An introduction to tuning VSAM file performance under CICS TS in z/VSE

Mike Poil CICS L3 Service
poilmike@uk.ibm.com

http://www.ibm.com/zVSE
http://twitter.com/IBMzVSE

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.
ASG-TMON is the registered trademark of the Allen Systems Group, Inc.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the 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.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the "Authorized Use Table for IBM Machines" provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html ("AUT").

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Abstract

The aim of this presentation is to explain some of the theory behind how VSE/VSAM works under CICS TS for VSE/ESA, to show how local file performance can be tuned using the IBM-provided free tools, and to identify limitations of these tools. *It is not designed to cover everything that you ever need to know about the subject!* Some Vendor products are also mentioned. The products will be referred to as "CICS" and "VSAM" from now on.

Remote files accessed by Function Shipping requests via MRO or ISC will *always* be a significant overhead compared to local files. I have seen a cpu *delta* of up to 100 times the cost of a local request. Using CICS Shared Data Tables may help with some KSDS - see the CICS TS for VSE/ESA Shared Data Tables Guide.

Similarly, VSAM Redirector file access is subject to network, remote database, other delays and any inherent design constraints imposed by z/VSE (especially before z/VSE 4.3), and will not be discussed.

It will not address the issue of tuning VSAM usage within the applications themselves. Developers need to be aware of the implications of the use of the various EXEC CICS requests. In many installations, rewriting applications is not an option anyway.

It assumes some familiarity with EXEC CICS VSAM requests, VSAM dataset concepts, IDCAMS DELETE/DEFINE parameters and CICS CSD resource definitions.

Agenda

Introduction.

Definitions that affect VSAM performance:

- CSD FILE resource buffering type selection.
- CSD LSRPOOL resource.
- IDCAMS.

CICS Statistics data.

Tuning LSR and case studies.

Rexx DFH0STAT analysis tool.

Tuning NSR files.

Contention and resource definition limit issues.

Tuning file contention.

Rexx DFH0STAT analysis tool File profiling.

Agenda

Defining a local KSDS as a Shared Data Table.

Vendor tuning products.

Additional reference material is provided at the end, and will not be discussed during the presentation itself.

Introduction

VSAM tuning is important but needs to be reviewed in the context of the whole environment.

VSAM performance impacts CICS, and a poorly tuned or overloaded CICS will impact VSAM. z/VSE, z/VM and others products' performance can impact CICS, and vice versa.

Tuning a 3-stage iterative process:

1. Collect an appropriate amount of performance data.

2. Analyze and look for performance problems that can be tuned.

3. Either tune (make changes) and start again at (1), or stop.

Tuning should not be based on one day's performance data.

A week is a good start. You also need to cover known regular and seasonal workload peaks. Don't assume that you only need to do it once for the lifetime of your system!

Remember the 80/20 rule and tune what will give you the most savings.

Stop if you have achieved your tuning objectives.

If the analysis shows that tuning still needs to be done, but the last changes did not achieve the expected improvement, you have either done all you can or you need to look for reasons elsewhere. If it is worse, you need to undo the changes!

Introduction

I am looking at how to tune VSAM to handle the workload that it is given to do by:

1. Reducing wait (elapsed) time and cpu time from unnecessary EXCPs - an EXCP is a z/VSE asynchronous I/O request.

2. Reducing wait time and cpu time due to VSAM contention and resource definition limits imposed by CICS definitions.

Tuning at this level is done by adjusting values in the:

- CICS CSD FILE and LSRPOOL resource definitions.
- IDCAMS definitions.

Be careful, changing a parameter without understanding how it works may impact performance in a positive way in one area but in a negative way in another.

Tuning at this level will help performance, but exactly how much it helps cannot be quantified from the data that we will look at, for this you need to look in detail at task response time and its components.

Tuning the workload given to VSAM may be required, which will require application changes.

Introduction

Important disclaimers:

All performance *comparisons* were done in a non-dedicated machine environment with pure random I/O driver transactions and do not guarantee what you will see in a production environment. Cpu speed, and hence usage, is a both a function of the cpu being used **and** the effectiveness of the caching of the workload. I/O times depend on the dasd subsystem model, it's configuration and features (e.g. PPRC and PAV), and the load placed on it. (Other VSAM performance data provided is similar to what I have seen in production systems.)

VSAM performance can be dependent on the release and the fixes currently applied to it.

CSD FILE Resource Definition Buffering Type

FILE LSRPOOLID=	Buffering	Notes
NONE (LSRPOOL 0 is shown in CICS statistics output)	NSR	NSR can be good for SHR(4) and output files (or those with a very high write/read ratio) e.g. an ESDS log file. VSAM resources are used **exclusively** by the file. STRINGS defines how many concurrent requests CICS *and* VSAM can handle. Use 1 for ESDS output files. DATABUFFERS and INDEXBUFFERS say how many buffers are allocated. Use the CICS defaults for SHR(4) and ESDS log files. Uses (mostly) 31-bit Partition Getvis storage, see the reference material at the end for more details.
1 to 15	LSR	LSR is designed to provide fast READ caching, and is normally more efficient than NSR in terms of cpu time and EXCPs. STRINGS limits how many concurrent requests *CICS* will handle without FILE string waits, even if VSAM can handle more due to the LSRPOOL STRINGs value. DATABUFFERS and INDEXBUFFERS are ignored, the LSRPOOL-defined buffers are **shared** by all files in it.

