sensing (RPS) delay occurs after the data has been found but the controller must wait to reconnect to the processor. There is only a small window of time to reconnect, otherwise another revolution is required before data is in position again. High disconnect time most often indicates high seek time, poor cache efficiency, or control unit contention.

Controlling I/O Resources

Throttling I/O Devices (SET THROTTLE)

The SET THROTTLE command allows you to limit the rate of I/Os issued to a real device. This can be used to minimize the impact a particular device has on system performance. For example, assume I/O to a particular DASD on a control unit is very high and greatly increases the contention at the control unit level. This contention might be increasing the service time of I/O to other DASD on this control unit and therefore increasing the user response time. If the I/O to the one very busy device is not critical I/O, you might want to throttle the I/O to this DASD to lower contention. It is important to note that I/O throttling does not affect CP I/O. Also, the throttling function can not distinguish between users. The throttle is done at the device level only.

I/O Priority Queueing

I/O Priority Queueing provides a method to favor DASD I/O of one virtual machine over another. This priority is used at two levels. When virtual machine I/O is targeted for the same real DASD volume at a rate that creates a queue for that device, the priority value is used in determining where in the queue the I/O should be placed. The priority value is also used by the hardware I/O subsystem when there is contention for channel paths. The higher priority I/Os will get better access to the channels. I/O priority effects are ineffective in cases where I/O resources are unconstrained.

When queueing occurs at the DASD volume level, increasing the I/O priority value for a virtual machine can improve the I/O performance. The problem of queueing can be seen in various performance monitor products by high response time where queue time is a significant portion of that response time. I/O priority settings can also help scenarios where there is contention for channels. High values in the function pending time portion of I/O service time often indicates contention at the channel level. The corresponding channel utilization will be high. In these cases, setting a higher I/O priority value may improve the performance for that virtual machine. For more information on I/O priority queueing, see the QUERY IOPRIORITY and SET IOPRIORITY commands in *z/VM: CP Commands and Utilities Reference*.

Setting the Dispatch Buffer (SET SRM DSPBUF)

The SET SRM DSPBUF command lets you control the number of users in the dispatch list. It also permits you to over- or undercommit resources of the processors and I/O devices. The three parameters of the SET SRM DSPBUF determine by transaction class (E1, E2, and E3) the number of users the system allows in the dispatch list. Recall that users with an eligible list transaction class of E1 are expected to have

short-running transactions. Users with a transaction class of E2 are expected to have medium-running transactions. Users with an E3 transaction class are expected to have long-running transactions.

For more information on the SET SRM DSPBUF command, see "Setting the Dispatch Buffer (SET SRM DSPBUF)" on page 115. For examples of using the SET SRM DSPBUF command, see "Sample Performance Problems and Solutions" on page 125 and "Using SET SRM DSPBUF—An Example" on page 115.

Setting the Minor Time Slice (SET SRM DSPSLICE)

The SET SRM DSPSLICE command controls the value of the minor *time slice* of a virtual machine. (A minor time slice, also called simply a time slice, is the length of time that a virtual machine stays in the dispatch list before the scheduler repositions it.) A smaller time slice causes the scheduler to sort the dispatch list more frequently. This causes more overhead for both the system and users. A larger time slice causes the scheduler to sort the dispatch list less frequently, thus causing less overhead. The user runs longer without being artificially disturbed.

Displaying the Size of the Current Minor Time Slice

To display the size of the minor time slice the scheduler assigns, enter:

```
query srm dspslice
```

Changing the Size of the Current Minor Time Slice

To change the size of the minor time slice the scheduler assigns, enter:

```
set srm dspslice minslice
```

Where *minslice* is the new size, in milliseconds, of the time slice the scheduler assigns.

Usage Notes

1. Decreasing the size of the minor time slice that the scheduler assigns causes the scheduler to sort the dispatch list more frequently; it also increases system overhead (the number of instructions CP must perform to accomplish a particular task). Decreasing the size of the minor time slice also causes CP to sort I/O bound work from processor bound work more effectively. This may produce better I/O throughput.
2. Increasing the size of the minor time slice that the scheduler assigns causes the scheduler to sort the dispatch list less frequently; it decreases system overhead but may also decrease throughput. Thus, if very few virtual machines are experiencing processor delays, DSPSLICE can be increased.
3. When you make a change to the DSPSLICE setting, you should also consider changing the duration parameter of SET SRM IABIAS. For an example of using SET SRM DSPSLICE together with SET SRM IABIAS, see "Sample Performance Problems and Solutions" on page 125.
4. Adjustments you make to your system with SET SRM DSPSLICE fall into the category of fine tuning. Such adjustments probably have only small effects on your system. To fix a major performance problem, you should probably try other adjustments before you try SET SRM DSPSLICE.
5. There are no guidelines for selecting an optimum size for the minor time slice. Selecting one that works best for your installation may require a certain amount of experimenting.

Tuning Hot I/O Detection

The hot I/O detection function protects CP from broken hardware that floods the system with unsolicited interrupts. Without this valuable protection, severe performance degradation or a HARD abend condition will occur. When the system detects a hot I/O problem, it attempts to recover. If recovery is not possible, the system removes the device from active I/O configuration. In either case, CP writes one or more messages to the primary system console. The messages include the address of the broken device.

HOT_IO_RATE is an optional system configuration file statement that allows you to define the maximum number of consecutive, unsolicited interrupts per second that CP should allow from an I/O device (the hot I/O rate) before refusing to accept input from it. By default, CP detects a hot I/O condition when a rate of 16 unsolicited interrupts per second occurs for a device. However, some devices are designed to generate a large number of interrupts per second. If you are using the default, CP can detect a hot I/O condition and unnecessarily remove the device from the active I/O configuration. In these cases, you can specify a hot I/O detection rate greater than 16. This can be done using either of the following:

- HOT_IO_RATE statement in the SYSTEM CONFIG file
- SET HOTIO command

See *z/VM: CP Planning and Administration* for additional information on the above methods.

If you do not know what hot I/O detection rate to use, you need to take an actual sampling of the amount of unsolicited interrupts per second that the device generates. For example, if the maximum observed rate is 25.4, then specify 26 for that device.

Sample Performance Problems and Solutions

The samples shown here describe how particular problems might be handled. For solutions to problems that actually affect your system, you must decide which tuning controls you should use as determined by monitoring your system. You can use these samples as a guide for some types of problems.

Poor Interactive Response Time (General)

If you notice relatively poor response time for short-running transactions, you may have a processor-constrained system. Adjustments to the SET SRM IABIAS value may improve the problem. For this situation, you should try increasing the IABIAS value.

To determine the IABIAS value, enter:

```
query srm iabias
```

Suppose the system responds with:

```
IABIAS: INTENSITY=90, DURATION=2
```

You might want to increase the IABIAS intensity by entering:

```
set srm iabias 95 2
```

These settings give interactive users a performance boost for the first time slice of their transaction (intensity of 95) and a greater-than-normal priority for the length of this performance boost (duration of 2). If necessary, increase both IABIAS values to see whether any further improvements can be obtained.

Note: To improve interactive response time, consider using the SET SRM STORBUF command in conjunction with the SET SRM IABIAS command. Refer to the next example for information on SET SRM STORBUF.

Poor Interactive Response Time (with a Large Number of Interactive Users)

If your system has a large number of highly interactive users, and you observe their response time to be unsatisfactory, first determine whether an excessive number of users are in page waits by using the INDICATE QUEUES and INDICATE PAGE commands. If this condition exists, refer to "Tuning the Paging Subsystem" on page 116 for information on tuning your paging subsystem. If the paging rate is satisfactory, consider using SET SRM STORBUF to reserve more storage for users with short-running transactions. First, determine your current STORBUF settings:

```
query srm storbuf
```

Suppose the system responds with:

```
STORBUF: Q1-100% Q2-85% Q3-75%
```

This response indicates that:

- One hundred percent of real storage is reserved for users with short, medium, and long-running transactions (E1, E2, and E3 users, respectively).
- Eighty-five percent of real storage is reserved for users with medium- and long-running transactions (E2 and E3 users, respectively).
- Seventy-five percent of real storage is reserved for users with long-running transactions (E3 users).

To reserve more storage for users with short-running transactions, enter:

```
set srm storbuf 100% 70% 60%
```

Rather than increasing the percentage for short-running transactions as shown previously, you may instead choose to decrease the value for long-running transactions:

```
set srm storbuf 80% 70% 40%
```

Note: As you tune your system, you may want to adjust the storage commitment level upward or downward by moving the STORBUF parameters over or under 100%. If you wish to overcommit real storage, set one or more of the parameters of the SET SRM STORBUF command over 100%. If you wish to under-commit real storage, set all parameters under 100%.

Poor Noninteractive Response Time

If you find that long-running jobs are taking too long to complete because of time spent in the eligible list, you should consider making the following tuning adjustments:

1. Increase the number of E3 users you allow in the dispatch list with the SET SRM DSPBUF command.
2. Adjust the long-running (and possibly the medium-running) percentages of the SET SRM STORBUF command upward.

First, determine your current DSPBUF setting:

```
query srm dspbuf
```

Suppose the system responds with:

```
DSPBUF: Q1=40 Q2=30 Q3=10
```

This response indicates that:

- Forty openings in the dispatch list are available to E1, E2, and E3 users. Of these 40 openings, 10 are guaranteed to E1 users. This value is determined by subtracting the second value in the command (30) from the first value (40): 40-30 = 10.
- Thirty openings in the dispatch list are available to E2 and E3 users. Of these 30, 20 are guaranteed not to be taken by E3 users (30-10 = 20). However, these 20 may be temporarily occupied by E1 users.
- Ten openings are available to E3 users, but these openings may be temporarily occupied by E1 and E2 users.

To improve the performance of noninteractive users, allow more E3 users into the dispatch list by increasing the Q3 value of the SET SRM DSPBUF command:

```
set srm dspbuf 40 30 20
```

If, after your adjustment, noninteractive work is still taking too long to complete, further tuning may be necessary. You may want to adjust the long-running (and possibly the medium-running) percentages of the SET SRM STORBUF command upward. First, determine the current setting:

```
query srm storbuf
```

Suppose the system responds with:

```
STORBUF: Q1-100% Q2-85% Q3-75%
```

A possible adjustment to aid noninteractive users is:

```
set srm storbuf 80% 75% 70%
```

Low Paging Rates (with the Number of Loading Users at a Maximum)

If your system is experiencing low paging rates, use the INDICATE LOAD command to determine whether the number of loading users your system can tolerate is at a maximum. If this is true, you will probably want to adjust your SET SRM LDUBUF settings. To determine what your current settings are, enter:

```
query srm ldubuf
```

For this example, suppose the system responds with:

```
LDUBUF: Q1-100% Q2-85% Q3-60%
```

This response indicates that:

- The count of Q3 loading users admitted to the dispatch list will be no higher than 60% of the number of paging exposures.
- The count of (Q3 or Q2) loading users admitted to the dispatch list will be no higher than 85% of the number of paging exposures.
- The count of (Q3 or Q2 or Q1) loading users admitted to the dispatch list will be no higher than 100% of the number of paging exposures.

It is generally advisable to overcommit on paging. Try entering:

```
set srm ldubuf 300 200 100
```

If your work consists almost exclusively of Q3 virtual machines, you may want to go even further, for example:

```
set srm ldubuf 500 400 300
```

To determine which users are loading users at a given time, enter the INDICATE QUEUES EXP command.

Tuning Summary

Table 4 on page 127 shows you which tuning controls (CP commands) apply to different environments you may face at your installation.

The controls for each environment are listed in approximate order of importance, though in some environments, the order of importance is difficult to predict.

Table 4. Summary of Tuning Controls

Situation	Appropriate Tuning Controls
Processor-constrained environment	1. SET SRM DSPBUF 2. SET SRM IABIAS
Paging-constrained environment	1. SET SRM LDUBUF 2. SET RESERVED
Storage-constrained environment	1. SET SRM STORBUF 2. SET RESERVED

Table 4. Summary of Tuning Controls (continued)

Situation	Appropriate Tuning Controls
I/O-constrained environment	1. SET SRM DSPBUF 2. SET SRM DSPSLICE 3. SET SRM IABIAS
Improved performance needed for a production guest	1. SET SHARE 2. SET QUICKDSP 3. SET RESERVED
Improved performance needed for a service virtual machine	1. SET QUICKDSP 2. SET SHARE 3. SET RESERVED

Chapter 13. SFS Tuning

This section discusses steps to help uncover SFS performance problems and the solutions to them.

Solving SFS Performance Problems

To solve a performance problem that you suspect SFS server processing might be causing, perform the following steps:

1. Confirm the problem.

   Decide whether there indeed is a performance problem. If you became aware of the problem because of user complaints, you might:

   a. Gather more information about the complaint.

   b. Look for confirmation from other users or from monitor data (if you regularly monitor performance).

   c. Attempt to reproduce the problem.

   If you suspect there is a performance problem because of trends you've noticed in monitor data, recheck the numbers and look at other monitor intervals to see if the problem occurs consistently.

   After you are satisfied that there is a problem, continue with the next step.

2. Isolate the problem.

   Narrow down the problem to a small number of possible causes by comparing the performance data that shows the problem to data gathered when performance was acceptable.

   a. Determine whether the problem is associated with one or more file pool servers or whether it is a general system problem. To make the determination, compare the percentage increase in average file pool request service time to the percentage increase in average response time. The average file pool request service time can be displayed using Performance Toolkit for z/VM. If the percentage increase in file pool service time is much greater than the percentage increase in overall response time, that server could be contributing to the problem. Otherwise, the problem is probably a general system problem.

      If a general problem is indicated, refer to Chapter 12, “Tuning Your System,” on page 103 for overall system tuning suggestions. Depending upon the nature of the problem, some tuning actions described in this section may also be helpful.

   b. If a file pool server problem is indicated, it is often helpful to next obtain the applicable file pool service time breakdowns. You might skip this step, however, if the problem you are investigating has well-defined, specific symptoms.

      Performance Toolkit for z/VM provides the following breakdown of average file pool request service time:

      - CPU time
      - Block I/O time
      - External security manager (ESM) time
      - DFSMS time
      - Lock wait time
      - Other time

      Average checkpoint time and a count of rollbacks because of deadlock are also provided.

      You will want to obtain this information for two time periods, when the system:

      - Was adequately performing
      - Shows the performance problem under investigation.

These service time breakdowns will show you where the server is spending most of its time and which activities are accounting for most of the increase relative to when performance was acceptable.

c. Use Table 5 on page 130 to identify possible causes of the symptoms you are observing. The problem description associated with each possible cause can help you confirm whether that problem applies to your situation or not.

Table 5. Index of Server Performance Problems

Symptom	Possible Causes	See ...
High CPU time	Excessive remote usage.	"Excessive Remote Usage" on page 132
	Data spaces not used.	"Data Spaces Not Used" on page 132
	Need more processing capacity.	"Need More Processing Capacity" on page 133
High block I/O time	Not enough catalog buffers.	"Not Enough Catalog Buffers" on page 133
	Not enough control minidisk buffers.	"Not Enough Control Minidisk Buffers" on page 133
	SFS file cache is too small.	"SFS Cache is Too Small" on page 134
	Minidisk caching not used.	"Minidisk Caching Not Used" on page 135
	Data spaces not used.	"Data Spaces Not Used" on page 132
	I/O activity not balanced	"I/O Activity Not Balanced" on page 135
	Catalogs are fragmented	"Catalogs Are Fragmented" on page 136
	Logs not on separate paths	"Logs Not on Separate Paths" on page 136
	Need more channels or control units.	"Need More Channels or Control Units" on page 136
	Need more DASD actuators.	"Need More DASD Actuators" on page 136
High ESM time	Excessive external security manager delays.	"Excessive External Security Manager Delays" on page 137
High DFSMS time	Excessive DFSMS delays.	"Excessive DFSMS Delays" on page 137
High lock wait time	Excessive logical unit of work holding time. Logs not on separate paths.	"Logs Not on Separate Paths" on page 136

Table 5. Index of Server Performance Problems (continued)		
Symptom	**Possible Causes**	**See ...**
High other time	Insufficient real agents	"Insufficient Real Agents" on page 138
	Too much server paging.	"Too Much Server Paging" on page 138
	Server code not in a saved segment.	"Server Code Not in a Saved Segment" on page 139
	Server priority is too low.	"Server Priority is Too Low" on page 139
	QUICKDSP not specified	"Server Utilization is Too High" on page 140
	Server utilization is too high.	"Server Utilization is Too High" on page 140
High system paging rate	Too many catalog buffers	"Too Many Catalog Buffers" on page 140
	SFS file cache is too large.	"SFS File Cache is Too Large" on page 141
	Server code not in a saved segment.	"Server Code Not in a Saved Segment" on page 139
	Excessive logical unit of work holding time.	"Users Not Running in XC Mode" on page 141
	Users not running in XC mode.	"Need More Real Storage" on page 141
	Need more real storage.	"Need More Real Storage" on page 141
High response times after system restart	ACCESS contention.	"ACCESS Contention" on page 142
Long checkpoint duration	Not enough control minidisk buffers.	"Not Enough Control Minidisk Buffers" on page 133
	Too many catalog buffers.	"Too Many Catalog Buffers" on page 140
Excessive rollbacks because of deadlock (See *z/VM: CMS File Pool Planning, Administration, and Operation*).	Logs not on separate paths.	"Logs Not on Separate Paths" on page 136
	Not enough catalog buffers.	"Not Enough Catalog Buffers" on page 133
	File pool capacity exceeded.	"File Pool Capacity Exceeded" on page 142

3. Take corrective action.

   In the previous step you identified one or more performance problems that might apply to your situation. Now review the suggested corrective actions indicated in Table 5 on page 130 and decide which actions (if any) to take. It may also be worthwhile to review the SFS performance guidelines provided in Chapter 4, "SFS Performance Guidelines," on page 61.

   For those problems having several possible corrective actions, it is often best to first try the actions that are easiest to implement and evaluate their effectiveness (see the next step).

4. Evaluate for effectiveness.

   Finally, determine whether the corrective actions taken actually produced the intended results. This is usually done by reviewing monitor data taken after the corrections have been made. Examine the key performance indicators to see if they are now in the acceptable range. If performance is still not acceptable, first make sure that corrective actions were properly made. You might then decide to look for additional improvements (go back to step 3).

Potential SFS Performance Problems

The following is a list of SFS file pool server related performance problems that might be encountered. See Table 5 on page 130 for an index into this list (by symptom).

Excessive Remote Usage

Problem description:

System processor overhead is higher than usual because of excessive file usage by users on other processors.

This problem is indicated if:

1. The file pool is heavily used; and,
2. A high percentage (more than 20%) of all server requests come from remote users. You can compute this percentage from the QUERY FILEPOOL COUNTER command output (or from the equivalent monitor data). Divide the Remote File Pool Requests value by Total File Pool Requests value.

Possible corrective actions:

- Move each user to the system that contains the file pool he or she uses the most often. You can determine the degree of usage by examining the SFS file pool server accounting records (provided the server is running with the ACCOUNT start-up parameter).
- Encourage remote users to duplicate the files they use frequently.

Data Spaces Not Used

Problem description:

VM data spaces are not being used for one or more directory control directories for one of the following reasons:

1. The directory has not been made eligible to use data spaces even though performance would benefit. Directories that are seldom updated and shared read-only by multiple users should experience greatly improved performance with data spaces. The QUERY ACCESSORS command can be used to find out how many users are accessing each of the file pool server's directory control directories.
2. The directory is eligible to use data spaces but SFS is not able to do so. If this is the case, you will have received message DMS2020I at the SFS operator's console.

Possible corrective actions:

- In the first case, use the DATASPACE command to make suitable directories eligible for use with data spaces. Note: Whenever you increase the number of eligible directories, also decide whether to increase the maximum number of data spaces that can be used by the server. This is controlled by the MAXNUMBER setting on the XCONFIG ADDRSPACE control statement in the server's CP directory entry.
- In the second case, review the messages to determine why data spaces are not being used and take corrective action.

  The most likely problem is that the maximum number of data spaces allocated to the server has been reached. If this is the case, you may choose simply to increase the maximum. You do this by

increasing the MAXNUMBER value on the XCONFIG ADDRSPACE control statement in the server's CP directory entry.

Before doing so, however, you may wish to see if there are any directories that are using an excessive number of data spaces. This happens when multiple versions of the directory get created as a result of a series of changes being made to the directory at the same time that users are accessing it. Each such version requires a separate data space. To check for this condition, enter the QUERY ACCESSORS command with the DATASPACE option. If the results show that there are one or more directories with an excessive number of levels, there are several actions you could take:

- Contact the person responsible for updating that z/VM system directory and ask that changes be grouped together and made, when practical, during times of low directory usage. This will minimize the number of directory levels that get created.
- Request users to reaccess the directory. When users reaccess, the server lets them see the most current level. The server discards an older level when all users have released that level of the directory.
- Use the DATASPACE RELEASE command to make the directory ineligible for use with data spaces. This may be the best course of action if the directory is updated frequently and neither of the actions listed above are practical. In this case, you should also consider changing the directory to be a file control directory, using the DIRATTR command.

Need More Processing Capacity

Problem description:
Your system does not have enough processing capacity to handle the work load.

Possible corrective actions:
Consider acquiring additional system processing capacity.

Not Enough Catalog Buffers

Problem description:
The server cannot use the file pool catalogs efficiently because there are not enough catalog block buffers. Sufficient real storage exists to support additional buffers.

To determine whether more buffers would help, use the QUERY FILEPOOL COUNTER output (or equivalent monitor data). Divide the Catalog Blocks Read value by the Total DASD Block Transfers value. The result should normally be less than 0.25. A higher value than this, coupled with a low system paging rate, suggests that overall performance would improve if more catalog buffers were specified.

Possible corrective actions:

- Make sure that the USERS start-up parameter has been set appropriately. If you do not specify the CATBUFFERS start-up parameter, an increase in USERS will automatically result in an appropriate increase in the number of catalog buffers. (See the descriptions of USERS and CATBUFFERS in *z/VM: CMS File Pool Planning, Administration, and Operation*.)
- Increase the CATBUFFERS file pool start-up parameter value.
- If real storage is plentiful (as indicated by a low system paging rate), you should also consider increasing the SFS file cache size. See "SFS Cache is Too Small" on page 134.

Not Enough Control Minidisk Buffers

Problem description:
The server cannot do checkpoint and other processing efficiently because there are not enough control minidisk buffers.

To verify whether checkpoint processing is taking a long time, divide the Checkpoint Time (ms) value by the Checkpoints Taken value. This should normally be less than four seconds.

Use QUERY FILEPOOL COUNTER command output (or the equivalent monitor data) to determine whether there are enough control minidisk buffers. Divide the Control Minidisk Blocks Read value by the Total File Pool Requests value. A result greater than 0.005 indicates that there are not enough control minidisk buffers.

Possible corrective actions:

- Make sure that the USERS start-up parameter has been set appropriately. If you let the CTLBUFFERS start-up parameter default, an increase in USERS will automatically result in a proportional increase in the number of control minidisk buffers. (See the descriptions of USERS and CTLBUFFERS in *z/VM: CMS File Pool Planning, Administration, and Operation*.)
- If USERS is properly set, increase the number of control minidisk buffers by specifying a CTLBUFFERS setting that is larger than the value your file pool is currently using.
- If real storage is plentiful (as indicated by a low system paging rate), you should also consider increasing the SFS file cache size and the number of catalog buffers. See "SFS Cache is Too Small" on page 134 and "Not Enough Catalog Buffers" on page 133.

SFS Cache is Too Small

Problem description:

The CMS SFS file cache size is smaller than its optimal value. This problem may be present if the system paging rate is low and the CMS SFS file cache size is less than the maximum (96 KB).

Background:

When a file is read sequentially, SFS reads as many blocks at a time as will fit into the SFS file cache. When a file is written sequentially, completed blocks are accumulated until they fill the SFS file cache and are then written together.

There is a separate SFS file cache for each sequentially opened file. SFS file cache size is specified on a system basis when the CMS nucleus is built. The optimal SFS file cache size depends upon how much real storage is available, as indicated by the system paging rate.

Although the maximum cache size is 96 KB, for any given file, the cache size will be limited by the following:

- The size of the file, rounded up to the next multiple of 4 KB. For example, if a file consists of 60 fixed length records, each 80 bytes long, its cache size will be limited to 8 KB.
- The maximum cache size supported by the server machine. This is a consideration when, for example, you are doing a remote request to a back-level server machine. Server machines that run on VM/SP CMS 1.6 support transmission of up to 28 KB of data per read or write request. When accessing a file in such a file pool, CMS will not use a cache larger than 28 KB.

Possible corrective actions:

- Increase the CMS SFS file cache size. The default CMS SFS cache size is 20 KB. You can check the coding of the DEFNUC macro in your CMS nucleus generation profile (DMSNGP for ESA/390 CMS, DMSZNGP for z/Architecture CMS) to determine the current setting for the cache buffers. To change the CMS SFS file cache size, you need to update the BUFFSIZ parameter in the DEFNUC macro in DMSNGP ASSEMBLE or DMSZNGP ASSEMBLE. Then assemble the file and rebuild the CMS nucleus. See *z/VM: CMS Planning and Administration* for information about DMSNGP, DMSZNGP, and DEFNUC. See *z/VM: Service Guide* for instructions on rebuilding the CMS nucleus.

  Note that increasing your system CMS SFS cache size to 96 KB may have little effect if most accessed files are small or if the Shared File System data resides in file pools that run on VM/SP CMS 1.6.

  In fact, indiscriminately increasing the maximum CMS SFS cache size could potentially increase the virtual storage demands of individual user machines. This could lead to *insufficient virtual storage* errors when running applications that access large files.

- If real storage is plentiful (as indicated by a low system paging rate), you should also consider increasing the number of catalog buffers and the number of control minidisk buffers. See "Not Enough Catalog Buffers" on page 133 and "Not Enough Control Minidisk Buffers" on page 133.

Minidisk Caching Not Used

Problem description:
Minidisk caching is not being used for this file pool's minidisks. This results in unnecessarily high block I/O times.

Possible corrective actions:
Run with minidisk caching. See "CP Tuning Parameters" on page 61 for more information.

I/O Activity Not Balanced

Problem description:
One (or more) of the channels or devices in the I/O subsystem is experiencing a high level of contention because of an excessive I/O rate. Other components of the I/O subsystem are under utilized.

Possible corrective actions:

- In the case of channel contention, redistribute the DASD volumes over the available channels to balance the load.
- In the case of device contention, consider doing one or more of the following:
  - Move one or more (SFS or non-SFS) minidisks from high usage devices to lower usage devices.
  - Move some users to a storage group on lower-usage devices. See the FILESERV MOVEUSER command in *z/VM: CMS File Pool Planning, Administration, and Operation*.
  - Rearrange the minidisks on the critical volume so as to minimize seek time.
  - Spread the storage group across more minidisks, some of which are on lower usage devices. This is not an immediate solution, but will reduce the problem over time.
- Decrease the overall amount of catalog I/O by increasing the number of catalog buffers. This will help if all or part of storage group 1 is on the channel or device that is experiencing the problem. To increase the number of catalog buffers, increase the CATBUFFERS start-up parameter value. The CATBUFFERS start-up parameter is described in *z/VM: CMS File Pool Planning, Administration, and Operation*.
- Decrease the overall amount of file block I/Os by increasing the CMS SFS file cache size. This will help if all or part of one of the user storage groups is on the channel or device that is experiencing the problem. To change the CMS SFS file cache size, you need to update the BUFFSIZ parameter in the DEFNUC macro, which is in DMSNGP ASSEMBLE (DMSZNGP ASSEMBLE for z/Architecture CMS). Then assemble the file and rebuild the CMS nucleus. See *z/VM: CMS Planning and Administration* for information about DMSNGP, DMSZNGP, and DEFNUC. See *z/VM: Service Guide* for instructions on rebuilding the CMS nucleus.
- Decrease the overall I/O load by using minidisk caching. Minidisk caching can be quite beneficial for SFS file pool servers.

  Be sure to make the SFS log minidisks ineligible for minidisk caching. This can be done by specifying NOMDC on the MINIOPT z/VM system directory control statement for those minidisks. The SFS logs do not benefit from minidisk caching because the I/O activity to them is almost entirely writes. Caching the SFS logs degrades system performance.

  The SFS control minidisks do not benefit from minidisk caching because internal buffering and the characteristics of I/O done to these minidisks. The SFS control minidisks should be disabled for minidisk cache.

  All other SFS file pool minidisks are eligible for, and benefit from, minidisk caching so they should be left eligible (which is the default). For more information on minidisk caching, see *z/VM: CMS Planning and Administration*.

- Reduce the impact of the server's read I/O activity by the use of a cached control unit.
- Reduce the impact of the server's write I/O activity by the use of the IBM DASD fast write function.

  Note: Using the IBM DASD fast write function is especially beneficial to the SFS logs because nearly all I/Os to them are writes.
- Put read-only files into directory control directories that are set up to use data spaces.

Catalogs Are Fragmented

Problem description:
The file pool is experiencing an excessive number of catalog I/Os because of index fragmentation and scattered catalog records.

You should suspect this problem if the average number of catalog blocks read per file pool request has gradually gotten higher over time. You can compute this from QUERY FILEPOOL COUNTER command output (or the equivalent monitor data). Divide the Catalog Blocks Read value by the Total File Pool requests value.

Possible corrective actions:
Reorganize the catalogs as described in the FILESERV REORG command section of *z/VM: CMS File Pool Planning, Administration, and Operation*.

Logs Not on Separate Paths

Problem description:
The two SFS logs have not been placed on separate I/O paths (different channels, control units, and devices). This reduces the degree to which the I/O request pairs made to these logs can be overlapped with each other. This will tend to increase I/O wait time, SFS lock wait time, and the likelihood of SFS rollbacks because of deadlock.

Possible corrective actions:

- Move one of the logs such that the logs are on different I/O paths. See *z/VM: CMS File Pool Planning, Administration, and Operation* for information on how to move an SFS log minidisk.
- I/O wait time, SFS lock wait time, and SFS rollbacks because of deadlock can be further reduced if each SFS log minidisk is placed on a DASD volume that experiences little or no other I/O activity. If this is done, seek time associated with SFS log I/O activity will be negligible.

  You could also place the SFS log minidisks on DASD in IBM DASD subsystems that support the DASD fast write function.

Need More Channels or Control Units

Problem description:
Your system does not have enough channels or control units to handle the work load.

Possible corrective actions:
Consider acquiring additional channel or control unit capacity.

Need More DASD Actuators

Problem description:
Your system does not have enough DASD actuators to handle the work load.

Possible corrective actions:
Consider acquiring additional DASD actuators.

Excessive External Security Manager Delays

Problem description:

A high percentage of the time spent servicing SFS requests is being spent in the External Security Manager (ESM) that you are using in conjunction with SFS. This delay may be in the ESM exits that the SFS server calls or in other server machines that the ESM exits may call.

You can use the output from the QUERY FILEPOOL COUNTER command to determine how much of this delay is in the exits themselves. This delay is given by Security Manager Exit Time. The total ESM delay (including the exits and any services they call) is given by External Security Manager Exit Time.

Note: The SFS server drives the ESM exits only when it has been started with the ESECURITY parameter specified in DMSPARMS.

Possible corrective actions:

Refer to performance guidance that is provided in your external security manager's documentation.

Excessive DFSMS Delays

Problem description:

A high percentage of the time spent servicing SFS requests is being spent in DFSMS/VM. This includes the time spent within the DFSMS exits themselves as well as the time they spend waiting for the completion of any requests they make to the DFSMS master machine and DFSMS server machines.

You can use the output from the QUERY FILEPOOL COUNTER command to determine how much of this time results from performing file recalls versus how much of this time results from calling all other DFSMS exits. Recall time is given by Recall DFSMS File Exit Time while all other DFSMS exit time is given by Other DFSMS Exit Time.

Note: The SFS server drives the DFSMS exits only when it has been started with the DFSMS parameter specified in DMSPARMS.

Possible corrective actions:

See the performance problem determination guidance provided in *z/VM: DFSMS/VM Storage Administration*.

Excessive Logical Unit of Work Holding Time

Problem description:

Some server logical units of work are being held for long periods of time without being used to process file pool requests. This typically occurs when applications remain in a logical unit of work over a terminal read (during user think time), but can also occur when applications remain in a logical unit of work while they do large amounts of unrelated processing.

To determine whether LUW holding time is excessive, use the QUERY FILEPOOL COUNTER command output (or equivalent monitor data). Divide the Agent Holding Time (ms) value by the File Pool Request Service Time (ms) value. A ratio less than 10 should be considered typical.

Background:

Certain resources are associated with a logical unit of work and are held until the logical unit of work is committed or rolled back. These resources include an agent (a server's internal representation of a user), some server storage, and implicit locks on objects in the file pool. SFS has been designed to minimize the adverse impact of holding these resources for long periods of time, but it is still best to commit work as soon as is practical.

Possible corrective actions:

- Increase the number of real agents by increasing the USERS file pool start-up parameter. This is a simple course of action, and may be advisable if the only symptom is that the file pool tends to run out of agents.
- Investigate the major SFS applications that are run with this file pool to see if there are any places where they unnecessarily remain in a logical unit of work for long periods of time. Especially look at

all points at which the applications read from the terminal to make sure that they are not in a logical unit of work at that time. Make all feasible corrections.

Insufficient Real Agents

Problem description:

File pool usage is so high that sometimes the demand for real agents exceeds the total number available. When this happens, SFS file pool requests sometimes have to wait for a real agent to become available.

This problem is indicated if:

1. The Active Agents Highest Value in the QUERY FILEPOOL AGENT command output (or equivalent monitor data) sometimes reaches the Total Number of Agents value which the server has created. See *z/VM: CMS File Pool Planning, Administration, and Operation* for more information.
2. Agent holding time is not excessive.

To determine whether agent holding time is excessive, use the QUERY FILEPOOL COUNTER command output. Divide the Agent Holding Time (ms) by the File Pool Request Service Time (ms).

This ratio will generally be 10 or less. Anything greater is an indication of excessive agent holding times, and you may wish to investigate further. See "Excessive Logical Unit of Work Holding Time" on page 137.

Possible corrective actions:

Increase the USERS file pool start-up parameter.

Too Much Server Paging

Problem description:

Excessive paging in the server virtual machine is resulting in increased response times.

Suspect this problem under the following combination of circumstances:

- High system paging rate.
- Significant paging in the server machine.
- High number of active agents in the server machine.
- Requests that do not use the file pool have adequate response times.

  This problem is directly indicated if the server utilization exceeds 50% and page fault resolution time is a large contributor to that overall server utilization. See "Server Utilization is Too High" on page 140.

Background:

When a page fault occurs in a user machine, only that user waits until the page fault is resolved. When a page fault occurs in a server machine, all users currently being processed by that server machine must wait until that page fault is resolved. Consequently, it is important to minimize the amount of paging that occurs in the server machine, especially when that server is supporting a large number of users.

Possible corrective actions:

- Reserve pages for exclusive use by the server machine. Use the CP SET RESERVED command, which is described in *z/VM: CP Commands and Utilities Reference*. The number of pages you reserve should correspond to or be somewhat greater than the server's average working set size during peak usage conditions. The INDICATE USER command can be used to display the working set size.
- Make sure that the server machine has been given a high dispatching priority by specifying SHARE REL 1500 in the z/VM system directory.

- Put the server code in a physical saved segment. Then specify the SAVESEGID start-up parameter to use that segment. See *z/VM: Installation Guide* for more about creating a physical saved segment for SFS file pool servers. (During installation, a physical saved segment named CMSFILES is automatically loaded for the SFS and CRR server code.)

  See *z/VM: CMS File Pool Planning, Administration, and Operation* for more about the SAVESEGID start-up parameter.
- Create another file pool and move some of the users to it.
- Take actions to reduce the system's overall real storage requirements.

Server Code Not in a Saved Segment

Problem description:
The system has two or more SFS and CRR servers active at the same time but the server code is not being run in a saved segment. This results in additional paging because multiple copies of the same code are being kept in storage.

Possible corrective actions:

- Put the server code in a physical saved segment. Then specify the SAVESEGID start-up parameter to use that segment. See *z/VM: Installation Guide* for more about creating a physical saved segment for SFS file pool servers and CRR recovery servers. (During installation, a physical saved segment named CMSFILES is automatically loaded for the SFS and CRR server code.)

  See *z/VM: CMS File Pool Planning, Administration, and Operation* for more about the SAVESEGID start-up parameter.

Server Priority is Too Low

Problem description:
The priority of the server virtual machine is too low.

Suspect this problem if all of the following are true:

- Processor utilization is high.
- The file pool is supporting a large number of users.
- A dispatching priority for the server has not been specified or has been set to a low priority.
- Requests that do not use the file pool have adequate response times.

Possible corrective actions:
Using your local operating procedures, specify SHARE REL 1500 in the z/VM system directory. For more information on the SHARE control statement, see *z/VM: CP Planning and Administration*.

You can also enter the SET SHARE command. For more information on the SET SHARE command, see *z/VM: CP Commands and Utilities Reference*.

QUICKDSP Not Specified

Problem description:
QUICKDSP has not been specified for the server virtual machine. As a result, the server machine can be placed on the eligible list, which can result in erratic, high server service times in real-storage-constrained environments.

Possible corrective actions:
Using your local operating procedures, update the server's z/VM system directory to add the QUICKDSP operand to the OPTION control statement. For more information on the QUICKDSP operand, see *z/VM: CP Planning and Administration*.

You can also specify QUICKDSP by issuing the SET QUICKDSP *userid* ON command. For more information on the SET QUICKDSP command, see *z/VM: CP Commands and Utilities Reference*.

Server Utilization is Too High

Problem description:

Delays are occurring because of excessive queueing on the server.

A SFS server machine is a performance resource in its own right. As such, excessive queueing for its services will occur when its utilization gets too high. The higher the server utilization, the more significant the queueing delays will be. As a practical guide, there is no significant problem if the total server utilization is below 50%.

Server utilization consists of that server's use of a processor plus all the time it spends waiting for serializing events that prevent it from using a processor. An example of such a serializing event is handling a page fault that occurs in the server.

There are four components to SFS server utilization: CPU utilization, page fault resolution time, checkpoint time, and QSAM time. Performance Toolkit for z/VM shows total server utilization and how much each of these four components contributes to that total. Therefore, if server utilization is identified as a contributing problem (that is, it exceeds 50%), you can use this breakdown to find the main cause.

Possible corrective actions:

- If most of the server utilization is because of CPU usage, a high level of server loading is usually indicated. The SFS server can only run on one processor at a time. A server CPU utilization of 100% would mean that the server is using all of one processor in the processing complex. Well before you get to that point, you should discontinue enrolling additional users in this server's file pool. If the server's CPU utilization is already excessively high, you may even wish to transfer some users from this file pool to another file pool that is less heavily used. See *z/VM: CMS File Pool Planning, Administration, and Operation* for instructions on how to do this.
- If page fault resolution time predominates, treat this as a server paging problem. See "Too Much Server Paging" on page 138 for suggestions.
- If checkpoint utilization predominates, consider actions that would reduce checkpoint time. See "Not Enough Control Minidisk Buffers" on page 133 and "Too Many Catalog Buffers" on page 140 for suggestions.
- If server utilization because of QSAM time is a major contributor, the problem is likely to be that the server is doing control data backups to tape or minidisk. If this is the case, you can remedy the situation by scheduling these backups to occur at times of low file pool usage. Alternatively, you could choose to do these backups to another SFS file pool. This resolves the problem because SFS will then do the control backups using asynchronous requests to the backup file pool rather than synchronous (and therefore serializing) QSAM I/O requests to tape or minidisk.