CSD LSRPOOL Resource Definition

Parameter	Notes
STRINGS	Defines how many file requests VSAM can handle concurrently before LSRPOOL string waits occur.
DATA and INDEX BUFFERS (0.5K, 1K, 2K, 4K and then 4K multiples to 32K)	Allocates a number of buffers by size. If the DEFINE CISZ does not match, the next larger allocated buffer size is used. DFHSTUP reports on the LSRPOOL buffer sizes used by each file, not the file's CISZ. Allocating a small number can cause errors when updating a file that has an AIX, e.g. AEIU abend with VSAM error code X'90' RC X'08'. Allocating more than 500 to 1,000 buffers of a given size *may* start to get expensive in terms of the cpu time used for buffer scanning. Allocate some 32K Data and Index buffers to avoid 4228I OPEN ERROR X'DC'(220) when the IDCAMS DEFINE CISZ is bigger that the largest buffer size that is allocated. Uses (mostly) 31-bit Partition Getvis storage, see the reference material at the end for more details about Getvis.

IDCAMS Definition

Parameter	Affects	Notes
CISZ Data	The number of EXCPs to process the data. Contention. Dasd utilisation.	Increasing the CISZ can reduce the number of EXCPs for browse requests, but may increase VSAM exclusive control waits. Use a special CISZ for some files to allow targeted tuning. Choose a Data CISZ for at least 10 records to allow CI FSPC to be set reasonably? For an AIX, use a CISZ that reduces spanning of the records. Less than 4K gives poor dasd utilisation.
CISZ Index	See the reference material.	
COMPRESSED	Cpu time. The number of EXCPs to process the data. Dasd utilisation.	More cpu is required for decompression and compression. More records per CI through compression may reduce the number of EXCPs. Reduces dasd utilisation.

IDCAMS Definition

Parameter	Affects	Notes
FSFC(CI%,CA%)	KSDS and AIX CI and CA split activity. The number of EXCPs. Cpu time. A split serializes *all* other access to the file until it is complete.	FSPC is reserved when the file is loaded or extended sequentially. May reduce EXCPs when records are added *before* End Of File (EOF) - LISTCAT counts them as inserts. Records added *after* EOF do not benefit, e.g. a log file - LISTCAT counts them in the total number of records in the file. A CI split requires about 7 EXCPs but a CA split can require 150+ EXCPs. LISTCAT will show the correct CI and CA split counts only *after* the file has been closed in CICS. Vendor products can show counts for splits due to CICS activity.

IDCAMS Definition

Parameter	Affects	Notes
SHR(4)	Number of EXCPs. Cpu time.	Normally produces the worst possible performance and does not respond to normal tuning techniques for reading data. In most cases VSAM will use an EXCP to read a CI without attempting look aside. A BROWSE operation is less affected when READNEXT accesses successive records in the same CI. VSAM uses more cpu per request than normal as it has to manage SHR(4) locks using the LOCK/UNLOCK SVC etc. If SHR(4) is used only to enable a PATH/AIX to be used, investigate the use of the DSNSHARING option in the PATH and the base FILE resource definition entries that allow SHR(2) to be used.

CICS Statistics Data

I use this for overall CICS tuning, but it is *not* perfect.

Statistics data is always collected and most counters are "reset" at times.

SIT STATRCD=OFF avoids the regular "interval" resets, which are every 3 hours by default.

Then consider the midnight reset, the time of which can be changed by a PLTPI program.

See the CICS Performance Guide chapter 5 for full details including what gets reset to what and when. FILE statistic counters are subject to timed resets to zero, but LSRPOOL statistic counters are not subject to timed resets.

Statistics data is sent to DMF before a reset and DFHSTUP can format it.

Statistics are "lost" when a file is closed and when an LSRPOOL is deleted after the last file is closed, but USS data is sent to DMF. Use the DFHSTUP SUMMARY parameter to combine timed reset and USS data, although some detail is lost in the report.

DFH0STAT (the STAT transaction) uses internal information from within CICS. For tuning VSAM, run it before you close files for batch or "midnight" to minimise data loss.

I will only be showing DFH0STAT output because the output from DFHSTUP is similar.

Tuning LSR

The aim is to improve VSAM "look aside", which means increasing the number of READ hits in VSAM's buffer caches to reduce cpu and elapsed time for I/O processing and EXCPs. We measure the "look aside" as a percentage of read hits compared to all reads.

How many LSRPOOLs should I use? There is no one "correct" answer.

*You should aim for a good lookaside % that does not require more than about 500 to 1,000 buffers for any single buffer allocation. The number of buffers affects the amount of cpu time for buffer management because VSE/VSAM does **not** use a Buffer Hashing algorithm to avoid the buffer pool size cpu effect.*

Consider placing SHR(4) datasets and associations in a separate LSRPOOL because they don't respond well to tuning and can get in the way of other files in the same LSRPOOL.

Avoid SHR(4) if you can, e.g. use DSNSHARING for the FILE and PATH definitions if that is what resulted in SHR(4) being required in the first place.

Use CSD LSRPOOL resource definitions and define the numbers of Strings, Data and Index buffers in order to make LSRPOOL tuning easier to do.

Letting CICS dynamically build the whole LSRPOOL results in no Index buffers and adjusting the numbers of strings and buffers requires changes to the FILE definitions!

Tuning LSR

The LSRPOOL-level look aside "rule of thumb" target is 95% for Index I/O and, if possible, 80% for Data I/O. If you must use just Data buffers, aim for at least 80%.

Look aside is tuned by adding buffers for CI sizes that are not performing well. The number to be added could be anywhere between +10% and +100% or even more depending on how much you need to improve the lookaside. You need to experiment. Index look aside is easier to improve than Data.