Too Many Catalog Buffers

Problem description:

1. The file pool is using more catalog buffers than can be efficiently backed by real storage.

   In the QUERY FILEPOOL COUNTER command output (or equivalent monitor data), the Catalog Blocks Read value divided by the Total DASD Block Transfers value should normally be greater than 0.20. A lower value than this, coupled with a high system paging rate, suggests that overall performance may improve if fewer buffers were specified.
2. Checkpoint duration has become excessive because large numbers of catalog buffers have been modified. Each such buffer must be written to disk as part of checkpoint processing.

   To verify whether checkpoint processing is taking a long time, divide the Checkpoint Time (ms) value by the Checkpoints Taken value. This should normally be less than four seconds.

Possible corrective actions:

- Make sure that the USERS start-up parameter has been set appropriately. If you let the CATBUFFERS start-up parameter default, a decrease in USERS will automatically result in an appropriate decrease in the number of catalog buffers. (See the descriptions of the USERS and CATBUFFERS start-up parameters in *z/VM: CMS File Pool Planning, Administration, and Operation*.)
- Decrease the CATBUFFERS start-up parameter value.
- If real storage is in short supply (as indicated by a high system paging rate), you should also consider decreasing the SFS file cache size. See "SFS File Cache is Too Large" on page 141.

SFS File Cache is Too Large

Problem description:

The CMS SFS file cache size is larger than its optimal value. This problem may be present if the system paging rate is high and the SFS file cache size is greater than the minimum (4 KB).

Background:

See "SFS Cache is Too Small" on page 134.

Possible corrective actions:

- Decrease the CMS SFS file cache size. The default SFS file cache size is 20 KB. You can check the coding of the DEFNUC macro in your CMS nucleus generation profile (DMSNGP for ESA/390 CMS, DMSZNGP for z/Architecture CMS) to determine the current setting for the cache buffers. To change the SFS file cache size, you need to update the BUFFSIZ parameter in the DEFNUC macro in DMSNGP ASSEMBLE or DMSZNGP ASSEMBLE. Then assemble the file and rebuild the CMS nucleus. See *z/VM: CMS Planning and Administration* for information about DMSNGP, DMSZNGP, and DEFNUC. See *z/VM: Service Guide* for instructions on rebuilding the CMS nucleus.

  Note that if the size of the SFS file cache is very large but most accessed files are small, reducing the size of the SFS file cache may have little, if any, effect.
- If real storage is in short supply (as indicated by a high system paging rate), you should also consider decreasing the number of catalog buffers. See "Too Many Catalog Buffers" on page 140.

Users Not Running in XC Mode

Problem description:

The full performance benefits of using data spaces with directory control directories are not being realized because all or some of the users who are accessing those directories are not running in XC mode.

Performance is better for XC mode users because 1) they use a shared copy of the file status table (FST) that resides in the data space instead of a private copy and 2) CMS can read files directly from the data space rather than having to read them by going through CP.

Possible corrective actions:

Encourage your CMS users to run in XC mode. CMS users that run XA or ESA mode should be able to run in XC mode without any problems. However, neither z/CMS nor guest operating systems like Linux, z/VM, z/OS, and z/VSE can run in XC mode.

Note: Users with 370-mode applications can use the CP 370 Accommodation Facility's SET 370ACCOM ON command or the SET GEN370 OFF command to execute in either XA, ESA, or XC mode.

Need More Real Storage

Problem description:

Your system does not have enough real storage to handle the work load.

Possible corrective actions:
Consider acquiring additional real storage.

ACCESS Contention

Problem description:
This problem is indicated by long response times during the execution of users' PROFILE EXECs immediately following system restart. It can occur when large numbers of users log on at the same time and those users access directories that contain large numbers of files.

Possible corrective actions:

- Put read-only files that are to be shared among large numbers of users into directory control directories that are set up to reside in data spaces. See *z/VM: CMS File Pool Planning, Administration, and Operation* for a discussion on how to do this.
- Identify all of the directories and minidisks that have a low frequency of change and are accessed read-only by large numbers of users. Consider implementing all or some of these files on minidisks with shared FSTs. See *z/VM: CMS Planning and Administration* for information on how to set up shared FSTs using the SAVEFD command.
- Encourage users to limit their initial search order (established by their PROFILE EXEC) to heavily-used directories and to access all other directories on an as-needed basis.

File Pool Capacity Exceeded

Problem description:
The rate of update activity the file pool is being asked to handle is too high, resulting in an increased likelihood of rollbacks because of deadlock. See *z/VM: CMS File Pool Planning, Administration, and Operation* for a discussion of deadlocks.

Possible corrective actions:

- Do not enroll any additional users in this file pool. This is the first step you should take.
- For a file pool experiencing a given update rate, certain tuning actions may reduce the likelihood of involuntary rollbacks. See “Logs Not on Separate Paths” on page 136 and “Not Enough Catalog Buffers” on page 133.
- Offload some of the users currently enrolled in this file pool to a new file pool or another existing file pool that has lower usage. See *z/VM: CMS File Pool Planning, Administration, and Operation* for the procedure for doing this.

Chapter 14. CRR Tuning

This section discusses steps to help uncover CRR performance problems and the solutions to them.

Solving CRR Performance Problems

To solve a performance problem that you suspect server processing might be causing, do the following:

1. Confirm the problem.

   Decide whether there indeed is a performance problem. If you became aware of the problem because of user complaints, you might:

   a. Gather more information about the complaint.

   b. Look for confirmation from other users or from monitor data (if you regularly monitor performance).

   c. Attempt to reproduce the problem.

   If you suspect there is a performance problem because of trends you've noticed in monitor data, recheck the numbers and look at other monitor intervals to see if the problem occurs consistently.

   After you are satisfied that there is a problem, continue with the next step.

2. Isolate the problem.

   Narrow down the problem to a small number of possible causes by comparing the performance data that shows the problem to data gathered when performance was acceptable.

   a. Determine whether the problem is associated with the CRR recovery server or whether it is a general system problem. To make the determination, compare the percentage increase in average file pool request service time to the percentage increase in average response time. Average file pool request service time is displayed in the FCX116, Shared File System Server Screen - SFS. It can also be calculated from the QUERY FILEPOOL COUNTER output for the CRR recovery server by dividing File Pool Request Service Time by Total File Pool Requests. If the percentage increase in file pool service time is much greater than the percentage increase in overall response time, the CRR server could be contributing to the problem. Otherwise, the problem is probably a general system problem.

      If a general problem is indicated, see Chapter 12, “Tuning Your System,” on page 103 for overall system tuning suggestions. Depending upon the nature of the problem, some tuning actions described in this section may also be helpful.

   b. If a CRR recovery server problem is indicated, it is often helpful to next obtain the applicable file pool service time breakdowns. You might skip this step, however, if the problem you are investigating has well-defined, specific symptoms.

      Performance Toolkit for z/VM provides the following CRR-related breakdown of average file pool request service time:

      - CPU time
      - Block I/O time
      - Other time.

      You will want to obtain this breakdown information for two time periods, when the system:

      - Was adequately performing
      - Shows the performance problem under investigation.

      These service time breakdowns will show you where the CRR recovery server is spending most of its time and which activities are accounting for most of the increase relative to when performance was acceptable.

c. Use Table 6 on page 144 to identify possible causes of the symptoms you are experiencing. The problem description associated with each possible cause can help you confirm whether that problem applies to your situation or not.

Table 6. Index of CRR Recovery Server Performance Problems

Symptom	Possible Causes	Location
High block I/O time	Minidisk caching being done for the CRR logs.	"Minidisk Caching Being Done for the CRR Logs" on page 144
	Logs not on separate paths	"Logs Not on Separate Paths" on page 145
High other time.	Insufficient real agents.	"Insufficient Real Agents" on page 145
	Too much server paging.	"Too Much Server Paging" on page 145
	Server code not in a saved segment.	"Server Code Not in a Saved Segment" on page 146
	Server priority is too low.	"Server Priority is Too Low" on page 146
	QUICKDSP not specified.	"QUICKDSP Not Specified" on page 146
High system paging rate.	Server code not in a saved segment.	"Server Code Not in a Saved Segment" on page 146

3. Take corrective action.

   In the previous step you identified one or more performance problems that might apply to your situation. Now review the suggested corrective actions at the location indicated by Table 6 on page 144 and decide which actions (if any) to take. It may also be worthwhile to review the CRR performance guidelines provided in Chapter 5, "CRR Performance Guidelines," on page 65.

   For those problems having several possible corrective actions, it is often best to first try the actions that are easiest to implement and evaluate their effectiveness (see the next step).

4. Evaluate for effectiveness.

   Finally, determine whether the corrective actions taken actually produced the intended results. This is usually done by reviewing monitor data taken after the corrections have been made. Examine the key performance indicators to see if they are now in the acceptable range. If performance is still not acceptable, first make sure that corrective actions were properly made. You might then decide to look for additional improvements (go back to step 3).

Potential CRR Performance Problems

The following is a list of CRR recovery server related performance problems that might be encountered. See Table 6 on page 144 for an index into this list (by symptom).

Minidisk Caching Being Done for the CRR Logs

Problem description:

Minidisk caching is being performed for the CRR logs, resulting in inefficient use of the cache. The CRR logs do not benefit from minidisk caching because the I/O activity to them is almost entirely writes. Caching the CRR logs degrades system performance.

Possible corrective actions:

Make the CRR log minidisks ineligible for minidisk caching. This can be done by specifying NOMDC on the MINIOPT z/VM system directory control statement for those minidisks.

All other CRR file pool minidisks are eligible for, but do not benefit from, minidisk caching so they should be made ineligible for caching. For more information on minidisk caching, see *z/VM: CMS Planning and Administration*.

Logs Not on Separate Paths

Problem description:

The two CRR logs have not been placed on separate I/O paths (different channels, control units, and devices). This reduces the degree to which the I/O request pairs made to these logs can be overlapped with each other, resulting in increased I/O wait time.

Possible corrective actions:

- Move one of the CRR logs such that the logs are on different I/O paths. See *z/VM: CMS File Pool Planning, Administration, and Operation* for information on how to move a log minidisk.
- I/O wait time can be further reduced if each CRR log minidisk is placed on a DASD volume that experiences little or no other I/O activity. If this is done, seek time associated with CRR log I/O activity will be negligible.

  You could also place the CRR log minidisks on DASD in an IBM DASD subsystem that supports the DASD fast write function.

Insufficient Real Agents

Problem description:

CRR recovery server usage is so high that sometimes the demand for real agents exceeds the total number available. When this happens, CRR logging requests sometimes have to wait for a real agent to become available.

This problem is indicated if the Active Agents Highest Value in the QUERY FILEPOOL AGENT command output (or equivalent monitor data) sometimes reaches the Total Number of Agents value which the server has created.

Possible corrective actions:

Increase the USERS server startup parameter.

Too Much Server Paging

Problem description:

Excessive paging in the CRR server virtual machine is resulting in increased response times.

Suspect this problem under the following combination of circumstances:

- High system paging rate.
- Significant paging in the server machine.
- High number of active agents in the server machine.

Background:

When a page fault occurs in a user machine, only that user waits until the page fault is resolved. When a page fault occurs in a server machine, all users currently being processed by that server machine must wait until that page fault is resolved. Consequently, it is important to minimize the amount of paging that occurs in the server machine, especially when that server is supporting a large number of users.

Possible corrective actions:

- Reserve pages for exclusive use by the server machine. Use the CP SET RESERVED command, which is described in *z/VM: CP Commands and Utilities Reference*. The number of pages you reserve should correspond to or be somewhat greater than the server's average working set size during peak usage conditions. The INDICATE USER command can be used to display the working set size.
- Make sure that the server machine has been given a high dispatching priority by specifying SHARE REL 1500 in the z/VM system directory.
- Put the SFS and CRR server code in a physical saved segment. Then specify the SAVESEGID startup parameter to use that segment.

  See *z/VM: Installation Guide* for more about creating a physical saved segment for the SFS and CRR recovery server code. See *z/VM: CMS File Pool Planning, Administration, and Operation* for more about the SAVESEGID startup parameter.
- Take actions to reduce the system's overall real storage requirements.

Server Code Not in a Saved Segment

Problem description:

The system has two or more servers active at the same time but the server code is not being run in a saved segment. This results in additional paging because multiple copies of the same code are being kept in storage.

Possible corrective actions:

- Put the SFS and CRR server code in a physical saved segment. Then specify the SAVESEGID startup parameter to use that segment.

  See *z/VM: Installation Guide* for more about creating a physical saved segment for the SFS and CRR recovery server code. See *z/VM: CMS File Pool Planning, Administration, and Operation* for more about the SAVESEGID startup parameter.

Server Priority is Too Low

Problem description:

The priority of the CRR recovery server virtual machine is too low.

Suspect this problem if all of the following are true:

- Processor utilization is high.
- The CRR recovery server is supporting a large number of users.
- A dispatching priority for the server has not been specified or has been set to a low priority.

Possible corrective actions:

Using your local operating procedures, specify SHARE REL 1500 in the z/VM system directory. For more information on the SHARE control statement, see *z/VM: CP Planning and Administration*.

You can also enter the SET SHARE command. For more information on the SET SHARE command, see *z/VM: CP Commands and Utilities Reference*.

QUICKDSP Not Specified

Problem description:

QUICKDSP has not been specified for the CRR recovery server virtual machine. As a result, the server machine can be placed on the eligible list, which can result in erratic, high CRR service times in real-storage-constrained environments.

Possible corrective actions:

Using your local operating procedures, update the CRR recovery server's z/VM system directory to add the QUICKDSP operand to the OPTION control statement. For more information on the QUICKDSP operand, see *z/VM: CP Planning and Administration*.

You can also specify QUICKDSP by issuing the SET QUICKDSP *userid* ON command. For more information on the SET QUICKDSP command, see *z/VM: CP Commands and Utilities Reference*.

Chapter 15. AVS Tuning Parameters

The AGWTUN ASSEMBLE file, which is a nonexecutable source module, contains APPC/VTAM Support (AVS) tuning parameters. This section describes these tuning parameters, recommendations and restrictions for their use, and the default values supplied by IBM. The tuning parameters are described in the order in which they are grouped in the module:

- Pause parameters
- Transformation control parameters
- Miscellaneous parameters.

The default tuning parameter values may be modified by editing the AGWTUN ASSEMBLE file. For the changes to take effect, you must reassemble the AGWTUN file and link the AVS load module again with the resulting AGWTUN TEXT file. However, unless you have identified performance problems within AVS, changing the default values of the parameters in AGWTUN ASSEMBLE is **not** recommended.

Note: Do not delete any of the declared constants in the module, rearrange their order, or add new constants. AVS initialization logic is completely dependent on the number and order of appearance of these constants. Simply change the values of the constants within recommended limits. Doing otherwise may cause abnormal execution of the AVS code and produce unpredictable results.

Pause Parameters

AVS consists of several subtasks that process events, such as commands, interrupts, and the generation of accounting records. The scheduling process for these events is nonpreemptive. When an AVS subtask receives control, it can continue its processing until it gives control to another AVS subtask. An AVS subtask always gives up control if it does not have work to perform.

Some AVS subtasks also give up control after processing a certain number of events. This prevents AVS from processing one class of events to the exclusion of all others. The pause parameters, which are described in Table 7 on page 149, set the values for these events. A pause parameter affects AVS performance only when the number of events queued for a subtask exceeds the value specified on that parameter.

Pause parameters provide a balance between the resources AVS uses to pass control to each subtask and the number of times an individual subtask receives control. If the parameter values are lower than the defaults provided, each subtask will receive control more often. This ensures that AVS can frequently process all classes of events. However, there is a corresponding increase in the amount of resources that AVS must use to pass control to the subtasks.

Conversely, if the parameter values are greater than the default values, processing time for some events may increase because the individual subtasks do not receive control as frequently. However, AVS uses fewer resources to pass control to each subtask.

Table 7. AVS Pause Parameters

Label	Default	Meaning
CMDPAUSE	20	Number of AVS commands, issued from the AVS console, that are processed before checking for other work.
VMSPAUSE	20	Number of APPC/VM functions or interrupts that AVS processes before checking for other work. Events such as receiving data sent by a local user and sending it through the gateway are limited by this parameter.

Table 7. AVS Pause Parameters (continued)

Label	Default	Meaning
VTSPAUSE	20	Number of VTAM commands or asynchronous events that AVS processes before checking for other work. These events include: accepting inbound allocates to a local resource from a remote user in the SNA network, and receiving data sent from a remote user in the SNA network.
IUCPAUSE	20	Number of IUCV interrupts from the *IDENT system service that AVS processes before checking for other work. Events in this class relate to gateway activation and deactivation.
ACTPAUSE	10	Number of accounting records that AVS creates before checking for other work to perform.

Note:

- The default values for VMSPAUSE and VTSPAUSE assume that the amount of inbound and outbound data that passes through the gateway is approximately equal.

  If the gateway receives more inbound data from a remote LU in the SNA network, the VTSPAUSE value should be greater than the VMSPAUSE value. This situation may occur if the gateway is dedicated to a server that logs requests to resources but does not start conversations or send data.
- In general, AVS will process more events relating to the VMSPAUSE or VTSPAUSE parameters than events relating to the CMDPAUSE, IUCPAUSE, or ACTPAUSE parameters.
- The queued events associated with the CMDPAUSE and IUCPAUSE parameters generally will not exceed the defaults provided. If you change the default CMDPAUSE or IUCPAUSE values, it is unlikely to affect AVS performance.

VTAM—VM Request Transformation Control Parameters

The transformation control parameters, described in Table 8 on page 150, balance processing and the resources used when AVS translates between the APPC/VM and APPC/VTAM protocols.

Table 8. VM–VTAM Request Transformation Control Parameters

Label	Default	Meaning
PIPBUFF	3	Number of buffers reserved to receive PIP data. Each buffer is 33022 bytes in length, which is the maximum size of PIP data plus the maximum size of the FMH5. Generally, this value should be slightly higher than the expected number of outstanding allocation requests through AVS. (An allocate request is considered to be outstanding from the time AVS receives it until VTAM notifies AVS that the allocation has completed.)
MAXRECV	10	Number of times that VTAM or z/VM receives data within a single conversation before another conversation is allowed to participate. This parameter prevents conversations that may be sending large amounts of data on the z/VM or VTAM side from monopolizing AVS resources.
CONVQUI	10	Conversation quiesce count. This parameter sets the maximum number of frames that AVS will queue for any conversation. (A frame contains approximately 1024 bytes of data). When the size of the queue for a conversation exceeds this value, AVS will temporarily quiesce the conversation.

Table 8. VM–VTAM Request Transformation Control Parameters (continued)

Label	Default	Meaning
CONVRES	2	Conversation resume count. This value specifies the point at which a quiesced conversation will be resumed; a conversation is quiesced when the number of data frames queued for it exceeds the CONVQUI value. When the number of frames queued for the conversation reaches the CONVRES value, AVS will accept new data frames for that conversation. **Note:** To prevent AVS from repeatedly quiescing and resuming conversations, the CONVRES value should be significantly less than the CONVQUI value.
CONVPRA	3	Number of conversations to be covered by one VTAM RECEIVE ANY request. This value must be at least 1. If you decrease the CONVPRA value, AVS will require more storage and processor time; however, AVS throughput may increase.
EXTRADAT	3	Number of data areas to initially allocate; this value is also the number of free data areas that AVS will retain before returning storage to GCS. When AVS uses all of the data areas that it has available, it tries to obtain storage from GCS. Conversely, when the number of free data areas exceeds the EXTRADAT value, AVS returns the excess storage to GCS. If you increase the EXTRADAT value, AVS throughput may increase but AVS may also require more storage. If you decrease the EXTRADAT value, AVS requires less storage but AVS throughput also decreases. EXTRADAT should always be at least 1. If you set the CONVPRA value lower than the default, EXTRADAT should be set above its default value.
MAXMSGSZ	15332	Maximum size of the buffer that AVS presents to VTAM when sending or receiving data. MAXMSGSZ defines the size of the data areas used for VTAM RECEIVE ANY requests; it must be a value between 1 and 15332, inclusive. The optimal MAXMSGSZ value depends on the logical record length of the data that is transferred through AVS. If you lower the MAXMSGSZ value to the logical record length, you can reduce the amount of storage that AVS requires. If the logical record length of some data exceeds the MAXMSGSZ value, additional RECEIVE ANY requests are required to send or receive the data. This may increase the time needed to send or receive the data. For example, if most applications that use AVS send data with a logical record length of 4096 bytes, the MAXMSGSZ value could be set to 4100. AVS will then allot 4096 bytes for RECEIVE ANY requests plus 4 bytes for header and length information.

Additional Tuning Parameters

Table 9 on page 152 describes the parameters that affect the AVS accounting facility and the generation of problem dumps.

Table 9. Additional AVS Tuning Parameters

Label	Default	Meaning
ACTINTV	1	Number of hours that the AVS accounting facility waits between creating accounting records for active conversations. AVS also creates accounting records when each conversation starts and ends, and any time a bytes-sent or bytes-received counter wraps around. This value must be an integer between 0 and 1192046; a value of 0 stops interval accounting.
MAXPROBD	20	Maximum number of AVS problem dumps taken during each AVS session. When AVS detects an internal error, it produces a dump and attempts to continue operating if possible. When the number of dumps produced reaches the MAXPROBD value, AVS will not produce problem dumps for additional errors. This is useful if AVS will be running in an unattended environment. Each time AVS is restarted, the current count of problem dumps taken is reset to 0.
REXTRYINT	0	Time, in hundredths of seconds [0.01], AVS waits before retrying a VTAM function that failed because of a temporary storage shortage. If AVS severs conversations because it does not have enough storage, increasing this parameter may alleviate the problem.
MAXRETRY	10	Maximum number of times AVS retries a VTAM function that failed because of a temporary storage shortage. If AVS severs conversations because it does not have enough storage, increasing this parameter may alleviate the problem.

Chapter 16. TCP/IP Tuning

This section describes several ways of tuning TCP/IP to its environment. It covers those aspects of TCP/IP tuning that are specific to TCP/IP for z/VM.

TCP/IP Server Virtual Machine Tuning

Each of the TCP/IP virtual machines should be run with the QUICKDSP option in effect. This is enabled by adding the QUICKDSP option to the OPTION control statement in the CP directory entries for these virtual machines. The QUICKDSP option ensures that the servers will not have to wait in the eligible list for system resources to become available.

The TCPIP virtual machine should be run with a SHARE setting that is significantly higher than the default (100). This is because it needs to provide responsive service to potentially large numbers of users. A relative SHARE of 3000 is suitable in many cases. Update the TCPIP virtual machine's CP directory entry to add a SHARE REL 3000 control statement.

It is sometimes beneficial to use the CP SET RESERVE command to reduce the amount of paging that occurs in the TCPIP virtual machine. This tuning action is indicated if the DASD page read rate for the TCP/IP virtual machine exceeds 5 pages per second. This information is available in the CP monitor data (user domain sample records) and from the INDICATE USER command. If SET RESERVE is used, the number of reserved pages should be set equal to the TCPIP virtual machine's typical working set size during peak usage conditions.

Configuration Parameters

The following configuration parameters have the largest effect on TCP/IP performance. Maximize these parameters where possible:

- Buffer size
- Number of buffers
- Window size
- Packet size

Configuration file statements can be modified to affect TCP/IP performance. For information about configuration file statements, see *z/VM: TCP/IP Planning and Customization*.

Buffer Size

The DATABUFFERPOOLSIZE statement describes the buffers used to hold data during TCPIP virtual machine processing. You can modify the number of buffers and the size of each one. The number of data buffers is limited only by the amount of virtual storage. You should be careful to specify a sufficient number of data buffers. Running out of data buffers causes degraded performance and the abnormal termination of Telnet server connections.

The default buffer size is 8192 bytes (8 KB). Other specific sizes are permitted up to 262144 bytes (256 KB). For a list of the supported values, see the DATABUFFERPOOLSIZE statement in *z/VM: TCP/IP Planning and Customization*. Increasing the size of the buffers results in increased throughput for file transfers, but it may waste storage for other applications such as Telnet that send and receive smaller blocks of data.

Number of Buffers

The LARGEENVELOPEPOOLSIZE statement describes the envelopes used to hold UDP datagrams larger than 2048 bytes while they are being processed for output or waiting for an application program to receive them. The number of large envelopes is 50 by default and can be increased as long as enough

virtual storage is available for the additional envelopes. The size of the large envelopes is 8192 bytes (8 KB) by default, but other specific sizes are permitted from 512 bytes to 65535 bytes (64 KB). For a list of the supported values, see the LARGEENVELOPEPOOLSIZE statement in *z/VM: TCP/IP Planning and Customization*.

Running out of large envelopes or regular data buffers causes TCP/IP to drop outgoing and incoming packets. This results in the retransmission of the lost packets, which leads to degraded performance. You should carefully consider the values you specify on the configuration statements to assess the impact on the system and virtual storage.

Window Size

The *window size* associated with a TCP connection determines the amount of unacknowledged data that can be outstanding between the sender and the receiver. Generally, it is the amount of data that can be in transit, somewhere in the network, between the two. The optimal window size for a particular connection can be calculated using the formula:

```
optimal window size = round-trip time * network bandwidth
```

For example, with a round-trip time of 15 milliseconds on a 16 megabit (that is, 2 MB) token ring, the optimal window size is

```
.015 seconds * 2,000,000 bytes/second = 30,000 bytes
```

The window size for data being received by TCP/IP for z/VM is determined by the size of the z/VM client's buffer. For example, the z/VM FTP client uses the size of the data buffer, specified by the DATABUFFERPOOLSIZE configuration statement, as its receive window size.

The window size for data being transmitted by TCP/IP for z/VM is set by the receiving system. Some systems have explicit controls that permit the window size to be set. For example, AIX® allows the sizes of its send and receive windows to be set either temporarily or permanently.

TCP/IP for z/VM provides a way to increase window sizes for incoming TCP data regardless of the client's buffer size. By implementing *window scaling*, it allows window sizes to be as large as 1 GB. In order to take advantage of window scaling, both the sending and receiving TCP/IPs must support it, as specified in RFC 1323. Without window scaling, the maximum window size is 65,535 bytes.

Window scaling in TCP/IP for z/VM is controlled by several parameters. Unless the NORFC1323 option of the ASSORTEDPARMS statement is specified, TCP/IP for z/VM attempts to establish window scaling for any TCP connection it initiates. Regardless of this setting, requests to support window scaling from other systems are accepted. The window size is set by multiplying the data buffer size, specified by the DATABUFFERPOOLSIZE statement, by the appropriate (i.e., send or receive) buffer limit, specified by the DATABUFFERLIMITS configuration statement.

The settings of these parameters can affect the number of data buffers required for a given TCP/IP workload and may require increasing the size of the data buffer pool. This in turn may increase TCP/IP storage requirements, leading to additional paging. As a result, the parameter settings should be selected with care.

When window scaling is active, the receive window size determines the size of the window that is advertised to the communicating partner. The send window size determines the maximum amount of data that the local z/VM client can have in transmission through the network at any one time.

Window scaling is most effective for networks that have large round trip times and large bandwidths (that is, networks whose optimal window size exceeds the actual window size when window scaling is not used).

Other Tuning Considerations

Here are some additional TCP/IP tuning considerations.

Multiple Servers

Maximum throughput sometimes becomes limited by the capacity of one of TCP/IP's server virtual machines. When this is the case, throughput can often be improved by configuring additional servers. TCPIP, SMTP, and FTPSERVE are examples of servers that can be replicated. For information on how to set up multiple servers, see *z/VM: TCP/IP Planning and Customization*.

Multiple TCP/IP Configurations

In some situations, using multiple TCP/IP configurations (that is, several TCP/IP stack machines and associated servers) can improve throughput. A typical scenario in which this approach can be beneficial has several networks whose users communicate primarily with one another and only occasionally communicate with users on another network. For example, one network within a single z/VM system, perhaps providing connections between CMS users and z/OS guests, and another network connecting PCs to the z/VM system via a local area network, might have this characteristic.

When multiple TCP/IP configurations are created, each requires its own home address. The two stack machines can be interconnected using either a virtual channel-to-channel adapter or an IUCV link. This allows traffic destined for a network managed by another stack machine to be forwarded there, although at the cost of some additional overhead. However, if most traffic is intra-network, the benefits of parallelism from using multiple configurations can be realized.

Another advantage that can be gained by using multiple configurations is the ability to limit the services provided to the users of a network. This may translate into performance benefits if, for example, the load associated with FTP activity can be reduced using this technique. In other situations, it may merely provide a convenient tool for network management.

Multiprocessor Host

The CPU capacity of the host has a significant effect on the performance of TCP/IP for z/VM. Because it uses multiple virtual machines, TCP/IP can make good use of multiprocessor configurations.

The TCPIP virtual machine can exploit a virtual multiprocessor configuration. It does so by using virtual processors you designate to run specific device drivers. This allows the TCPIP load to be spread across multiple real processors and, in high-activity environments, can improve responsiveness and throughput. If your TCPIP load ordinarily uses a substantial portion of a single processor, there may be benefits to creating a multiprocessor configuration.

Chapter 17. VMRM Tuning Parameters

This section describes the Virtual Machine Resource Manager (VMRM) service virtual machine (SVM), the statements used to configure this SVM, and rules for adjusting users in a workload.

VMRM provides functions to dynamically tune the z/VM system. Groups of virtual machines can be defined to be part of a workload. The workloads can then be managed by VMRM to goals that are also defined. The system administrator can use VMRM to create a form of group scheduling for a set of virtual machines. There can be multiple workloads (groups of virtual machines), each managed to different goals. VMRM automatically adjusts performance parameters when there is contention between virtual machines for a resource. Therefore, VMRM is not effective in environments where resources are unconstrained.

Consider the following example. There are several virtual machines running web servers that are critical to the business. Another set of virtual machines are used by development and test groups that consume large amounts of resources. One workload could be defined that contains the virtual machines running the web servers and another workload that contains the development and test systems. Because the web serving workload is critical, the goals for this workload would be high. Lower goals could be created and assigned to the development and test workload. This example would ensure that development work would not interfere with business-critical workloads.

VMRM uses z/VM monitor data to obtain regular measurements of virtual machine resource consumption. Based on a customer-supplied definition of workloads and workload goals from a configuration file, VMRM will adjust virtual machine tuning parameters to achieve those goals.

Approximately once a minute (the recommended setting for the MONITOR SAMPLE interval) VMRM will compute the achievement levels for each workload. It selects a workload based on importance that is not within a reasonable percent of its CPU velocity goal (as described in “GOAL Statement” on page 165) to improve or degrade. If the workload has been selected recently for CPU velocity adjustment, VMRM skips it and attempts to select another one. VMRM then selects a workload that is not within a reasonable percent of its DASD velocity goal (as described in “GOAL Statement” on page 165) and adjusts the workload accordingly.

Note: Do not relocate a Linux guest that is being monitored by VMRM. System and guest performance results are unpredictable and very likely to be unsatisfactory.

Overview of Managing Memory with VMRM Cooperative Memory Management

VMRM Cooperative Memory Management (VMRM-CMM) between a z/VM system and Linux guests assists in managing memory constraint in the system. Based on several variables obtained from the system and storage domain CP monitor data, VMRM detects when there is such constraint, and notifies specific Linux virtual guests when this occurs. The guests can then take the appropriate action to adjust their memory utilization in order to relieve this constraint on the system, such as issuing a CP DIAGNOSE X'10' instruction to release pages of storage.

In addition to the workload management functions for CPU and DASD I/O provided by VMRM, the following is provided for Linux guests:

- A NOTIFY statement with a MEMORY keyword in the Virtual Machine Resource Manager configuration file. Following the keyword is a user ID or a list of user IDs to be notified when virtual memory becomes constrained. For the format of this statement, see “NOTIFY Statement” on page 166.
- System and storage domains are monitored for data to be used for calculating memory constraint, as well as how much memory to request the guest machine to release.
- When memory is constrained, VMRM issues a CP SMSG to notify the specified guests with the amount required to release in order to relieve the constraint. For the format of the SMSG buffer, see Usage Note “1” on page 167.

- A message is logged in the VMRM log file, indicating which users were sent an SMSG, and the text of the SMSG buffer. Also, if MSGUSER is specified on the VMRM ADMIN statement, the same message written to the log is written to the MSGUSER user ID's console as well.

For more information on VMRM Cooperative Memory Management, refer to VM Resource Manager (VMRM) Cooperative Memory Management (https://www.ibm.com/vm/sysman/vmrm/vmrmcmm.html).

This web page includes a link to *Linux on System z®: Device Drivers, Features, and Commands*. The manual contains a chapter entitled "Cooperative memory management" that describes how to set up VMRM Cooperative Memory Management. In particular, refer to the `modprobe` command or the `cmm.sender` kernel parameter in *Linux on System z: Device Drivers, Features, and Commands* for information on enabling VMRMSVM to send messages to Linux guests.

Note: Do not confuse VMRM Cooperative Memory Management with Collaborative Memory Management Assist. The latter is a machine feature that allows z/Architecture guests with the appropriate support to exchange memory usage and status information with z/VM. See "Collaborative Memory Management Assist" on page 36.

Monitoring System Domains

In addition to the monitor records being used for CPU and DASD velocity goals, VMRM enables and monitors some system domain records. There are two types of problems to solve using this data: 1) determining when there is memory constraint, and 2) when memory is constrained, calculating how much memory to notify the guest to release.

Once system memory constraint has been detected, VMRM calculates how much memory each Linux guest should release to relieve the constraint. Using the SHRINK keyword on the SMSG command, a message indicating the amount of storage to release is sent to each logged on Linux guest in the notify list.

When system memory is no longer constrained, another SHRINK message with a smaller absolute value is issued. A smaller SHRINK request than the previous one effectively instructs the guest to reclaim some of the storage previously released.

VMRM User Definitions

VMRMSVM is a predefined multiconfiguration virtual machine that can be enabled to run the VMRM code. Because it uses some privileged system functions to monitor virtual machine performance and adjust virtual machine performance settings, the user definition of the VMRMSVM user ID requires some special settings. VMRMSVM requires privilege Class A because it will be using the CP SET SHARE and CP SET IOPRIORITY commands. It also requires the "NAMESAVE MONDCSS" and "IUCV *MONITOR MSGLIMIT 255" directory statements. You may want to have VMRMSVM logged on automatically at system IPL.

The VMRM configuration file may define a user ID to which error messages are sent. This user ID requires no special privileges and is optional. Error messages and status information are always written to the log file on the VMRMSVM A-disk.

The following is an example of the user ID definitions required by VMRM. These users are defined when the z/VM product is installed.

Figure 11. Sample VMRM User Definitions

```
IDENTITY VMRMSVM pppppppp 64M 128M AG
 BUILD ON * USING SUBCONFIG VMRMSV-1
 IPL CMS
 OPTION APPLMON QUICKDSP
 SHARE ABSOLUTE 3%
 ACCOUNT MONITOR
 MACHINE ESA
 IUCV *MONITOR MSGLIMIT 255
 NAMESAVE MONDCSS
 CONSOLE 0009 3215 T
 SPOOL 000C 2540 READER *
 SPOOL 000D 2540 PUNCH  A
 SPOOL 000E 1403 A
```

```
SUBCONFIG VMRMSV-1
 LINK  MAINT 190 190 RR
 LINK  MAINT 193 193 RR
 LINK  MAINT 19D 19D RR
 LINK  MAINT 19E 19E RR
 MDISK 191 3390 3036 040 MO1RES MR READ     WRITE    MULTIPLE

IDENTITY VMRMADMN pppppppp 16M 16M G
 BUILD ON * USING SUBCONFIG VMRMAD-1
 IPL CMS
 MACHINE ESA
 SPOOL   000C RDR   *
 SPOOL   000D PUN   A
 SPOOL   000E 1403  A
 CONSOLE 0009 3215  T

SUBCONFIG VMRMAD-1
 LINK  MAINT 190 190 RR
 LINK  MAINT 193 193 RR
 LINK  MAINT 19D 19D RR
 LINK  MAINT 19E 19E RR
 MDISK 191 3390 3035 001 MO1RES MR READ     WRITE    MULTIPLE
```

VMRM Configuration File

VMRM uses a customer-managed configuration file to define workloads to be managed and other VMRM configuration information. The default configuration file name is VMRM CONFIG, and may reside at any file mode that is accessible to the VMRMSVM user ID. You may specify another file name, file type, and file mode of your choice when starting the SVM.

The following table gives a brief description of each VMRM configuration file statement and tells you where to find detailed information about each statement.

Table 10. VMRM SVM Configuration Statements

Statement	Description	Reference
ADMIN	Specifies a user ID on the same system where messages can be sent from the SVM. Can also specify the name of a new configuration file to be used when you update a configuration file statement. The ADMIN statement is optional.	“ADMIN Statement” on page 163
GOAL	Defines a goal that is comprised of velocity targets for CPU or DASD or both.	“GOAL Statement” on page 165
MANAGE	Associates a workload to a goal with the importance of achieving that goal.	“MANAGE Statement” on page 166
NOTIFY	Lists the Linux guest virtual machine user IDs to be notified when memory constraint is detected by VMRM.	“NOTIFY Statement” on page 166
WORKLOAD	Defines a workload comprised of one or more virtual machines identified by a user name, account ID, or ACI group name.	“WORKLOAD Statement” on page 168

Dynamically Changing the Configuration File

You can dynamically change to a new configuration file by doing the following:

1. Specify a file name, file type, and an SFS directory name on the ADMIN statement. The file name you specify is monitored for changes.
2. Update an ADMIN, GOAL, MANAGE or WORKLOAD statement.

For example, suppose you specify an alternate configuration file on the ADMIN statement. If you update a WORKLOAD statement to add virtual machines to a workload, VMRM will detect the updated configuration file and put that file into production while VMRM is running. (The new configuration file is put into production when the timestamp of the file changes and no syntax errors are found in the file. It is recommended that you use the SYNCHECK option on the IRMSERV EXEC to check the syntax of the new configuration file before putting it into production.) For more information see "Starting and Stopping VMRM" on page 161 and "ADMIN Statement" on page 163.

Configuration File Rules

The following rules are applied to the parsing and structure of the VMRM configuration file. VMRM assumes that an error in the configuration file would result in VMRM mismanaging workloads. Therefore, the penalty for most errors in the configuration file is severe—usually VMRM shuts down and does not proceed to manage workloads. Errors found in the configuration file are recorded in the VMRM log file and optionally sent to the user ID specified by the MSGUSER operand of the ADMIN statement.

1. The configuration file is read during VMRM initialization. You can dynamically change to a new configuration file by specifying a new configuration file name on the ADMIN statement. The file you specify is monitored for changes. The new configuration file is put into production only if the file is updated (that is, if the timestamp of the file changes and no syntax errors are found in the file) while VMRM is running. Note that this can be accomplished by simply opening the file with Xedit and then filing it.
2. The configuration file can be a fixed-length or variable-length record file. Any record length that is supported by the CMS file system is permitted.
3. Statements can be continued on multiple lines, using a comma at the end of the line being continued.
4. Statements may be specified in the configuration file in any order.
5. Line comments are allowed and must start with '*', ';' or '/*'.
6. Blank lines are allowed.
7. You can enter information into the configuration file in either upper case or mixed case. Entries in the configuration file are converted to upper case before they are processed.

Configuration File Example

The following is an example of the VMRM configuration file:

Figure 12. Sample VMRM Configuration File

```
/*********************************************************************/
/*                                                                   */
/*   Licensed Materials - Property of IBM                            */
/*   This product contains "Restricted Materials of IBM"             */
/*   5739-A03 (C) Copyright IBM Corp. - 2003                         */
/*   All rights reserved.                                            */
/*   US Government Users Restricted Rights -                         */
/*   Use, duplication or disclosure restricted by GSA ADP Schedule   */
/*   Contract with IBM Corp.                                         */
/*   See IBM Copyright Instructions.                                 */
/*                                                                   */
*--------------------------------------------------------------------*
*  This is a sample VMRM Service                                      *
*  Virtual Machine configuration file.                                *
*                                                                     *
*  -  ADMIN is an optional statement that must contain                *
*     either one or both of the keywords: MSGUSER and NEWCFG, to      *
*     specify a userid where error messages can be sent and a new     *
*     configuration file to use.                                      *
*   - WORKLOAD is a required statement, specifying the workload       *
*     name followed by either a USER, ACCOUNT, or ACIGROUP keyword    *
*     and the appropriate value.  Multiple users, account IDs,        *
*     or Acigroup names may be specified on one line for each type,   *
*     or continued on the next line using a comma as a continuation   *
*     character at the end of the continuing line.                    *
*  -  GOAL is a required statement, specifying the goal name,         *
*     the goal type keyword, CPU or DASD keyword, followed by the     *
*     target percentage value.                                        *
```

```
*  -  MANAGE is a required statement that associates a WORKLOAD        *
*     with a GOAL.  An importance value between 1-10 must be           *
*     specified for managing this workload.  A workload may be         *
*     managed to only one goal at a given time.                        *
*                                                                      *
*----------------------------------------------------------------------*

*   This is a valid comment line  *
/*  So is this                    */
;   and this

/* ADMIN STATEMENT */
/* This will cause messages to be sent to VMRMADMN's console */
ADMIN MSGUSER vmrmadmn

/* GOAL STATEMENTS */
GOAL MAX VELOCITY CPU 100 DASD 100
GOAL MIDDASD VELOCITY DASD 50
GOAL MINCPU VELOCITY CPU 1

/* WORKLOAD statements followed by corresponding MANAGE statement */
* workload 1
WORKLOAD WORK1 USER linux* manfred fredrick usera,
    userb chris kurt doug jon
MANAGE WORK1 GOAL MAX IMPORTANCE 5

* workload 2
WORKLOAD WORK2 USER payroll
MANAGE WORK2 GOAL MAX IMPORTANCE 10

* workload 3
WORKLOAD WORK3 USER webcount
MANAGE WORK3 GOAL MIDDASD IMPORTANCE 7

* workload 4
WORKLOAD WORK4 USER theboss
MANAGE WORK4 GOAL MINCPU IMPORTANCE 1
```

Starting and Stopping VMRM

You may log on to the VMRMSVM user ID, or xautolog VMRMSVM using parameters set up in the user directory statement for your z/VM system. A sample PROFILE EXEC and sample VMRM CONFIG file are supplied for you to use on the VMRMSVM A-disk, as well as files necessary to run VMRM. You may run the profile to start VMRM or specify the exec name, IRMSERV, on the command line, and optionally the name of a configuration file. If no configuration file is specified, the default name VMRM CONFIG * will be used.

Your startup parameters may be specified similar to the following examples:

```
     IRMSERV MYCONFIG FILE A
```

or

```
     IRMSERV MYCONFIG FILE A (SYNCHECK
```

where MYCONFIG FILE A is a user-defined configuration file on the VMRMSVM user's A-disk.

To stop VMRM, log on to the VMRMSVM user ID and issue HMONITOR. HMONITOR is an immediate command handler that is set up when VMRM is receiving monitor data. Using HX, IPL CMS, or forcing off the VMRMSVM user ID is not recommended because normal SVM termination will not occur and the VMRM log file will not be closed. The data in the log file will be incomplete.

When you specify SYNCHECK the configuration file is checked for errors and the server is not started. Any errors are logged in the log file.

Interaction with CP Monitor

VMRM requires CP MONITOR SAMPLE data in order to monitor virtual machine performance. VMRM assumes that the MONDCSS saved segment exists. This segment was defined and saved during system installation. VMRM will issue "SEGMENT LOAD MONDCSS" to load the MONDCSS saved segment into its virtual storage. See “The Monitor Saved Segment” on page 82 for details.

VMRM will start sample monitoring with an interval of 1 minute if sample monitoring is inactive. If sample monitoring is already active, then VMRM does not start it and uses whatever interval is already set.

Note that a sample monitor interval of less than 1 minute or more than 5 minutes is not recommended. A sample monitor interval that is less than 1 minute does not allow enough time for VMRM to collect a sufficient amount of monitor data. A sample monitor interval that is more than 5 minutes may cause VMRM to adjust workload performance settings too slowly. The system programmer must negotiate a sample monitor interval that satisfies the requirements of all sample monitor data users on the system.

When the HMONITOR immediate command is issued to stop VMRM, VMRM will stop sample monitoring if VMRM started it and no other user is connected to sample monitoring.

Problem Conditions

VMRM assumes that certain conditions exist in order for it to effectively use dynamic tuning. Use of explicit tuning for these same virtual machines may cause unexpected results. The following are examples of conditions that should be avoided if VMRM is being used:

- Running in a capped LPAR
- Dedicating processors to virtual machines that are assigned to a workload
- Using the CP SET THROTTLE command for a DASD device to which VMRM–managed virtual machines are doing I/O
- Using non-default values for the SET SRM DSPBUF settings

Rules for Adjusting Users

If a workload was not adjusted for CPU during the last interval and is selected by VMRM, the following rules are used for adjusting users within that workload.

For CPU goals, all of the following must be true:

The user has a Relative share setting.
The user does not have LIMITHARD specified on the CPU SHARE setting.
The user is not already within 5% of the goal for the user.

If a workload was not adjusted for DASD during the last interval and is selected by VMRM, the following rules are used for adjusting users within that workload.

For DASD goals, the following must be true:

The user has a Relative I/O priority setting.
The high value has not already reached 255 for adjusting upward.
The low value has not already reached 0 for adjusting downward.
The user is more or less than 5 points from the goal for the user.

VMRM Log File

While VMRM runs, it creates a log file of major events as they occur. All messages that are sent to the message user (MSGUSER) defined in the ADMIN statement are logged. (If that user is not logged on, no messages are sent, but the messages are still logged.) Other information that is not sent to the MSGUSER is also logged. See "ADMIN Statement" on page 163 for more information.

The information in the log file is useful for fixing syntax problems in the configuration file and many other problems encountered by VMRM. The messages for VMRM can be found in *z/VM: CMS and REXX/VM Messages and Codes*.

A log file is stored on the VMRMSVM user's A-disk even if the MSGUSER operand on the ADMIN statement is not specified. The primary log file is called VMRM LOG1 A. Once the log file reaches 10,000 records, it will be renamed and saved as VMRM LOG2 A. A previously existing VMRM LOG2 A file will be erased.

Logging will continue and be written to a new VMRM LOG1 A file. This process is repeated each time the log record limit is reached. Figure 13 on page 163 shows an example of a VMRM log file.

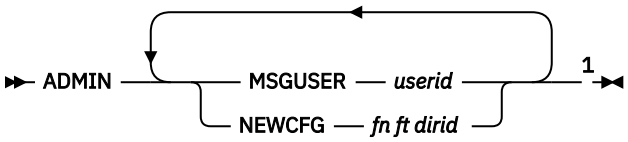
```
 2002-06-20 15:30:58 ServExe  Entry     ------------------------------------------
 2002-06-20 15:30:58 ServExe  PCfg       MAINTEST CONFIG A1 4/02/04 17:24:32
 2002-06-20 15:30:59 ServExe  MSG       IRMSER0032I No errors found in VMRM configuration file
 2002-06-20 15:30:59 ServExe  MSG       IRMSER0033I SYNCHECK option was specified. The server will
not be started.
 2002-06-20 15:31:18 ServExe  Entry     ------------------------------------------
 2002-06-20 15:31:18 ServExe  MSG       IRMSER0022I VM Resource Manager Service Virtual Machine
initialization started
 2002-06-20 15:31:18 ServExe  PCfg       MAINTEST CONFIG A1 4/02/04 17:24:32
 2002-06-20 15:31:18 ServExe  InitEnv  Monitor sample started -- recording is pending
 2002-06-20 15:31:18 ServExe  InitEnv  HCPMNR6224I Sample recording is pending because there are
no users connected to *MONITOR for this data.
 2002-06-20 15:31:18 ServExe  InitEnv  MONITOR EVENT INACTIVE    BLOCK    4     PARTITION    0
 2002-06-20 15:31:18 ServExe  InitEnv  MONITOR DCSS NAME - NO DCSS NAME DEFINED
 2002-06-20 15:31:18 ServExe  InitEnv  CONFIGURATION SIZE       68 LIMIT         1 MINUTES
 2002-06-20 15:31:18 ServExe  InitEnv  CONFIGURATION AREA IS FREE
 2002-06-20 15:31:18 ServExe  InitEnv  USERS CONNECTED TO *MONITOR - NO USERS CONNECTED
 2002-06-20 15:31:18 ServExe  InitEnv  MONITOR   DOMAIN ENABLED
 2002-06-20 15:31:18 ServExe  InitEnv  PROCESSOR DOMAIN DISABLED
 2002-06-20 15:31:18 ServExe  InitEnv  STORAGE   DOMAIN DISABLED
 2002-06-20 15:31:18 ServExe  InitEnv  SCHEDULER DOMAIN DISABLED
 2002-06-20 15:31:18 ServExe  InitEnv  SEEKS     DOMAIN DISABLED
 2002-06-20 15:31:18 ServExe  InitEnv  USER      DOMAIN DISABLED
 2002-06-20 15:31:18 ServExe  InitEnv  I/O       DOMAIN DISABLED
 2002-06-20 15:31:18 ServExe  InitEnv  APPLDATA  DOMAIN DISABLED
 2002-06-20 15:31:18 ServExe  InitEnv  MONITOR SAMPLE PENDING
 2002-06-20 15:31:19 ServExe  InitEnv  INTERVAL    1 MINUTES
 2002-06-20 15:31:19 ServExe  InitEnv  RATE     2.00 SECONDS
 2002-06-20 15:31:19 ServExe  InitEnv  MONITOR DCSS NAME - NO DCSS NAME DEFINED
 2002-06-20 15:31:19 ServExe  InitEnv  CONFIGURATION SIZE      241 LIMIT         1 MINUTES
 2002-06-20 15:31:19 ServExe  InitEnv  CONFIGURATION AREA IS FREE
 2002-06-20 15:31:19 ServExe  InitEnv  USERS CONNECTED TO *MONITOR - NO USERS CONNECTED
 2002-06-20 15:31:19 ServExe  InitEnv  MONITOR   DOMAIN ENABLED
 2002-06-20 15:31:19 ServExe  InitEnv  SYSTEM    DOMAIN ENABLED
 2002-06-20 15:31:19 ServExe  InitEnv  PROCESSOR DOMAIN DISABLED
 2002-06-20 15:31:19 ServExe  InitEnv  STORAGE   DOMAIN DISABLED
 2002-06-20 15:31:19 ServExe  InitEnv  USER      DOMAIN ENABLED
 2002-06-20 15:31:19 ServExe  InitEnv  ALL USERS ENABLED
 2002-06-20 15:31:19 ServExe  InitEnv  I/O       DOMAIN DISABLED
 2002-06-20 15:31:19 ServExe  InitEnv  APPLDATA  DOMAIN DISABLED
 2002-06-20 15:31:19 ServExe  MSG IRMSER0023I VMRM Service Virtual Machine initialization complete.
Proceeding to connect to Monitor.
 2002-06-20 15:31:19 MonRexx  Entry   MonIntCtr= 1 , Record= ENDR B7CEEE7F6551F741 , Processing this
record at 20 Jun 2002 15:31:19
 2002-06-20 15:32:18 MonRexx  Entry   MonIntCtr= 2 , Record= ENDR B7CEEEB79D0C9E80 , Processing this
record at 20 Jun 2002 15:32:18
 2002-06-20 15:33:18 MonRexx  Entry   MonIntCtr= 3 , Record= ENDR B7CEEEF0D235BE80 , Processing this
record at 20 Jun 2002 15:33:18
 2002-06-20 15:33:18 MonRexx  Select  IRMMON0028I Workload WORK7 selected to adjust UP for CPU
 2002-06-20 15:33:18 MonRexx  Select  IRMMON0028I Workload WORK2 selected to adjust DOWN for DASD
 2002-06-20 15:33:18 MonRexx  CPCMD   SET SHARE IRD00011 RELATIVE 769
 2002-06-20 15:33:18 MonRexx  CPCMD   SET IOPRIORITY IRD00002 RELATIVE 5 7
 2002-06-20 15:33:53 MonExec  Exit         IRMMON0026I VM Resource Manager processing of monitor records
ended. Pipe RC= 0
 2002-06-20 15:33:53 ServExe  MSG          IRMSER0012I VM Resource Manager Service Virtual Machine
shutdown in progress
 2002-06-20 15:33:53 ServExe  ExitSVM   Monitor sample stopped
 2002-06-20 15:33:53 ServExe  MSG          IRMSER0027I VM Resource Manager Service Virtual Machine
shutdown complete
```

Figure 13. Sample VMRM Log File

VMRM Configuration File Statements

This section describes the statements used to configure VMRM. The default configuration file name is VMRM CONFIG.

ADMIN Statement



Notes:

[1] The MSGUSER and NEWCFG operands can be specified only once.

Purpose

Use the ADMIN statement to specify:

- A user ID on the same system where messages can be sent from VMRMSVM if necessary

- A new configuration file

How to Specify

- You can place ADMIN statements anywhere in the configuration file.
- You may specify only one ADMIN statement.
- If you specify more than one ADMIN statement, only the last one will be used.

Operands

MSGUSER *userid*
specifies the user ID that will receive messages from VMRMSVM. Only one message user ID is allowed.

NEWCFG *fn ft dirid*
specifies the file name, file type, and fully qualified SFS directory name for the new configuration file. The new configuration file is put into production only if the file is updated (that is, if the timestamp of the file changes) while VMRM is running.

Usage Notes

1. The ADMIN statement is optional.
2. You can specify either MSGUSER or NEWCFG, or both. They can be specified in either order.
3. The new configuration file must be on an SFS directory that VMRMSVM has read/new read authority to. (See *z/VM: CMS File Pool Planning, Administration, and Operation* for information on setting up SFS directories.)
4. The SFS directory used by VMRM is the default file pool and directory shipped with z/VM unless changed by an administrator. The constant VMRM_SFSDir is set to 'VMSYS:VMRMSVM.' in the IRMCONS COPY file used by VMRM. If the administrator changes the default file pool or directory, this constant must be updated to match the changed directory name. For more information on naming SFS directories, see "Introduction and General Concepts" in *z/VM: CMS Commands and Utilities Reference*. The updates should be made as local modifications using the automated local modification procedure. See *z/VM: Installation Guide* for more information on using this procedure.
5. When a new configuration file is put into production, the current processing of monitor records stops, and VMRMSVM is restarted with the new configuration file if no errors are found in the file. If any errors are found, the errors are logged and VMRMSVM is shut down.
6. Each subsequent new configuration file can have an ADMIN statement that specifies another new configuration file. This allows you to have several files on a directory that can be put into production at various times.
7. A log file will always be stored on the VMRMSVM user's A-disk as VMRM LOG1 A, even if the ADMIN statement is not specified.

Examples

1. To cause user CHRIS to receive messages from VMRMSVM, use the following ADMIN statement::

```
admin msguser chris
```

2. To specify a new configuration file using the default file pool and directory, use the following ADMIN statement:

```
admin newcfg vmrm2 config vmsys:vmrmsvm.
```

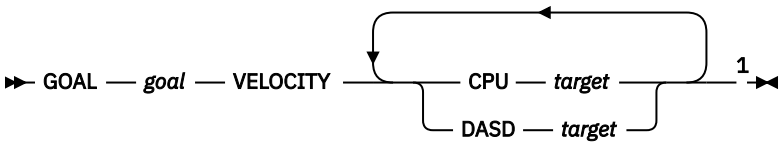
To specify a different file pool and directory, use the an ADMIN statement similar to this:

```
admin newcfg vmrm2 config mysrv:chris.vmrm
```

3. To cause user CHRIS to receive messages from VMRMSVM and to specify a new configuration file, use the following ADMIN statement:

```
admin msguser chris newcfg vmrm2 config vmsys:vmrmsvm.
```

GOAL Statement



Notes:

[1] The CPU and DASD operands can be specified only once.

Purpose

Use the GOAL statement to define a goal that is comprised of CPU and DASD velocity targets. A goal may have either or both targets defined.

How to Specify

Include as many statements as needed. You can place GOAL statements anywhere in the configuration file.

Operands

goal
specifies the name of the goal. The variable *goal* must be an alphanumeric string 1 to 16 characters long.

CPU *target*
specifies a CPU velocity goal as a percentage of time that the workload should receive CPU resources when it is ready to consume them. This is computed by taking the time the users are running and dividing the sum of time the users are running or waiting for CPU. The variable *target* must be an integer from 1 to 100.

DASD *target*
specifies a DASD velocity goal as a percentage of time that the virtual DASD I/O requests are processed without waiting because of higher priority I/O requests. The variable *target* must be an integer from 1 to 100.

Usage Notes

1. If a user is within a reasonable percent of the target goal, they will be considered to have met the goal, and no adjustments will be made for this user.
2. The goal defined on a GOAL statement can be used by more than one MANAGE statement. Extra unused GOAL statements are not allowed.

Examples

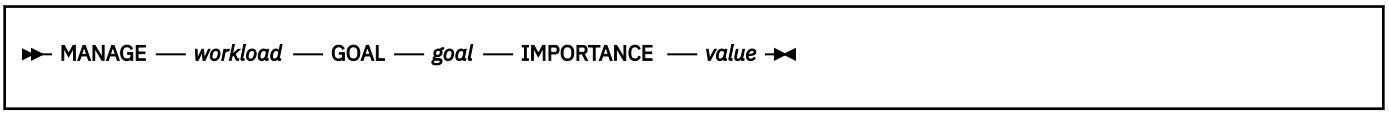
1. To define a goal that contains both CPU and DASD velocity targets, use the following GOAL statement in the configuration file:

```
goal middle velocity cpu 50 dasd 50
```

2. To define a goal where only a CPU velocity target is required, use the following GOAL statement in the configuration file:

```
goal high velocity cpu 90
```

MANAGE Statement



Purpose

Use the MANAGE statement to associate a workload to a goal and the importance of achieving that goal.

How to Specify

Include as many statements as needed. You can place MANAGE statements anywhere in the configuration file.

Operands

workload
specifies a workload that is defined by a WORKLOAD statement.

GOAL *goal*
specifies a goal that is defined by a GOAL statement.

IMPORTANCE *value*
specifies the whole number from 1 to 10 that indicates the relative importance of having this workload achieve the goal when compared to other MANAGE statements. The larger the number, the more important it is.

Usage Notes

1. A workload may be managed to only one GOAL at a given time.
2. For each WORKLOAD statement, there must be only one MANAGE statement that uses it. For each MANAGE statement, there must be only one associated WORKLOAD statement. Extra unused MANAGE statements are not allowed.

Examples

1. To assign goal *highcpu* to the *webserve* workload with an importance value of 9, use the following MANAGE statement in the configuration file:

```
manage webserver goal highcpu importance 9
```

NOTIFY Statement

Purpose

Use the NOTIFY statement with the MEMORY keyword to list the Linux guest virtual machine user IDs to be notified when memory constraint is detected by the resource manager.

How to Specify

Include as many statements as needed. You can specify the NOTIFY statement as the only statement in a configuration file, or specify it in addition to other VMRM configuration statements.

Operands

MEMORY
is the system object being managed.

userid
is a virtual machine user ID to be notified though SMSG.

user_list
is a list of virtual machine user IDs, separated by blanks, to be notified through SMSG. Instead of a user ID, a list member can contain a * wildcard character at the end, representing several user IDs. For example `linux*` represents any user ID beginning with a 5-character string ("linux") followed by 1 to 3 characters.

Usage Notes

1. The CP SMSG buffer sent to the guest has the following general format:

   ```
   CP SMSG userid buffer
   ```

   where

   userid
   is a user ID from the NOTIFY MEMORY list.

   buffer
   has the form CMM SHRINK *value*.

   CMM
   indicates VMRM Cooperative Memory Management.

   SHRINK
   is a keyword indicating pages are to be released or reclaimed.

   value
   is the number of pages in decimal to release (through DIAGNOSE X'10'). If a SHRINK message is issued with a smaller absolute value than the value previously issued, the guest can reclaim some of the memory previously released.
2. The use of VMRM Cooperative Memory Management can significantly improve overall system performance in cases where the overall z/VM system is constrained for real storage and much of that storage is being held by one or more Linux guests. However, use of VMRM-CMM can sometimes reduce the performance of one or more of the participating Linux guests. Accordingly, it is recommended that the performance of the Linux guests be monitored before and after the use of VMRM-CMM, allowing you to determine whether any performance-critical guests are being adversely affected and, if so, to remove them from the NOTIFY list. For a Linux guest excluded from VMRM-CMM, the best way to constrain the storage usage is to reduce its virtual storage size as much as is practical.
3. Support in Linux for VMRM-CMM is available on the developerWorks®® website at IBM developerWorks: Linux: Linux on IBM Z and LinuxONE (https://www.ibm.com/developerworks/linux/linux390/).
4. Use of VMRM-CMM can potentially reduce the performance of one or more of the participating Linux guests. VMRM can ask the Linux guests to give up too much memory, resulting in the Linux guests using excessive CPU time trying to find more storage to give up, leaving little CPU time available for useful work. To prevent this situation from happening, VMRM has a configurable constant that defines

a safety net value below which VMRM will not ask the Linux guests to shrink. By default, this safety net value is 64 MB, but for certain workloads, the value might be too low, causing poor performance. The safety net value is defined by the constant MinRequired in the VMRM constants file, IRMCONS COPY, and by default is set to 16,384 pages. To change the safety net value, update the value of the constant MinRequired (set in pages) in file IRMCONS COPY. Make the update as a local modification by following the automated local modification procedure documented in "Appendix D. Apply or Rework Local Service and Modifications" in *z/VM: Installation Guide*.

Examples

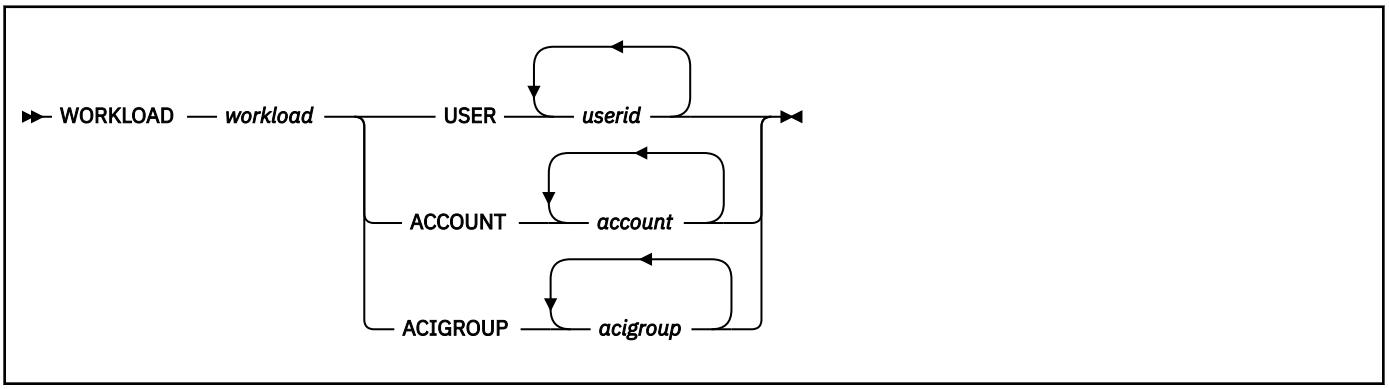
1. To set up memory management notification for users LINUX1, LINUX2, CHRIS, and ABCUSER, code the following NOTIFY statement:

```
NOTIFY MEMORY LINUX1 LINUX2 CHRIS ABCUSER
```

or

```
NOTIFY MEMORY LINUX* CHRIS ABCUSER
```

WORKLOAD Statement



Purpose

Use the WORKLOAD statement to define a workload that is comprised of one or more virtual machines identified by user name, account ID, or ACI group name.

How to Specify

Include as many statements as needed. You can place WORKLOAD statements anywhere in the configuration file.

Operands

workload
specifies the name of the workload. The variable *workload* must be an alphanumeric string 1 to 16 characters long.

USER *userid*
specifies the user ID that you want to place in the workload. A wildcard character (*) can be used as the last character in a *userid* string. For example, `erin*` represents matching a user ID beginning with a 4-character string ('erin') followed by 1 to 4 more characters.

ACCOUNT *account*
specifies an account ID that is used to select user IDs that you want to place in the workload.

ACIGROUP ***acigroup***
specifies the Access Control Interface (ACI) group name that is used to select user IDs that you want to place in the workload.

Usage Notes

1. Multiple users, account IDs, or ACI group names may be specified on one line for each type.
2. For each WORKLOAD statement, there must be only one MANAGE statement that uses it. For each MANAGE statement, there must be only one associated WORKLOAD statement. Extra unused WORKLOAD statements are not allowed.
3. If the current values of user ID, account ID, or ACI group for a user match more than one workload statement, the user will be included in only one workload based on the following priority: user ID, account ID, and finally ACI group.
4. Long lines can be continued using a comma (,) at the end of the line, and then continuing onto the next line. The resulting lines are treated as one statement.

Examples

1. To assign users LINUX1, LINUX2, JON, ABCUSER, and CHRIS to workload BUDDIES, use one of the following WORKLOAD statement in the configuration file:

```
workload buddies user Linux1 Linux2 jon abcuser chris
```

or

```
workload buddies user Linux* jon abcuser chris
```

Appendix A. Monitor System Service (*MONITOR)

This section describes the monitor system service and tells you about:

- Establishing data link with *MONITOR
- IUCV functions used in conjunction with *MONITOR.

A virtual machine uses the monitor system service (*MONITOR) as a data link with CP. Understanding the monitor system service is necessary if you desire to write an application like IBM's MONWRITE.

Establishing Communication with *MONITOR

PI

A virtual machine communicates with the *MONITOR through IUCV functions. To establish an IUCV connection with *MONITOR, a virtual machine must:

- Have an IUCV directory control statement in its directory entry that specifies *MONITOR as the system service ID.
- Load the monitor saved segment into its virtual storage. This is done through the DIAGNOSE code X'64' function or CMS SEGMENT LOAD command.
- Issue the IUCV DCLBFR (Declare Buffer) function to initialize the virtual machine for communication with IUCV.

For more information on the IUCV functions mentioned in this section, see *z/VM: CP Programming Services*.

Connect Interface

*MONITOR accepts two different connect interfaces. The connect interface determines the mode that monitor runs in, shared or exclusive.

When in exclusive mode, only one user is allowed to connect to *MONITOR. When in shared mode, one or more users can connect using the connect interface for shared mode. In addition to limiting the number of connections, exclusive mode differs from shared mode in that, by its use of quiesce or by delinquent replies, an exclusive mode user can cause monitor to suspend or stop data recording, whereas a shared mode user never suspends or stops monitor.

IUCV Functions Used by a Virtual Machine

A virtual machine uses some or all of the following IUCV functions: CONNECT, QUIESCE, REPLY, RESUME, and SEVER.

IUCV CONNECT

Number of Connections

The number of permissible concurrent connections is limited to:

- When connecting using the interface for exclusive mode:
  - One path
- When connecting using the interface for shared mode:
  - A maximum of 65535 paths. (This is an architectural limit. The practical limit depends on available system resources.)
  - One path for each virtual machine.

Format of CONNECT Interface

A virtual machine connects to *MONITOR by issuing the IUCV CONNECT macro with the following specified:

- USERID=*label* (or *register* containing the address) of an 8-byte area containing the EBCDIC characters for **MONITOR*.
- MSGLIM=*label* (or *register* containing the address) of a half-word containing any number between 4 and the maximum message limit (65535 messages) allowed by IUCV.

  Note:

  1. The message limit provided here by the user limits the number of outstanding messages from *MONITOR that the virtual machine can have at any one time.
  2. **Warning:** If a user's message limit is reached, further IUCV messages from *MONITOR will be delayed or missed completely until the user replies to or rejects one or more IUCV messages.
- PRMDATA=YES to indicate that you want to receive data in the parameter list rather than in a buffer.
- USERDTA=*label* (or *register* with address) of 16-byte field which is formatted following the exclusive or shared CONNECT interface. Both interfaces are described by the following.

CONNECT Interface for Exclusive Mode

For the exclusive interface, the 16-byte field pointed to by USERDTA is formatted as follows:

DCSS name
is the name of monitor DCSS (EBCDIC). The name must be left-justified and padded with blanks (X'40').

8 - 15
Reserved

When the exclusive connect interface is used, the connection is for both sample and event data. This means that IUCV SENDs are issued to the application for sample and event monitoring, depending on what type of monitoring is active.

CONNECT Interface for Shared Mode

The shared interface tailors the monitor environment of the application. For the shared interface, the 16-byte field pointed to by USERDTA is formatted as follows:

Byte 0: Version
is the version code. It must be X'01'.

Byte 1: Data
is the type of data to be collected:

Bits
Meaning

0
Sample data

1
Event data

2-6
Reserved for IBM use

7
Subinterval sample data

Note: Either sample data (bit 0) or event data (bit 1), or both, must be specified. If neither is specified, then the connection request is rejected. Subinterval sample data is provided only if sample data collection is enabled.

Bytes 2-7: Reserved

Bytes 8-15: DCSS name
is the DCSS name in EBCDIC, left-justified and padded with blanks.

IUCV QUIESCE

An application can quiesce its path to *MONITOR.

When running in exclusive mode:

- Monitor suspends data recording until the user resumes. All events being monitored that occur while the user is quiesced are lost.
- Data currently residing in the DCSS remains there until the user resumes.

When running in shared mode:

- Monitor does not suspend data recording. Event monitoring continues, and any event data is saved for the user. Sample recording also continues; however, no data is saved for the user.
- While quiesced, monitor purges any messages that become due.

  Note: If a message is purged, the content of the corresponding pages in the DCSS is unpredictable.
- When the user resumes, he receives any saved event data. He receives new sample data when the next sample interval expires.

IUCV REPLY

REPLYs are expected for IUCV messages that provide the location of records (configuration or monitor data) to indicate that the application is finished accessing the records. Termination messages (IPTRGCLS='ST' or 'ET' do not require a REPLY, although it is recommended to release the storage used by CP for the messages and to decrement the user's count of outstanding IUCV messages.

Note: If the user's count of outstanding messages reaches his IUCV message limit, *MONITOR cannot send any more IUCV messages until the user issues an IUCV REPLY or REJECT.

The length of time that a user has to reply to record notification messages depends on the type of data (configuration or monitor data). The action monitor takes when a reply is delinquent depends on the mode monitor is running in.

The application should always check the condition code after an IUCV REPLY to a record notification message. A zero condition code indicates all is well; a condition code of 1 with an IPRCODE=9, indicates that *MONITOR purged the message and that the DCSS pages pointed to by the message were released for reuse.

Reply to Configuration Notification

The user has until the configuration time limit expires, as established by the MONITOR SAMPLE/EVENT CONFIG LIMIT command, to reply. If the user does not reply within the limit, then monitor stops when running in exclusive mode and issues a message. When running in shared mode, monitor purges the message, releases the corresponding DCSS pages, and issues another message. Data recording continues.

Reply to Data Notification

Exclusive Mode

When a user in the exclusive mode is late in replying to a sample data send, monitor issues message 6243I and gives the user an additional interval in which to reply. If he does not reply by the end of the second interval, message 6244I is issued and sample monitor is stopped. If the user is quiesced when late, the reply is not due until the end of the interval during which the path is resumed.

For event monitoring, the exclusive user has until the event portion of the DCSS becomes full. When full, data recording suspends and waits for a reply. If a reply is not received after one minute, a CP console

message is issued, indicating event recoding has suspended. If no reply is issued after an additional five minutes, event monitoring stops and the message is again issued, indicating that monitoring has stopped this time.

Shared Mode

When a user in the shared mode is late in replying to a sample send, the notification message is purged and message 6274I is issued. New sample records replace the current records in the DCSS. No grace periods are ever given to a user in shared mode, and monitoring is never stopped.

Monitor considers a user in shared mode late for event replies when the DCSS has been completely allocated. At this time, notification messages of users with the most outstanding or pending messages are purged, the corresponding DCSS pages released, and a message issued to the user. Messages are purged even if the user is quiesced. Again, monitor does not suspend or stop for a user when running in shared mode.

IUCV RESUME

When an application resumes its path to *MONITOR, IUCV messages to this application are resumed, unless monitoring is stopped. If the user was quiesced when monitoring was started, new configuration records are created and an IUCV message is sent to the user. If monitor is running in exclusive mode, data recording resumes.

If monitor is running in shared mode, a message is issued if he had missed any sample data sends while he was quiesced. Also, any event messages that were saved for him are sent.

IUCV SEVER

An application issues IUCV SEVER to end communication with *MONITOR.

For a usual termination, the application should place a X'FFFF' in the first two bytes of the field indicated by the USERDTA parameter. If the first 2 bytes contain anything other than X'FFFF', a CP message is issued to the primary system operator and to the virtual machine severing from *MONITOR, indicating that the monitor IUCV path was abnormally terminated by the user and displaying the hexadecimal value found in the 2 bytes.

Monitor returns to *pending* state if no other users are connected; and, if other users are connected, the SEVER is transparent to them.

IUCV Functions Used by the *MONITOR

*MONITOR uses the following IUCV functions: ACCEPT, SEND, and SEVER.

IUCV ACCEPT

If all the protocol to connect to *MONITOR has been followed by the virtual machine, *MONITOR responds to a connect request by issuing IUCV ACCEPT with:

- QUIESCE=YES, which prevents the virtual machine from being able to SEND messages to *MONITOR.
- IPMSGLIM= the message limit specified by the virtual machine on its CONNECT. This limits the number of outstanding messages that *MONITOR can SEND to the virtual machine at any one time.

If the virtual machine used the shared CONNECT parameter list (see “CONNECT Interface for Shared Mode” on page 172), *MONITOR provides the following information in the IPUSER field. Otherwise, the IPUSER field of the ACCEPT parameter list contains hexadecimal zeros.

Version
is the version code. It must be X'01'.

Event
is the event status at the time the IUCV ACCEPT is issued:

Code
Meaning

X'00'
Event monitoring is inactive (has not been started).

X'04'
Event recording is active.

X'08'
Event recording is pending. If this connection is for event data, and sample data is not currently in the event pages of the DCSS, event recording becomes active when this connection completes.

X'0C'
Event recording is suspended because there are no CP frames available to hold event records.

Sample
is the sample status at the time the IUCV ACCEPT is issued:

Code
Meaning

X'00'
Sample monitoring is inactive (has not been started).

X'04'
Sample recording is active.

X'08'
Sample recording is pending. If this connection is for sample data, sample recording becomes active when this connection completes.

IUCV SEND

*MONITOR issues IUCV SENDs to provide:

- The location of records in the DCSS. The records may be either configuration records or monitor data records.
- Termination notification. Termination messages (for sample or event) indicate that there are no more record notification messages for this type of data.

*MONITOR sends all messages with TYPE=2WAY, PRTY=NO, DATA=PRMMSG. Information is provided to the application in the IPTRGCLS and IPRMMSG fields of the IPARML.

The IPTRGCLS Field

This field indicates whether the message is for sample or event monitoring, and whether it is a record notification or a termination message. IPTRGCLS is a fullword which is formatted as follows:

Bytes
Meaning

0 - 1
Message type (EBCDIC)

2 - 3
Reserved.

Descriptions of the first 2 bytes of the IPTRGCLS field for each message type:

C'S'
Sample records are available.

C'SP'
Sample records are available, but the data is incomplete because there was not enough space in the sample area of the DCSS to contain all the records. Any records which could not be written are lost.

C'ST'
Sample data notification has been terminated.

C'E'
Event data is available.

C'EI'
Event data is available for the sample interval that has just expired.

C'EP'
Event configuration records are available, but the data is incomplete because there was not enough space in the event partition of the DCSS to contain all the records. Any records which could not be written are lost.

Note: 'EP' messages do not apply to event data recording. If there is not enough room for event data, event monitoring is suspended (exclusive mode), or event IUCV messages are purged and corresponding pages are made available for re-use (shared mode).

C'ET'
Event data notification has been terminated.

The IPRMMSG Field

This field is 2 fullwords. For a record notification message, the first word contains the guest real address of the first byte of the monitor control area, and the second word contains the guest real address of the last byte of the monitor control area. For termination messages, the IPRMMSG field contains hexadecimal zeros. "The Monitor Control Area" on page 177 gives the layout of the monitor control area.

Note: If the number of outstanding messages on a path reaches the message limit specified for that path on the Connect request, *MONITOR cannot send any further IUCV messages on that path until the application issues an IUCV REPLY or REJECT to one or more of the messages from *MONITOR.

While a user's path is at its message limit, event data messages and the corresponding records in the DCSS will be saved, unless the event area in the DCSS becomes full. Any termination or configuration messages or both will be delayed until the user responds and these messages can be sent. Configuration records will be created at that time. Any sample data messages will be lost.

IUCV SEVER

*MONITOR severs a path in the following circumstances:

- To reject a CONNECT request
- In response to an IUCV SEVER by a virtual machine
- In response to certain error conditions:
  - If the user logs off or purges the monitor DCSS while connected to *MONITOR
  - If a monitor soft abend occurs.

IPUSER SEVER Codes

When *MONITOR severs a path, it places one of the following hex codes in the left-most byte of the IPUSER field.

Code
Meaning

X'00'
User initiated sever.

X'04'
The IUCV message limit is less than 6.

X'08'
The connecting virtual machine cannot accept data in a parameter list.

X'0C'
The first type 'SC' page range in the DCSS *dcssname* is less than 11 pages.

X'10'
DCSS *dcssname* not found or the *dcssname* is a space name.

X'14'
DCSS *dcssname* does not contain a page range of the type 'SC'.

X'18'
Connecting virtual machine does not have DCSS *dcssname* loaded.

X'1C'
Monitor has taken a soft abend.

X'20'
The virtual machine is already connected to *MONITOR.

X'24'
The version code in the IPUSER field of the IUCV CONNECT parameter list is unusable. It must be X'01' unless the first 8 bytes of the field contain a *dcssname* in EBCDIC.

X'28'
DCSS *dcssname* does not match the DCSS name already established by a current connection to *MONITOR.

X'2C'
DCSS *dcssname* is not large enough to satisfy the CONFIG, PARTITION, and BLOCK requirements currently established by options (or defaults) on the MONITOR command for the types of monitoring that has been started.

X'30'
No data type was specified in a shared CONNECT parameter list. At least one type must be specified.

X'34'
Exclusive connection request is rejected because monitor is currently running in shared mode.

X'38'
Connection request is rejected because another virtual machine is currently connected in exclusive mode.

X'3C'
dcssname is not a saved segment.

The Monitor Control Area

*MONITOR informs the virtual machine through IUCV SEND when sample or event data are available. The information is in the form of an immediate message containing two 31-bit addresses. IPRMMSG1 in the IPARML contains the first address; IPRMMSG2 contains the second address. These addresses point to the start and end of a monitor control area. This area is made up of one or more monitor control elements. These elements are three full words in size.

Word
Meaning

0
A type field made up of the following four bytes:

Byte
Definition

0
Flag for sample or event data where *S* stands for sample and *E* stands for event.

1-2
Monitor domains contained in between the addresses given. The following represents the bit definitions for this byte:

Bit
Domain

0
System

1
Monitor

2
Scheduler

3
Storage

4
User

5
Processor

6
I/O

7
Seek

8
Virtual Network

9
(Reserved for IBM use)

10
APPLDATA

11-15
(Reserved for IBM use)

3
Sequence number of sample in interval.

1
31-bit guest real address of the start of the monitor records.

2
31-bit guest real address of the last byte of the last monitor record.