You need about 5% more 31-bit Getvis storage than the sum of each (CI size * added buffers). Reference material at the end shows how to calculate available Getvis storage.

SHR(4) files will typically consume extra buffers without any noticeable improvement.

Heavily used large files can "hog" buffers, and adding more buffers may not produce a worthwhile improvement. Use statistics to identify them, and try moving them to a low usage LSRPOOL, or perhaps to a "Hog" LSRPOOL. Another option is to define them with different CISZ values compared to "normal" files so that you can isolate their activity within an LSRPOOL. You need to experiment. You could even try using LSRPOOLID=NONE for NSR, but profile the FILE performance before with an appropriate amount of data.

A simple test to demonstrate the possible effects of LSR buffering

A non-dedicated environment was used, with z/VSE 5.2 using one 900 MIP virtual z10 cpu under z/VM with MDC inactive and fast dasd (3K+ requests per second). It exaggerates the differences as there is no business logic. The relative values and the effect of EXCPs are of more interest than the numbers. Actual MIPs depends on cpu cache performance. **YMWV.**

The test is 4M randomised reads to a KSDS with 100 Data and 101 Index CIs. The file was browsed by another transaction before each test to prime buffers. The baseline minimised the effect of LSR buffer scanning in the VSAM code. Buffer counts are Data/Index

Test	Cpu seconds	Elapsed seconds	EXCPs	Lookaside
Baseline	12	12	0	100%
50/50 buffers	307.5	810	4,068,110	49% Data 74% Index
80/90 buffers	110.5	448	1,296,031	79% Data 94% Index
100/101 buffers	17.5	18	0	100%
500/500 buffers	24.5	27	0	100%
1000/1000 buffers	38.5	40	0	100%
4000/4000 buffers	95.5	98	0	100%
CICS Data Table	4	4	0	No VSAM access

A simple test to demonstrate the possible effects of LSR buffering

SIR SMF data from one test that shows how fast the dasd was even without 100% dasd cache hits (there is disconnect time from cache misses) - ¼ millisecond per I/O!

See the z/VSE publication "Hints and Tips from L2". SSCH is the instruction that starts I/O.

```
sir smf=32d
AR 0015 TIMING VALUES FOR 32D BASED ON    2053754 I/O INSTRUCTIONS
AR 0015   QUEUED      PENDING     CONNECT     DISCONN    DEV.BUSY      TOTAL
AR 0015  msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH
AR 0015      0.001       0.004       0.128       0.108       0.000       0.242
sir smf,vse,32d
AR 0015 TIMING VALUES FOR 32D BASED ON    2053677 I/O INSTRUCTIONS
AR 0015 MAXIMUM I/O QUEUE   8
AR 0015   QUEUED      PENDING     CONNECT     DISCONN    DEV.BUSY      TOTAL
AR 0015  msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH
AR 0015      0.001       0.000       0.354       0.000       0.000       0.355
```

Tuning LSR Case Study 1

I realise that these case studies are based on one day's data, but I only have 90 minutes to talk about the subject! However, it does give you an idea about how much time you would need to tune VSAM properly without some kind of automation.

```
LSR Pools

  Pool Number :   2      Time Created :   05:17:23.10465

     Maximum key length . . . . . :         255

     Total number of strings  . . . . :          65

     Peak concurrently active strings :          12

     Total requests waited for string :           0

     Peak requests waited for string. :           0
```

This is where you look for LSR contention issues.

The number of strings defined in the CSD LSRPOOL definition is a good value - peak active is not close to the total number so there were no string waits.

Tuning LSR Case Study 1

```
Buffer Totals

     Data Buffers . . . . . . . . . . :         288      Index Buffers. . . . . . . . . . :         320
       Successful look asides . . . . :  11,504,852        Successful look asides . . . . :  29,751,487
       Buffer reads . . . . . . . . . :   3,467,370        Buffer reads . . . . . . . . . :      58,223
       User initiated writes. . . . . :   3,210,888        User initiated writes. . . . . :     209,187
       Non-user initiated writes. . . :           0        Non-user initiated writes. . . :           0
```

This is where you start looking for potential EXCP savings.

Data and Index buffers are defined, which is Best Practice.

Look aside % = (Successful look asides*100)/(Successful look asides + Buffer reads)

*Data look aside (11,504,852 * 100) / (11,504,852 + 3,467,370) = 77%, which is very close to the suggested 80%.*

*Index look aside (29,751,487 * 100) / (29,751,487 + 58,223) = 99%, which is more than the suggested 95%.*

Ignore writes.

This LSRPOOL is working very well.

Tuning LSR Case Study 1

Data Buffer Statistics

Size	Buffers	Look Asides	Reads	User Writes	Writes
2048	160	94,344	426,164	56,597	0
4096	128	11,410,508	3,041,206	3,154,291	0

Index Buffer Statistics

Size	Buffers	Look Asides	Reads	User Writes	Writes	
512	32	0	0	0	0	delete since it is not used
2048	128	14,593,314	484	0	0	
4096	128	12,628,859	55,989	203,712	0	
8192	24	2,529,314	1,750	5,475	0	
16384	8	0	0	0	0	

This is where you look to see which buffer sizes to tune.

Consider tuning the number of 4K Data buffers because there are 3M EXCPs to avoid versus 400K EXCPs for the 2K buffers.

I would add some 32K Data and Index buffers to allow for rogue CISZ changes.