For the virtual machine to obtain the monitor records it must do the following:

1. From the control area element, get the start address of the monitor record.
2. The monitor records then can be crossed through by picking up the length field from the header of the monitor record and adding it to the virtual address. This gives the virtual machine the address of the next monitor record.
3. If the entire monitor record will not fit in the current guest's virtual page, then an end-of-frame record is placed into this virtual page; the application program must move to the next virtual page to start obtaining the monitor records again.
4. The next monitor record address calculated by the previous step being greater than the end address given indicates that the last record pertaining to this control area element has been reached.

A picture of the control area follows:

Table 11. Control area

Fullword 1	**Fullword 2**	**Fullword 3**
Type	Start Address	End Address
Type	Start Address	End Address
Type	Start Address	End Address

A picture of the monitor record data area follows:

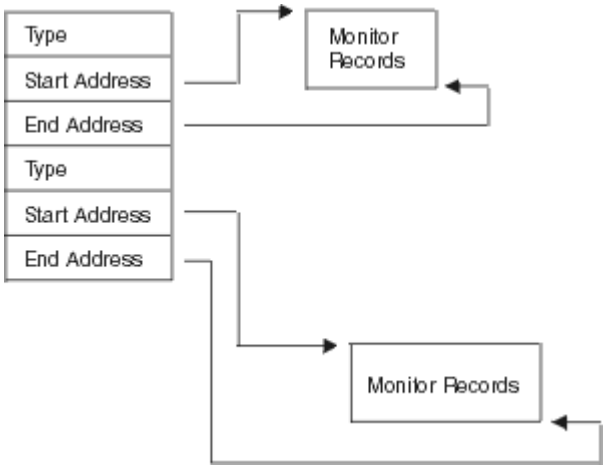


Figure 14. Vertical View of the Monitor Control Area

The virtual machine must be aware of the following conditions:

- Because a MONITOR STOP command can be issued at any time from a privileged user, the monitor control area and the data pointed to by the control area may become zero at any time. An application program processing the monitor data should always check the IUCV return code from each REPLY issued. A return code of 1 with an IPRCODE=09 indicates that the data processed for that REPLY may have become zero or may have been replaced.
- The control area and each set of monitor records pointed to by the control area is contiguous in storage. However, it may not be true that the control area and the set of monitor records as a whole group are contiguous in storage.
- The TOD clock times in the monitor records for the interval will not all be exactly the same time. Each monitor record will have its own TOD time when it was created. Therefore, it is natural for there to be some finite amount of time difference in the first monitor record's TOD and the last monitor record's TOD.
- The monitor records in the discontiguous saved segment may not necessarily be contiguous by domain nor by record number within a domain.

Appendix B. The MONWRITE Program

PI

An application program can read the records from the file that MONWRITE creates and perform data reduction on the performance data found within those records. This section presents information on how to do this.

MONWRITE is an IBM-supplied program that can serve as the application program required for retrieving monitor data from the monitor saved segment. MONWRITE runs in a virtual machine and is implemented in a CMS module which can be called by the MONWRITE command. MONWRITE processes records in the monitor save segment.

For more information on the MONWRITE command, see *z/VM: CP Commands and Utilities Reference*. For a sample of directory entries that a virtual machine running MONWRITE would need, see "The Virtual Machine to Collect Data Records" on page 89.

For a full description of the *MONITOR system service, see Appendix A, "Monitor System Service (*MONITOR)," on page 171.

What Output from MONWRITE Looks Like

The MONWRITE output file contains control records and monitor data records. MONWRITE writes the pages containing these records to the output file in the same order as they appear in the monitor saved segment.

Output File Order

The sequence of output file records is as follows:

1. A 4 KB control record
2. The monitor data records associated with that control record
3. The next 4 KB control record
4. The monitor data records associated with that control record
5. Succeeding control records and the monitor data records associated with each control record
6. A 4 KB end-of-data record.

The maximum record size is 28 KB. (The maximum number of pages in a record is 7.)

Figure 15 on page 181 illustrates the MONWRITE output file order.

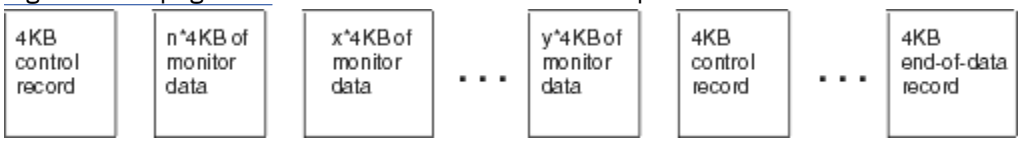


Figure 15. Sequence of Records in the Monitor Writer Output File

n, x and y indicate the number of 4 KB pages that have been grouped into a single output file record.

Contents of Each 4 KB Monitor Control Record

Each 4 KB control record contains:

- A message pending interrupt buffer
- The control area, which contains one or more control area entries, which are copied unchanged from the monitor saved segment. Each control area entry contains:

– The guest real address (location in the saved segment) of the first byte of monitor data

  The offset of this address is also the offset in the next output record where the monitor records start.

– The guest real address (location in the saved segment) of the last byte of monitor data.

From these two addresses, you can determine the number of 4 KB records associated with this control area entry (by subtracting the starting page number from the ending page number and then adding one).

Putting *MONITOR Data into the MONWRITE Control Record

Figure 16 on page 182 illustrates how the MONWRITE program puts the data presented by the *MONITOR system service into the MONWRITE control record. The *MONITOR system service sends the control area delimiters by means of the external interrupt buffer that results from the IUCV SEND command. The control area and the pages with the monitor data reside in the monitor saved segment.

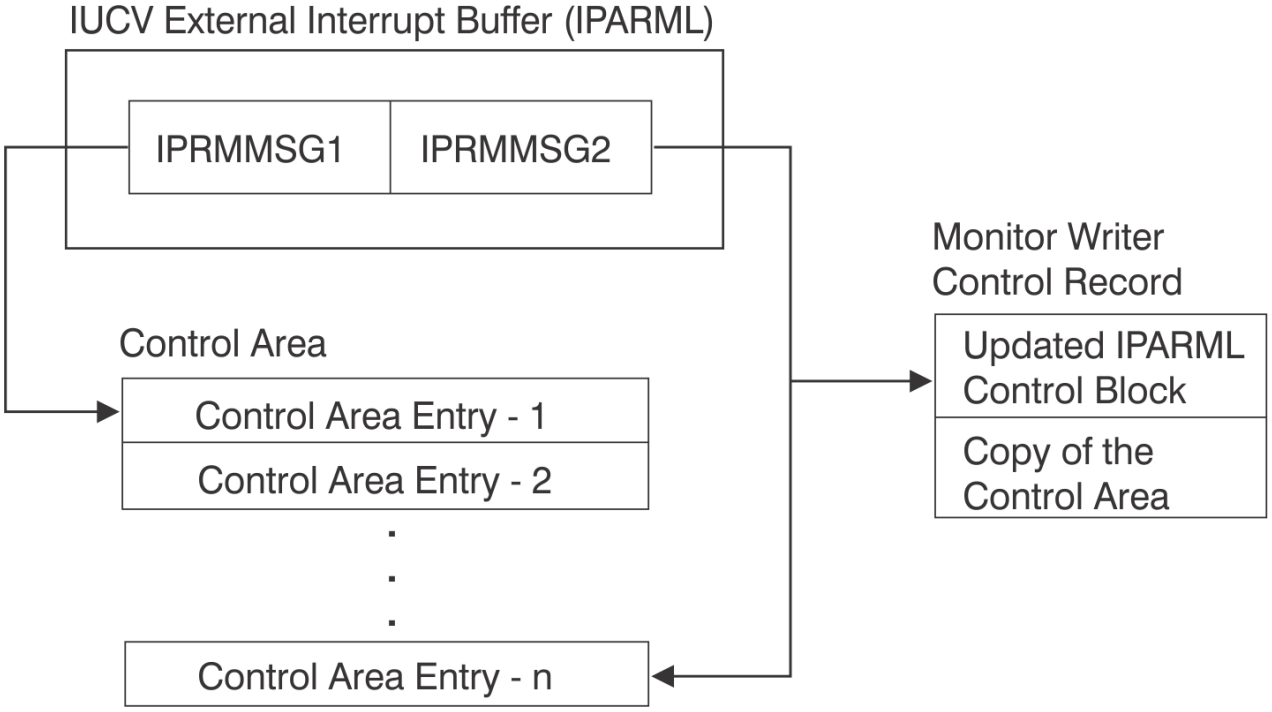


*Figure 16. Putting *MONITOR Data into the MONWRITE Control Record*

Figure 17 on page 183 illustrates the details of the control records.

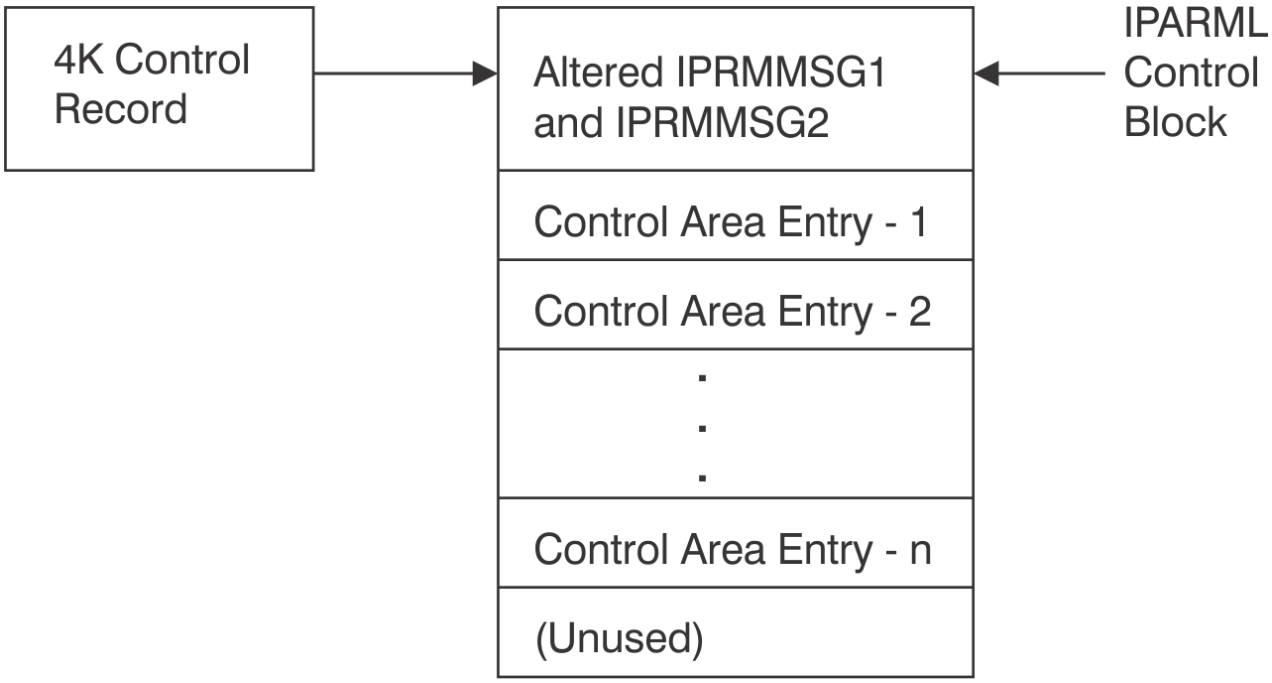


Figure 17. Details of the Control Record within the Output File

The control record includes an updated version of the message pending external interrupt buffer, received as a result of an IUCV SEND. This area, mapped by the IPARML control block, contains:

- IPRMMSG1 - the displacement to the first byte of the control area. IPRMMSG1 is a fullword containing the offset from the start of this 4 KB control record to the start of the control area.
- IPRMMSG2 - the displacement to the last byte of the control area. IPRMMSG2 is a fullword containing the offset from the start of this 4 KB control record to the last byte of the control area.
- All other fields from the external interrupt buffer, which are copied unchanged.

The End-of-Data Record

A special version of the control record represents end-of-data. When the IPRMMSG1 and IPRMMSG2 fields are zero, the control record marks end-of-data. All other fields in the end-of-data control record are unpredictable.

Note: There may be multiple end-of-data records between monitoring sessions. An application processing the MONWRITE output should continue until the physical end-of-file is detected.

MWTBK DSECT and IPARML DSECT Files

The COPY files, HCPMWTBK and IPARML, for the MWTBK DSECT (the MONWRITE control record) and the IPARML DSECT are located in the HCPGPI macro library.

The MONWRITE Control Record Block (MWTBK)

The MONWRITE Control Record Block describes the monitor data stored immediately after this record in the output file created by the MONWRITE program. See "MWTBK - Monitor Writer Control Record Block" on page 183 for a detailed description of MWTBK DSECT.

MWTBK - Monitor Writer Control Record Block

```
DSECT NAME - HCPMWTBK
DESCRIPTIVE NAME - Monitor Writer Control Record Block
DSECT NAME - MWTBK
FUNCTION - The monitor writer control record is the record
           that describes the monitor data stored
```

```
             immediately after this record in the output file
             created by the monitor writer.
  LOCATED BY - none.
  CREATED BY - HCPMOWTR, monitor writer function (MONWRITE)
  DELETED BY - none.
```

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	BITSTRING	40	MWTEXTBF	Message pending external interrupt buffer. This is mapped by the IPARML control block.
40	(28)	BITSTRING	4056	MWTCAREA	Control area for Monitor data that follows this record.
40	(28)	SIGNED	4	MWTCAENT (3)	Control area entry
40	(28)	SIGNED	4	MWTCADOM	Domain information
44	(2C)	SIGNED	4	MWTCASTR	Start address for the monitor records associated with this control area entry.
48	(30)	SIGNED	4	MWTCAEND	End address for the monitor records associated with this control area entry.
		EXPRESSION		MWTCALEN	"*-MWTCAENT" Length of each control area entry
40	(28)	BITSTRING	1	MWTCAFLG	Type of monitor data
41	(29)	BITSTRING	1	MWTCADMA	Domains whose data is given in this control area entry
42	(2A)	BITSTRING	1	MWTCADMB	Second byte of domains whose data is given in this control area entry
43	(2B)	BITSTRING	1		Reserved for IBM use
		EXPRESSION		MWTLENTH	"*-MWTBK" The MWTBK is always 4 KB long but may have unused areas depending on the number of control area entries within a given record.

CODES DEFINED IN MWTCAFLG

Code	Name	Description
111. ..1.	MWTCASMP	X'E2' Sample data
11.. .1.1	MWTCAEVT	X'C5' Event data

BITS DEFINED IN MWTCADMA

Bits	Name	Description
1...	MWTSYSTM	X'80' System domain
.1..	MWTMONTR	X'40' Monitor domain
..1.	MWTSCHED	X'20' Scheduler domain
...1	MWTSTORE	X'10' Storage domain
.... 1...	MWTUSER	X'08' User domain
.... .1..	MWTPROC	X'04' Processor domain

Bits	Name	Description
.... ..1.	MWTIO	X'02' I/O domain
....1	MWTSEEKS	X'01' Seeks domain

BITS DEFINED IN MWTCADMB

Bits	Name	Description
1...	MWTVND	X'80' Virtual Network domain
.1..	MWTISFC	X'40' ISFC domain
..1.	MWTAPPL	X'20' Appldata domain
...1	MWTSSI	X'10' SSI domain

PI end

Appendix C. CP Monitor Records

PI

This topic provides a general description of the CP monitor records.

Where to Find the Layouts of the CP Monitor Records

The layouts of the CP monitor records can be found at:

z/VM Data Areas, Control Blocks, and Monitor Records (https://www.vm.ibm.com/pubs/ctlblk.html)

General Description of the CP Monitor Records

The monitor records generated by the z/VM Monitor Facility are placed in the saved segment defined for Monitor. The starting and ending addresses of the data are communicated to application programs by the *MONITOR CP system service.

The records are arranged by domain number and, within each domain, by record number. The domains associated with each number are:

Number	Domain Name
0	System
1	Monitor
2	Scheduler
3	Storage
4	User
5	Processor
6	I/O
7	Seek
8	Virtual Network
9	ISFC
10	Application data
11	SSI

The placement of the records within the saved segment is not guaranteed to be in any particular order by domain or within domains. This is especially true for event processing. Unless otherwise stated, all counters are cumulative counters. That is, the application program processing the data must subtract an interval's data from the previous interval value to determine how much a particular counter changed from one interval to the next. Cardinal counters are counters whose values are incremented or decremented, such as the number of logged-on users. Cardinal counters, therefore, represent a state of the system at the time it is sampled.

Monitor Records File Content

The information given in Monitor Records includes:

- Monitor Records
  - General information about the monitor records
  - A layout of each record.

Table 12 on page 188 shows the layout of the header data used for each monitor record, which consist of the following:

Table 12. Format for the CP Monitor Records Header

Data Item	Number of Bytes
Record length in bytes	2
Field of zeros	2
Domain identifier	1
Reserved for IBM use	1
Record identifier	2
Time at which this record was built.	8
Reserved for IBM use	4

List of CP Monitor Records

A list of z/VM CP monitor records follows.

Note: If MONITOR EVENT ENABLE COMMAND is in effect, the records marked with an asterisk (*) in this list are collected as command events whether or not event recording for their domains is enabled.

Header

MRRECHDR - Monitor Record Header

Domain 0 (System)

Record 1 - MRSYTSYP - System Data (per processor)
Record 2 - MRSYTPRP - Processor Data (per processor)
Record 3 - MRSYTRSG - Real Storage Data (global)
Record 4 - MRSYTRSP - Real Storage Data (per processor)
Record 5 - MRSYTXSP - Expanded Storage Data (per processor) (No longer available)
Record 6 - MRSYTASG - Auxiliary Storage (global)
Record 7 - MRSYTSHS - Shared Storage Data
Record 8 - MRSYTUSR - User Data
Record 9 - MRSYTCPC - Channel Path Contention Data
Record 10 - MRSYTSCG - Scheduler Activity (global)
Record 11 - MRSYTCOM - Processor Communication Activities (per processor)
Record 12 - MRSYTUWT - User wait states
Record 13 - MRSYTSCP - Scheduler Activity (per processor)
Record 14 - MRSYTXSG - Minidisk Cache
Record 15 - MRSYTCUG - Logical Partition Configuration
Record 16 - MRSYTCUP - CPU Utilization in a Logical Partition
Record 17 - MRSYTCUM - Physical CPU Utilization Data for LPAR Management
Record 18 - MRSYTCPM - Channel Path Measurement Data
Record 19 - MRSYTSYG - System Data (global)
Record 20 - MRSYTEPM - Extended Channel Path Measurement Data (per channel)
Record 21 - MRSYTSXG - System Execution Space (global)
Record 22 - MRSYTSXP - System Execution Space (per processor)
Record 23 - MRSYTLCK - Formal Spin Lock Data (global)
Record 24 - MRSYTSPT - Scheduler Activity (per processor type)
Record 25 - MRSYTPOW - Power Consumption Information

Domain 1 (Monitor)

Record 1 - MRMTREPR - Event Profile
Record 2 - MRMTRECM - Event Alteration command
Record 3 - MRMTRSUS - Suspension Record
Record 4 - MRMTRSYS - System Configuration Data
Record 5 - MRMTRPRP - Processor Configuration (per processor)
Record 6 - MRMTRDEV - Device Configuration Data
Record 7 - MRMTRMEM - Memory Configuration Data
Record 8 - MRMTRPAG - Paging Configuration Data
Record 9 - MRMTRSPR - Sample Profile
Record 10 - MRMTRSCM - Sample Alteration command
Record 11 - MRMTREND - Interval End
Record 12 - MRMTRSOS - Event Record Start of suspend
Record 13 - MRMTREOF - End of Frame Indicator
Record 14 - MRMTRDDR - Domain Detail
Record 15 - MRMTRUSR - Logged on User
Record 16 - MRMTRSCH - Scheduler Settings - Sample Record
Record 17 - MRMTRXSG - Expanded Storage Data (no longer available)
Record 18 - MRMTRCCC - CPU Capability Change
Record 19 - MRMTRQDC - QDIO Device Configuration
Record 20 - MRMTRHPP - HyperPAV Pool Definition
Record 21 - MRMTRMCC - Memory Configuration Change
Record 22 - MRMTRSTP - Server Time Protocol Event Record
Record 23 - MRMTRISC - ISFC End Point Configuration
Record 24 - MRMTRILC - ISFC Logical Link Configuration
Record 25 - MRMTRSSI - SSI Configuration
Record 26 - MRMTRTOP - System Topology Configuration
Record 27 - MRMTRPCI - PCI function Configuration Data
Record 28 - MRMTRCPC - CPU Pool Configuration
Record 29 - MRMTRCPD - CPU Pool Definition - Event Record
Record 30 - MRMTRSRV - Service Configuration Sample Record
Record 32 - MRMTRCHC - CHPID in use by EDEVICEs - Configuration Record
Record 33 - MRMTRFCC - FCP device in use by EDEVICEs - Configuration Record
Record 34 - MRMTRENC - Encrypted Service Event
Record 35 - MRMTRPCC - Protection Change Command
Record 36 - MRMTRAZN - Available Zone Information

Domain 2 (Scheduler)

Record 1 - MRSCLRDB - Begin Read - Event Record
Record 2 - MRSCLRDC - Read Complete - Event Record
Record 3 - MRSCLWRR - Write Response - Event Record
Record 4 - MRSCLADL - Add User To Dispatch List - Event Record
Record 5 - MRSCLDDL - Drop User From Dispatch List - Event Record
Record 6 - MRSCLAEL - Add User To Eligible List - Event Record
*Record 7 - MRSCLSRM - SET SRM Changes - Event Record
Record 8 - MRSCLSTP - System Timer Pop - Event Record
*Record 9 - MRSCLSHR - SET SHARE Changes - Event Record
*Record 10 - MRSCLSQD - SET QUICKDSP Changes - Event Record
*Record 11 - MRSCLIOP - I/O Priority Changes
*Record 12 - MRSCLSCA - SET CPFAFFINITY Changes
Record 13 - MRSCLALL - Add VMDBK to the limit list - Event Record
Record 14 - MRSCLDLL - Drop VMDBK from the limit list - Event Record

Domain 3 (Storage)

Record 1 - MRSTORSG - Real Storage Management (global)
Record 2 - MRSTORSP - Real Storage Activity (per processor)

Record 3 - MRSTOSHR - Shared Storage Management (per NSS or DCSS)
Record 4 - MRSTOASP - Auxiliary Storage Management (per exposure)
*Record 5 - MRSTOSHS - NSS/DCSS Saved
*Record 6 - MRSTOSHP - NSS/DCSS Successfully Purged
*Record 7 - MRSTOATC - Attach of CP Volume
Record 8 - MRSTOBPG - Block Paging Data
Record 9 - MRSTOXSG - Expanded Storage Data (no longer available)
Record 10 - MRSTOXSU - Expanded Storage Data (per user) (no longer available)
Record 11 - MRSTOASS - Auxiliary Storage/Shared Device Mgmt (per exposure)
Record 12 - MRSTOASC - Address Space Created
Record 13 - MRSTOASD - Address Space Deleted
Record 14 - MRSTOASI - Address Space Information Record
Record 15 - MRSTOSHL - NSS/DCSS/SSP Loaded into Storage
Record 16 - MRSTOSHD - NSS/DCSS/SSP Removed From Storage
Record 17 - MRSTOVDK - Virtual Disk in Storage Information Record
Record 18 - MRSTOSCS - SCSI Storage Pool Sample
Record 19 - MRSTOSXG - System Execution Space (global)
Record 20 - MRSTOSXP - System Execution Space (per processor)
*Record 21 - MRSTOADD - Central Storage Added to Real Memory
Record 22 - MRSTORST - Central Storage Add or Remove Started
Record 23 - MRSTOREM - Central Storage Removed from Real Memory
Record 24 - MRSTORCP - Reconfigurable Storage Converted to Permanent Storage
Record 25 - MRSTOAZN - Available Zone Information (global)

Domain 4 (User)

*Record 1 - MRUSELON - User Logon - Event Record
*Record 2 - MRUSELOF - User Logoff Data - Event Record
Record 3 - MRUSEACT - User Activity Data
Record 4 - MRUSEINT - User Interaction Data
*Record 5 - MRUSEDFC - DEFINE CPU - Event Record
*Record 6 - MRUSEDTC - DETACH CPU - Event Record
*Record 7 - MRUSERDC - DEFINE CPU n AS - Event Record
Record 8 - MRUSETRE - User Transaction End - Event Record
Record 9 - MRUSEATE - User Activity data at Transaction End - Event Record
Record 10 - MRUSEITE - User Interaction data at Transaction End - Event Record
*Record 11 - MRUSERLS - Guest Relocation Started - Event Record
*Record 12 - MRUSERLE - Guest Relocation Ended - Event Record
*Record 13 - MRUSECPC - CPU Pool Change - Event Record
*Record 14 - MRUSESCP - SCP Identification

Domain 5 (Processor)

*Record 1 - MRPRCVON - Vary On Processor - Event Data
*Record 2 - MRPRCVOF - Vary Off Processor - Event Data
Record 3 - MRPRCPRP - Processor Data (per processor)
Record 4 - MRPRCVFN - Vary On Vector Facility (no longer available)
Record 5 - MRPRCVFF - Vary Off Vector Facility (no longer available)
Record 6 - MRPRCCFN - Vary On Crypto Facility Event Data (no longer available)
Record 7 - MRPRCCFF - Vary Off Crypto Facility Event Data (no longer available)
Record 8 - MRPRCIOP - I/O Processor (IOP) Utilization
Record 9 - MRPRCAPC - Crypto Performance Counters
Record 10 - MRPRCAPM - Crypto Performance Measurement Data
Record 11 - MRPRCINS - Instruction Counts (per processor)
Record 12 - MRPRCDIA - Diagnose Counts (per processor)
Record 13 - MRPRCMFC - CPU-Measurement Facility Counters
Record 14 - MRPRCTOP - System Topology
Record 15 - MRPRCDSV - Dispatch Vector Assignments (Event)

Record 16 - MRPRCPUP - Park/Unpark Decision (Event)
Record 17 - MRPRCRCD - Real CPU Data (per CPU) (Sample)
Record 18 - MRPRCDHF - Dispatch Vector High Frequency Data (Sample)
Record 19 - MRPRCCPU - CPU Pool Utilization (Sample)
Record 20 - MRPRCMFM - MT CPUMF Counters
*Record 21 - MRPRCSMT - SMT Configuration Change Event
Record 22 - MRPRCSXL - Shared-Exclusive Spin Lock Utilization (per-processor)

Domain 6 (I/O)

*Record 1 - MRIODVON - Vary On Device - Event Data
*Record 2 - MRIODVOF - Vary Off Device - Event Data
Record 3 - MRIODDEV - Device Activity
Record 4 - MRIODCAD - Cache Activity Data
*Record 5 - MRIODATD - Attach Device - Event Data
*Record 6 - MRIODDTD - Detach Device - Event Data
*Record 7 - MRIODENB - Enable terminal - Event Data
*Record 8 - MRIODDSB - Disable terminal - Event Data
*Record 9 - MRIODATS - Attach Shared Device
Record 10 - MRIODALS - Automatic Tape Library Statistics - Event
*Record 11 - MRIODSON - Vary on subchannel- Event
*Record 12 - MRIODSOF - Vary off subchannel - Event
*Record 13 - MRIODMON - Set subchannel measurement on - Event
*Record 14 - MRIODMOF - Set subchannel measurement off - Event
*Record 15 - MRIODDDV - Delete device - Event
*Record 16 - MRIODDDV - Modify device - Event
*Record 17 - MRIODDCH - Delete CHPID - Event
*Record 18 - MRIODTON - Set throttle rate - Event
*Record 19 - MRIODTOF - Set throttle off - Event
Record 20 - MRIODSTC - State change
Record 21 - MRIODVSW - Virtual Switch Activity
*Record 22 - MRIODVSF - Virtual Switch Failure
*Record 23 - MRIODVSR - Virtual Switch Recovery
Record 24 - MRIODSZI - SCSI Device Activity
*Record 25 - MRIODQDA - QDIO Device Activation Event
Record 26 - MRIODQDS - QDIO Device Activity Sample
*Record 27 - MRIODQDD - QDIO Device Deactivation Event
Record 28 - MRIODHPP - HyperPAV Pool Activity
*Record 29 - MRIODHPC - HyperPAV Pool Creation
Record 30 - MRIODLPT - LSS PAV Transition
Record 31 - MRIODMDE - Minidisk Activity
Record 32 - MRIODHPF - HPF Feature Change
*Record 33 - MRIODBPA - Virtual Switch Bridge Port Activation
*Record 34 - MRIODBPD - Virtual Switch Bridge Port Deactivation
Record 35 - MRIODBPS - Virtual Switch Bridge Port Activity
*Record 36 - MRIODPAT - Attach PCI Function
*Record 37 - MRIODPDT - Detach PCI Function
Record 38 - MRIODPEN - Guest Enables a PCI Function
Record 39 - MRIODPAC - PCI Activity
Record 40 - MRIODPDS - Guest Disables a PCI Function
Record 41 - MRIODPER - PCI function error
*Record 42 - MRIODPAD - PCI function added to the system
*Record 43 - MRIODPDL - PCI function deleted from the system
*Record 44 - MRIODPMD - PCI function program controls modified
*Record 45 - MRIODPON - Real PCI function varied on
*Record 46 - MRIODPOF - Real PCI function varied offline
*Record 47 - MRIODCHA - CHPID in use for EDEVICE activity - Event Record
*Record 48 - MRIODCHD - CHPID no longer in use for EDEVICE activity - Event Record

Record 49 - MRIODCHS - EDEVICE FCP CHPID Activity
Record 50 - MRIODFCS - EDEVICE FCP Device Activity
*Record 51 - MRIODFCA - FCP device in use by online EDEVICEs - Event Record
*Record 52 - MRIODFCD - FCP device no longer in use for EDEVICE activity - Event Record
*Record 53 - MRIODSEC - Store Event Channel Report
Record 54 - MRIODVSE - Virtual Switch EQDIO Activity

Domain 7 (Seek)

Record 1 - MRSEKSEK - Seek Data - Event Data

Domain 8 (Virtual Network)

Record 1 - MRVNDSES - Virtual NIC Session Activity
Record 2 - MRVNDLSU - Virtual NIC Guest Link State - Link Up
Record 3 - MRVNDLSD - Virtual NIC Guest Link State - Link Down
Record 4 - MRVNDGLB - Global Virtual Switch Activity

Domain 9 (ISFC)

Record 1 - MRISFISC - ISFC End Point Status Change - Event
Record 2 - MRISFISA - ISFC End Point Activity - Sample
Record 3 - MRISFILC - ISFC Logical Link Definition Change - Event
Record 4 - MRISFNOD - ISFC Logical Link Activity - Sample

Domain 10 (ApplData)

Record 1 - MRAPLEDT - Event Application Data
Record 2 - MRAPLSDT - Sample Application Data

Domain 11 (SSI)

Record 1 - MRSSISCS - State Change Synchronization Activity
Record 2 - MRSSISMI - State/Mode Information
*Record 3 - MRSSISCH - State Change - Event
*Record 4 - MRSSISLT - Slot Definition - Event
Record 6 - MRSSIXLK - XDISK Serialization Activity
Record 7 - MRSSIXDI - XDISK Activity
*Record 8 - MRSSIPDR - SSI PDR volume change

PI end

Appendix D. SFS and CRR Server Monitor Records

PI

Both SFS file pool servers and CRR recovery servers contribute to the CP monitor data by using the APPLDATA domain.

The CRR recovery server's functions reside in an SFS file pool server; therefore, you could have the same server performing both SFS functions and CRR functions (although this is not recommended). Hereafter, when *server* is referred to, it could mean one of the following:

- Dedicated SFS file pool server
- Dedicated CRR recovery server
- A server used for both SFS and CRR.

If you followed the instructions in *z/VM: CMS File Pool Planning, Administration, and Operation*, your server is already properly configured for contributing data to the CP monitor file. The IBM-supplied servers VMSERVU, VMSERVS, and VMSERVR are also properly configured.

To begin data collection, enable the APPLDATA domain and start monitoring (use the CP MONITOR command).

The data records for servers are domain X'A' APPLDATA records, the general format of which is described in HTML format at z/VM Data Areas, Control Blocks, and Monitor Records (https://www.vm.ibm.com/pubs/ctlblk.html).

Each data record consists of these parts:

- Monitor record header
- SFS/CRR APPLDATA record header
- Server header data
- Counter data

The SFS/CRR APPLDATA header data consists of the following:

Table 13. SFS/CRR APPLDATA Header Data

Data Item	Number of Bytes
Byte offset to application data relative to start of this record	2
Length in bytes of application data	2
User ID of the server machine (in EBCDIC)	8
Product identification (in EBCDIC). For SFS and/or CRR records, this field contains “5684112SF1010100”.	16
Status	1
Reserved	3

Following the SFS/CRR APPLDATA header data is the server header data:

Table 14. Server Header Data

Data Item	Number of Bytes
Reserved	4

Table 14. Server Header Data (continued)

Data Item	Number of Bytes
Flags: • Dedicated Maintenance Mode Flag • Coordinated Resource Recovery Flag	1
Reserved	3

The dedicated maintenance mode flag indicates whether the SFS server generates the record while processing in dedicated maintenance mode. You should ignore these records when you are analyzing multiple user mode performance. When the high-order bit (bit 0) of the flag byte is set to 1 (X'80'), the record is generated during dedicated maintenance mode. When the high-order bit is 0, the record is generated during multiple user mode operation.

The Coordinated Resource Recovery flag indicates whether the CRR recovery server generates the record. When bit 1 of the flag byte is set to 1 (X'40'), the CRR recovery server generates the record. When bit 1 of the flag byte is 0, only an SFS file pool server generates the record.

Following the CP header data and the server header data is the counter data. The counters in the CP monitor record are essentially the same counters that are displayed when you enter a QUERY FILEPOOL REPORT command. The QUERY FILEPOOL REPORT command displays a superset of the counters in the CP monitor record. These additional counters are derived from the same data that is made available in the CP monitor records.

Each counter occupies 8 bytes in the CP monitor record. The first byte contain the counter's length (which is always 8) and a number that identifies the counter. The last 4 bytes contain the counter data itself. Table 15 on page 194 shows the data format used for each of the counters.

Table 15. Data Format for Counters

Data Item	Number of Bytes
Length of the counter data (always has a value of 8).	1
Counter ID	2
Reserved	1
Counter data	4

Table 16 on page 194 shows the numeric counter IDs and a descriptions for each counter. All fields are four bytes, unsigned.

If processing back-level file pool information, only the counters applicable to the back-level release are recorded in the CP monitor data.

Table 16. Server Counters

Dec	Hex	Name	Description
1	1	AGENTHWM	CounterID = 1 Active Agents Highest Value - This is the highest number of user agents that were concurrently in use during the current multiple user mode session. The primary use of agents (for both SFS and CRR) is to function as the server's internal representation of a user. When a user requests something of the server, the server assigns that user to an agent and does the work. After completing the work, the server frees the agent and can reuse it for another user.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
2	2	STORHWM	CounterID = 2 Virtual Storage Highest Value - This is greatest amount of virtual storage (in KB) used at any one time. This value includes all virtual storage acquired by the SFS or CRR server since it was last started.
3	3	STORDENY	CounterID = 3 Virtual Storage Requests Denied - This is the number of times the SFS file pool server tried to get virtual storage but could not. In most cases, the server cannot get virtual storage because there is none available in the virtual machine (it is all in use).
4	4	CHKPNTNUM	CounterID = 4 Checkpoints Taken - This is the number of SFS checkpoints taken during the current execution of the SFS file pool server. A checkpoint is an internal server operation during which the changes recorded on the log minidisks are permanently made to the file pool. The SFS file pool server takes checkpoints after a certain number of log minidisk blocks have been written.
5	5	CHKPNTTIM	CounterID = 5 Checkpoint Time (tenths of a millisecond) - This is the total amount of elapsed time (in tenths of milliseconds) spent in doing SFS checkpoints.
6	6	RACFNUM	CounterID = 6 Security Manager Exit Calls - This is the number of times the SFS file pool server called an external security manager.
7	7	RACFTIM	CounterID = 7 Security Manager Exit Time (tenths of a millisecond) This is the total amount of elapsed time (in tenths of a millisecond) spent in processing security manager calls. (The SFS file pool server notes the clock time, calls the external security manager, and notes the clock time again upon return from the external security manager.)
8	8	ERACFNUM	CounterID = 8 External Security Manager Exit Calls - This is the number of times the SFS file pool server called an external security manager and that manager needed to use IUCV to communicate with a security manager running in another virtual machine.
9	9	ERACFTIM	CounterID = 9 External Security Manager Exit Time (tenths of a millisecond) - This is the total amount of elapsed time (tenths of a millisecond) spent in processing authorization requests by an external security manager that needed to use IUCV to communicate with a security manager running in another virtual machine.
10	A	ADDSTOR	CounterID = 10 Add Storage Requests - This is the number of times file pool administrators have requested an increase in a user's space allocation. The MODIFY USER command is used to increase storage.
11	B	CACHEREL	CounterID = 11 Cache Release Requests - This is the number of times the SFS file pool server has been requested to stop sending cache updates to a user machine. The cache in a user machine contains information about the user's accessed directories.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
12	C	CHGTHRESH	CounterID = 12 Change Threshold Requests - This is the number of times that the SFS file pool server was requested to change a user's space allocation threshold percentage (the threshold percentage at which the SFS file pool server produces a space allocation warning message). This counter is incremented when the SET THRESHOLD command is executed.
13	D	CLOSEDIR	CounterID = 13 Close Directory Requests - This is the number of times the SFS file pool server was requested to close a directory.
14	E	CLOSE	CounterID = 14 Close File Requests - This is the number of times the SFS file pool server was requested to close a file.
15	F	COMMIT	CounterID = 15 Commit Requests - This is the number of times the SFS file pool server was requested to commit work.
16	10	CONNECT	CounterID = 16 Connect Requests - This is the number of times the SFS file pool server was requested to accept a connection from a user machine.
17	11	CRALIAS	CounterID = 17 Create Alias Requests - This is the number of times the SFS file pool server was requested to create an alias.
18	12	CRDIRECT	CounterID = 18 Create Directory Requests - This is the number of times the SFS file pool server was requested to create a new directory.
19	13	DELDIRECT	CounterID = 19 Delete Directory Requests - This is the number of times the SFS file pool server was requested to delete a directory.
20	14	DELETE	CounterID = 20 Delete File Requests - This is the number of times the SFS file pool server was requested to erase a base file or alias residing in the file pool.
21	15	DELSTOR	CounterID = 21 Delete Storage Requests - This is the number of times file pool administrators have requested a decrease in a user's space allocation. The MODIFY USER command is used to decrease storage.
22	16	FILECOPY	CounterID = 22 File Copy Requests - This is the number of times that the SFS file pool server was requested to copy a source file in the file pool it is managing to a target file in the same file pool.
23	17	GETDIR	CounterID = 23 Get Directory Requests - This is the number of times the SFS file pool server was requested to read directory records. This is the request that is made by the DMSGETDI Get Directory, DMSGETDA Get Directory - Searchall, DMSGETDD Get Directory - Dir, DMSGETDK Get Directory - Lock, DMSGETDL Get Directory - Alias, DMSGETDS Get Directory - Searchauth, DMSGETDT Get Directory - Auth, and DMSGETDX Get Directory - File Extended CSL routines.
24	18	GETDIRENT	CounterID = 24 Get Directory Entry Requests - This is the number of times that the SFS file pool server was requested to provide information about a single directory entry. This request is made by the DMSEXIST Exist, DMSEXIDI Exist Directory, and DMSEXIFI Exist Files CSL routines.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
25	19	GRANTADMIN	CounterID = 25 Grant Administrator Authorization Requests - This is the number of times the SFS file pool server has been requested to enroll a file pool administrator. This request is caused by ENROLL ADMINISTRATOR command.
26	1A	GRANTAUTH	CounterID = 26 Grant Authorization Requests - This is the number of times the SFS file pool server has been requested to grant authority on an object to a user.
27	1B	GRANTUSER	CounterID = 27 Grant User Connect Requests - This is the number of times the SFS file pool server was requested to enroll a user in the file pool. This request is caused by ENROLL USER command.
28	1C	LOCK	CounterID = 28 Lock Requests - This is the number of times the SFS file pool server has been requested to lock objects in the file pool. (Note that DISABLE operator commands are considered lock requests.) This counter does not include the number of implicit locks the SFS file pool server has created.
29	1D	OPENDIR	CounterID = 29 Open Directory Requests - This is the number of times the file pool SFS file pool server was requested to open a directory (to be subsequently used for reading).
30	1E	OPENNEW	CounterID = 30 Open File New Requests - This is the number of times that the SFS file pool server was requested to open a new file (that is, create a new file).
31	1F	OPENREAD	CounterID = 31 Open File Read Requests - This is the number of times the SFS file pool server was requested to open a file for read access.
32	20	OPENREP	CounterID = 32 Open File Replace Requests - This is the number of times the SFS file pool server was requested to open an existing file such that existing records are replaced by records that are added. If the file does not exist, it is created.
33	21	OPENWRITE	CounterID = 33 Open File Write Requests - This is the number of times the SFS file pool server was requested to open a file for write access. (In this case, records can be changed or added to the file without affecting other records in the file.)
34	22	QADMIN	CounterID = 34 Query Administrator Requests - This is the number of times the SFS file pool server was requested to provide information about users with file pool administrator authority.
35	23	QCONNECT	CounterID = 35 Query Connected Users Requests - This is the number of times the SFS file pool server was requested to provide information about users that were connected to it.
36	24	QENROLL	CounterID = 36 Query Enrolled Users Requests- This is the number of times the SFS file pool server was requested to provide information about users that are enrolled in the file pool.
37	25	QLOCK	CounterID = 37 Query Lock Conflicts Requests - This is the number of times the SFS file pool server was requested to provide information about implicit lock conflicts. This request is caused by the QUERY FILEPOOL CONFLICT command.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
38	26	QPOOL	CounterID = 38 Query File Pool Requests - This is the number of times the SFS file pool server was requested to provide information about the status of the file pool. This request is caused either by the QUERY FILEPOOL STATUS, QUERY FILEPOOL REPORT, QUERY FILEPOOL STORGRP, QUERY FILEPOOL OVERVIEW, QUERY FILEPOOL MINIDISK, QUERY FILEPOOL AGENT, QUERY FILEPOOL LOG, QUERY FILEPOOL CATALOG, QUERY FILEPOOL COUNTER, or QUERY FILEPOOL CRR command.
39	27	QSPACE	CounterID = 39 Query User Space Requests - This is the number of times the SFS file pool server was requested to provide information about the space that is allocated to file pool users. This request is caused by the QUERY LIMITS command, DMSQLIMA Query Limits for all Enrolled Users CSL routine, or DMSQLIMU Query Limits for a Single User CSL routine.
40	28	READ	CounterID = 40 Read File Requests - This is the number of times the SFS file pool server was requested to read data from files.
41	29	MRCLOSE	CounterID = 41 Recovery Close Catalog Requests - This is the number of times that the SFS file pool server was requested to close a previously-opened catalog object. These requests are generally made by programs that recover user data.
42	2A	MRGET	CounterID = 42 Recovery Get Catalog Requests - This is the number of times that the SFS file pool server was requested to get catalog information. These requests are generally made by programs that recover user data.
43	2B	MROPEN	CounterID = 43 Recovery Open Catalog Requests - This is the number of times that the SFS file pool server was requested to open a catalog object. These requests are generally made by programs that recover user data.
44	2C	MRPUT	CounterID = 44 Recovery Put Catalog Requests- This is the number of times that the SFS file pool server was requested to write to a catalog object. These requests are generally made by programs that recover user data.
45	2D	REFRSHDIR	CounterID = 45 Refresh Directory Requests - This is the number of times that the SFS file pool server was requested to refresh the information that is maintained in the user machine cache. CMS in the user machine decides when it is necessary to have the cache refreshed based on what the user is doing.
46	2E	RELOCATE	CounterID = 46 Relocate Requests - This is the number of times that the SFS file pool server was requested to move a file or directory sub-tree from one directory to another. This request is caused by the RELOCATE command or DMSRELOC CSL routine.
47	2F	RENAME	CounterID = 47 Rename Requests - This is the number of times the SFS file pool server was requested to rename a file or directory. This request is caused by the RENAME command or DMSRENAM CSL routine.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
48	30	REVOKADMIN	CounterID = 48 Revoke Administrator Authorization Requests - This is the number of times the SFS file pool server was requested to delete a file pool administrator.
49	31	REVOKAUTH	CounterID = 49 Revoke Authorization Requests- This is the number of times the SFS file pool server was requested to revoke authority on a file or directory from a user.
50	32	REVOKEU	CounterID = 50 Revoke User Requests - This is the number of times the SFS file pool server was requested to delete a user from the file pool.
51	33	ROLLBACK	CounterID = 51 Rollback Requests - This is the number of times the SFS file pool server was requested to roll back a logical unit of work. (These are known as voluntary rollbacks.)
52	34	UNLOCK	CounterID = 52 Unlock Requests - This is the number of times the SFS file pool server was requested to delete an explicit lock on a file pool object. This request is caused by DELETE LOCK command and ENABLE command.
53	35	WRITEACCT	CounterID = 53 Write Accounting Requests - This is the number of times the SFS file pool server was requested to write accounting records for a file pool.
54	36	WRITE	CounterID = 54 Write File Requests - This is the number of times the SFS file pool server was requested to write to a file in the file pool.
55	37	REQUESTTIM	CounterID = 55 File Pool Request Service Time (tenths of a millisecond) - This is the total amount of time (in tenths of a millisecond) the SFS file pool server took to process requests. To calculate the service time, the SFS file pool server makes a note of the time whenever it receives a request from a user machine. When it sends the result of the request back to the user machine it again notes the time. It then subtracts the time the request was received from the time the response was sent to determine the service time for that particular request. This counter is the sum of the service times of all the requests the SFS file pool server received since it was last started.
56	38	REMOTEREQ	CounterID = 56 Remote File Pool Requests - This is the total number of file pool requests made by users on different processors.
57	39	ALIASREAD	CounterID = 57 Alias Definitions Examined - This is the number of times the SFS file pool server had to read catalog information about aliases during the processing of all requests.
58	3A	ALIASUPD	CounterID = 58 Alias Definitions Updated - This is the total number of times an alias definition was created, changed, or deleted. An alias definition is changed when any of its attributes are changed (for example, number of records in the file, date, time, record format, and so on).
59	3B	BEGINLUW	CounterID = 59 Begin LUWs - This is the total number of logical units of work (LUWs) that the SFS file pool server has started.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
60	3C	LUWTIME	CounterID = 60 Agent Holding Time (tenths of a millisecond) - This is the total amount of time (in tenths of a millisecond) the server spent holding user agents. The server calculates agent holding time by subtracting the time at which the agent is acquired from the time it is released (made available for other users). This counter is the sum of the agent holding times for all agents that the server has used since it was last started. Note that this counter applies to both SFS file pool servers and CRR recovery servers.
61	3D	LUWROLLB	CounterID = 61 LUW Rollbacks - This is the total number of logical units of work (LUWs) that the server rolled back.
62	3E	SACCALLS	CounterID = 62 SAC Calls - This is the total number of times the server called its Storage Access Component (SAC). SAC is the portion of server code that accesses the catalogs and user files. It also provides support for locking, catalog indexes, logging, file pool recovery, and file pool generation.
63	3F	GRPXLOCK	CounterID = 63 Storage Group Explicit Lock Conflicts - This is the number of times that a request for a lock on a storage group was denied because an explicit lock had already been created on that storage group. The request for the lock could have been either an implicit or an explicit request.
64	40	FSXLOCK	CounterID = 64 File Space Explicit Lock Conflicts - This is the number of times that a request for a lock on a file space was denied because an explicit lock had already been created on that file space. The request for the lock could have been either an implicit or an explicit request.
65	41	DIRXLOCK	CounterID = 65 Directory Explicit Lock Conflicts- This is the number of times that a request for a lock on a directory was denied because an explicit lock had already been created on that directory. The request for the lock could have been either an implicit or an explicit request.
66	42	FILEXLOCK	CounterID = 66 File Explicit Lock Conflicts - This is the number of times that a request for a lock on a file was denied because an explicit lock had already been created on that file. The request for the lock could have been either an implicit or an explicit request.
67	43	GRPLLOCK	CounterID = 67 Storage Group Logical Lock Conflicts - This is the number of times that a request for a lock on a storage group was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that storage group. The request for the lock could have been either an implicit or an explicit request.
68	44	FSLLOCK	CounterID = 68 File Space Logical Lock Conflicts - This is the number of times that a request for a lock on a file space was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that file space. The request for the lock could have been either an implicit or an explicit request.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
69	45	DIRLLOCK	CounterID = 69 Directory Logical Lock Conflicts- This is the number of times that a request for a lock on a directory was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that directory. The request for the lock could have been either an implicit or an explicit request.
70	46	FILELLOCK	CounterID = 70 File Logical Lock Conflicts - This is the number of times that a request for a lock on a file was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that file. The request for the lock could have been either an implicit or an explicit request.
71	47	CATLLOCK	CounterID = 71 Catalog Lock Conflicts - This is the number of times that a request for a lock on a part of a catalog was delayed or denied because an implicit lock had already been created on the desired item.
72	48	LOCKTIME	CounterID = 72 Lock Wait Time (tenths of a millisecond) - This is the total amount of time (in tenths of a millisecond) spent waiting for locks. For each case of a lock wait, the server calculates the lock wait time by subtracting the time at which it suspended processing the request from the time at which it resumed processing the request. This counter is the sum of the lock wait times for all lock wait conditions that occurred since the server was started.
73	49	DEADLOCK	CounterID = 73 Deadlocks - This is the number of deadlocks that the server detected and corrected. A deadlock occurs when two applications are each holding file pool resources that the other needs. The server detects this condition and rolls back the logical unit of work that started most recently.
74	4A	QSAMREQ	CounterID = 74 QSAM Requests - This is the number of QSAM requests the SFS file pool server has made. QSAM requests occur for control data backups and restores. They also occur for security audit trace output.
75	4B	QSAMTIME	CounterID = 75 QSAM Time (tenths of a millisecond) - This is the time (in tenths of a millisecond) that the SFS file pool server waited for all QSAM requests to complete. The SFS file pool server notes the time immediately before making a QSAM request and immediately upon return from QSAM. The difference between the two times is the QSAM Time for a QSAM request.
76	4C	FILBLKRD	CounterID = 76 File Blocks Read - This is the total number of 4 KB blocks read from user storage groups.
77	4D	FILBLKWR	CounterID = 77 File Blocks Written - This is the total number of 4 KB blocks written to user storage groups.
78	4E	CATPAGRD	CounterID = 78 Catalog Blocks Read - This is the total number of 4 KB blocks read from the file pool catalogs (storage group 1).
79	4F	CATPAGWR	CounterID = 79 Catalog Blocks Written - This is the total number of 4 KB blocks written to the file pool catalogs (storage group 1).

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
80	50	CTLBLKRD	CounterID = 80 Control Minidisk Blocks Read - This is the total number of 512-byte blocks read from the file pool control minidisk.
81	51	CTLBLKWR	CounterID = 81 Control Minidisk Blocks Written- This is the total number of 512-byte blocks written to the file pool control minidisk.
82	52	LOGREAD	CounterID = 82 Log Blocks Read - This is the total number of 4 KB blocks read from the SFS log minidisks.
83	53	LOGWRITE	CounterID = 83 Log Blocks Written - This is the total number of 4 KB blocks written to the SFS log minidisks.
84	54	BIOFILRD	CounterID = 84 BIO Requests to Read File Blocks - This is the total number of times block I/O was used to read blocks from user storage groups. Where possible, a SFS file pool server tries to read multiple blocks using a single block I/O request. Therefore, this number is usually less than the File Blocks Read.
85	55	BIOFILWR	CounterID = 85 BIO Requests to Write File Blocks - This is the total number of times block I/O was used to write blocks to user storage groups. Where possible, a SFS file pool server tries to write multiple blocks using a single block I/O request. Therefore, this number is usually less than the File Blocks Written.
86	56	BIOCATRD	CounterID = 86 BIO Requests to Read Catalog Blocks - This is the total number of times block I/O was used to read blocks from the file pool catalogs (storage group 1). The SFS file pool server reads catalog blocks one at a time, as needed.
87	57	BIOCATWR	CounterID = 87 BIO Requests to Write Catalog Blocks - This is the total number of times block I/O was used to write blocks to the file pool catalogs (storage group 1). The SFS file pool server writes catalog blocks one at a time, as needed.
88	58	BIOCTLRD	CounterID = 88 BIO Requests to Read Control Minidisk Blocks - This is the total number of times block I/O was used to read blocks from the file pool control minidisk. In general, the SFS file pool server reads control minidisk blocks one at a time, as needed.
89	59	BIOCTLWR	CounterID = 89 BIO Requests to Write Control Minidisk Blocks - This is the total number of times block I/O was used to write blocks to the file pool control minidisk. Where possible, a SFS file pool server tries to write multiple blocks using a single block I/O request. Therefore, this number is usually less than the Control Minidisk Blocks Written.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
90	5A	BIOTIME	CounterID = 90 Total BIO Request Time (tenths of a millisecond) - This is the total amount of time (in tenths of a millisecond) that agents had to wait because they initiated a block I/O request in the SFS File pool server. When a user agent requests a block I/O operation, the SFS file pool server checks the processor clock, starts the operation, and (rather than waiting for the I/O to complete) begins doing work for other users. After handling requests for other active agents, and the block I/O request has completed, the SFS file pool server dispatches the agent that was waiting. The SFS file pool server then checks the clock again and subtracts the start time from this time to determine how long the user agent waited. It is this value that is added to the counter.
91	5B	SIOFILRD	CounterID = 91 I/O Requests to Read File Blocks - This is the number of I/O requests issued by CP to read blocks from user storage groups. This count will typically be larger than BIO Requests to Read File Blocks because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and all requested blocks are found in the minidisk cache.
92	5C	SIOFILWR	CounterID = 92 I/O Requests to Write File Blocks - This is the number of I/O requests issued by CP to write blocks to user storage groups. This count will typically be larger than BIO Requests to Write File Blocks because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD.
93	5D	SIOCATRD	CounterID = 93 I/O Requests to Read Catalog Blocks - This is the number of I/O requests issued by CP to read blocks from the file pool catalogs (storage group 1). Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and the requested block is found in the minidisk cache.
94	5E	SIOCATWR	CounterID = 94 I/O Requests to Write Catalog Blocks - This is the number of I/O requests issued by CP to write blocks to the file pool catalogs (storage group 1). Each I/O request will normally result in a physical I/O to DASD.
95	5F	SIOCTLRD	CounterID = 95 I/O Requests to Read Control Minidisk Blocks - This is the number of I/O requests issued by CP to read blocks from the control minidisk. Each I/O request will normally result in a physical I/O to DASD.
96	60	SIOCTLWR	CounterID = 96 I/O Requests to Write Control Minidisk Blocks - This is the number of I/O requests issued by CP to write blocks to the control minidisk. This count will typically be larger than BIO Requests to Write Control Minidisk Blocks because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
97	61	RELBLKS	CounterID = 97 Release Blocks Requests - This is the total number of times the SFS file pool server was requested to release (deallocate) blocks that encountered write I/O errors in a specified storage group.
98	62	TCLOSE	CounterID = 98 Temporary Close Requests - This is the total number of times the SFS file pool server was requested to prepare an open file for commit processing.
99	63	SETESMUD	CounterID = 99 SFS Send User Data Requests- This is the total number of times the SFS file pool server was requested to pass user data information to an external security manager.
100	64	PREPARE	CounterID = 100 Prepare Requests - This is the number of times the SFS file pool SFS file pool server was requested to do the first phase of a two-phase commit.
101	65	CHGATTR	CounterID = 101 Change File Attributes Requests - This is the total number of times the SFS file pool server was requested to modify the overwrite and/or recoverability attributes of SFS files.
102	66	MAXCONNHWM	CounterID = 102 Highest MAXCONN used - This is the highest number of APPC/VM (and IUCV) connections currently in by the server machine.
103	67	CRRGETCAP	CounterID = 103 Get Capability Requests - This is the total number of times the CRR recovery server had requests to get the capabilities of a protected conversation partner in a coordinated commit.
104	68	GETLOGNAME	CounterID = 104 Get Logname Requests - This is the total number of times the CRR recovery server had requests to get the CRR recovery server's log name.
105	69	GETLUWID	CounterID = 105 Get LUWID Requests - This is the total number of times the CRR recovery server had requests to create a new SNA LU 6.2 logical unit of work ID (LUWID).
106	6A	RESYNCINIT	CounterID = 106 Resync Init Requests - This is the total number of times the CRR recovery server had requests to do resynchronization activity for a resource, following a resource failure during a coordinated commit.
107	6B	RESYNCPV	CounterID = 107 Resync Protocol Violations Requests - This is the total number of times the CRR recovery server was given information that a communications protocol violation was detected during a synchronization (sync) point.
108	6C	RESYNCQDIR	CounterID = 108 Resync Query Direction Requests - This is the total number of times the CRR recovery server had requests to determine the sync point direction (commit or rollback) by means of resynchronization activity, following a failure during communications with the sync point initiator.
109	6D	CRRLOGWR	CounterID = 109 Write Log Requests - This is the total number of times the CRR recovery server had requests to write to the CRR log.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
110	6E	CRRREQTIME	CounterID = 110 CRR Request Service Time (tenths of milliseconds) - This is the total amount of time (in tenths of milliseconds) the CRR recovery server spent handling CRR requests. For each CRR request, the server calculates the holding time by subtracting the time at which the request begins from the time it ends. This counter is the sum of all CRR requests that the CRR recovery server processed since it was last started.
111	6F	SYNCPOINT	CounterID = 111 Syncpoints - This is the total number of times the CRR recovery server was requested to process a CRR sync point.
112	70	SYNCPTTIME	CounterID = 112 Syncpoint Time (tenths of milliseconds) This is the total amount of time (in tenths of milliseconds) the CRR recovery server spent handling sync points. For each sync point request, the server calculates the holding time by subtracting the time at which the request begins from the time it ends. This counter is the sum of all sync point requests that the CRR recovery server processed since it was last started.
113	71	CRRLOGRES	CounterID = 113 Participating Resources - This is the number of resources that have participated in coordination or recovery. Participating Resources divided by Syncpoints is the average number of different resource managers that have participated in each sync point.
114	72	CRRLOGCKPT	CounterID = 114 Log Checkpoints Taken - This is the total number of times the CRR recovery server had done a CRR log checkpoint.
115	73	CRRLOGIO	CounterID = 115 Log I/O Requests - This the number of I/O requests issued by CP to read or write CRR recovery server log minidisk blocks. Each I/O request will normally result in a physical I/O to DASD.
116	74	CRRBIOTIME	CounterID = 116 BIO Request Time (tenths of milliseconds) - This is the total amount of time (in tenths of milliseconds) that agents had to wait because they initiated block I/O requests to the CRR logs. When a user agent initiates a BIO request, the server checks the processor clock, starts the operation, and (rather than waiting for the BIO request to complete), begins doing work for other users. After handling requests for other active agents, and the block I/O request has completed, the server dispatches the agent that was waiting. The server then checks the clock again and subtracts the start time from this time to determine how long the agent waited. It is this value that is added to the counter.
117	75	DATASPACE	CounterID = 117 Dataspace Requests - This is the total number of times the SFS file pool server was requested to assign or remove a directory from the data space eligibility list. This request is caused by the DATASPACE command.
118	76	DIRATTR	CounterID = 118 Dirattr Requests - This is the total number of times the SFS file pool server was requested to change directory control attributes. This request is caused by the DIRATTR command and by the DMSDIRAT CSL routine.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
119	77	QACCESSORS	CounterID = 119 Query Accessors Requests - This is the total number of times the SFS file pool server was requested to return information about users who are accessing directory control directories. This request is caused by the QUERY ACCESSORS command.
120	78	QDATASPACE	CounterID = 120 Query Dataspace Requests - This is the total number of times the SFS file pool server was requested to return information about the eligibility of directories for use in data spaces. This request is caused by the QUERY DATASPACE command.
121	79	SETREFDATE	CounterID = 121 Set Reference Date Requests- This is the number of times CMS requested the SFS file pool server to change the date of last reference for files that are in a directory mapped to a data space.
122	7A	DIRRESLOCK	CounterID = 122 DIRCONTROL Resource Lock Conflicts - This is the number of times that a request to lock a directory control directory was delayed or denied because an implicit lock had already been created on that directory. SFS uses internal locking mechanisms to maintain the consistency of directory control directories.
123	7B	DEADROLL	CounterID = 123 Rollbacks Due to Deadlock - This is the total number of logical units of work (LUWs) that the SFS file pool server rolled back due to a deadlock that could not be corrected through a retry process.
124	7C	CHANGEDRA	CounterID = 124 Change DFSMS Related Attribute Requests - This is the number of times that the SFS file pool server was requested to change the DFSMS/VM Related Attributes (DRAs) for a specified file or directory.
125	7D	CREATEEO	CounterID = 125 Create External Object Requests - This is the total number of times the SFS file pool server was requested to create an external object in an SFS directory. This request is caused by the DMSCROB CSL routine.
126	7E	CREATEFILE	CounterID = 126 Create File Requests - This is the number of times the SFS file pool server was requested to add a new (empty) file to an SFS directory. This request is caused by the DMSCRFIL CSL routine or the CREATE FILE command.
127	7F	QUERYSG	CounterID = 127 Query User Storage Group Requests - This is the total number of times the SFS file pool server was requested to query a file pool for information about user storage groups. This request is caused by the DMSQUSG CSL routine.
128	80	SENDSMS	CounterID = 128 Send DFSMS Data Requests - This is the number of times that the SFS file pool server was requested to pass user-defined data to the DFSMS/VM exit.
129	81	OPENMIGRAT	CounterID = 129 Migrate Requests - This is the number of times that the SFS file pool server was requested (using the DFSMS MIGRATE or DFSMS MANAGE command) to open a file or alias and prepare to migrate file blocks.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
130	82	OPENRECALL	CounterID = 130 Recall Requests - This is the number of times that the SFS file pool server was requested (using the DFSMS RECALL command, or automatic recalls when files in migrated status are referenced) to open a file or alias and prepare to recall file blocks from migrated status.
131	83	RECALLNUM	CounterID = 131 Recall DFSMS File Exit Calls - This is the number of times that the SFS file pool server called a DFSMS/VM Recall File exit point (to automatically recall a file in migrated status that has been referenced).
132	84	RECALLTIM	CounterID = 132 Recall DFSMS File Exit Time (tenths of milliseconds) - This is the total amount of elapsed time (in tenths of milliseconds) spent in processing DFSMS/VM Recall File Exit calls.
133	85	OSMSEXNUM	CounterID = 133 Other DFSMS Exit Calls - This is the total number of times the SFS file pool server called one of the following predefined DFSMS/VM exits.
134	86	OSMSEXTIM	CounterID = 134 Other DFSMS Exit Time (tenths of milliseconds) - This is the total amount of time (in tenths of milliseconds) spent in processing one of the following predefined DFSMS/VM exits.
135	87	EXITNUM	CounterID = 135 DMSSFSEX Exit Calls - This is the number of times the SFS file pool server called the DMSSFSEX CSL routine. When usage of SFS file space or a storage group reaches a predefined point, the SFS file pool server will call one of the two exits of this type: File Space Usage or User Storage Group Full. The exits are called when the SFS file pool server calls the IBM-supplied CSL routine DMSSFSEX.
136	88	EXITTIM	CounterID = 136 DMSSFSEX Exit Time (tenths of milliseconds) - This is the total amount of elapsed time (in tenths of milliseconds) spent in routine processing for DMSSFSEX requests.
137	89	RCLEXITCON	CounterID = 137 Recall Exit Lock Conflicts - This is the number of times that a request for a lock involving a DFSMS/VM Recall Exit resource was delayed or denied because an implicit lock has already been created on that item.
138	8A	FILERCLCON	CounterID = 138 File Recall Lock Conflicts - This is the number of times that a request for a lock involving a DFSMS/VM File Recall resource was delayed or denied because an implicit lock had already been created on the desired item.
139	8B	CONNECTU	CounterID = 139 Connect User Requests - This is the number of times the server was requested to establish a connection to a file pool for the specified user ID, or set the associated user ID for an already established connection to the specified user ID or the number of times the server was requested to connect user IDs that differ from the user ID of the connecting machine.
140	8C	PRECOORD	CounterID = 140 Precoordination Requests - This is the total number of times the SFS file pool server was called during the precoordination phase of a syncpoint, to see if a user's file space is still exceeded. (The Precoordination request is issued only if a user's file space was exceeded during the current LUW).

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
141	8D	RENAMEUS	CounterID = 141 Rename Userid Requests - This is the number of times the SFS file pool server was requested to rename a file space. This request is made by the FILEPOOL RENAME command.
142	8E	FILEPOOLCTL	CounterID = 142 File Pool Control Backup Requests - This is the number of times the SFS file pool server was requested to take a control data backup using the FILEPOOL CONTROL BACKUP command.
143	8F	SFSLOGIO	CounterID = 143 BIO Requests to Write Log Blocks - This is the total number of times block I/O was used to write blocks to the SFS log minidisks. There are cases where the server can group multiple log blocks into one block I/O request. Therefore, this number may be less than Log Blocks Written.
144	90	SFSCPLOGIO	CounterID = 144 I/O Requests to Write Log Blocks - This is the number of I/O requests issued by CP to write blocks to the SFS log minidisks. Each I/O request will normally result in a physical I/O to DASD.
145	91	ADDMDISK	CounterID = 145 Add Minidisk Requests - This is the number of times the SFS file pool server was requested to add minidisk(s) while the server is in multiple user mode. This request is caused by the FILEPOOL MINIDISK command.
146	92	QDISABLE	CounterID = 146 Query Disable Requests - This is the number of times the SFS file pool server was requested to query a storage group or a file space or all file spaces and all storage groups in a file pool to determine if they have been previously disabled. This request is caused by the QUERY FILEPOOL DISABLE command, or DMSQFPDS CSL routine, or QUERY DISABLE operator command.
147	93	LOCKTIMEOUT	CounterID = 147 Locks Denied Due to Timeout- This is the number of times a lock could not be obtained within the timeout interval. When a lock is waited for, an internal timer may be set to prevent excessive wait times. If the timer expires, the lock is denied, and the request fails.
148	94	FSRECLAIM	CounterID = 148 Virtual Storage Reclaim Value- This is the total number of times the SFS or CRR server has reclaimed its unused free storage. Periodically, when the server is running low on storage, it performs a reclaim process to free up additional storage.
149	95	FSREOPEN	CounterID = 149 Create A Migrated File. - This is the total number of times an SFS administrator has created migrated files while restoring files from a userdata backup to a storage group.
150	96	BFACCESS	CounterID = 150 Byte File Check File Accessibility Requests - This is the number of times the BFS file pool was requested to check the accessibility of a BFS object.
151	97	BFCHMOD	CounterID = 151 Byte File Change Mode Requests - This is the number of times the BFS file pool was requested to change the mode associated with a BFS object.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
152	98	BFCHOWN	CounterID = 152 Byte File Change Owner Requests - This is the number of times the BFS file pool was requested to change the owner (UID/GID) of a BFS object.
153	99	BFCLOSE	CounterID = 153 Byte File Close File Requests- This is the number of times the BFS file pool was requested to close a regular file.
154	9A	BFCLOSEDIR	CounterID = 154 Byte File Close Directory Requests - This is the number of times the BFS file pool was requested to close a directory.
155	9B	BFZAPCAT	CounterID = 155 Byte File ZAPCAT Requests - This is the number of times the BFS file pool was requested to read or alter catalogs using the ZAPCAT request.
156	9C	BFLINK	CounterID = 156 Byte File Create Link Requests - This is the number of times the BFS file pool was requested to create a hard link.
157	9D	BFLOCKBY	CounterID = 157 Byte File Lock Byte Requests- This is the number of times the BFS file pool was requested to lock a byte range.
158	9E	BFLOOKUP	CounterID = 158 Byte File Lookup Requests - This is the number of times the BFS file pool was requested to look up a BFS object.
159	9F	BFMKCAT	CounterID = 159 Byte File Makecat Requests - This is the number of times the BFS file pool was requested to make a BFS object.
160	A0	BFMKREGFILE	CounterID = 160 Byte File Create Regular file Requests - This is the number of times the BFS file pool was requested to create a regular byte file.
161	A1	BFMKDIR	CounterID = 161 Byte File Create Directory Requests - This is the number of times the BFS file pool was requested to create a byte file directory.
162	A2	BFMKSYMLINK	CounterID = 162 Byte File Create Symbolic Link Requests - This is the number of times the BFS file pool was requested to create a symbolic link.
163	A3	BFMKEXTLINK	CounterID = 163 Byte File Create External Link Requests - This is the number of times the BFS file pool was requested to create an external link.
164	A4	BFMKFIFO	CounterID = 164 Byte File Create Named Pipe (FIFO) Requests - This is the number of times the BFS file pool was requested to create a named pipe (FIFO).
165	A5	BFMKCHARSPEC	CounterID = 165 Byte File Create Character Special File Requests - This is the number of times the BFS file pool was requested to create a character special file.
166	A6	BFMKBLKSPEC	CounterID = 166 Byte File Create Block Special File Requests - This is the number of times the BFS file pool was requested to create a block special file.
167	A7	BFOPENNEWR	CounterID = 167 Byte File Open File New With Intent Read Requests - This is the number of times the BFS file pool was requested to create a new file with intent READ.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
168	A8	BFOPENNEWW	CounterID = 168 Byte File Open File New With Intent Write Requests- This is the number of times the BFS file pool was requested to create a new file with intent WRITE.
169	A9	BFOPENREAD	CounterID = 169 Byte File Open File Read Requests - This is the number of times the BFS file pool was requested to open a file for read access.
170	AA	BFOPENWRIT	CounterID = 170 Byte File Open File Write Requests - This is the number of times the BFS file pool was requested to open a file for write/trunc access.
171	AB	BFOPENDIR	CounterID = 171 Byte File Open Directory Requests - This is the number of times the BFS file pool was requested to open a directory (to be subsequently used for reading).
172	AC	BFREAD	CounterID = 172 Byte File Read File Requests - This is the number of times the BFS file pool was requested to read data from files.
173	AD	BFREADDIR	CounterID = 173 Byte File Read Directory Entry Requests - This is the number of times the BFS file pool was requested to read directory entries.
174	AE	BFREADLINK	CounterID = 174 Byte File Read Link Contents Requests - This is the number of times the BFS file pool was requested to read the contents of a link.
175	AF	BFRENAME	CounterID = 175 Byte File Rename Requests - This is the number of times the BFS file pool was requested to rename a BFS object.
176	B0	BFRMDIR	CounterID = 176 Byte File Remove Directory Requests - This is the number of times the BFS file pool was requested to remove a directory.
177	B1	BFSREGFILE CLEANUP	CounterID = 177 Byte File Unlinked File Cleanup Requests - This is the number of unlinked files removed during FILESERV START.
178	B2	BFTOKRETRN	CounterID = 178 Byte File Token Return Requests - This is the number of times a client requested the BFS file pool to free a vnode token or block token.
179	B3	BFTSLOCKBY	CounterID = 179 Byte File Test Locked Bytes Requests - This is the number of times the BFS file pool was requested to test byte range locks held on a specific byte range.
180	B4	BFUNLINK	CounterID = 180 Byte File Unlink Requests - This is the number of times the BFS file pool was requested to remove a BFS object.
181	B5	BFUNLOCKBY	CounterID = 181 Byte File Unlock Byte Requests - This is the number of times the BFS file pool was requested to unlock a byte range.
182	B6	BFUTIME	CounterID = 182 Byte File Change Access/Modification Time Requests - This is the number of times the BFS file pool was requested to change the access/modification times of a BFS object.
183	B7	BFWRITE	CounterID = 183 Byte File Write File Requests- This is the number of times the BFS file pool was requested to write to a regular file.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
184	B8	TOKCONFLCAUSCBS	CounterID = 184 Byte File Token Conflicts Causing Callbacks - This is a count of the number of callbacks of tokens due to token conflicts. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token that is held by another user, the requestor must wait until the client machine that holds the token returns the it (responds to a callback of that token).
185	B9	GLOBCBWTIME	CounterID = 185 Byte File Callback Wait Time - This is the time (in tenths of milliseconds) spent waiting for callbacks of tokens. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token that is held by another user, the requestor must wait until the client machine that holds the token returns the it (responds to a callback of that token).
186	BA	TOKCBTO RETRIES	CounterID = 186 Byte File Token Callback Timeout Retries - This is a count of the number of retries of callbacks because of a delay of the holding client machine to respond to the token callback request. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token that is held by another user, the requestor must wait until the client machine that holds the token returns the it (responds to a callback of that token).
187	BB	TOKCBREQ RETRIES	CounterID = 187 Byte File Token Callback Requestor Retries - This is a count of the number of requestor retries because of extended delays in call back response. This occurs when it is necessary to give up waiting for a normal callback completion because of exceeding the retry limit for callback retries (see "Token Callback Timeout Retries" counter). Return code to requestor suggests retry by the client application. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token that is held by another user, the requestor must wait until the client machine that holds the token returns the it (responds to a callback of that token).
188	BC	DOIDCLLOCK	CounterID = 188 Byte File Directory Creation/Deletion Logical Lock Conflicts - This is the number of times that a request for a lock on an object (file, directory, link or symbolic link) to be created or deleted was denied (or waited for) because an implicit lock was already held on the object in a byte file system.
189	BD	OIDBLLOCK	CounterID = 189 Byte File Token Manager Logical Lock Conflicts - This is the number of times the Token Manager requested a WRITE VNODE lock but had to wait for the lock because the implicit lock was already held.
190	BE	NAMTIDLLOCK	CounterID = 190 Byte File NAMECAT Unallocated Logical Lock Conflicts - This is the number of times that a request for a lock on an unallocated NAMECAT row was denied because the implicit lock was already held on the row.
191	BF	OIDGSLLOCK	CounterID = 191 Byte File Global Storage Logical Lock Conflicts - This is the number of times that a request for a lock on the object was denied (or waited for) because an implicit lock was already held on the object.

Table 16. Server Counters (continued)

Dec	Hex	Name	Description
192	C0	OIDLLLOCK	CounterID = 192 Byte File File Logical Lock Conflicts - This is the number of times that a request for a lock, unlock, or close on a file for serializing byte range lock/unlock and file closes was denied (or waited for) because an implicit lock had already been created on that file. The request for the lock was from an implicit request.
193	C1	BFSLOCK RETRIES	CounterID = 193 Byte File Logical Lock Retries- This is the number of times a retry was attempted to obtain the BFS requests logical locks.
194	C2	BFSLOCK EXCEEDED	CounterID = 194 Byte File Logical Lock Retries Exceeded - This is the number of times the request was denied logical locks due to the lock retry count being exceeded.
195	C3	BFSBRLockWaits	CounterID = 195 Byte File Byte Range Lock Waits - This is the number of times that a Byte Range Lock Request had to wait before being awarded the requested lock.
196	C4	BFPIPEOPEN READ	CounterID = 196 Byte File Pipe Open For Read Requests - This is the number of times the BFS FIFO file pool was requested to open a named pipe for read.
197	C5	BFPIPEOPEN WRITE	CounterID = 197 Byte File Pipe Open For Write Requests - This is the number of times the BFS FIFO file pool was requested to open a named pipe for write.
198	C6	BFPIPEREAD	CounterID = 198 Byte File Pipe Read Requests- This is the number of times the BFS FIFO file pool was requested to read from a named pipe.
199	C7	BFPIPEWRITE	CounterID = 199 Byte File Pipe Write Requests- This is the number of times the BFS FIFO file pool was requested to write to a named pipe.
200	C8	BFPIPECLOSE	CounterID = 200 Byte File Pipe Close Requests- This is the number of times the BFS FIFO file pool was requested to close a named pipe.
201	C9	BFPIPEACCESS	CounterID = 201 Byte File Pipe Access Requests- This is the number of times the BFS file pool was requested to verify the access authorization to a named pipe.
202	CA	BFPIPEUTIME	CounterID = 202 Byte File Pipe Utime Requests- This is the number of times the BFS file pool was requested to update the timestamps associated with a named pipe.
203	CB	BFPIPESTAT	CounterID = 203 Byte File Pipe Stat Requests- This is the number of times the BFS FIFO file pool was requested to obtained the current status information about a named pipe.
204	CC	BFCANCEL	CounterID = 204 Byte File Cancel Requests - This is the number of times the BFS file pool was requested to cancel a specified BFS request.
205	CD	BFCHAUDIT	CounterID = 205 Byte File Change Audit Requests - This is the number of times the BFS file pool was requested to change the audit flags for a BFS object.

PI end

Appendix E. CMS APPLDATA Monitor Records

PI

The CMS nucleus contributes to the CP monitor data by using the APPLDATA domain.

To begin data collection, enable the APPLDATA domain and start monitoring (use the CP MONITOR command). The directory option APPLMON is also required for the selected users.

The data records for servers are domain X'A' APPLDATA records, the general format of which is described in HTML format at z/VM Data Areas, Control Blocks, and Monitor Records (https://www.vm.ibm.com/pubs/ctlblk.html).

Each data record consists of these parts:

- Monitor record header
- CMS APPLDATA record header
- CMS Multitasking application data

The CMS APPLDATA header data consists of the following:

Table 17. CMS APPLDATA Header Data

Data Item	Number of Bytes
Byte offset to application data relative to start of this record	2
Length in bytes of application data	2
User ID of the server machine (in EBCDIC)	8
Product identification (in EBCDIC). For CMS Multitasking records, this field contains “5684030MT1020100”.	16
Status	1
Reserved	3

Following the CP header data is the counter data. The counters in the CP monitor record are essentially the same counters those available through the MonitorBufferGet as described in *z/VM: CMS Application Multitasking*.

Table 18 on page 213 shows record layout for the CMS supplied application data. The offset values listed are the offsets into the application data area of the monitor record (field APLSDT_ADATA). Always use the byte offset and length fields in the standard domain 10 records to locate the start and end of the application data within the record.

Table 18. CMS Appldata Data

Dec	Hex	Type	Len	Name	Description
0	0	SIGNED	4	CRCOUNT	This is number of times a thread has been created.
4	4	CHAR	8	CRTIME	This is total amount of elapsed time spent creating threads. Accumulated in TOD clock units.
12	C	SIGNED	4	DLCOUNT	This is number of times a thread has been deleted.

Table 18. CMS Appldata Data (continued)

Dec	Hex	Type	Len	Name	Description
16	10	CHAR	8	DLTIME	This is total amount of elapsed time spent deleting threads. Accumulated in TOD clock units.
24	18	SIGNED	4	SWSCOUNT	This is the number of times the regular path switch was executed.
28	1C	SIGNED	4	SWFCOUNT	This is the number of times the fast path switch was executed.
32	20	SIGNED	4	BLOCKED	This is the count of threads currently blocked.
36	24	SIGNED	4	PROCHIGH	This is the highest number of processes that were concurrently defined.
40	28	SIGNED	4	THDHIGH	This is the highest number of threads that were concurrently defined.
44	2C	SIGNED	4	PSXMAX	This is the number of times POSIX process creation failed due to an attempt to exceed the maximum allowable POSIX processes.
48	30	SIGNED	8	*	Reserved and available for IBM use.