Tuning LSR Case Study 2

```
LSR Pools

 Pool Number :   3      Time Created :   00:05:15.09522

     Maximum key length . . . . . :         100

     Total number of strings  . . . . :          90

     Peak concurrently active strings :          90

     Total requests waited for string :      42,987

     Peak requests waited for string. :          21
```

What you see here is normally the result of the number of strings allocated being too small.

It is also possible that there is a problem that is stopping strings being released fast enough.

If you have a CSD LSRPOOL definition, consider increasing the number of strings to avoid string waits. If you have no LSRPOOL definition, you would need to increase STRINGS in FILE definitions that are allocated to LSRPOOLID 3!

Increasing the number of strings may increase buffer stealing and reduce look aside!

Note: The maximum number of strings that can be defined anywhere is 255.

Tuning LSR Case Study 2

```
Buffer Totals

    Data Buffers . . . . . :          435        Index Buffers. . . . . :          0
      Successful look asides . . . . :  44,689,132   Successful look asides . . . . :  0
      Buffer reads . . . . . :          56,267,410   Buffer reads . . . . . :          0
      User initiated writes. . . . . :   3,756,970   User initiated writes. . . . . :  0
      Non-user initiated writes. . . :           0   Non-user initiated writes. . . :  0
```

Index buffers are not defined! (Because CICS dynamically defined the LSR pool.)

Look aside % = (Successful look asides*100)/(Successful look asides + Buffer reads)

Combined Data and Index = 44%, which is very bad.

Define an LSRPOOL in the CSD and copy the 4 buffer allocations on the next slide as both Data *and* Index, test, then delete unused buffer sizes.

Tuning LSR Case Study 2

Data and Index Buffer Statistics

Size	Buffers	Look Asides	Reads	User Writes	Writes	
2048	110	451,493	7,401,268	1,453,394	0	6% look aside
4096	200	27,530,104	45,221,982	552,349	0	38% look aside
8192	110	16,701,475	3,642,161	1,750,503	0	82% look aside
16384	15	6,060	1,999	724	0	75% look aside

The 4K buffers are the main problem as they use 45M EXCPs.

The 2K buffers are a problem, but not as much as the 4K.

The 8K and 16K buffers are performing reasonably well.

The look aside % values will change when Index buffers are allocated, only then should you tune the buffer sizes.

Add some 32K Data and Index buffers to allow for rogue CISZ changes or replace the 16K.

Rexx DFH0STAT Analysis Tool

A Rexx program that analyses DFH0STAT output and runs on z/VM or z/VSE, although it is easier to run and is more flexible on z/VM. It is not perfect and it has to work with data that can have imperfections, but it saves a LOT of time.

Its analysis covers about 90 cases of actual or potential configuration and performance issues. It will also profile the workload.

The report can be cumulative and can contain many days of DFH0STAT data from different CICS systems. It will distinguish between them based on the VTAM "applid".

It can produce a number of cumulative CSV files for analysis using your favourite Spreadsheet or Relational Database PC application.

There are many comments that describe how it should be used and you can tailor all thresholds etc. that result in messages.

Some thresholds exclude activity that is at a level where spending time on tuning is probably not worth your while even if their performance is not optimum.

I have included its analysis of Case Study 2's data for LSR pools 1 to 4, not just 3.

Rexx DFH0STAT Analysis Tool

"Best Practice" messages tell you that something has been configured in a way that is not optimum or may cause a problem. You should make changes unless you know that what is defined *is* correct for your system, e.g. you have an LSRPOOL with only one file in it and you know that the Data and Index CI sizes are different.

```
Best Practice LSRPOOL  1 does not have index buffers allocated, LSR performance will be degraded
Best Practice LSRPOOL  2 does not have index buffers allocated, LSR performance will be degraded
Best Practice LSRPOOL  3 does not have index buffers allocated, LSR performance will be degraded
Best Practice LSRPOOL  4 data  CI size 01024 has no I/O activity, consider setting zero buffers
Best Practice LSRPOOL  4 does not have index buffers allocated, LSR performance will be degraded
```

Rexx DFH0STAT Analysis Tool

"Limit" messages tell you that you have hit a configuration limit and that is likely to result in avoidable waits. The output below is based on case study 2.

Increasing the reported "limit" to the suggested value may be the solution. However, this may not always work because it may have been reached due to other problems slowing CICS or VSAM down.

```
Limit         LSRPOOL  1 peak string usage 100%, consider increasing number of strings    80 (max. strings
255)

Limit         LSRPOOL  3 peak concurrent string waits         21, total string waits      42987, consider
increasing LSRPOOL strings to 116

Limit         LSRPOOL  3 peak string usage 100%, consider increasing number of strings    90 (max. strings
255)
```

LSRPOOL 1 has reached the string limit but has not caused string waits this time, consider increasing LSRPOOL 1 STRINGs by a small number.

The LSRPOOL 3 STRINGS value has resulted in a very large number of string waits and needs action to be taken.

Note: LSRPOOLs can be excluded from analysis if required, and remember to ignore messages that result from deliberate choices by you.

Rexx DFH0STAT Analysis Tool

"Threshold" messages tell you that something is probably not working in the optimum way, or is warning of something that is close to becoming a "limit".

The output is based on case study 2 and shows that LSRPOOL buffering is not optimum.

These LSRPOOLs need to have Index buffers allocated first, then check again.