PI end

Appendix F. TCP/IP Monitor Records

PI

The TCP/IP stack contributes to the CP monitor data by creating records in the APPLDATA domain.

To begin data collection:

1. Add the APPLMON directory option to the directory entry of the TCPIP virtual machine.
2. Add a MONITORRECORDS configuration statement to the PROFILE TCPIP of the TCPIP virtual machine.
3. From an authorized user, issue the CP MONITOR command to enable the APPLDATA domain for sample and event recording and to start monitoring.

The data records for servers are domain X'A' APPLDATA records, the general format of which are described on this web page:

z/VM Data Areas, Control Blocks, and Monitor Records (https://www.vm.ibm.com/pubs/ctlblk.html)

Each data record consists of these parts:

- Monitor record header
- TCP/IP APPLDATA record header
- TCP/IP application data.

The TCP/IP APPLDATA header data consists of the following:

Table 19. TCP/IP APPLDATA Header Data

Data Item	Number of Bytes
Byte offset to application data relative to start of this record	2
Length in bytes of application data	2
User ID of the service machine (in EBCDIC)	8
Product identification (in EBCDIC). For TCP/IP records, this field contains **5735FALSTx0ttt00**. • **5735FALST** — Constant • **x** — sub—record number • **0** — reserved • **ttt** — TCP/IP level • **00** – reserved	16
Status	1
Reserved	3

Following the header data is the TCP/IP data. TCP/IP produces a variety of record formats. Each record is identified with a one-digit hexadecimal sub-record number in the product identification field of the record header.

Table 20 on page 216 shows the record layout for the TCP/IP MIB Record. This record is produced as Sample-class data and has a TCP/IP sub-record type of '00'x.

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	ipInReceives	IP packets received
4	4	Unsigned	4	ipInHdrErrors	IP packets received that had header errors
8	8	Unsigned	4	ipInAddrErrors	IP packets received that had addressing errors
12	C	Unsigned	4	ipForwDatagrams	IP datagrams forwarded
16	10	Unsigned	4	ipInUnknownProtos	IP datagrams that specified an unknown protocol
20	14	Unsigned	4	ipInDiscards	IP datagrams discarded
24	18	Unsigned	4	ipInDelivers	IP datagrams delivered to IP user protocols
28	1C	Unsigned	4	ipOutRequests	IP datagrams supplied by IP user-protocols for delivery
32	20	Unsigned	4	ipOutDiscards	Outgoing IP datagrams discarded before delivery
36	24	Unsigned	4	ipOutNoRoutes	Outgoing IP datagrams that had no route to their destination
40	28	Unsigned	4	ipReasmReqds	IP fragments received requiring reassembly
44	2C	Unsigned	4	ipReasmOKs	IP datagrams reassembled
48	30	Unsigned	4	ipReasmFails	IP datagram reassembly errors
52	34	Unsigned	4	ipFragOKs	IP datagrams fragmented
56	38	Unsigned	4	ipFragFails	IP datagram fragmentation failures
60	3C	Unsigned	4	ipFragCreates	IP datagram fragments created
64	40	Unsigned	4	icmpInMsgs	ICMP messages received
68	44	Unsigned	4	icmpInErrors	ICMP messages received that had errors
72	48	Unsigned	4	icmpInDestUnreachs	ICMP destination unreachable messages received
76	4C	Unsigned	4	icmpInTimeExcds	ICMP time exceeded messages received
80	50	Unsigned	4	icmpInParmProbs	ICMP parameter problem messages received
84	54	Unsigned	4	icmpInSrcQuenchs	ICMP source quench messages received

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
88	58	Unsigned	4	icmpInRedirects	ICMP redirect messages received
92	5C	Unsigned	4	icmpInEchos	ICMP echo messages received
96	60	Unsigned	4	inEchoReps	ICMP echo reply messages received
100	64	Unsigned	4	icmpInTimestamps	ICMP timestamp messages received
104	68	Unsigned	4	icmpInTimestampReps	ICMP timestamp reply messages received
108	6C	Unsigned	4	icmpInAddrMasks	ICMP address mask messages received
112	70	Unsigned	4	icmpInAddrMaskReps	ICMP address mask reply messages received
116	74	Unsigned	4	icmpOutMsgs	ICMP messages sent
120	78	Unsigned	4	icmpOutErrors	ICMP message transmission errors
124	7C	Unsigned	4	icmpOutDestUnreachs	ICMP destination unreachable messages sent
128	80	Unsigned	4	icmpOutTimeExcds	ICMP time exceeded messages sent
132	84	Unsigned	4	icmpOutParmProbs	ICMP parameter problem messages sent
136	88	Unsigned	4	icmpOutSrcQuenchs	ICMP source quench messages sent
140	8C	Unsigned	4	icmpOutRedirects	ICMP redirect messages sent
144	90	Unsigned	4	icmpOutEchos	ICMP echo messages sent
148	94	Unsigned	4	icmpOutEchoReps	ICMP echo reply messages sent
152	98	Unsigned	4	icmpOutTimestamps	ICMP timestamp messages sent
156	9C	Unsigned	4	icmpOutTimestampReps	ICMP timestamp reply messages sent
160	A0	Unsigned	4	icmpOutAddrMasks	ICMP address mask messages sent
164	A4	Unsigned	4	icmpOutAddrMaskReps	ICMP address mask reply messages sent
168	A8	Unsigned	4	tcpActiveOpens	TCP connection opens initiated
172	AC	Unsigned	4	tcpPassiveOpens	TCP connection opens accepted

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
176	B0	Unsigned	4	tcpAttemptFails	TCP connection open failures
180	B4	Unsigned	4	tcpEstabResets	TCP connections reset
184	B8	Unsigned	4	tcpInSegs	TCP segments received
188	BC	Unsigned	4	tcpOutSegs	TCP segments transmitted
192	C0	Unsigned	4	tcpRetransSegs	TCP segments retransmitted
196	C4	Unsigned	4	tcpInErrs	TCP segments received that had errors
200	C8	Unsigned	4	tcpOutRsts	TCP segments transmitted that included a reset
204	CC	Unsigned	4	udpInDatagrams	UDP datagrams received
208	D0	Unsigned	4	udpNoPorts	UDP datagrams received for ports that had no listener
212	D4	Unsigned	4	udpInErrors	UDP datagrams received that had errors
216	D8	Unsigned	4	udpOutDatagrams	UDP datagrams transmitted
220	DC	Unsigned	4	arpInRequests	ARP requests received
224	E0	Unsigned	4	arpOutReplies	ARP replies transmitted
228	E4	Unsigned	4	arpOutRequests	ARP requests transmitted
232	E8	Unsigned	4	ioReads	Read requests
236	EC	Unsigned	4	ioWrites	Write requests
240	F0	Unsigned	4	ioInOctets	Bytes received
244	F4	Unsigned	4	ioOutOctets	Bytes transmitted
248	F8	Unsigned	4	iucvReceives	IUCV receives
252	FC	Unsigned	4	iucvRejects	IUCV rejects
256	100	Unsigned	4	iucvReplies	IUCV replies
260	104	Unsigned	4	iucvSends	IUCV sends
264	108	Unsigned	4	vmcfSendsOK	VMCF successful sends
268	10C	Unsigned	4	vmcfSendsAbnormal	VMCF abnormal sends
272	110	Unsigned	4	vmcfSendsFatal	VMCF send failures
276	114	Unsigned	4	dosLand	LAND denial-of-service packet discards

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
280	118	Unsigned	4	ioDirectReads	QDIO/EQDIO inbound data transfers This field is incremented by the QDIO service whenever an "Input Buffer Empty" to "Input Buffer Primed" state change occurs on a QDIO input queue or by the EQDIO service whenever an Ethernet frame is received from an EQDIO input queue.
284	11C	Unsigned	4	ioDirectWrites	QDIO/EQDIO outbound data transfers This field is incremented by the QDIO service whenever an "Output Buffer Primed" to "Output Buffer Empty" state change occurs on a QDIO output queue or by the EQDIO service whenever a datagram is sent to an EQDIO output queue.
288	120	Unsigned	4	QDIOpolls	QDIO polling operations This field is incremented by the QDIO service for each occurrence of a QDIO polling operation performed by QDIO_Poll.
292	124	Unsigned	4	ioPCI	QDIO PCI interrupts This field is incremented by the QDIO service whenever a PCI is received from a QDIO data device. The PCI interrupt is used by the hardware adapter to request additional QDIO buffers for inbound data transfers. This event should only occur if TCP/IP enters a wait state or is not polling the queues at a sufficient rate .
296	128	Unsigned	4	ioIdlePollCnt	This field is incremented by the QDIO service whenever the TCP/IP scheduler performs a QDIO polling operation in which no QDIO data transfer has taken place.

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
300	12C	Unsigned	4	dosSmurf	Smurf denial-of-service packet discards
304	130	Unsigned	4	dosFraggle	Fraggle denial-of-service packet discards
308	134	Unsigned	4	dosPOD	Ping-o-death denial-of-service packet discards
312	138	Unsigned	4	dosBlat	Blat denial-of-service packet discards
316	13C	Unsigned	4	dosStream	Stream denial-of-service packet discards
320	140	Unsigned	4	dosR4P3D	R4P3D denial-of-service packet discards
324	144	Unsigned	4	dosKod	KOD denial-of-service packet discards
328	148	Unsigned	4	dosKox	KOX denial-of-service packet discards
332	14C	Unsigned	4	dosSynflood	Synflood denial-of-service packet discards
336	150	Unsigned	4	dosPMtu	Path MTU denial-of-service packet discards
340	154	Unsigned	4	ipv6InReceives	IPv6 datagrams received
344	158	Unsigned	4	ipv6InHdrErrors	IPv6 datagrams received that had header errors
348	15C	Unsigned	4	ipv6InTooBigErrors	IPv6 datagrams too big to forward on a link
352	160	Unsigned	4	ipv6InNoRoutes	IPv6 datagrams that had no route for forwarding
356	164	Unsigned	4	ipv6InAddrErrors	IPv6 datagrams received that had addressing errors
360	168	Unsigned	4	ipv6InUnknownProtos	IPv6 datagrams that specified an unknown protocol
364	16C	Unsigned	4	ipv6InTruncatedPkts	IPv6 datagrams not carrying enough data in frame
368	170	Unsigned	4	ipv6InDiscards	IPv6 datagrams discarded
372	174	Unsigned	4	ipv6InDelivers	IPv6 packets delivered to IP user-protocols
376	178	Unsigned	4	ipv6OutForwDatagrams	IPv6 packets forwarded
380	17C	Unsigned	4	ipv6OutRequests	IPv6 datagrams supplied by IP user-protocols for deliver
384	180	Unsigned	4	ipv6OutDiscards	Outgoing IPv6 datagrams discarded before delivery

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
388	184	Unsigned	4	ipv6OutNoRoutes	Outgoing IPv6 datagrams that had no route to their destination
392	188	Unsigned	4	ipv6OutFragOKs	IPv6 datagrams fragmented
396	18C	Unsigned	4	ipv6OutFragFails	IPv6 datagram fragmentation failures
400	190	Unsigned	4	ipv6OutFragCreates	IPv6 datagram fragments created
404	194	Unsigned	4	ipv6ReasmReqds	IPv6 fragments received requiring reassembly
408	198	Unsigned	4	ipv6ReasmOKs	IPv6 datagrams reassembled
412	19C	Unsigned	4	ipv6ReasmFails	IPv6 datagram reassembly errors
416	1A0	Unsigned	4	ipv6InMcastPkts	IPv6 multicast packets received
420	1A4	Unsigned	4	ipv6OutMcastPkts	IPv6 multicast packets sent
424	1A8	Unsigned	4	ipv6IcmpInMsgs	ICMPv6 messages received
428	1AC	Unsigned	4	ipv6IcmpInErrors	ICMPv6 messages received that had errors
432	1B0	Unsigned	4	ipv6IcmpInDestUnreachs	ICMPv6 messages received
436	1B4	Unsigned	4	ipv6IcmpInAdminProhibs	ICMPv6 administratively-prohibited messages received
440	1B8	Unsigned	4	ipv6IcmpInTimeExcds	ICMPv6 time exceeded messages received
444	1BC	Unsigned	4	ipv6IcmpInParmProblems	ICMPv6 parameter problem messages received
448	1C0	Unsigned	4	ipv6IcmpInPktTooBigs	ICMPv6 packet too big messages received
452	1C4	Unsigned	4	ipv6IcmpInEchos	ICMPv6 echo request messages received
456	1C8	Unsigned	4	ipv6IcmpInEchoReplies	ICMPv6 echo reply messages received
460	1CC	Unsigned	4	ipv6IcmpInRouterSolicits	ICMPv6 router solicitation messages received
464	1D0	Unsigned	4	ipv6IcmpInRouterAdvertisements	ICMPv6 router advertisement messages received
468	1D4	Unsigned	4	ipv6IcmpInNeighborSolicits	ICMPv6 neighbor solicitation messages received
472	1D8	Unsigned	4	ipv6IcmpInNeighborAdvertisements	ICMPv6 neighbor advertisement messages received

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
476	1DC	Unsigned	4	ipv6IcmpInRedirects	ICMPv6 redirect messages received
480	1E0	Unsigned	4	ipv6IcmpInGroupMembQueries	ICMPv6 group membership query messages received
484	1E4	Unsigned	4	ipv6IcmpInGroupMembResponses	ICMPv6 group membership responses (Report) messages received
488	1E8	Unsigned	4	ipv6IcmpInGroupMembReductions	ICMPv6 group membership reduction (Done) messages received
492	1EC	Unsigned	4	ipv6IcmpOutMsgs	ICMPv6 messages sent
496	1F0	Unsigned	4	ipv6IcmpOutErrors	ICMPv6 message transmission errors
500	1F4	Unsigned	4	ipv6IcmpOutDestUnreachs	ICMPv6 destination unreachable messages sent
504	1F8	Unsigned	4	ipv6IcmpOutAdminProhibs	ICMPv6 administratively-prohibited messages sent
508	1FC	Unsigned	4	ipv6IcmpOutTimeExcds	ICMPv6 time exceeded messages sent
512	200	Unsigned	4	ipv6IcmpOutParmProblems	ICMPv6 parameter problem messages sent
516	204	Unsigned	4	ipv6IcmpOutPktTooBigs	ICMPv6 packet too big messages sent
520	208	Unsigned	4	ipv6IcmpOutEchos	ICMPv6 echo request messages sent
524	20C	Unsigned	4	ipv6IcmpOutEchoReplies	ICMPv6 echo reply messages sent
528	210	Unsigned	4	ipv6IcmpOutRouterSolicits	ICMPv6 router solicitation messages sent
532	214	Unsigned	4	ipv6IcmpOutRouterAdvertisements	ICMPv6 router advertisement messages sent
536	218	Unsigned	4	ipv6IcmpOutNeighborSolicits	ICMPv6 neighbor solicitation messages sent
540	21C	Unsigned	4	ipv6IcmpOutNeighborAdvertisements	ICMPv6 neighbor advertisement messages sent
544	220	Unsigned	4	ipv6IcmpOutRedirects	ICMPv6 redirect messages sent
548	224	Unsigned	4	ipv6IcmpOutGroupMembQueries	ICMPv6 group membership query messages sent
552	228	Unsigned	4	ipv6IcmpOutGroupMembResponses	ICMPv6 group membership responses (Report) messages sent

Table 20. TCP/IP MIB Record - Type '00'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
556	22C	Unsigned	4	ipv6IcmpOutGroupMembReductions	ICMPv6 group membership reduction (Done) messages sent
560	230	Unsigned	4	dosZeroWin	Zero window denial-of-service packet discards
564	234	Unsigned	4	SSLMaxSessions	Maximum number of secure connections allowed across all SSL servers
568	238	Unsigned	4	SSLActiveConnections	Number of connections that are currently secure
572	23C	Unsigned	4	SSLHighWater	Highest number of connections that have been secure at any one time
576	240	Unsigned	4	SSLConnFailConfig	Secure connection request failed - configuration problem
580	244	Unsigned	4	SSLConnFailNoResources	Secure connection request failed - no resources
584	248	Unsigned	4	dosSockStress	SSTRESS denial-of-service packet discards
588	24C	Unsigned	4	UdpLimitErr	UDP packet discards due to queue limit
592	250	Unsigned	4	EQDIOpolls	EQDIO polling operations This field is incremented by the EQDIO service for each occurrence of an EQDIO polling operation performed by the device driver.
596	254	Unsigned	4	EQDIOFruitlessPollCnt	EQDIO polling operations This field is incremented by the EQDIO service for each occurrence of an EQDIO polling operation performed by the device driver that does not find any work to do.

Table 21 on page 223 shows the record layout for the TCP/IP TCB Open Record. This record is produced as Event-class data and has a TCP/IP sub-record type of '01'x.

Table 21. TCP/IP TCB Open Record - Type '01'x - Event Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	TcbTag	TCP connection identifier
4	4	Hex	4	ForeignAddress	Foreign IP address
8	8	Unsigned	2	ForeignPort	Foreign port number
10	A	Unsigned	2	LocalPort	Local port number

Table 21. TCP/IP TCB Open Record - Type '01'x - Event Data (continued)

Dec	Hex	Type	Len	Name	Description
12	C	Character	8	ClientName	Client user identifier
20	14	Unsigned	2	MaximumSegmentSize	Maximum segment size
22	16	Hex	2	LogicalDevice	Logical device number
24	18	Unsigned	4	MaxRcvWnd	Maximum receive window size
28	1C	Unsigned	4	MaxSndWnd	Maximum send window size
32	20	Unsigned	4	MinRcvWnd	Minimum receive window size
36	24	Unsigned	2	WindowSendScale	Send window scale factor
38	26	Unsigned	2	WindowReceiveScale	Receive window scale factor
40	28	Hex	4	LocalAddress	Local IP address
44	2C	Character	16	ForeignAddressIPv6	IPv6 format of the foreign address
60	3C	Character	16	LocalAddressIPv6	IPv6 format of the local address

Table 22 on page 224 shows the record layout for the TCP/IP TCB Close Record. This record is produced as Event-class data and has a TCP/IP sub-record type of '02'x.

Table 22. TCP/IP TCB Close Record - Type '02'x - Event Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	TcbTag	TCP connection identifier
4	4	Hex	4	ForeignAddress	Foreign IP address
8	8	Unsigned	2	ForeignPort	Foreign port number
10	A	Unsigned	2	LocalPort	Local port number
12	C	Character	8	ClientName	Client user identifier
20	14	Unsigned	2	MaximumSegmentSize	Maximum segment size
22	16	Hex	2	LogicalDevice	Logical device number
24	18	Unsigned	4	MaxRcvWnd	Maximum receive window size
28	1C	Unsigned	4	MaxSndWnd	Maximum send window size
32	20	Unsigned	4	MinRcvWnd	Minimum receive window size
36	24	Unsigned	4	BytesIn	Bytes received
40	28	Unsigned	4	BytesOut	Bytes sent
44	2C	Unsigned	4	SegmentTotal	Segments
48	30	Unsigned	4	MaxNumberUnacked	Maxmimum number of unacknowledged segments
52	34	Float	4	SmoothTime	Smoothed round trip time (in milliseconds)
56	38	Float	4	SmoothVariance	Smoothed round trip time variance (in milliseconds)
60	3C	Unsigned	4	TotalAcked	Total acknowledgments sent

Table 22. TCP/IP TCB Close Record - Type '02'x - Event Data (continued)

Dec	Hex	Type	Len	Name	Description
64	40	Unsigned	4	TotalTime	Total round trip time (in milliseconds)
68	44	Unsigned	4	DroppedFutureTotal	Future segments dropped
72	48	Unsigned	4	DupTotal	Duplicate segments
76	4C	Float	4	SmoothTime1323	Smoothed round trip time from RFC 1323 timestamps (in milliseconds)
80	50	Float	4	SmoothVariance1323	Smoothed round trip time variance from RFC 1323 timestamps (in milliseconds)
84	54	Unsigned	4	TotalAcked1323	Total acknowledgments sent
88	58	Unsigned	4	TotalTime1323	Total round trip time from RFC 1323 timestamps (in milliseconds)
92	5C	Unsigned	2	MaxInputQueueSize	Maximum input buffer queue size
94	5E	Unsigned	2	MaxOutputQueueSize	Maximum output buffer queue size
96	60	Unsigned	4	NewBytesInInMeg	Bytes received in millions of bytes (see note)
100	64	Unsigned	4	NewBytesIn	Additional bytes received (see note)
104	68	Unsigned	4	NewBytesOutInMeg	Bytes sent in millions of bytes (see note)
108	6C	Unsigned	4	NewBytesOut	Additional byes sent (see note)
112	70	Unsigned	2	WindowSendScale	Send window scale factor
114	72	Unsigned	2	WindowReceiveScale	Receive window scale factor
116	74	Hex	4	LocalAddress	Local IP address
120	78	Unsigned	4	PredictedSegmentTotal	Total segments predicted correctly
124	7C	Character	16	ForeignAddressIPv6	IPv6 format of the foreign address
140	8C	Character	16	LocalAddressIPv6	IPv6 format of the local address

Note: To calculate the total bytes received or sent, use the following formula:

Total bytes Received= (NewBytesInInMeg x 1000000 + NewBytesIn)Total Bytes Sent= (NewBytesOutInMeg x 1000000 + NewBytesOut)

Table 23 on page 226 shows the record layout for the TCP/IP Pool Limit Record. This record is produced as Configuration-class data and has a TCP/IP sub-record type of '03'x.

Table 23. TCP/IP Pool Limit Record - Type '03'x - Configuration Data

Dec	Hex	Type	Len	Name	Description
0	0	Structure	36	AcbPoolInformation	Activity control block pool information
36	24	Structure	36	CcbPoolInformation	Client control block pool information
72	48	Structure	36	DataBufferPoolInformation	Data buffer pool information
108	6C	Structure	36	EnvelopePoolInformation	Envelope pool information
144	90	Structure	36	HostPoolInformation	Host pool information
180	B4	Structure	36	LargeEnvelopePoolInformatio n	Large envelope pool information
216	D8	Structure	36	RcbPoolInformation	Raw IP control block pool information
252	FC	Structure	36	ScbPoolInformation	Socket control block pool information
288	120	Structure	36	SkcbPoolInformation	BSD-style socket control block pool information
324	144	Structure	36	SmallDataBufferPool Information	Small data buffer pool information
360	168	Structure	36	TcbPoolInformation	TCP control block pool information
396	18C	Structure	36	TinyDataBufferPool Information	Tiny data buffer pool information
432	1B0	Structure	36	UcbPoolInformation	UDP control block pool information
468	1D4	Structure	36	AddressTranslation PoolInfo	Address translation pool information
504	1F8	Structure	36	IPRoutePoolInfo	IP route pool
540	21C	Structure	36	WhenSentPoolInfo	Segment acknowledgment pool information
576	240	Unsigned	4	MachineSize	TCPIP virtual machine storage size
580	244	Unsigned	4	FreeStorage	Amount of storage available for allocation
584	248	Unsigned	4	LargestBlock	Size of largest block of available storage
588	24C	Reserved	24		Reserved
612	264	Structure	36	FpspPoolInformation	Fixed page storage pool information
648	228	Structure	36	NcbInformation	Neighbor cache control block pool information
684	2AC	Structure	36	IPv6RoutePoolInfo	IPv6 IP route pool

Table 24 on page 227 shows the layout of the structure containing information for each pool.

Table 24. Pool Structure Layout

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	FreePoolSize	Blocks allocated
4	4	Unsigned	4	LimitSize	Unrestricted allocation limit
8	8	Unsigned	4	PermitSize	Restricted allocation limit
12	C	Unsigned	4	FreePoolCurrentSize	Blocks available
16	10	Unsigned	4	FreePoolLowWater	Minimum depth reached
20	14	Reserved	12	.	Reserved
32	20	Unsigned	4	FreePoolElementSize	Pool element size

Table 25 on page 227 shows the record layout for the TCP/IP Pool Size Record. This record is produced as Sample-class data and has a TCP/IP sub-record type of '04'x.

Table 25. TCP/IP Pool Size Record - Type '04'x - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	AcbQSize	Activity control block pool level
4	4	Unsigned	4	CcbQSize	Client control block pool level
8	8	Unsigned	4	DatBufQSize	Data buffer pool level
12	C	Unsigned	4	EnvQSize	Envelope pool level
16	10	Unsigned	4	LrgEnvQSize	Large envelope pool level
20	14	Unsigned	4	RcbQSize	Raw IP control block pool level
24	18	Unsigned	4	ScbQSize	Socket control block pool level
28	1C	Unsigned	4	SkcbQSize	BSD-style socket control block pool level
32	20	Unsigned	4	SdbQSize	Small data buffer pool level
36	24	Unsigned	4	TcbQSize	TCP control block pool level
40	28	Unsigned	4	TdbQSize	Tiny data buffer pool level
44	2C	Unsigned	4	UcbQSize	UDP control block pool level
48	30	Unsigned	4	AcbQMin	Activity control block pool minimum level
52	34	Unsigned	4	CcbQMin	Client control block pool minimum level
56	38	Unsigned	4	DatBufQMin	Data buffer pool minimum level
60	3C	Unsigned	4	EnvQMin	Envelope pool minimum level
64	40	Unsigned	4	LrgEnvQMin	Large envelope pool minimum level
68	44	Unsigned	4	RcbQMin	Raw IP control block pool minimum level
72	48	Unsigned	4	ScbQMin	Socket control block pool minimum level

Table 25. TCP/IP Pool Size Record - Type '04'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
76	4C	Unsigned	4	SkcbQMin	BSD-style socket control block pool minimum level
80	50	Unsigned	4	SdbQMin	Small data buffer pool minimum level
84	54	Unsigned	4	TcbQMin	TCP control block pool minimum level
88	58	Unsigned	4	TdbQMin	Tiny data buffer pool minimum level
92	5C	Unsigned	4	UcbQMin	UDP control block pool minimum level
96	60	Unsigned	4	WhenSentQSize	Segment acknowledgment pool level
100	64	Unsigned	4	WhenSentQMin	Segment acknowledgment pool minimum level
104	68	Unsigned	4	FpspSize	FPSP pool level This field specifies the number of 4 KB blocks currently allocated within the storage pool and in use by functions within the stack. FPSM_TotalPages is the value being sampled.
108	6C	Unsigned	4	FpspMin	FPSP pool minimum level This is the lowest number of pages available within the storage pool since the start up of TCP/IP. FPSM_MinAvailPages is the value being sampled.
112	70	Unsigned	4	FpspCommitted	FPSP available locked pages This is the number of locked 4 KB pages that are currently available, but not being used within the storage pool. FPSM_TotalCommitted is the value being sampled.
116	74	Unsigned	4	FpspInUse	FPSP locked pages in use This is the number of locked 4 KB pages that are currently allocated by users of the storage pool. FPSM_TotalInUse is the value being sampled.
120	78	Unsigned	4	FpspCommit2G	FPSP available locked pages above 2 GB This is the number of 4 KB pages that are locked in storage above 2 GB that are currently available, but not being used within the storage pool. FPSM_2GTotCommit is the value being sampled.

Table 25. TCP/IP Pool Size Record - Type '04'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
124	7C	Unsigned	4	FpspInUse2G	FPSP locked pages above 2 GB This is the number of 4 KB pages that are locked in storage above 2 GB that are currently allocated by users of the storage pool. FPSM_TotInUse2G is the value being sampled.

Table 26 on page 229 shows the record layout for the TCP/IP LCB Record. This record is produced as Sample-class data and has a TCP/IP sub-record type of '05'x. An LCB record is produced for each link.

Table 26. TCP/IP LCB Record - Type '05'x - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	ifInOctets	Bytes received. Number of bytes that are received by the TCP/IP stack over this link device. For most devices, this field is updated on completion of a SSCH read request. For OSD devices and EQDIO devices, the respective service will update this field when creating a TCP/IP envelope. This field will be incremented by the number of bytes transferred via a QDIO input queue or an EQDIO input queue.
4	4	Unsigned	4	ifInUcastPkts	Unicast packets received
8	8	Unsigned	4	ifInNUcastPkts	Non-unicast packets received
12	C	Unsigned	4	ifInDiscards	Incoming packets discarded
16	10	Unsigned	4	ifInErrors	Incoming packets that had errors
20	14	Unsigned	4	ifInUnknownProtos	Incoming packets that had unknown protocols
24	18	Unsigned	4	ifOutOctets	Bytes transmitted. Number of bytes that are transmitted by the TCP/IP stack over this link device. For most devices, this field is updated by the driver when scheduling a SSCH write request. For OSD devices and EQDIO devices, the respective service will update this field when copying a TCP/IP envelope to an output buffer. This field will be incremented by the number of bytes transferred via a QDIO output queue or an EQDIO output queue.
28	1C	Unsigned	4	ifOutUcastPkts	Unicast packets transmitted
32	20	Unsigned	4	ifOutNUcastPkts	Non-unicast packets transmitted
36	24	Unsigned	4	ifOutDiscards	Outgoing packets discarded
40	28	Unsigned	4	ifOutErrors	Outgoing packets that had errors
44	2C	Signed	2	ifDescrLength	Link description length

Table 26. TCP/IP LCB Record - Type '05'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
46	2E	Character	94	ifDescr	Link description
140	8C	Unsigned	4	ifType	Interface types: (defined by RFC 1213)
144	90	Unsigned	4	ifMtu	The value of the maximum transmission unit (MTU) for a device that has been started. This value can be one of these: 1. Specified on the MTU option of the LINK statement 2. Returned by the device (as in the case of OSD, HiperSockets, or EQDIO devices). 3. A default provided by the TCPIP virtual machine.
148	94	Unsigned	4	ifSpeed	Estimate of the interface's current bandwidth (bits per second). If the value of this field is X'FFFFFFFF', refer to the ifHSpeed field in the record for the true estimate of the interface's bandwidth.
152	98	Unsigned	4	InterfaceNumber	Interface number
156	9C	TOD	8	LastChange	Time of last state change
164	A4	Unsigned	4	LinkNumber	Link number
168	A8	Character	16	LinkName	Link name
184	B8	Unsigned	4	NetNumber	Network number

Table 26. TCP/IP LCB Record - Type '05'x - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
188	BC	Unsigned	1	NetworkType	Interface types: 1 - InternalLoopback 2 - ProNet 3 - Ethernet 4 - EtherOr802.3 5 - Only802.3 6 - TokenRing 7 - Util 8 - IUCV 9 - CTC 10 - DDN1822 12 - A220 13 - HIPPI 14 - FDDI 15 - CLAWip 16 - ControlTask 17 - OffloadLink1 18 - OffloadApiBroadcastMedia 19 - OffloadApiPointToPoint 20 - OffloadApiOtherKinds 21 - Virtual Device (VIPA) 22 - OSA ATM native mode 23 - QDIO Ethernet mode 24 - QDIO ATM mode 25 - QDIO Token Ring 26 - HiperSockets 28 - EQDIO Ethernet
189	BD	Reserved	3	.	Reserved
192	C0	Unsigned	4	ifInOctetsWrapped	For NetworkType 23 (QDIO Ethernet mode), 26 (HiperSockets), and 28 (EQDIO Ethernet), the number of times the ifInOctets counter has wrapped.
196	C4	Unsigned	4	ifOutOctetsWrapped	For NetworkType 23 (QDIO Ethernet mode), 26 (HiperSockets), and 28 (EQDIO Ethernet), the number of times the ifOutOctets counter has wrapped.
200	C8	Unsigned	4	ifHSpeed	Estimate of the interface's current bandwidth (million bits per second).
204	CC	Unsigned	4	ifInBuffers	For NetworkType 28 (EQDIO Ethernet), the current number of input buffers for the device.
208	D0	Unsigned	4	StallDetected	The number of times a stall was detected and force initiative called for (output data plane only)
212	D4	Unsigned	4	FISSet	The number of times force initiative state was set on an SCPQC (output data plane only)

Table 27 on page 232 shows the record layout for the TCP/IP UCB Open Record. This record is produced as Event-class data and has a TCP/IP sub-record type of '06'x.

Table 27. TCP/IP UCB Open Record - Type '06'x - Event Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	UcbTag	UDP connection identifier
4	4	Hex	4	LocalAddress	Local IP address
8	8	Unsigned	2	LocalPort	Local port number
10	A	Reserved	2	.	Reserved
12	C	Character	8	ClientName	Client user identifier
20	14	Character	16	LocalAddressIPv6	IPv6 format of the local address

Table 28 on page 232 shows the record layout for the TCP/IP UCB Close Record. This record is produced as Event-class data and has a TCP/IP sub-record type of '07'x.

Table 28. TCP/IP UCB Close Record - Type '07'x - Event Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	UcbTag	UDP connection identifier
4	4	Hex	4	LocalAddress	Local IP address
8	8	Unsigned	2	LocalPort	Local port number
10	A	Reserved	2	.	Reserved
12	C	Character	8	ClientName	Client user identifier
20	14	Unsigned	4	BytesIn	Bytes received
24	18	Unsigned	4	BytesOut	Bytes transmitted
28	1C	Unsigned	4	NewBytesInInMeg	Bytes received in millions of bytes (see note)
32	20	Unsigned	4	NewBytesIn	Additional bytes received (see note)
36	24	Unsigned	4	NewBytesOutInMeg	Bytes sent in millions of bytes (see note)
40	28	Unsigned	4	NewBytesOut	Additional bytes sent (see note)
44	2C	Character	16	LocalAddressIPv6	IPv6 format of the local address

Note: To calculate the total bytes received or sent, use the following formula:

Total bytes Received= (NewBytesInInMeg x 1000000 + NewBytesIn)Total Bytes Sent= (NewBytesOutInMeg x 1000000 + NewBytesOut)

Table 29 on page 232 shows the record layout for the TCP/IP Link Definition Record. This record is produced as Configuration-class data and has a TCP/IP sub-record type of '08'x. A link Definition Record is produced for each link.

Table 29. TCP/IP Link Definition Record - Type '08'x - Configuration Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	InterfaceNumber	Interface number

Table 29. TCP/IP Link Definition Record - Type '08'x - Configuration Data (continued)

Dec	Hex	Type	Len	Name	Description
4	4	Unsigned	4	LinkNumber	Link number
8	8	Hex	4	DeviceNumber	Device number
12	C	Unsigned	1	NetworkType	Network types: 1 - InternalLoopback 2 - ProNet 3 - Ethernet 4 - EtherOr802.3 5 - Only802.3 6 - TokenRing 7 - Util 8 - IUCV 9 - CTC 10 - DDN1822 12 - A220 13 - HIPPI 14 - FDDI 15 - CLAWip 16 - ControlTask 17 - OffloadLink1 18 - OffloadApiBroadcastMedia 19 - OffloadApiPointToPoint 20 - OffloadApiOtherKinds 21 - Virtual Device (VIPA) 22 - OSA ATM native 23 - QDIO Ethernet mode 24 - QDIO ATM mode 25 - QDIO Token Ring 26 - HiperSockets 28 - EQDIO Ethernet
13	D	Unsigned	1	DeviceType	Device types: 1 - LCS 4 - DDN1822 6 - PVMIUCV 9 - CTC 10 - HCH 11 - HIPPI 12 - CLAW 13 - SNALU62 14 - Virtual (VIPA) 15 - ATM 16 - OSD (OSA Direct Express Adapter) 17 - HiperSockets 18 - LocalIUCV 19 - VswitchIUCV 20 - OSD Vswitch 22 - EQDIO

Table 29. TCP/IP Link Definition Record - Type '08'x - Configuration Data (continued)

Dec	Hex	Type	Len	Name	Description

14	E	Unsigned	2	TransportType	Transport Types:
					`2 - Ethernet (layer 2)`
					`3 - IP (layer 3)`

16	10	Character	16	NetworkName	Network name
32	20	Character	16	DeviceName	Device name
48	30	Signed	2	ifDescrLength	Link description length
50	32	Character	94	ifDescr	Link description
144	90	Unsigned	4	ifType	Interface type
148	94	Unsigned	4	ifMtu	The value of the maximum transmission unit (MTU) for a device that has been started. This value can be one of these: 1. Specified on the MTU option of the LINK statement 2. Returned by the device (as in the case of OSD, HiperSockets, or EQDIO devices). 3. A default provided by the TCPIP virtual machine.
152	98	Unsigned	4	ifSpeed	An estimate of the interface's current bandwidth (bits per second). If the value of this field is X'FFFFFFFF', refer to the ifHSpeed field in the record for the true estimate of the interface's bandwidth.
				For EQDIO devices:	
156	9C	Unsigned	4	MinInputBuffers	The minimum number of input buffers available for use by an EQDIO device
160	A0	Unsigned	4	MaxInputBuffers	The maximum number of input buffers available for use by an EQDIO device
				For PVMIUCV and SNALU62 devices:	
156	9C	Character	8	LocalNode	Local node name
164	A4	Character	8	LocalUser	Local user identifier
172	AC	Character	8	RemoteNode	Remote node name
180	B4	Character	8	RemoteUser	Remote user identifier
				For CLAW devices:	
156	9C	Character	8	ClawHostName	Host name
164	A4	Character	8	ClawAdapterName	Adapter name
172	AC	Character	8	ClawControlTask	Control task name
180	B4	Unsigned	4	ClawReadBuffers	Read buffers

Table 29. TCP/IP Link Definition Record - Type '08'x - Configuration Data (continued)

Dec	Hex	Type	Len	Name	Description
184	B8	Unsigned	4	ClawWriteBuffers	Write buffers
188	BC	Unsigned	4	ClawReadSize	Read buffer size
192	C0	Unsigned	4	ClawWriteSize	Write buffer size
				For all devices:	
196	C4	Unsigned	4	ifHSpeed	Estimate of the interface's current bandwidth (million bits per second).

Table 30 on page 235 shows the record layout for the TCP/IP ACB Record. This record is produced as Sample-class data and has a TCP/IP sub-record type of '09'x.

Table 30. TCP/IP ACB Record - Type '09'x - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	RtNowQSize	High-priority queue size
4	4	Unsigned	4	PriorQSize	Medium-priority queue size
8	8	Unsigned	4	ToDoQSize	Low-priority queue size
12	C	Unsigned	4	RtNowMaxQSize	High-priority queue maximum size
16	10	Unsigned	4	PriorMaxQSize	Medium-priority queue maximum size
20	14	Unsigned	4	ToDoMaxQSize	Low-priority queue maximum size
24	18	Unsigned	4	MarkedForDeathAcbs	ACBs requiring purge
28	1C	Unsigned	4	MoveToTimerQueueAcbs	ACBs moved to timer queue
32	20	Unsigned	4	QueuesEmpty	Times no ACBs to schedule
36	24	Signed	2	SchedulerStatsSize	Process scheduler statistics area size
38	26	Reserved	2	.	Reserved
40	28	Array of structures	*	SchedulerStatistics	Scheduler statistics for each process
*	*	Array of structures	*	DeviceStatistics	Device statistics for each device type

Table 31 on page 235 shows the layout of each structure element of the process and device statistics arrays.

Table 31. Process and Device Statistics Structure Layout

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	AcbsScheduled	ACBs scheduled
4	4	Unsigned	4	ElapsedTime	Elapsed time ACB (active in microseconds)
8	8	Unsigned	4	VirtualCpuTime	Virtual CPU time ACB (active in microseconds)

Table 31. Process and Device Statistics Structure Layout (continued)

Dec	Hex	Type	Len	Name	Description
12	C	Unsigned	4	ElapsedMax	Maximum elapsed time ACB (active in microseconds)
16	10	Unsigned	4	VirtualCpuMax	Maximum virtual CPU time ACB (active in microseconds)

Table 32 on page 236 shows the record layout for the TCP/IP CPU Record. This record is produced as Sample-class data and has a TCP/IP sub-record type of '0A'x. One record is generated for each virtual CPU in the configuration.

Table 32. TCP/IP CPU Record - Type '0A'x - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	TOD	8	VirtualCPU	Virtual CPU time used
8	8	TOD	8	DispatchTOD	Time of last dispatch
16	10	Integer	8	AccumulatedWaitTime	Idle time (active in microseconds)
24	18	TOD	8	WaitStartTOD	Wait start time (if idle at sample) or 0
32	20	Unsigned	2	CpuAddress	Virtual processor address
34	22	Character	2	.	Reserved

Table 33 on page 236 shows the record layout for the TCP/IP CCB Record. This record is produced as Sample-class data and has a TCP/IP sub-record type of '0B'x.

Table 33. TCP/IP CCB Record - Type '0B'x - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	Character	8	Name	Client user identifier
8	8	Character	8	SubtaskName	Subtask name
16	10	TOD	8	StartTime	Connection start time
24	18	TOD	8	ResponseSendTime	Last response transmission time
32	20	TOD	8	NoticeSendTime	Last notice transmission time
40	28	Unsigned	4	RequestsReceived	Requests received from client
44	2C	Unsigned	4	ResponsesSent	Responses transmitted to client
48	30	Unsigned	4	NoticesSent	Notices transmitted to client
52	34	Unsigned	4	RequestTime	Cumulative request time (in microseconds)
56	38	Unsigned	4	QueueTime	Cumulative request queue time (in microseconds)
60	3C	Unsigned	4	NoticeTime	Cumulative notice time (in milliseconds)
64	40	Unsigned	4	ReceiveDelay	Cumulative receive delay (in milliseconds)

Table 34 on page 237 shows the record layout for the TCP/IP Tree Size Record. This record is produced as Sample-class data and has a TCP/IP sub-record type of '0C'x.

Table 34. TCP/IP Tree Size Record - Type '0C'x - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	Array of unsigned	6*4	Sizes	Tree sizes: 0 - Reserved 1 - IP routing 2 - TCP connections 3 - UDP 4 - Address translation
24	18	Array of unsigned	6*4	FreeEntries	Free entry counts: 0 - Reserved 1 - IP routing 2 - TCP connections 3 - UDP 4 - Address translation
48	30	Array of unsigned	6*4	MinFreeEntries	Free entry count minima: 0 - Reserved 1 - IP routing 2 - TCP connections 3 - UDP 4 - Address translation
72	48	Unsigned	4	NumArrayElements (nn)	Number of array elements in the following three arrays.
76	4C	Array of unsigned	nn*4	AdditionalSizes	Additional tree sizes: 0 - Neighbor cache 1 - IPv6 routing
a = (nn*4) + 76	x = (nn*4) + 4C	Array of unsigned	nn*4	AdditionalFree	Additional free entry counts: 0 - Neighbor cache 1 - IPv6 routing
b = (nn*4) + a	y = (nn*4) + x	Array of unsigned	nn*4	AdditionalMinFree	Additional free entry count minima: 0 - Neighbor cache 1 - IPv6 routing

Table 35 on page 237 shows the record layout for the TCP/IP Home Record. This record is produced as Configuration-class data and has a TCP/IP sub-record type of '0D'x.

Table 35. TCP/IP Home Record - Type '0D'x - Configuration Data

Dec	Hex	Type	Len	Name	Description
0	0	Hex	4	HomeAddress1	IP address
4	4	Reserved	4	.	Reserved

Table 35. TCP/IP Home Record - Type '0D'x - Configuration Data (continued)

Dec	Hex	Type	Len	Name	Description
8	8	Character	16	HomeLinkName1	Link name
24	18	Hex	4	HomeAddress2	IP address
28	1C	Reserved	4	.	Reserved
32	20	Character	16	HomeLinkName2	Link name
				⋮	
nn	xx	Hex	4	HomeAddressN	IP address
nn	xx	Reserved	4	.	Reserved
nn	xx	Character	16	HomeLinkNameN	Link name

Table 36 on page 238 shows the record layout for the TCP/IP IPv6 Home Record. This record is produced as Configuration-class data and has a TCP/IP sub-record type of '0E'x.

Table 36. TCP/IP IPv6 Home Record - Type '0E'x - Configuration Data

Dec	Hex	Type	Len	Name	Description
0	0	Hex	16	Home6Address1	Home address (first home entry)
16	10	Reserved	4	.	Reserved
20	14	Character	16	Home6LinkName1	Home link name
36	24	Character	1	Home6AutoConfig1	AutoConfig flag
37	25	Character	1	Home6DadState1	Duplicate address detection flag
38	26	Character	1	Home6Depricated1	Deprecated flag
39	27	Character	1	Home6JoinedSolicited1	Joined solicited flag
40	28	Unsigned	4	Home6ValidLifetime1	Valid lifetime value
44	2C	Reserved	4	.	Reserved
48	30	Hex	8	Home6ValidStartTime1	Valid start time
56	38	Unsigned	4	Home6PreferredLifeTime1	Preferred lifetime
60	3C	Reserved	4	.	Reserved
64	40	Unsigned	4	Home6Scope1	Scope of address
68	44	Character	1	Home6Origin1	Origin flag
69	45	Character	3	.	Reserved
72	48	Hex	16	HomeAddress2	Home address (second home entry)
88	58	Reserved	4	.	Reserved
92	5C	Character	16	HomeLinkName2	Link name
108	6C	Character	1	Home6AutoConfig2	AutoConfig flag
109	6D	Character	1	Home6DadState2	Duplicate address detection flag
110	6E	Character	1	Home6Deprecated2	Deprecated flag

Table 36. TCP/IP IPv6 Home Record - Type '0E'x - Configuration Data (continued)

Dec	Hex	Type	Len	Name	Description
111	6F	Character	1	Home6JoinedSolicited2	Joined solicited flag
112	70	Unsigned	4	Home6ValidLifetime2	Lifetime value
116	74	Reserved	4	.	Reserved
120	78	Hex	8	Home6ValidStartTime2	Valid start time
128	80	Unsigned	4	Home6PreferredLifeTime2	Preferred lifetime
132	84	Reserved	4	.	Reserved
136	86	Unsugned	4	Home6Scope2	Scope of address
140	8C	Character	1	Home6Origin2	Origin flag
141	8D	Character	3	.	Reserved
				...	
				...	
				...	
nn*72	nn*48	Hex	16	HomeAddress*nn*	Home address (*nn*=last home entry)
(nn*72) +16	(nn*48)+10	Reserved	4		Reserved
(nn*72)+20	(nn*48)+14	Character	16	HomeLinkName*nn*	Link name
				...	
				...	
				...	
(nn*72)+68	(nn*48)+44	Character	1	Home6Origin*nn*	Origin flag
(nn*72)+69	(nn*48)+45	Character	3	.	Reserved

Table 37 on page 239 shows the record layout for the TCP/IP Takeover Record. This record is produced as Configuration-class data and has a TCP/IP sub-record type of '0F'x.

Table 37. TCP/IP Takeover Record - Type ''0F'x - Event Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	IpProtocolVersion	The IP protocol version this takeover event represents
4	4	Unsigned	4	FailingInterfaceNumber	The interface number of the failing interface
8	8	Unsigned	4	TakeoverInterfaceNumber	The interface number of the interface that is performing the takeover
12	C	Unsigned	4	TakeoverInterfaceSpeed	The speed of the interface that is performing the takeover

Table 37. TCP/IP Takeover Record - Type ''0F'x - Event Data (continued)

Dec	Hex	Type	Len	Name	Description
16	10	Unsigned	1	TakeoverInterfaceType	The network type of the interface performing takeover. Possible values: 3 - Ethernet 4 - EtheOr802.3 5 - Only802.3 23 - QDIO Ethernet mode

Table 38 on page 240 shows the record layout for the TCP/IP Link Deletion Record. This record is produced as Event-class data and has a TCP/IP subrecord type of '10'x. One record is generated each time a link is deleted from the TCP/IP configuration.

Table 38. TCP/IP Link Deletion Record - Type '10'x - Event Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	InterfaceNumber	Interface number
4	4	Unsigned	4	LinkNumber	Link number
8	8	Hex	4	DeviceNumber	Device number
12	C	Unsigned	1	NetworkType	Network types: 3 - Ethernet 4 - EtherOr802.3 5 - Only802.3 6 - TokenRing 8 - IUCV 9 - CTC 12 - A220 14 - FDDI 15 - CLAWip 21 - Virtual Device (VIPA) 22 - OSA ATM native 23 - QDIO Ethernet mode 24 - QDIO ATM mode 25 - QDIO Token Ring 26 - HiperSockets 28 - EQDIO Ethernet
13	D	Unsigned	1	DeviceType	Device types: 1 - LCS 6 - PVMIUCV 9 - CTC 10 - HCH 12 - CLAW 14 - Virtual (VIPA) 15 - ATM 16 - OSD (OSA Direct Express Adapter) 17 - HiperSockets 18 - Local IUCV 22 - EQDIO
14	E	Reserved	2	.	Reserved

Table 38. TCP/IP Link Deletion Record - Type '10'x - Event Data (continued)

Dec	Hex	Type	Len	Name	Description
16	10	Character	16	NetworkName	Network name
32	20	Character	16	DeviceName	Device name
48	30	Character	8	UserId	User performing the deletion

PI end

Appendix G. VMRM APPLDATA Monitor Records

PI

APPLDATA for VMRM is returned in Domain 10 (APPLDATA domain) Record 2 (Application Data Sample Record).

To begin data collection, enable the APPLDATA domain and start monitoring (using the CP MONITOR command). Also, the OPTION APPLMON statement must be specified in the directory for VMRMSVM.

The data records for servers are domain X'A' APPLDATA records, the general format of which is described in HTML format at IBM: z/VM data areas, control blocks, and monitor records (https://www.ibm.com/vm/pubs/ctlblk.html).

Each data record consists of these parts:

- Monitor record header
- VMRM APPLDATA record header
- VMRM application data: see Table 40 on page 243.

The VMRM APPLDATA header data consists of the following:

Table 39. VMRM APPLDATA Header Data

Data Item	Number of Bytes
Byte offset to application data relative to start of this record	2
Length in bytes of application data	2
User ID of the server machine (in EBCDIC)	8
Product identification (in EBCDIC). For VMRM records, this field contains **5739A03RM0000000**.	16
Status	1
Reserved	3

VMRM Application Data

The VMRM application data follows the APPLDATA header data. Table 40 on page 243 shows the record layout for the VMRM application data. The offset values listed are the offsets into the application data area of the monitor record (field APLSDT_ADATA). Always use the byte offset and length fields in the standard domain 10 records to locate the start and end of the application data within the record.

Note that up to 143 workload (VMRMSVM_WRKLD) entries are reported, depending on the number of workloads that have had associated users logged on at anytime since VMRMSVM started. If a workload was updated during a sample interval, it is flagged as being active (VMRMSVM_ACWKLD) during that interval. Otherwise, the workload information is still reported, but this flag is reset and the workload is considered inactive.

Table 40. VMRM APPLDATA

Dec	Hex	Type	Len	Name (Dim)	Description
52	34	CHAR	8	VMRMSVM_HDR	VMRM header.
52	34	CHAR	1	VMRMSVM_VERSION	The version of VMRM in use.

Table 40. VMRM APPLDATA (continued)

Dec	Hex	Type	Len	Name (Dim)	Description
53	35	UNSIGNED	1	VMRMSVM_DATAOFF	The byte offset, relative to the start of the actual SVM data.
54	36	UNSIGNED	1	VMRMSVM_LENTRS	The length, in bytes, of each workload entry.
55	37	BITSTRING	1	VMRMSVM_CTLFLG	Control header flags.
		1.....		VMRMSVM_NEWCFG	A bit that when set indicates that a new configuration file has been put into production. This bit is reset after the first sample interval containing workload information has been reported.
		.1111111		*	Reserved and available for IBM use.
56	38	CHAR	1	*	Reserved and available for IBM use.
57	39	UNSIGNED	1	VMRMSVM_CWRKLDS	The count of workloads, in the VMRM configuration file, in production at the time of this sample interval. The maximum number of workload entries reported will not exceed 143.
58	3A	CHAR	2	*	Reserved and available for IBM use.
60	3C	CHAR	*	VMRMSVM_WRKLD	The start of the VMRM configuration file workload entries.

- VMRMSVM_WRKLD per entry content:

Table 41. VMRM APPLDATA – VMRMSVM_WRKLD per entry content

Type	Len	Name	Description
CHAR	16	VMRMSVM_NWRKLD	The name of the workload record.
UNSIGNED	4	*	Reserved and available for IBM use.
UNSIGNED	1	VMRMSVM_GCPU	The target CPU velocity GOAL associated with this workload.
UNSIGNED	1	VMRMSVM_ACPU	The actual CPU velocity GOAL associated with this workload that was achieved.
UNSIGNED	1	VMRMSVM_GDASD	The target DASD velocity GOAL associated with this workload.
UNSIGNED	1	VMRMSVM_ADASD	The actual DASD velocity GOAL associated with this workload that was achieved.
UNSIGNED	1	VMRMSVM_IMPVAL	The importance value, ranging from 1–10, associated with this workload.

Table 41. VMRM APPLDATA – VMRMSVM_WRKLD per entry content (continued)

Type	Len	Name	Description
BITSTRING	1	VMRMSVM_WKLFLG	Workload flag byte
1.....		VMRMSVM_ACWKLD	A bit that when set indicates that this workload was recently active.
.1111111		*	Reserved and available for IBM use.
CHAR	2	*	Reserved and available for IBM use.

PI end

Appendix H. SSL Server Monitor Records

PI

The SSL Server contributes to the CP monitor data by creating records in the APPLDATA domain. To begin data collection:

1. Add the APPLMON directory option to directory profile entry of the pertinent SSL server pool.
2. From an authorized user, issue the CP MONITOR command to enable the APPLDATA domain for sample and event recording and to start monitoring.

The data records for servers are domain X'A' APPLDATA records. For a description of their general format, see:

z/VM Data Areas, Control Blocks, and Monitor Records (https://www.vm.ibm.com/pubs/ctlblk.html)

Each data record consists of these parts:

- Monitor record header
- SSL APPLDATA record header
- SSLSERV application data.

The SSL APPLDATA header data consists of the following:

Table 42. SSL APPLDATA Header Data

Data Item	Number of Bytes
Byte offset to application data relative to start of this record	2
Length in bytes of application data	2
User ID of the service machine (in EBCDIC)	8
Product identification (in EBCDIC). For SSL server records, this field contains **5735FALSSLx00000**. • **5735FALSSL** — Constant • **x** — record identifier (C or S) • **00000** — reserved	16
Status	1
Reserved	3

Following the header data is the SSL server data. SSLSERV produces two record formats: config record and sample record.

Table 43 on page 247 shows the record layout for the SSL server monitor. This record is produced as Config-class data.

Table 43. SSL Server Monitor - Config Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	cacheSize	Total number of cache elements
4	4	Unsigned	4	cacheLife	Hours cache is valid

Table 43. SSL Server Monitor - Config Data (continued)

Dec	Hex	Type	Len	Name	Description
8	8	Hex	4	sessPoolSize	Maximum number of secure connections
12	C	Character	8	userID	VM TCP/IP stack user ID

Table 44 on page 248 shows the record layout for the SSL server monitor. This record is produced as Sample-class data.

Table 44. SSL Server Monitor - Sample Data

Dec	Hex	Type	Len	Name	Description
0	0	Unsigned	4	countThBk	Reserved
4	4	Unsigned	4	timeThBk_sec	Reserved
8	8	Unsigned	4	timeThBk_usec	Reserved
12	C	Unsigned	4	countToDo	Reserved
16	10	Unsigned	4	timeToDo_sec	Reserved
20	14	Unsigned	4	timeToDo_usec	Reserved
24	18	Unsigned	4	countCert	Reserved
28	1C	Unsigned	4	timeCert_sec	Reserved
32	20	Unsigned	4	timeCert_usec	Reserved
36	24	Unsigned	4	countWork	Reserved
40	28	Unsigned	4	timeWork_sec	Reserved
44	2C	Unsigned	4	timeWork_usec	Reserved
48	30	Unsigned	4	sessNumActive	Number of active sessions
52	34	Unsigned	4	sessNumHWM	The highest number of active sessions
56	38	Unsigned	4	sessTimeActive_sec	Cumulative time sessions have been open (updated when a session closes)
60	3C	Unsigned	4	sessTimeActive_usec	Cumulative time sessions have been open (updated when a session closes)
64	40	Unsigned	4	thrdClosed	Reserved

68	44	Character	1	currentState	State of the SSL Server:
					`A=Available`
					`N=Not available`

69	45	Character	1	traceStatus	Trace status:
					`'N'ormal`
					`'C'onnections`
					`'F'low  'O'ff`

Table 44. SSL Server Monitor - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description

70	46	Character	1	traceScope	Trace scope:
					`0:off`
					`1:one`
					`0xff:all`

71	47	Character	1	reserved1	Reserved
72	48	Unsigned	4	xfersIn	Number of inbound (decryption) operations See note “3” on page 252
76	4C	Unsigned	4	xfersOut	Number of outbound (encryption) operations See note “3” on page 252
80	50	Unsigned	4	MbytesIn	Millions of bytes of data inbound See notes “2” on page 252 and “3” on page 252
84	54	Unsigned	4	bytesIn	Bytes in addition to MbytesIn See notes “2” on page 252 and “3” on page 252
88	58	Unsigned	4	MbytesOut	Millions of bytes of data outbound See notes “2” on page 252 and “3” on page 252
92	5C	Unsigned	4	bytesOut	Bytes in addition to MbytesOut See notes “2” on page 252 and “3” on page 252
96	60	Unsigned	4	timeIn_sec	Time spent processing inbound data See notes “2” on page 252 and “3” on page 252
100	64	Unsigned	4	timeIn_usec	Time spent processing inbound data See note “3” on page 252
104	68	Unsigned	4	timeOut_sec	Time spent processing outbound data See note “3” on page 252
108	6C	Unsigned	4	timeOut_usec	Time spent processing outbound data See note “3” on page 252
112	70	Unsigned	4	countStatic	Static connection requests

Table 44. SSL Server Monitor - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
116	74	Unsigned	4	countDynamic	Dynamic connection requests
120	78	Unsigned	4	countNewStart	New connect requests
124	7C	Unsigned	4	timeNewStart_sec	Time spent on new requests
128	80	Unsigned	4	timeNewStart_usec	Time spent on new requests
132	84	Unsigned	4	countResumed	Number of resumed connect requests
136	88	Unsigned	4	timeResumed_sec	Time spent on resumed requests
140	8C	Unsigned	4	timeResumed_usec	Time spent on resumed requests
144	90	Unsigned	4	countDisconnect	Number of requests to close connections
148	94	Unsigned	4	timeDisconnect_sec	Time spent on disconnection requests
152	98	Unsigned	4	timeDisconnect_usec	Time spent on disconnection requests
156	9C	Unsigned	4	countMiscReqs	Number of "Other" requests (e.g., queries)
160	A0	Unsigned	4	timeMiscReqs_sec	Time spent on Other requests
164	A4	Unsigned	4	timeMiscReqs_usec	Time spent on Other requests
168	A8	Unsigned	4	countNstrength	Number of connections with cipher strength 'None'
172	AC	Unsigned	4	countLstrength	Number of connections with cipher strength 'Low'
176	B0	Unsigned	4	countMstrength	Number of connections with cipher strength 'Medium'
180	B4	Unsigned	4	countHstrength	Number of connections with cipher strength 'High'
184	B8	Unsigned	4	count40bits	Number of connections with 40-bit session cipher
188	BC	Unsigned	4	count56bits	Number of connections with 56-bit session cipher
192	C0	Unsigned	4	count128bits	Number of connections with 128-bit session cipher
196	C4	Unsigned	4	count168bits	Number of connections with 168-bit session cipher
200	C8	Unsigned	4	countOtherbits	Number of connections with Other cipher length

Table 44. SSL Server Monitor - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
204	CC	Unsigned	4	count512bitsPK	Number of connections with 512 bit PK cipher See note “4” on page 252
208	D0	Unsigned	4	count1024bitsPK	Number of connections with 1024 bit PK cipher See note “4” on page 252
212	D4	Unsigned	4	count2048bitsPK	Number of connections with 2048 bit PK cipher See note “4” on page 252
216	D8	Unsigned	4	count4096bitsPK	Number of connections with 4096 bit PK cipher See note “4” on page 252
220	DC	Unsigned	4	countOtherbitsPK	Number of connections with Other PK cipher bit length See note “4” on page 252
224	E0	Unsigned	4	avgEvPerSel	Running averages of I/O events per select()
228	E4	Unsigned	4	avgTimePerSel_sec	Running averages of time spent outside select() per one cycle
232	E8	Unsigned	4	avgTimePerSel_usec	Running averages of time spent outside select() per one cycle
236	EC	Unsigned	4	avgTimeInSel_sec	Running averages of time spent in select() per one cycle
240	F0	Unsigned	4	avgTimeInSel_usec	Running averages of time spent in select() per one cycle
244	F4	Unsigned	4	maxEvPerSel	Max I/O event per select()
248	F8	Unsigned	4	maxTimePerSel_sec	Max time spent outside select() per one cycle
252	FC	Unsigned	4	maxTimePerSel_usec	Max time spent outside select() per one cycle
256	100	Unsigned	4	maxTimeInSel_sec	Max time spent in select() per one cycle
260	104	Unsigned	4	maxTimeInSel_usec	Max time spent in select() per one cycle
264	108	Unsigned	4	wakePerSec	Number of select() returns per previous second
268	10C	Unsigned	4	eventPerSec	Number of I/O events reported per previous second

Table 44. SSL Server Monitor - Sample Data (continued)

Dec	Hex	Type	Len	Name	Description
272	110	Unsigned	4	consecPerSec	Number of consecutive events belonging to the same session per previous second
276	114	Unsigned	4	sleepToggles	Number of times sleep optimization was toggled on or off
280	118	Unsigned	4	sleepSeconds	Time in seconds when sleep optimization was on
284	11C	Unsigned	4	count256bits	Number of connections with 256-bit cipher session
288	120	Unsigned	4	count160EC	Number of connections with 160-bit elliptical curve key
292	124	Unsigned	4	count192EC	Number of connections with 192-bit elliptical curve key
296	128	Unsigned	4	count224EC	Number of connections with 224-bit elliptical curve key
300	12C	Unsigned	4	count256EC	Number of connections with 256-bit elliptical curve key
304	130	Unsigned	4	count320EC	Number of connections with 320-bit elliptical curve key
308	134	Unsigned	4	count384EC	Number of connections with 384-bit elliptical curve key
312	138	Unsigned	4	count512EC	Number of connections with 512-bit elliptical curve key
316	13C	Unsigned	4	count521EC	Number of connections with 521-bit elliptical curve key
320	140	Unsigned	4	countOtherEC	Number of connections with other elliptical curve key

Notes:

1. Times are measured in two fields: _sec fields, which give the number of seconds, and _usec fields, which gives the number of additional microseconds.

   To calculate the total time in microseconds, use the following formula:

   Total time in microseconds = (Time_sec * 1000000) + Time_usec
2. Bytes are calculated in two fields: Mbytes, which gives how many millions of bytes; and Bytes, which gives the value of any additional bytes.

   To calculate the total bytes received or sent, use the following formula:

   Total Bytes = (Mbytes * 1000000) + Bytes
3. These fields are not updated until after a given connection has been closed.
4. These fields refer to the size of the Public Key from the certificate that is used in the SSL handshake.

PI end

Appendix I. z/VM Performance Data Pump

z/VM Performance Data Pump (Data Pump) converts machine-readable z/VM monitor and SFS data into a generic text-based data stream. Modern tools can use the data stream to display real-time performance dashboards, aggregate real-time data for long-term usage analysis, or integrate with existing enterprise observability solutions.

Purpose

Data Pump provides high-quality z/VM performance data to enterprise monitoring tools that are already deployed for application monitoring or capacity planning. Such tools align with skills and experiences of many users and offer integration with other tools and solutions.

Data Pump by itself does not deliver value that a user can readily use. To take advantage of Data Pump the customer must provision, configure, and deploy other services to process the data stream. While instructions for deploying the open source solutions are available, these components are not delivered with the z/VM product. For more information, see z/VM Performance Data Pump (https://www.vm.ibm.com/related/perfkit/datapump/).

Relationship to Performance Toolkit

z/VM Performance Data Pump is licensed with Performance Toolkit for z/VM but does not support, depend upon, or interact with Performance Toolkit for z/VM in any way. Data Pump does not depend on function that is provided by Performance Toolkit, and Performance Toolkit does not process data from Data Pump. Data Pump is an alternative way to process the same CP monitor data that Performance Toolkit can process. A customer might decide to use Performance Toolkit or Data Pump or both, depending on the requirements. When a customer wants to use both tools, a separate virtual machine is required to run the Data Pump program.

Setting up the DATAPUMP virtual service machine

Data Pump can run in any CMS virtual machine that has sufficient privilege and resources. For the typical use case of continuously collecting z/VM performance data, it is practical to create a dedicated virtual service machine.

It is convention (but not a necessity) that Data Pump runs on the DATAPUMP user ID. The following instructions use the DATAPUMP user ID convention and assume a single system image (SSI) cluster.

Enabling Data Pump

The z/VM Performance Data Pump program requires a license for Performance Toolkit for z/VM. Before you use z/VM Performance Data Pump, you must enable Performance Toolkit for z/VM as described in *MEMO TO USERS for IBM Performance Toolkit for z/VM*. The *MEMO TO USERS for IBM Performance Toolkit for z/VM* is provided to customers who obtain a Performance Toolkit for z/VM license.

Updating the user directory

Create a user directory entry like the following example:

```
IDENTITY DATAPUMP LBYONLY 128M 512M EG
 INCLUDE IBMDFLT
 ACCOUNT IBM
 NAMESAVE MONDCSS
 IUCV *MONITOR MSGLIMIT 255
 SHARE ABS 1%
 IPL CMS PARM AUTOCR FILEPOOL VMSYS:
 OPTION SVM
 LOGONBY IBMVM1
```

```
CONSOLE 0009 3215 T
LINK PERFSVM 201 201 RR
```

Notes on the example user directory:

1. The user ID is defined as IDENTITY because the virtual service machine runs on each member of a single system image (SSI) cluster. Because there are no per-member resources (beyond the VMSYS file space), no SUBCONFIG entries are required.
2. The user ID allows LOGON BY IBMVM1 to diagnose problems during customization. When Data Pump runs without error, you might change the LBYONLY parameter to AUTOONLY.
3. The user ID is defined with privilege class E. If the CP monitor does not start by the time that Data Pump starts, privilege class E permits Data Pump to issue the **MONITOR** command to start the CP monitor.
4. The NAMESAVE and IUCV statements are required to collect data from the CP monitor.
5. The entries define space in the local VMSYS file pool for an A-disk.

If you use an external security manager (ESM), then grant the DATAPUMP user ID read access to PERFSVM 201, which is where the Data Pump code is located.

Basic tasks to configure the DATAPUMP virtual service machine

To configure the DATAPUMP virtual service machine, the following basic tasks must be completed for each cluster member. You can complete the tasks manually for each cluster member. As an alternative, you can use the MDXSETUP EXEC, which automates some configuration tasks.

- On each member of the cluster, enroll the DATAPUMP user ID with 1000 blocks in the member's local VMSYS file pool.
- Create a PROFILE EXEC file in the VMSYS:DATAPUMP directory. Include statements that access the PERFSVM 201 disk and a statement that issues the **DATAPUMP** command.
- In the VMSYS:DATAPUMP directory, create the configuration files for the Data Pump program. For more information, see “Configuring Data Pump ” on page 258.

Using the MDXSETUP EXEC to configure the DATAPUMP virtual service machine

The MDXSETUP EXEC, which is included with Data Pump, configures the DATAPUMP virtual service machine in a way that simplifies administration among the members of a single system image (SSI) cluster. Data Pump configuration files are shared among cluster members by sharing a configuration file in a global file pool. In the rare case where a single Data Pump configuration is not appropriate for every member, you can specify a system-specific Data Pump configuration in a local file pool.

Note: VMPSFS is the default global file pool, and Data Pump uses VMPSFS if it exists. If VMPSFS does not exist, Data Pump finds and uses another global file pool. In the following example, the VMPSFS global file pool exists.

Run MDXSETUP EXEC once on each member of the SSI. You can run MDXSETUP EXEC when you run PUT2PROD to install the code, or you can log on and run MDXSETUP EXEC on one member at a time later. The following instructions assume that you run MDXSETUP EXEC on one member at a time.

Update the first member of the SSI cluster

Use a version-appropriate MAINT*xxx* user ID (for example, MAINT730) to run MDXSETUP EXEC on the first member. In this example, the system name is VMA. The following screen is an example of running MDXSETUP EXEC.

```
vmlink perfsvm 201 perfsvm 1cc ( invoke mdxsetup
DMSVML2060I PERFSVM 201 linked as 0120 file mode Z
DMSVML2060I PERFSVM 1CC linked as 0121 file mode X
Updating files:
DEFAULT DATAPUMP 2023-08-17 03:51:50 VMPSFS:DATAPUMP.
PROFILE EXEC 2023-08-17 03:51:50 VMPSFS:DATAPUMP.
Customize configuration files with VMLINK .DIR VMPSFS:DATAPUMP. ( FILEL
DMSVML2061I PERFSVM 201 detached
DMSVML2061I PERFSVM 1CC detached
Ready; T=0.05/0.07 03:51:50
```

The MDXSETUP EXEC completes the following tasks:

- The MDXSETUP EXEC enrolls DATAPUMP in the global VMPSFS file pool and in the local VMSYS file pool.
- The MDXSETUP EXEC creates the following objects in the shared VMPSFS:DATAPUMP directory:
  - A sample PROFILE EXEC file for the DATAPUMP user ID
  - A default configuration file for the Data Pump program (DEFAULT DATAPUMP)
  - A system-specific subdirectory whose name matches the system name (VMA)
- The MDXSETUP EXEC replicates the following files from the shared VMPSFS:DATAPUMP directory to the local VMSYS:DATAPUMP directory:
  - The sample PROFILE EXEC file for the DATAPUMP user ID
  - The default configuration file for the Data Pump program (DEFAULT DATAPUMP)

Access the new shared VMPSFS:DATAPUMP directory to continue the configuration. For example, use the **VMLINK** command:

```
VMLINK .DIR VMPSFS:DATAPUMP. ( WRITE FILEL
 MAINT FILELIST A0 V 169 Trunc=169 Size=3 Line=1 Col=1 Alt=0
Directory = VMPSFS:DATAPUMP.
Cmd Filename Filetype Fm Format Lrecl Records Blocks Date Time
DEFAULT DATAPUMP Z1 V 16 2 1 4/21/23 3:51:50
PROFILE EXEC     Z1 V 38 9 1 4/21/23 3:51:50
VMA              Z DIR - - - 4/21/23 3:51:50
```

In the shared VMPSFS:DATAPUMP directory, update the DEFAULT DATAPUMP configuration file with information for your environment. For more information, see “Configuring Data Pump ” on page 258.

Notes:

- The PROFILE EXEC file contains statements that start the MDXSETUP EXEC and then start the Data Pump program. Every time that the virtual service machine is started and the PROFILE EXEC runs, the MDXSETUP EXEC replicates the Data Pump configuration files from the shared VMPSFS:DATAPUMP directory and the system-specific subdirectory to the local VMSYS:DATAPUMP directory. All SSI members can use the most recent configuration files that are in the shared VMPSFS:DATAPUMP directory.
- You can specify another shared file pool when you start MDXSETUP EXEC. For example, you can specify IPGATE and share the Data Pump configuration files over multiple clusters. Specify the file pool name as an operand of MDXSETUP EXEC. The default file pool is VMPSFS.

Tip: To allow another user ID to customize the Data Pump configuration, grant that user ID write access to the directories and files.

Issue the **XAUTOLOG DATAPUMP** command to start the Data Pump program. If you made significant changes to the PROFILE EXEC file, then you might have to **FORCE** and then **XAUTOLOG** a second time to activate the changes in the file.

If you make more changes to the Data Pump configuration files, then use **FORCE** and **XAUTOLOG** or issue a **SEND CP DATAPUMP IPL** command to restart.

After the configuration of the first member is tested, configure the other members of the SSI cluster.

Run the MDXSETUP EXEC on other members of the SSI cluster

After the setup and configuration of the first member is complete, run the MDXSETUP EXEC once on each other member of the SSI cluster. You can simplify the process if you issue **XAUTOLOG** to a privileged user ID on the other members.

For example, for the VMB system, use the following command:

```
AT VMB CMD XAUTOLOG MIGMAINT CMD VMLINK PERFSVM 201 ( INVOKE MDXSETUP
```

The **XAUTOLOG** command creates a VMB subdirectory in the shared VMPSFS:DATAPUMP directory.

If required, create system-specific Data Pump configuration files

If any system requires a unique Data Pump configuration, create a DEFAULT DATAPUMP file in the system-specific subdirectory of the shared VMPSFS:DATAPUMP directory. (In this example, the system-specific sub directories are VMPSFS:DATAPUMP.VMA and VMPSFS:DATAPUMP.VMB.)

Usually, you use the same Data Pump configuration file for all SSI cluster members. When all systems feed data into the same InfluxDB or Splunk instance, a single shared configuration file is usually sufficient.

A system-specific configuration is for the rare scenario where a single common configuration file is not appropriate for Data Pump on all SSI members. System-specific Data Pump configuration files can contain MDXEXTRA files as required for advanced instrumentation.

The PROFILE EXEC file contains statements that start the MDXSETUP EXEC and then start the Data Pump program. The DEFAULT DATAPUMP file from the shared system-specific subdirectory replaces the DEFAULT DATAPUMP file from the shared parent directory (VMPSFS:DATAPUMP). Data Pump starts and uses the system-specific DEFAULT DATAPUMP file.

The setup is complete. When the DATAPUMP virtual service machine starts on any member of the cluster, the Data Pump configuration files are replicated from the shared VMPSFS:DATAPUMP directory to the local VMSYS:DATAPUMP directory. The members are synchronized on a common Data Pump configuration but will use their system-specific Data Pump configurations, if they exist.

After configuration

You can remove DEBUG parameters that you set to help resolve problems while testing the configuration files. When DEBUG parameters are cleared, fewer messages are issued from the Data Pump program.

If you run IBM Operations Manager or a similar program to monitor virtual machines, you might want to make that service machine secondary console for DATAPUMP. That service machine will capture messages from the Data Pump program.

After configuration is complete and Data Pump runs as required, update the AUTOLOG1 or AUTOLOG2 user ID so that the DATAPUMP virtual machine starts after z/VM IPLs. In order for data collection to commence, DATAPUMP must be started after the TCPIP virtual machine starts.

Starting and stopping Data Pump

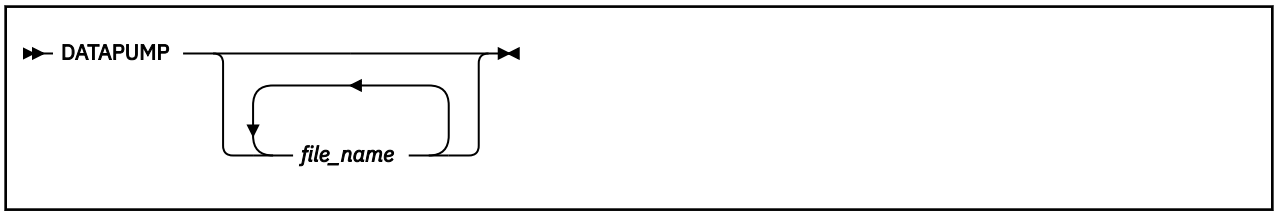
Start the Data Pump program by using the DATAPUMP command. You can stop the Data Pump program several ways:

- Use the **CP FORCE** command to end the service machine that runs Data Pump.
- When you are logged on to the user ID that is running Data Pump, you can use one of the following methods:
  - Use the **CP HX** command to halt execution.
  - You can **IPL CMS** (#CP IPL CMS) to restart Data Pump.

Other options to start and stop Data Pump depend on the configuration of the DATAPUMP virtual machine. For more information, see “Setting up the DATAPUMP virtual service machine” on page 253.

The DATAPUMP command

Format



Purpose

Use the **DATAPUMP** command to start the z/VM Performance Data Pump (Data Pump). Data Pump collects performance data and sends selected metrics from that data to the configured services.

Operands

file_name

The *file_name* operand specifies the file name of a Data Pump configuration file. The file type must be DATAPUMP. You can specify one or more configuration files. The configuration information of the files is concatenated.

If no configuration file is specified, then Data Pump searches the accessed file modes for a single default configuration file. The file type must be DATAPUMP. Data Pump searches for the following file names in the order specified:

1. A file name that matches the system name or the name of the user ID that runs the **DATAPUMP** command
2. DEFAULT

Note: The sample PROFILE EXEC file that the MDXSETUP EXEC creates issues the **DATAPUMP** command with no file operands. In that example, Data Pump uses the DEFAULT DATAPUMP configuration file. See “Setting up the DATAPUMP virtual service machine” on page 253.

Messages

The following messages can be returned from the **DATAPUMP** command. Data Pump messages are documented in *z/VM: Other Components Messages and Codes*. See Data Pump Messages.

- FCXxxx3001E This system is not licensed to use the program.
- FCXxxx3002E Incorrect option "*parm*" in "*process*".
- FCXxxx3003E TCP/IP connection failed reason "*reason*" error "*error*"
- FCXxxx3004E Incorrect section name "*string*".
- FCXxxx3005E Missing parameter "*parm*".
- FCXxxx3006E Duplicate values for parameter "*parm*".
- FCXxxx3007E Unknown parameter "*parm*".
- FCXxxx3008E Incorrectly formed parameter "*parm*".
- FCXxxx3009E Input file "*file*" not found.
- FCXxxx3010E Plug-in type "*type*" not found.
- FCXxxx3011E Plug-in collection "*collection*" not found.
- FCXxxx3012E Plug-in handler ended with RC=*retcode*.
- FCXxxx3013E Multiple sections of "*type*" not supported.

- FCXxxx3014E Value for "*parm*" must be a whole number (is "*string*").
- FCXxxx3015E Value for "*parm*" is "*value*", must be "*value*" or more.
- FCXxxx3016E Loading *MONITOR segment "*mondcss*" failed with RC=*retcode*.
- FCXxxx3017E Value for "*parm*" must be a boolean value, not "*string*".
- FCXxxx3018E Connection to *MONITOR System Service failed with RC=*retcode*.
- FCXxxx3019E Not authorized to start MONITOR Service.
- FCXxxx3020E Shared File Pool "*filepool*" not found.
- FCXxxx3021E Error *retcode* from SFS QUERY*pool*: "*string*".
- FCXxxx3024E File "*file*" already exists; specify REPLACE to overwrite.
- FCXxxx3025E File "*file*" is not a supported MONWRITE file.
- FCXxxx3030I Started "*task*", parameters "*string*".
- FCXxxx3031I Finished "*task*", response "*string*".
- FCXxxx3099E Unknown message "*id*" issued by "*module*".

Configuring Data Pump

Data Pump is configured by using configuration files. One or more configuration files can be specified as operands on the DATAPUMP command. The specified configuration files constitute the configuration when the Data Pump program is started.

The Data Pump configuration specifies where to collect data and what to do with the data. You can collect data from multiple sources. You can manipulate (filter, display, save, and distribute) the collected data. Typically, Data Pump distributes data through a TCP/IP connection to services like InfluxDB and Splunk. The services can be on remote systems or on the same z/VM system as the virtual machine where Data Pump runs.

Plug-in Types

The configuration defines instances of different types of plug-in modules. The different plug-in types work together to collect, filter, display, save, and distribute performance data. Although a configuration that defines only one plug-in instance is valid, it produces no output. To produce output, a configuration must include at least one plug-in that collects data and at least one plug-in that manipulates the collected data. Data Pump provides the following plug-in types:

MONITOR
The plug-in connects to the *MONITOR system service to collect real-time performance data. The data can be passed to other plug-ins that consume the monitor data.