It shows look aside ratios for the LSRPOOL and buffers, and the potential EXCP savings help you to decide the relative importance of tuning. LSRPOOL 3 and its 4K buffers are the obviously the worst performers at this point in time. Any text in blue is a note added by me.

```
Threshold     LSRPOOL  1 data  hit ratio  70% is < recommended 80%, Read EXCPs   52703156

Threshold     LSRPOOL  1 data  CI size 01024 hit ratio  60% EXCP saving at 80%     310801, consider
increasing number of data  buffers    35

Threshold     LSRPOOL  1 data  CI size 02048 hit ratio  67% EXCP saving at 80%    5418263, consider
increasing number of data  buffers   250

Threshold     LSRPOOL  1 data  CI size 04096 hit ratio  74% EXCP saving at 80%    5212766, consider
increasing number of data  buffers   300

Threshold     LSRPOOL  1 data  CI size 08192 hit ratio  65% EXCP saving at 80%    6701366, consider
increasing number of data  buffers    55

Threshold     LSRPOOL  1 data  EXCP saving at 80%   17643196  (about 33% of the LSRPOOL total EXCPs)
```

Rexx DFH0STAT Analysis Tool

```
Threshold     LSRPOOL  2 data  hit ratio  53% is < recommended 80%, Read EXCPs   14397267

Threshold     LSRPOOL  2 data  CI size 01024 hit ratio   2% EXCP saving at 80%    1667732, consider
increasing number of data  buffers    70

Threshold     LSRPOOL  2 data  CI size 02048 hit ratio  69% EXCP saving at 80%     258956, consider
increasing number of data  buffers   130

Threshold     LSRPOOL  2 data  CI size 04096 hit ratio  56% EXCP saving at 80%    5001964, consider
increasing number of data  buffers   210

Threshold     LSRPOOL  2 data  CI size 08192 hit ratio  58% EXCP saving at 80%    1288335, consider
increasing number of data  buffers    55

Threshold     LSRPOOL  2 data  EXCP saving at 80%    8216987  (about 14% of the LSRPOOL total EXCPs)


Threshold     LSRPOOL  3 data  hit ratio  44% is < recommended 80%, Read EXCPs   56267410

Threshold     LSRPOOL  3 data  CI size 02048 hit ratio   6% EXCP saving at 80%    5830716, consider
increasing number of data  buffers   110

Threshold     LSRPOOL  3 data  CI size 04096 hit ratio  38% EXCP saving at 80%   30671565, consider
increasing number of data  buffers   200

(The 16K buffer size was ignored due to it's usage being below an analysis trigger even though the look aside is 75%.)

Threshold     LSRPOOL  3 data  EXCP saving at 80%   36502281        (about 65% of the LSRPOOL total EXCPs)
```

Rexx DFH0STAT Analysis Tool

The "PROF" option can be helpful. LSRPOOLs 1 and 3 are obviously the busiest in terms of EXCPs, but look at the ratio of EXCPs/LSR activity for LSRPOOL 3 compared to 1!

```
Profile       EXCPS for LSR  123391045 is  98.77% of total EXCPS reported
Profile       EXCPS for NSR    1535857 is   1.23% of total EXCPS reported
Profile       LSRPOOL    Buffer Hit        per transaction is    46.96
Profile       LSRPOOL    Buffer I/O        per transaction is    35.11
Profile       LSRPOOL    Buffer Read       per transaction is    31.52
Profile       LSRPOOL    Buffer Write      per transaction is     3.59
Profile       LSRPOOL  1 is responsible for   42.71% of all LSR Read EXCPs and    57.06% of all LSR activity
Profile       LSRPOOL  2 is responsible for   11.67% of all LSR Read EXCPs and    10.06% of all LSR activity
Profile       LSRPOOL  3 is responsible for   45.60% of all LSR Read EXCPs and    32.86% of all LSR activity
Profile       LSRPOOL  4 is responsible for    0.02% of all LSR Read EXCPs and     0.02% of all LSR activity
```

Getting to now the typical profile over time can be useful for spotting problems, and is always worth checking after a release change.

Rexx DFH0STAT Analysis Tool

This LSRPOOL has Data and Index buffers defined.

```
Threshold     LSRPOOL  1 data  CI size 01024 hit ratio   0% EXCP saving at 80%     270608, consider
increasing number of data  buffers    50 if the saving is worthwhile
Threshold     LSRPOOL  1 data  CI size 02048 hit ratio   0% EXCP saving at 80%     239028, consider
increasing number of data  buffers    50 if the saving is worthwhile
Threshold     LSRPOOL  1 data  CI size 08192 hit ratio  29% EXCP saving at 80%    7586403, consider
increasing number of data  buffers   100 if the saving is worthwhile
Threshold     LSRPOOL  1 data  CI size 16384 hit ratio  17% EXCP saving at 80%    2768599, consider
increasing number of data  buffers   100 if the saving is worthwhile
Threshold     LSRPOOL  1 data  EXCP saving at 80%   10864638

Threshold     LSRPOOL  1 index hit ratio  58% is < recommended 95%, Read EXCPs   19748769
Threshold     LSRPOOL  1 index CI size 01024 hit ratio  61% EXCP saving at 95%    2087616, consider
increasing number of index buffers    40 if the saving is worthwhile
Threshold     LSRPOOL  1 index CI size 02048 hit ratio  59% EXCP saving at 95%   13211002, consider
increasing number of index buffers   100 if the saving is worthwhile
Threshold     LSRPOOL  1 index CI size 04096 hit ratio  51% EXCP saving at 95%    1373873, consider
increasing number of index buffers   100 if the saving is worthwhile
Threshold     LSRPOOL  1 index CI size 08192 hit ratio  36% EXCP saving at 95%     716877, consider
increasing number of index buffers   100 if the saving is worthwhile
Threshold     LSRPOOL  1 index EXCP saving at 95%   17389368
```

The Data look aside is good because there is no threshold message. The 4K activity is not reported because there were 215M look asides and only 19M EXCPs. Increase the 8K and possibly the 16K buffers if you want to save some more EXCPs.