MONWRITE
The plug-in reads z/VM monitor data that was previously recorded by the CP MONWRITE utility. The data can be passed to other plug-ins that consume the monitor data.

SFS
The plug-in issues a QUERY FILEPOOL command at regular intervals to collect usage information about the storage groups in the specified file pools. When combined with performance data, the SFS usage information can provide more insight about excessive usage and long-term trends.

INFLUXDB
The plug-in formats data according to the InfluxDB Line Write Protocol and sends the data to an InfluxDB instance by using TCP/IP.

SPLUNK
The plug-in uses the Splunk Enterprise HTTP Event Collector (HEC) to send the data to a Splunk service.

STATISTICS
The plug-in reports the volume of data and can help determine the effect of enabling or disabling certain domains or resources. The plug-in can also be used to diagnose other problems in the configuration.

DATADUMP
The plug-in writes CP monitor records to a separate disk file, which can be used for further diagnosis.

Data from the data-collecting plug-ins can be manipulated by the other plug-ins in the following ways:

Table 45. Interaction between data-collecting and other plug-ins

The data from this data-collecting plug-in:	**... can be consumed by these data-manipulating plug-ins:**
MONITOR	INFLUXDB, SPLUNK, STATISTICS, DATADUMP
MONWRITE	INFLUXDB, STATISTICS, DATADUMP
SFS	INFLUXDB, SPLUNK, STATISTICS

Configuration File Organization

A Data Pump configuration file is composed of one or more sections. Each section defines one instance of one type of plug-in module. A section has the following elements:

Section name
Each section begins with a name, which is a one word label that is enclosed by square brackets ([]). Some error messages include the name of the section that yields the error. Hence, unique section names can help identify the source of a problem.

The section name can use characters that are valid for CMS file names: A-Z, a-z, 0-9, $ (dollar sign), # (number sign), @ (at sign), + (plus sign), - (hyphen), : (colon), and _ (underscore). No blank space is allowed within the label or between the label and the brackets. The section name must be on a line that has no parameters.

The following examples are valid section names:

```
[influxdb#1]
[Live_Data]
```

Parameters

Type parameter
Each section contains one TYPE parameter. The TYPE parameter identifies a plug-in type.

The following examples are valid TYPE parameters:

```
type=monitor
type = influxdb
Type= Splunk
```

Other parameters
A section can contain other parameters that are appropriate for the plug-in type. Required and optional parameters for each plug-in type are listed in the plug-in topics. See “Related Topics” on page 261.

Comments
Comments are optional. A semicolon indicates a comment. After a semicolon, the remaining data on the line is not processed.

Tips for defining parameters:

The following tips might improve readability and manageability of configuration files:

- Indenting parameter lines might improve readability but is not required.

- Indent the parameter lines with two spaces so that the first space can easily be changed into a semicolon. Use the first space to "disarm" a parameter temporarily (or to leave the previous setting in the configuration file as documentation).
- A value that is a series of tokens can be expressed in portrait form with a trailing comma for continuation. Tokens can be removed by adding a semicolon and added again by removing the semicolon. The following parameter is an example of portrait form.

```
filepool = ,
  gplsrv1 ,
; gplsrv2 , ; Waiting for an admin to enroll the user
  gplsrv3
```

Configuration Requirements

Configurations have the following requirements:

- A configuration can contain a single section of TYPE=MONITOR or a single section of TYPE=MONWRITE but not both.
- Each section name and each parameter must be on a separate line.
- A parameter definition is a key-value pair, which has the format `parameter_name=value`
- When the spaces in a value are significant, enclose the value in double quotation marks.
- Configuration files are plain text files in "ini-file format" and must be file type DATAPUMP.

Plug-in topics include requirements that are specific to the plug-in. See "Related Topics" on page 261.

Configuration Notes

The following properties apply to a configuration:

- A configuration allows multiple instances (multiple sections) of the following plug-ins: SFS, INFLUXDB, SPLUNK, STATISTICS, DATADUMP.
- A data-manipulating plug-in can consume data from more than one data-collecting plug-in.
- Some parameters require a boolean value. The case of the value (uppercase or lowercase) is ignored.
  - The following boolean values can be used interchangeably: true, yes, on, 1.
  - The following boolean values can be used interchangeably: false, no, off, 0.
- A parameter definition can span two or more lines if a continuation character is used:
  - A comma (,) at the end of a line of data continues a parameter definition on the next line. When the parameter is processed, the lines are joined and the comma is replaced by a space.
  - A backslash (/) at the end of a line of data continues a parameter definition on the next line. When the parameter is processed, the lines are joined and the backslash is replaced by no space.
- Leading and trailing blank spaces around a section name, a key, or a one-word value are ignored.
- The order of sections and the order of parameters within a section does not affect the program.
- The configuration file is a flat hierarchy. Sections are not nested.
- Casing in section names and parameter names is ignored. Casing might be important for some parameter values that are unique to a user's environment such as user credentials and parts of URLs.
- Information from several configuration files can be combined. Specify the configuration files as operands when the DATAPUMP command is issued.

Plug-in topics include notes that are specific to the plug-in. See "Related Topics" on page 261.

Examples

The following configuration content is a minimal example that provides data from the *MONITOR system service to an InfluxDB time-series database:

```
; z/VM Performance Data Pump Configuration File

[input]
   type = monitor         ; Read data from *MONITOR CP System Service

[output_influxdb]
   type = InfluxDB
   url = https://lnxrmh01.pok.ibm.com:8086  ; InfluxDB server and port
```

Related Topics

For more information about configuring Data Pump, see the following topics.

- “Data Pump MONITOR plug-in” on page 262
- “Data Pump MONWRITE plug-in” on page 264
- “Data Pump SFS plug-in” on page 266
- “Data Pump INFLUXDB plug-in” on page 268
- “Data Pump SPLUNK plug-in” on page 273
- “Data Pump STATISTICS plug-in” on page 277
- “Data Pump DATADUMP plug-in” on page 278

Data Pump MONITOR plug-in

The Data Pump MONITOR plug-in reads data from the *MONITOR system service. Data Pump can pass the data to plug-ins that manipulate the monitor data.

Parameters

A configuration section that defines an instance of the MONITOR plug-in uses the following parameters.

TYPE=MONITOR
The TYPE parameter specifies the plug-in type. In this case, the plug-in type is MONITOR.

The TYPE=MONITOR parameter is required.

MONDCSS=*mondcss*
The MONDCSS parameter specifies the name of the monitor shared segment.

The default value is MONDCSS.

REFRESH=*intervals*
The REFRESH parameter specifies the number of monitor sample intervals before the monitor configuration data is refreshed. For example, if the monitor sample interval is defined as one minute and the MONITOR plug-in is configured for REFRESH=30, then the plug-in reads new monitor configuration data every 30 minutes. For more information, see usage note “1” on page 262.

The default value is 15.

START=*boolean*
The START parameter specifies whether Data Pump starts the CP monitor when the CP monitor is not started. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

For more information, see usage note “2” on page 262.

The default value is FALSE.

DEBUG=STARTUP
DEBUG=VERBOSE
The DEBUG parameter issues extra informational messages for the section.

- DEBUG=STARTUP issues diagnostic messages when the plug-in loads the DCSS and connects to *MONITOR.
- DEBUG=VERBOSE issues detailed diagnostic messages.

Example

For many installations, the following example is sufficient to enable data collection.

```
[section1]
   type=monitor
```

Usage Notes

1. The REFRESH parameter is similar to the CLOSE operand of the CP **MONWRITE** utility. The plug-in briefly disconnects from the *MONITOR system service and then reconnects again to get a fresh set of configuration records. Because some dashboard panels use data from configuration records (for example, release level data), set the refresh interval shorter than the minimal time range that you want to see in dashboards.
2. If another user does not start the *MONITOR system service after a system IPL, specify START=YES for Data Pump to enable the necessary sample domains. The user that runs Data Pump must be

authorized to issue the **CP MONITOR** command, which requires CP privilege class A or E and external security manager permission. If the monitor system service is started before Data Pump is started, then the START parameter is ignored.

3. If Data Pump is the only consumer of monitor data, then message HCP6224I is issued when Data Pump briefly disconnects from the *MONITOR service. You can ignore the messages or suppress the messages in your automated operations tables.
4. A configuration can define at most one instance of the MONITOR plug-in. If a configuration defines an instance of the MONWRITE plug-in, then the configuration cannot also define an instance of the MONITOR plug-in.

Data Pump MONWRITE plug-in

The Data Pump MONWRITE plug-in reads z/VM monitor data that was previously recorded by the CP MONWRITE utility. Data Pump can pass the data to plug-ins that manipulate the monitor data.

Parameters

A configuration section that defines an instance of the MONWRITE plug-in uses the following parameters.

TYPE=MONWRITE
The TYPE parameter specifies the plug-in type. In this case, the plug-in type is MONWRITE.

The TYPE=MONWRITE parameter is required.

FILE=*file*
The FILE parameter points to data that the MONWRITE utility created:

- If the section uses the LIST=YES option, then the FILE parameter must specify a file that contains a list of MONWRITE files.

  See usage notes "3" on page 265, "4" on page 265, "5" on page 265, "6" on page 265.
- If the section uses the LIST=NO option or does not specify the LIST parameter, then the FILE parameter must specify a single MONWRITE file.

The FILE parameter is required.

LIST=*boolean*
The LIST parameter specifies whether the file that is specified by the FILE parameter is a list of MONWRITE files. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is FALSE.

DEBUG=STARTUP
DEBUG=VERBOSE
The DEBUG parameter issues extra informational messages for the section.

- DEBUG=STARTUP issues informational messages about the input MONWRITE files.
- DEBUG=VERBOSE shows extra information about the input files.

Example

The following example reads a single MONWRITE file and reports a summary of the content by using the STATISTICS plug-in.

```
[input]
  type = monwrite
  file = D060722 T074530 B1
[stats]
  type = statistics
```

The following example reads a series of MONWRITE files. The INFLUXDB plug-in processes only the SYTCUP and SYTCUM monitor records. The STATISTICS plug-in processes all records.

```
[input]
  type = monwrite
  file = CMS EXEC A
  list = true
[influx]
  type = InfluxDB
  url = https://lnxrmh01.pok.ibm.com:8086/
  db = oldzvm
  include = sytcup sytcum
[stats]
```

```
  type = statistics
```

The following example reads a MONWRITE file from an SFS directory. It is not necessary to first use the CMS **ACCESS** command to access the directory:

```
[input]
  type = monwrite
  file = D2MC7740 MONDATA GPLSRV2:PERFTEAM.WAREHOUSE.REGRESSION.0912.CMS1
```

Usage Notes

1. When you use older MONWRITE data, verify the retention period that is set for the database. InfluxDB discards the data when it is older than the retention period for the database. You can either adjust the retention period temporarily, or use a separate database. A separate database requires a new set of Grafana dashboards that are imported to point to the new database.
2. When you import MONWRITE files for analysis, you can include the STATISTICS plug-in. The STATISTICS plug-in displays the time range of the data, which you can use to set the time range in your dashboard to see the data. The STATISTICS plug-in displays the date and time in Coordinated Universal Time format (UTC). Consider the time setting of your dashboard.
3. When you use multiple MONWRITE files, they must be processed in the order that they were created. A sort by file name is likely not an effective method to sort by creation date. In the list file, list the MONWRITE files in the correct order.

   The EXEC option of the CMS **LISTFILE** command can produce a list of files. However, the order of the files might not be the order in which the files were created.
4. When meter-type or counter-type metrics are computed, the first monitor sample of each file does not produce metrics. For example, if 15-minute MONWRITE files that contain 1-minute sample intervals are processed one by one, then every 15th sample is lost. When the LIST=YES parameter is used and the MONWRITE files are listed in a list file, only the first sample in the first file is lost.
5. The MONWRITE files that are specified in a list file must contain data from the same system and must comprise a contiguous time range. If data from more than one system or from discontiguous time ranges are used, then the computed interval values are incorrect.
6. If you specify a long SFS directory name, ensure the integrity of the directory name in the Data Pump configuration file. The Data Pump configuration file must be wide enough to contain the directory name on one line or the directory name must be continued to one or more lines by using the backslash (\) continuation character.
7. The MONWRITE data files can be packed or tersed. Compressed files do not have to be manually decompressed before Data Pump processes them.
8. A configuration can define at most one instance of the MONWRITE plug-in. If a configuration defines an instance of the MONITOR plug-in, then the configuration cannot also define an instance of the MONWRITE plug-in.

Data Pump SFS plug-in

The Data Pump SFS plug-in issues a **QUERY FILEPOOL** command at regular intervals to collect usage information about the storage groups in specified file pools. Data Pump can pass the data to plug-ins that manipulate the SFS data. When combined with performance data, the SFS usage information can provide more insight about excessive usage and long-term trends.

Parameters

A configuration section that defines an instance of the SFS plug-in uses the following parameters.

TYPE=SFS
The TYPE parameter specifies the plug-in type. In this case, the plug-in type is SFS.

The TYPE=SFS parameter is required.

FILEPOOL=*pools*
The FILEPOOL parameter specifies the resource names of the files pools to monitor.

The FILEPOOL parameter is required.

INTERVAL=*minutes*
The INTERVAL parameter specifies the delay (in minutes) between samples that the plug-in collects. The default value is 15.

For more information, see usage note "5" on page 267.

MIN=*blocks*
The MIN parameter specifies the minimum space usage (in blocks) that triggers a record for that file space. If a file space uses fewer blocks than the trigger value, then no record is created for that file space. The value must be a whole number. The default value is 1.

For more information, see usage note "4" on page 267.

TOP=*number*
The TOP parameter limits the sample to the top *number* file spaces in the storage group that use the most space. The value must be a whole number. The value defaults to 0, which means that the parameter is ignored and all file spaces that meet the minimum size are processed.

DEBUG=STARTUP
DEBUG=VERBOSE
The DEBUG parameter issues extra informational messages for the section.

- DEBUG=STARTUP issues diagnostic messages when the plug-in searches for file pools.
- DEBUG=VERBOSE issues detailed diagnostic messages.

You can specify DEBUG=STARTUP or DEBUG=VERBOSE in a section but you cannot specify both in the same section.

Examples

1. The following example limits the samples to the five file spaces that use the most space.

```
[vmpsfs]
  type = sfs
  filepool = vmpsfs
  top = 5  ; Limit samples to the top 5 file spaces that use the most space.
```

2. The following example shows how to divert SFS metrics to a different InfluxDB database. Two plug-ins (TYPE=MONITOR, TYPE=SFS) collect data for two instances of the INFLUXDB plug-in ([zvm_default], [zvm_long]). The [zvm_long] section identifies an InfluxDB database that is intended for SFS data and has a longer retention period than the zvm_default database. Because the [zvm_long] configuration section limits the data to only three record types (sfsgrp, sfsuse, sfslog), the data volume is less than the volume from the MONITOR plug-in. The lower volume enables longer retention. The records

from the SFS plug-in are excluded from the zvm_default database, but are included in the zvm_long database.

```
[live]
  type = monitor
[zvm_default]
  type = influxdb
  url = https://lnxrmh01.pok.ibm.com:8086/
  db = zvm_default
  exclude = sfsgrp sfsuse sfslog
[sfs]
  type = sfs
  filepool = vmsys vmsysu vmpsfs
[zvm_long]
  type = influxdb
  url = https://lnxrmh01.pok.ibm.com:8086/
  db = zvm_long
  include = sfsgrp sfsuse sfslog
```

Usage Notes

1. Given similar sample intervals and the default set of monitor records, the SFS data is a lower volume than monitor data. You might be able to retain SFS data longer than the monitor data. The changes in SFS usage are normally small enough that you can study trends over longer periods. Example “2” on page 266 shows how to use a separate database with a longer retention policy for SFS data. When you import the SFS Usage dashboard, you can select the Grafana data source that refers to that separate database.
2. By default, Data Pump monitors file pools only of SFS servers that are running on the same system (SSI cluster member). The motivation for this default behavior is that for a shared file pool, you want Data Pump on only one member of the SSI to monitor the file pool. Dashboards might show incorrect information when data from different systems is combined and Data Pump on multiple systems is collecting the same data. Because the file pool might run on any member of the cluster, you don't want to specify in advance where the server runs. By coding the shared file pool VMPSFS in the configuration on each member, only the Data Pump on the same member monitors the file pool. The other Data Pump instances ignore the entry. Because Live Guest Relocation is not supported for SFS servers, Data Pump does not “follow” the SFS server. When you restart the SFS server on a different member of the cluster, the file pool is still monitored by the Data Pump on the member where the SFS server was initially.
3. You can monitor a remote file pool when no Data Pump is running on the system where the SFS server is running. To monitor a remote file pool, specify the value by using the format *filepool*:*node*.
4. Enrolling a user ID creates a file space for that user ID, even if only with a size of 0 blocks. Because these user IDs are not relevant for reporting the usage of the storage groups, use `min=1` to filter these empty file spaces out of the reported data.
5. When activity in SFS is moderate, 15-minute intervals are usually sufficient to show at least some data in the dashboard. If the dashboard shows longer periods, hourly intervals might work.

Data Pump INFLUXDB plug-in

The Data Pump INFLUXDB plug-in formats data according to the InfluxDB Line Write Protocol and sends the data to an InfluxDB instance by using TCP/IP.

Parameters

A configuration section that defines an instance of the INFLUXDB plug-in uses the following parameters.

TYPE=INFLUXDB
The TYPE parameter specifies the plug-in type. In this case, the plug-in type is INFLUXDB.

The TYPE=INFLUXDB parameter is required.

BUNDLES=*files*
The BUNDLES parameter specifies the names of one or more MDXEXTRA files that specify extra sets of metrics to extract.

DB=*name*
The DB parameter specifies the name of the database as created in InfluxDB. The default value is ZVM.

When authentication is enabled in InfluxDB, the user that is specified by the USERID parameter must be granted write access to the database.

The DB parameter is required if the section does not specify the FILE parameter.

DICTIONARIES=*files*
The DICTIONARIES parameter specifies alternative data dictionaries and is typically used only for diagnosis and testing. See usage note "7" on page 271.

EXCLUDE=*record_types*
The EXCLUDE parameter specifies record types to exclude from processing. Record types that are specified are excluded from processing. Specify the record types. For example, `exclude=sytprp useact`.

If you use the EXCLUDE parameter, then do not use the INCLUDE parameter.

See usage note "3" on page 271.

INCLUDE=*record_types*
The INCLUDE parameter processes only the specified record types. Record types that are not specified are excluded from processing. Specify the record types. For example, `include=sytprp useact`.

If you use the INCLUDE parameter, then do not use the EXCLUDE parameter.

See usage note "3" on page 271.

FILE=*file*
The FILE parameter specifies an output file. If the FILE parameter is specified, then data is not sent to the InfluxDB service but is written to a file. The data is formatted according to the InfluxDB Line Write Protocol. The FILE option is intended for diagnostic purposes.

REPLACE=*boolean*
The REPLACE parameter specifies whether Data Pump overwrites an existing file. The REPLACE parameter affects processing only when the FILE parameter is specified. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is FALSE.

TERSE=*boolean*
The TERSE parameter specifies whether Data Pump compresses the output file. If you specify a "true" value for the TERSE parameter, then you must also specify the FILE parameter. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is FALSE.

MONTIME=*boolean*
The MONTIME parameter specifies whether the timestamp from the monitor record is used instead of the database time. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is TRUE. See usage note “6” on page 271.

PASSWORD=*password*
The PASSWORD parameter specifies credentials for the user ID that is specified by the USERID parameter. The user must be authorized to write record types (InfluxDB "measurements") in the InfluxDB database. An alternative method to specify the user ID is described in usage note “2” on page 271.

RELEASE=*zvm_release*
The RELEASE parameter specifies the release of the z/VM system where Data Pump runs. The RELEASE parameter is required in the rare case that no monitor data preview is possible. Use a format like RELEASE=ZVM730. See usage note “7” on page 271.

RETENTION=*policy*
The RETENTION parameter specifies an InfluxDB retention policy. See usage note “5” on page 271.

STARTUP=*seconds*
The STARTUP parameter specifies the time (in seconds) that is allowed for Data Pump to connect to the InfluxDB service.

The default value is 0, which means that the plug-in does not wait for the remote service to start. When STARTUP=0, the service must be available when Data Pump starts.

TAGS=*tags*
The TAGS parameter lists optional tags for the rare case that no monitor data preview is possible. The default tags are retrieved from monitor data and include the z/VM release level, the CPC serial number, and the CPC model number. See usage note “7” on page 271.

TCPIP_OPTIONS=*options*
The TCPIP_OPTIONS parameter specifies extra options for the CMS Pipelines tcpclient stage, which is Data Pump uses to connect to the InfluxDB service. Common options are USER TCPIPxxx and UNSAFE.

See usage note “4” on page 271.

URL=*url*
The URL parameter specifies the URL of the InfluxDB service. The URL must specify the protocol, the hostname, and the port number.

The URL parameter is required. A well-formed URL is required, even if the FILE parameter is specified.

USERID=*userid*
The USERID parameter specifies a user that can authenticate with the InfluxDB database. An alternative method to specify the user ID is described in usage note “2” on page 271.

DEBUG=CONFIG
DEBUG=STARTUP
DEBUG=VERBOSE
The DEBUG parameter issues extra informational messages for the section.

- DEBUG=CONFIG displays configuration information about selected record types (InfluxDB "measurements").
- DEBUG=STARTUP issues diagnostic messages when the configuration is processed and when Data Pump connects to the InfluxDB service.
- DEBUG=VERBOSE issues detailed diagnostic messages.

You can specify the DEBUG parameter only one time in a section.

The INFLUXDB plug-in can display information about the following diagnostic tasks.

Table 46. Diagnostic startup tasks for the INFLUXDB plug-in

Task name	Description
Preview	The plug-in analyzes the first part of the monitor data to determine the z/VM release and data tags. When the plug-in appears unresponsive after this task is started, perhaps the z/VM monitor was not started. The z/VM monitor immediately presents the configuration records that are required for the preview when Data Pump connects to the *MONITOR service. No delay is expected.
Preflight	The plug-in verifies that InfluxDB is operational and that data can be written.
Ping	The plug-in verifies connectivity and authentication (when authentication is required).
Dbwrite	The plug-in verifies that data can be written to the specified database with stated credentials.
Sslcheck	The plug-in verifies that SSL support is installed and that TCP/IP options are correct.
Resolve	The plug-in resolves the hostname that is specified in the URL. If the plug-in appears unresponsive, the DNS configuration might be incorrect. The Resolve task times out after 30 seconds.
Connect	The plug-in establishes a TCP/IP connection to the specified service. Addresses and firewalls are checked.
Rest	The plug-in traces the REST transactions that are attempted during the Preflight task.

Example

The following example assumes that authentication is not required and uses the default database name. The URL parameter indicates an SSL connection to the database.

```
[prod]
  type = InfluxDB
  url = https://lnxrmh01.pok.ibm.com:8086/
```

The following example assumes that authentication is enabled in InfluxDB. The example specifies user credentials to use Basic Authentication.

```
[secure]
  type = InfluxDB
  url = https://lnxrmh01.pok.ibm.com:18086/
  db = zvm
  userid = monitor
  password = “SecretData”
```

The following example might be appropriate for a debugging exercise. To reduce the data volume, the IODDEV metrics are excluded.

```
[debug]
  type = InfluxDB
  url = https://lnxrmh01.pok.ibm.com:8086/
  exclude = ioddev
```

Usage Notes

1. The following conditions are required for a secure connection to an InfluxDB service:
   - The URL parameter uses HTTPS.
   - The z/VM system is configured for SSL.
   - The InfluxDB service is configured for secure connections.
2. You can encode the use credentials (user ID and password) in the URL directly. For example, `https://`*`userid`*`:`*`password`*`@lnxrmh01.pok.ibm.com:18086/`. If you encode the user credentials in the URL, you do not have to use the USERID and PASSWORD parameters. Use the user ID and password parameter when either value contains characters that are unsafe for a URL.

   Remember: Parameter values are converted from EBCDIC to ASCII (code pages 1047 and 850). Special characters can get corrupted.
3. The include and exclude criteria are used to filter the record types from the monitor data. The most common use case is when you want to divert some of the record types to a different InfluxDB database. The requirement for different retention periods often motivates diversion to different databases.
4. When you need to specify a different TCP/IP stack to use, you must also set appropriate parameters in the `TCPIP DATA` file. The REXX sockets API is used to resolve the hostname in the URL. For more information, see the following topics:
   - Configuring the TCPIP DATA File in *z/VM: TCP/IP Planning and Customization*
   - REXX Sockets Application Program Interface in *z/VM: REXX/VM Reference*

   Documentation of the CMS Pipelines tcpclient program is available. See z/VM: CMS Pipelines User's Guide and Reference.
5. The retention policy specifies how long data is kept in InfluxDB. When the database is created, normally a retention policy AUTOGEN is defined with value 0 (no expiration) and is set as the default.

   Use the **`ALTER RETENTION POLICY`** command in InfluxDB to set the retention period such that the database does not fill all disk space. As a general guideline, a month of data from a single z/VM system could fit in less than 1 GB. Because disk space requirements depend on system configuration and workload, observe the usage over several days. Use your observations to set a practical expiration period and add disk space if required.
6. When `MONTIME=TRUE` (the default), Data Pump includes the timestamp from the monitor record in the data stream for InfluxDB. Usually, the default setting of MONTIME provides complete and correct timestamps in dashboards.

   When MONTIME=FALSE, Data Pump omits the timestamp. The InfluxDB service adds a timestamp when InfluxDB receives the data.
7. When the RELEASE and TAGS parameters are not configured, the default values can be retrieved from the monitor data (MTRSYS and MTRSSI). If Data Pump collects only SFS data, you can specify the RELEASE and TAGS values in the Data Pump configuration file.

   When the parameters are not retrieved from monitor data, you can usually set the value of the RELEASE parameter to the z/VM release where Data Pump runs. Use a format like `RELEASE=ZVM730`. The release also defines the correct data dictionary and overrides the default value, which is retrieved from the running system.

The default TAGS parameter contains the VM release level, the CPC serial number, and the CPC model number. You can use the identifiers to select and aggregate data from different systems in various ways.

8. During startup, write access to the InfluxDB service is verified by writing a Data Pump record type (InfluxDB "measurement") in the specified database. Because write access doesn't necessarily allow the user to delete the data, the measurement is retained and expired according to the retention policy. A convenient way to determine exactly when Data Pump was started or restarted is to check the initial write action.
9. Particular tasks of the plug-in are marked by a pair of 3030 and 3031 messages when the DEBUG=STARTUP option is set. The startup task message might be useful when you diagnose a problem in the configuration that does not yield an error message. For example, when the Resolve task appears to stall, you might expect a problem with the DNS configuration. See the startup tasks in Table 46 on page 270.
10. When you investigate a problem with data for a specific record type, you might want to save the data to a file for review if the problem reappears. You can add a section to the configuration file to include only that record type and write the data to a file. You can save the data to a file by creating another instance of the INFLUXDB plug-in (another configuration section) that uses the FILE parameter.

    An alternative method to save data to a file is to use the DATADUMP plug-in. For more information, see “Data Pump DATADUMP plug-in” on page 278.

Data Pump SPLUNK plug-in

The Data Pump SPLUNK plug-in uses the Splunk Enterprise HTTP Event Collector (HEC) to send data to a Splunk service.

Parameters

A configuration section that defines an instance of the SPLUNK plug-in uses the following parameters.

TYPE=SPLUNK
The TYPE parameter specifies the plug-in type. In this case, the plug-in type is SPLUNK.

The TYPE=SPLUNK parameter is required.

BUNDLES=*files*
The BUNDLES parameter specifies the names of one or more MDXEXTRA files that specify extra sets of metrics to extract.

EXCLUDE=*record_types*
The EXCLUDE parameter specifies record types to exclude from processing. Record types that are specified are excluded from processing. Specify the record types. For example, `exclude=sytprp useact`.

If you use the EXCLUDE parameter, then do not use the INCLUDE parameter.

See usage note "3" on page 275.

INCLUDE=*record_types*
The INCLUDE parameter processes only the specified record types. Record types that are not specified are excluded from processing. Specify the record types. For example, `include=sytprp useact`.

If you use the INCLUDE parameter, then do not use the EXCLUDE parameter.

See usage note "3" on page 275.

FILE=*file*
The FILE parameter specifies an output file. If the FILE parameter is specified, then data is not sent to the Splunk service but is written to a file. The data is formatted according to the Splunk Enterprise HTTP Event Collector (HEC) specification. The FILE option is intended for diagnostic purposes.

REPLACE=*boolean*
The REPLACE parameter specifies whether Data Pump overwrites an existing file. The REPLACE parameter affects processing only when the FILE parameter is specified. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is FALSE.

PASSWORD=*password*
The PASSWORD parameter specifies credentials for the user that is specified by the USERID parameter.

RELEASE=*zvm_release*
The RELEASE parameter specifies the release of the z/VM system where Data Pump runs. The RELEASE parameter is required in the rare case that no monitor data preview is possible. See usage note "5" on page 275.

SOURCETYPE=*name*
The SOURCETYPE parameter specifies a Splunk source type. The source type must match the specification in the Splunk HTTP Event Collector (HEC). The default value is ZVM.

See usage note "2" on page 275.

STARTUP=*seconds*
The STARTUP parameter specifies the time (in seconds) that is allowed for Data Pump to connect to the Splunk service.

The default value is 0, which means that the plug-in does not wait for the remote service to start. When STARTUP=0, the service must be available when Data Pump starts.

TAGS=*tags*
The TAGS parameter lists optional tags for the rare case that no monitor data preview is possible. The default tags are retrieved from monitor data and include the z/VM release level, the CPC serial number, and the CPC model number. See usage note "5" on page 275.

TCPIP_OPTIONS=*options*
The TCPIP_OPTIONS parameter specifies extra options for the tcpclient stage that connects to the Splunk service. Common options are USER TCPIPxxx and UNSAFE.

See usage note "6" on page 276.

TOKEN=*SPLUNK_TOKEN*
The TOKEN parameter specifies a Splunk-generated authentication token that is required to write to Splunk.

See usage note "2" on page 275.

URL=*url*
The URL parameter specifies the URL of the Splunk service. The URL must specify the protocol, the hostname, and the port number.

The URL parameter is required. A well-formed URL is required, even if the FILE parameter is specified.

USERID=*userid*
The USERID parameter specifies a user that can authenticate with the Splunk service.

DEBUG=STARTUP
DEBUG=VERBOSE
The DEBUG parameter issues extra informational messages for the section.

- DEBUG=STARTUP issues diagnostic messages when the configuration is processed and when Data Pump connects to the Splunk service.
- DEBUG=VERBOSE issues detailed diagnostic messages.

You can specify the DEBUG parameter only one time in a section.

The SPLUNK plug-in can display information about the following diagnostic tasks.

Table 47. Diagnostic startup tasks for the SPLUNK plug-in

Task name	Description
Preview	The plug-in analyzes the first part of the monitor data to determine the z/VM release and data tags. When the plug-in appears unresponsive after this task is started, perhaps the z/VM monitor was not started. The z/VM monitor immediately presents the configuration records that are required for the preview when Data Pump connects to the *MONITOR service. No delay is expected.
Splunk_hec	The plug-in verifies that data can be written to Splunk.
Tcp/ip	The plug-in verifies connectivity and TCP/IP options.
Hec_stats	The plug-in verifies that the Splunk HEC service is available. Credentials are checked.
Dbwrite	The plug-in verifies that data can be written to the HEC service for the specified source type.

Table 47. Diagnostic startup tasks for the SPLUNK plug-in (continued)	
Task name	**Description**
Sslcheck	The plug-in verifies that SSL support is installed and that TCP/IP options are correct.
Resolve	The plug-in resolves the hostname that is specified in the URL. If the plug-in appears unresponsive, the DNS configuration might be incorrect. The Resolve task times out after 30 seconds.
Connect	The plug-in establishes a TCP/IP connection to the specified service. Addresses and firewalls are checked.
Rest	The plug-in traces the REST transactions that are attempted during the Splunk_hec task.

Example

The following example is a simple plug-in configuration for a Splunk service that runs at the typical port number. The STARTUP parameter can be turned off or on by removing or adding the comment character (;).

```
[output]
  type = Splunk
  url = https://lnxrmh01.pok.ibm.com:8088/
  sourcetype = "zvm_index"
; startup = 300 ; Number of seconds to wait for service
  token = Splunk e8c57903-b956-4dc8-a575-4aa5a6438dc6
```

Usage Notes

1. Splunk Enterprise supports the API through SSL connections. The url parameter starts with https to enable SSL. Unless a specific server certificate is defined in the Splunk instance, the service uses a default server certificate that is signed by the SplunkCommonCA root certificate. Your security policy probably requires that you acquire a proper server certificate instead of the default server certificate. If you do use the default server certificate, you must install the SplunkCommonCA root certificate in your z/VM SSL certificate database as trusted root certificate. You must also specify the UNSAFE option to disable hostname validation.
2. The Splunk administrator defines a Splunk source type as an HEC parameter. The Data Pump SOURCETYPE value must match that source type parameter.

   When the HEC is defined, Splunk generates a token. The value of the Data Pump TOKEN parameter must match the generated token. An example token has a format like `Splunk e8c57903-b956-4dc8-a575-4aa5a6438dc6`.
3. The include and exclude criteria are used to filter the record types to extract from the monitor data. The most common use case is when you want to divert some of the record types to a different Splunk instance. The requirement for different retention periods often motivates diversion to different databases.
4. Your Splunk administrator might appreciate (or require) a file with a representative example of the data that you plan to send to Splunk. Add the FILE parameter to the section and the data is saved to a file instead of sent to Splunk. Use the data in the file to test the Splunk setup.
5. When the RELEASE and TAGS parameters are not configured, the default values can be retrieved from the monitor data (MTRSYS and MTRSSI). If Data Pump collects only SFS data, you can specify the RELEASE and TAGS values in the Data Pump configuration file.

   When the parameters are not retrieved from monitor data, you can usually set the value of the RELEASE parameter to the z/VM release where Data Pump runs. Use a format like `RELEASE=ZVM730`.

The default TAGS parameter contains the VM release level, the CPC serial number, and the CPC model number. You can use the identifiers to select and aggregate data from different systems in various ways.

6. When you need to specify a different TCP/IP stack to use, you must also set appropriate parameters in the `TCPIP DATA` file. The REXX sockets API is used to resolve the hostname in the URL. For more information, see the following topics:
   - Configuring the TCPIP DATA File in *z/VM: TCP/IP Planning and Customization*
   - REXX Sockets Application Program Interface in *z/VM: REXX/VM Reference*

   Documentation of the CMS Pipelines tcpclient program is available. See z/VM: CMS Pipelines User's Guide and Reference.
7. Particular tasks of the plug-in are marked by a pair of 3030 and 3031 messages when the `DEBUG=STARTUP` option is set. The startup task message might be useful when you diagnose a problem in the configuration that does not yield an error message. For example, when the Resolve task appears to stall, you might expect a problem with the DNS configuration. See the startup tasks in Table 47 on page 274.
8. When you investigate a problem with data for a specific record type, you might want to save the data to a file for review if the problem reappears. You can add a section to the configuration file to include only that record type and write the data to a file. You can save the data to a file by creating another instance of the SPLUNK plug-in (another configuration section) that uses the FILE parameter.

   An alternative method to save data to a file is to use the DATADUMP plug-in. For more information, see "Data Pump DATADUMP plug-in" on page 278.

Data Pump STATISTICS plug-in

The Data Pump STATISTICS plug-in reports the volume of data and can help determine the effect of enabling or disabling certain domains or resources. The plug-in can also be used to diagnose other problems in the configuration.

Parameters

A configuration section that defines an instance of the STATISTICS plug-in uses the following parameters.

TYPE=STATISTICS
The TYPE parameter specifies the plug-in type. In this case, the plug-in type is STATISTICS.

The TYPE=STATISTICS parameter is required.

INTERVAL=*minutes*
The INTERVAL parameter specifies the delay (in minutes) after which a report is generated. The value defaults to 60.

TOP=*number*
The TOP parameter limits the report to the top *number* record types that have the greatest volume of data. The value must be a whole number. A value of 0 displays the time range but no record types. The value defaults to 5.

Note: Record types are identified by their domain and record numbers. For example, the plug-in output of 1,6 indicates the Device Configuration record, which has DSECT name MTRDEV.

Example

The following example reads a MONWRITE file and reports only the record type with the greatest data volume (top=1). The time range is for all records (not for only the top record type).

```
[data]
  type = monwrite
  file = ASTL1 MONWRITE B
[stat]
  type = statistics
  top = 1
```

The output shows that the record type with the greatest volume of data is 1,6, which is responsible for 30% of the data volume. Record type 1,6 indicates domain 1 record 6, which is the MRMTRDEV (Device Configuration) record type.

```
FCXMDX3000I z/VM Performance Data Pump - Service Level 1.0 - FC02385
[stat] Time range: 2021-11-08 20:38:31 2021-11-08 20:42:30
[stat] Total 38166 records, 10.5 MB
[stat] Top-1 by data volume
[stat]    3129920 1,6
[stat] Top-1 by record count
[stat]       9781 1,6
Ready; T=0.09/0.10 10:59:23
```

Data Pump DATADUMP plug-in

The Data Pump DATADUMP plug-in writes CP monitor records to a separate disk file, which can be used for further diagnosis.

Parameters

A configuration section that defines an instance of the DATADUMP plug-in uses the following parameters.

TYPE=DATADUMP
The TYPE parameter specifies the plug-in type. In this case, the plug-in type is DATADUMP.

The TYPE=DATADUMP parameter is required.

FILE=*file*
The FILE parameter specifies the output file where CP monitor records are saved. Specify the file name. The file type is optional and defaults to DATADUMP. The file mode is optional and defaults to A.

The FILE parameter is required.

REPLACE=*boolean*
The REPLACE parameter specifies whether Data Pump overwrites an existing file. The REPLACE parameter affects processing only when the FILE parameter is specified. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is FALSE.

MONVIEW=*boolean*
The MONVIEW parameter specifies whether Data Pump creates the output file in MONVIEW format. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is FALSE.

CONS=*boolean*
The CONS parameter specifies whether Data Pump writes the header of the CP monitor records. The following values are valid and casing is ignored:

- The following values indicate Boolean "true": TRUE, YES, ON, 1.
- The following values indicate Boolean "false": FALSE, NO, OFF, 0.

The default value is FALSE.

ONLY=*domain*,*record*
The ONLY parameter limits the record types that are processed. Only the specified record types are processed. Specify the monitor domain numbers and record numbers. For example, to process only MTRDEV records, specify ONLY=1,6.

Example

The following example reads a MONWRITE file and writes only the MRSYTEPM records to an output file in MONVIEW format. The MONWRITE input file is ASTLI MONWRITE B. The name of the output file is specified (TEST) but the output file type and file mode use the default values. The ONLY parameter selects MRSYTEPM records by using the domain,record notation (0,20).

```
[data]
  type = monwrite
  file = ASTL1 MONWRITE B
[dump]
  type = datadump
```

```
file = test
replace = true
monview = true
only = 0,20
```

Usage Notes

1. Performance Toolkit cannot process the output file that is created by the DATADUMP plug-in or any other Data Pump plug-in.
2. The output records can be filtered to use in other data analytics programs or to quickly verify that particular records are present in the data.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

This book primarily documents information that is NOT intended to be used as Programming Interfaces of z/VM.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

PI

<...Programming Interface information...>

PI end

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See z/VM Performance Data Pump.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See Data Pump Messages.

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_3.1.0/pdf/ickug00_v3r1.pdf), GC35-0033

Related Products

XL C®++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Additional Documents

The *z/VM Performance Report* is available at IBM: z/VM Performance Report (https://www.ibm.com/vm/perf/reports/).

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Product Number: 5741-A09

Printed in USA

SC24-6301-74