The Index performance is very bad. Try doubling the 2K and the 1K buffers.

Tuning NSR Files

Reported as LSRPOOL 0 FILEs.

NSR files will provide look aside, and adding buffers in the FILE definition may result in fewer EXCPs. The look aside seems to be more costly in cpu time than LSR.

This example shows no look aside as every read requires an EXCP to the Index and to the Data because the FILE is SHR(4) with only has 1 Index level. This is normal SHR(4) behaviour and cannot be tuned.

Filename	Access Method	Type	LSR Pool	Str Waits Max	Str Waits Total	Read Requests	Get Update Requests	Browse Requests	Add Requests	Update Requests	Delete Requests	Data EXCPs	Index EXCPs
VSMSHR4	VSAM	KSDS	0	0	0	156517	0	0	0	0	0	156517	156517

The DFH0STAT analyzer shows:

```
Best Practice FILE VSMSHR4  is using NSR buffering, this is not normally recommended unless it is SHR(4) or
add-only ESDS
```

Tuning NSR Files

VFI0005 has poor Index buffering, which is shown in its Index/Data EXCP ratio (3975192/849812). Using LSR should help.

VFI0009 has a very high EXCPs/Request ratio (1875284/246443) due to NSR buffering and split activity. Using LSR may help to reduce EXCPs, but you would need to profile before and after and be prepared to change back.

Filename	Access Method	Type	LSR Pool	Str Waits Max	Str Waits Total	Read Requests	Get Update Requests	Browse Requests	Add Requests	Update Requests	Delete Requests	Data EXCPs	Index EXCPs
VFI0005	VSAM	KSDS	0	0	0	254,913	214,777	8,940,197	76,754	148,154	149	849,812	3,975,192
VFI0009	VSAM	KSDS	0	0	0	17,616	114,721	0	105,354	55	8,697	1,091,308	783,976

The DFH0STAT analyzer "PROF" option shows:

```
Profile  EXCPs    4825004 NSR        KSDS VFI0005 Index/Data EXCP ratio  4.68 EXCPs/Request  0.50
File requests     9634944 Read/Write ratio      41.81 Read    4.87% Browse   92.79% Update    1.54% Add
0.80% Delete     0.00%

Profile  EXCPs    1875284 NSR        KSDS VFI0009 Index/Data EXCP ratio  0.72 EXCPs/Request  7.61
File requests      246443 Read/Write ratio      1.16  Read   53.70% Browse    0.00% Update    0.02% Add
42.75% Delete    3.53%
```

A simple test to demonstrate the possible effects of NSR buffering

A repeat of the previous 4M random read test. I have included some LSR buffering (with only one file in the LSRPOOL) as a comparison. There are 2 levels in the Index.

You would never expect 0 Data EXCPs from NSR. I found that the number of Index EXCPs could be reduced by using 1,000 buffers, but the results varied a lot between runs and I have not included them.

If nothing else, it shows the relative cost of a large number of EXCPs in terms of cpu and elapsed time even at only ¼ millisecond per I/O.

The results will be exaggerated compared to normal operation. **YMWV**.

LSR Test	Cpu seconds	Elapsed seconds	EXCPs	Lookaside
50/50 buffers	307.5	810	4,068,110	49% Data 74% Index

NSR Test	Cpu seconds	Elapsed seconds	Data EXCPs	Index EXCPs
50/50 buffers	375	1735	2,0160,000	3,648,000
100/101 buffers	225.5	812	0	3,245,999

Contention and Resource Definition Limit Issues

A CICS task is normally placed in an FCIOWAIT while the I/O is running asynchronously in the I/O subsystem. This allows CICS to dispatch other tasks until such time as CICS notices that the I/O is complete and the task can be re-dispatched. The reference material contains a simplified description of the process.

But other wait states can occur, e.g.

- FCCIWAIT - access to the **FILE** is blocked until the (CA and) CI split has completed.
- FCXCWAIT - access to the **CI** is blocked until an update to the same CI has completed.

CICS does *not* report the number of FCCIWAIT and FCXCWAIT waits.

- FCSRSUSP - **LSRPOOL** access is blocked until an LSRPOOL string becomes available.
- FCPSWAIT - **FILE** access is blocked until a FILE string becomes available.

CICS reports on FCSRSUSP is the LSR pools report.

CICS reports on FCPSWAIT waits in the Files report.

See the CICS Problem Determination Guide chapter 6 for full details.

Contention and Resource Definition Limit Issues

CICS forms a queue of tasks waiting on the CI/FILE/LSRPOOL.

When the issue is resolved, *all* of the tasks in the queue are resumed to retry their request, and the highest priority task gets dispatched first.

As the maximum queue depth grows, wait times become less linear and there is more chance of a retry failure for a task.

For example, tasks A and B are resumed an FCXCWAIT is resolved. Task A's update becomes an FCIOWAIT. This allows B to be dispatched to retry its request before task A frees the CI. Task B goes into another FCXCWAIT having used cpu time in CICS *and* VSAM to achieve nothing! This is CICS W.A.D.

General or task performance problems that slow things down may result in these waits occurring more frequently than is normal.

FCXCWAITs may be reduced by using a smaller Data CISZ in order to reduce the potential for concurrent updates to the same CI, but you need to know that they are occurring!

Tuning FILE Contention

The data that is available is not good for proper statistical analysis. It suggests that there could be a problem and that changing FILE definition values may help.

Filename	Access Method	LSR Type	Str Pool	Waits Max	Waits Total	Read Requests	Get Requests	Update Requests	Browse Requests	Add Requests	Update Requests	Delete Requests	Data EXCPs	Index EXCPs
MSTR001	VSAM	KSDS	1	79	>999	211234	213416	0	5445	213416	0	304412	59993	
MSTR002	VSAM	KSDS	1	155	>999	8	278	864650	27107	277	3	213887	62001	

The files had *more than* 999 FCPSWAIT string waits, probably due to the number of STRINGS in the FILE definition being too small. At one point in time, 79 and 155 tasks were queued in FCPSWAIT, which *is* a major concern. The DFH0STAT analyzer reports:

```
Limit          FILE MSTR001 peak concurrent string waits          79, total string waits         999+ consider
increasing the FILE definition STRINGS value

Limit          FILE MSTR001 peak string waits is high, please check the file and overall VSAM performance

Limit          FILE MSTR002 peak concurrent string waits        155, total string waits         999+ consider
increasing the FILE definition STRINGS value

Limit          FILE MSTR002 peak string waits is high, please check the file and overall VSAM performance

Threshold      MONITOR exception records were produced, 97289 SOS/DFHTEMP/VSAM resource limit waits occurred
(if CICS exception recording is active, this message shows the extent of the problem)
```

Rexx DFH0STAT Analysis Tool File Profiling

The raw data can be stored in a cumulative CSV file for DIY profiling of many days' data.

The tool profiles files other than ESDS that exceed the threshold TFILEPIO=250000 EXCPs. As seen previously, this may help to explain why performance is not as expected.

```
Profile  EXCPs   4587897 LSRPOOL  3 KSDS MAS0002 Index/Data EXCP ratio  0.92 EXCPs/Request 1.92
File requests    2388259 Read/Write ratio       9.99 Read   90.90% Browse    0.00% Update    9.10% Add
0.00% Delete    0.00%
```

FILE MAS0002 has a high EXCPs/Request ratio *even with a high random Read %.* High Index EXCP counts could be due to poor LSRPOOL 3 buffering that we saw earlier. Check the difference after tuning LSR.

```
Profile  EXCPs  26869764 LSRPOOL  3 KSDS TAB0001 Index/Data EXCP ratio  0.93 EXCPs/Request 1.14
File requests   23583307 Read/Write ratio    6105.50 Read   54.43% Browse   45.55% Update    0.00% Add
0.01% Delete    0.01%
```

FILE TAB0001 (27M EXCPs!) is similar, but has a lot of browse requests that could cause Data and Index EXCPs and may not be so easy to tune.

Rexx DFH0STAT Analysis Tool File Profiling

```
Profile  EXCPs  13110280 LSRPOOL  3 RRDS MASTRR1  Index/Data EXCP ratio   N/A EXCPs/Request  1.00
File requests   13110280 Read/Write ratio 9999999.99 Read  100.00% Browse    0.00% Update    0.00% Add
0.00% Delete    0.00%
```

FILE MASTRR1 has a 1:1 EXCPs/Request ratio even with a high Read %. This RRDS appears to have no look aside whatsoever. Is it too large to benefit from LSR, are there too few buffers in LSRPOOL 3, or what? It may be an LSR "hog"!

```
Profile  EXCPs  12013393 LSRPOOL  2 KSDS J01F002 Index/Data EXCP ratio  1.00 EXCPs/Request  2.00
File requests    6007728 Read/Write ratio 9999999.99 Read    0.00% Browse  100.00% Update    0.00% Add
0.00% Delete    0.00%
```

This file is always browsed and shows no obvious look aside with every request reading one Index and Data record. Browse uses the Sequence Set, which is the biggest part of the Index Component, so just a small increase in LSRPOOL 2 Index buffers may not help.

Rexx DFH0STAT Analysis Tool File Profiling

```
Profile       EXCPs     333823 LSRPOOL  1 KSDS ATM0001 Index/Data EXCP ratio  0.01 EXCPs/Request  0.08 File
requests    4308182 Read/Write ratio   59834.86 Read    0.00% Browse  100.00% Update    0.00% Add    0.00%
Delete    0.00%


Profile       EXCPs    4192010 LSRPOOL  4 KSDS SUBMIT1 Index/Data EXCP ratio  0.24 EXCPs/Request  0.05 File
requests   81588207 Read/Write ratio     186.38 Read    0.00% Browse   99.46% Update    0.00% Add    0.53%
Delete    0.00%
```

As a contrast, these files are working well even for the high percentage of browse activity.

Note: The number of EXCPs reported for a PATH in the FILE statistics does *not* include the EXCPs that are used internally by VSAM to access the AIX, the EXCPs are those that VSAM uses to access the Base Cluster mapped by the PATH - this is not a bug. EXCP counts for an AIX accessed by a PATH *are* included in the LSRPOOL statistics.

Defining a Local KSDS as a Shared Data Table

A "small" base KSDS (not a PATH) that is predominantly accessed by READ without UPDATE and BROWSE requests may benefit from being defined as a Shared Data Table even if it is not shared between CICS partitions.

The Shared Data Table records are cached in a Data Space, but it does require a significant amount of Partition Getvis storage to map every key and for internal indexes to the data.

A CICS-Maintained Table (CMT) works much like any KSDS, it must be mapped to an LSRPOOL and all changes to the Data Space are made to the source KSDS.

A User-Maintained Table (UMT) is slightly restricted in terms of the API etc. and no changes are made to the source KSDS.

There is a one-time cost for loading the data records when the file is opened. Subsequently, CICS uses less cpu time to access the data in the Data Table compared to CICS accessing it in an LSRPOOL. Any file change uses more cpu time with a CMT because CICS has to update its cache *and* call VSAM to write the changes to the KSDS using normal LSR access. Data Table cache misses will access the dataset using normal LSR processing.

EXEC CICS request counts in the FILE statistics will normally just be from the initial load of the Data Table.

For full details, see the CICS TS for VSE/ESA Shared Tables Guide.

Vendor Tuning Products

The information here has been supplied by those Vendors who responded to my request for details about how their product could help with tuning VSAM under CICS.

The presentation will not explain how to interpret the data values in the following slides.

Vendor Tuning Products

C\TREK from the C\TREK corporation in the USA is available from various business partners, and is a product that some customers use instead of, or in a complimentary fashion with other Vendor CICS tuning products to tune CICS systems and VSAM files.

It is primarily on online tool.

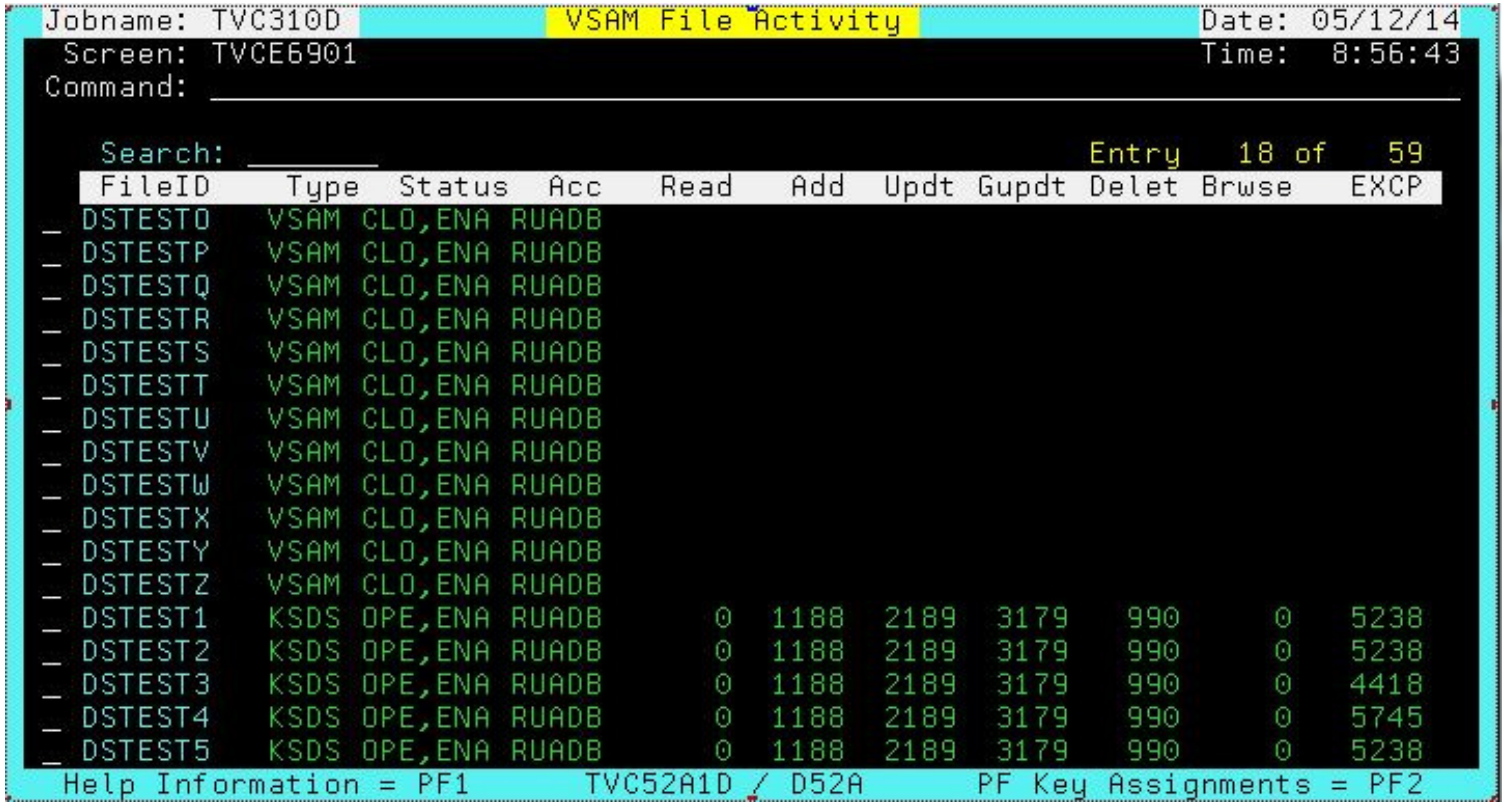
It is able to identify approximately 150 different performance issues within CICS as a whole and provide suggestions on how to fix them.

It has the ability to view CICS control blocks to look at various issues, and is able to help with certain types of CICS task transaction abends.

Control block displays provide information about LSRPOOLs and VSAM files that CICS is unable to.

Vendor Tuning Products

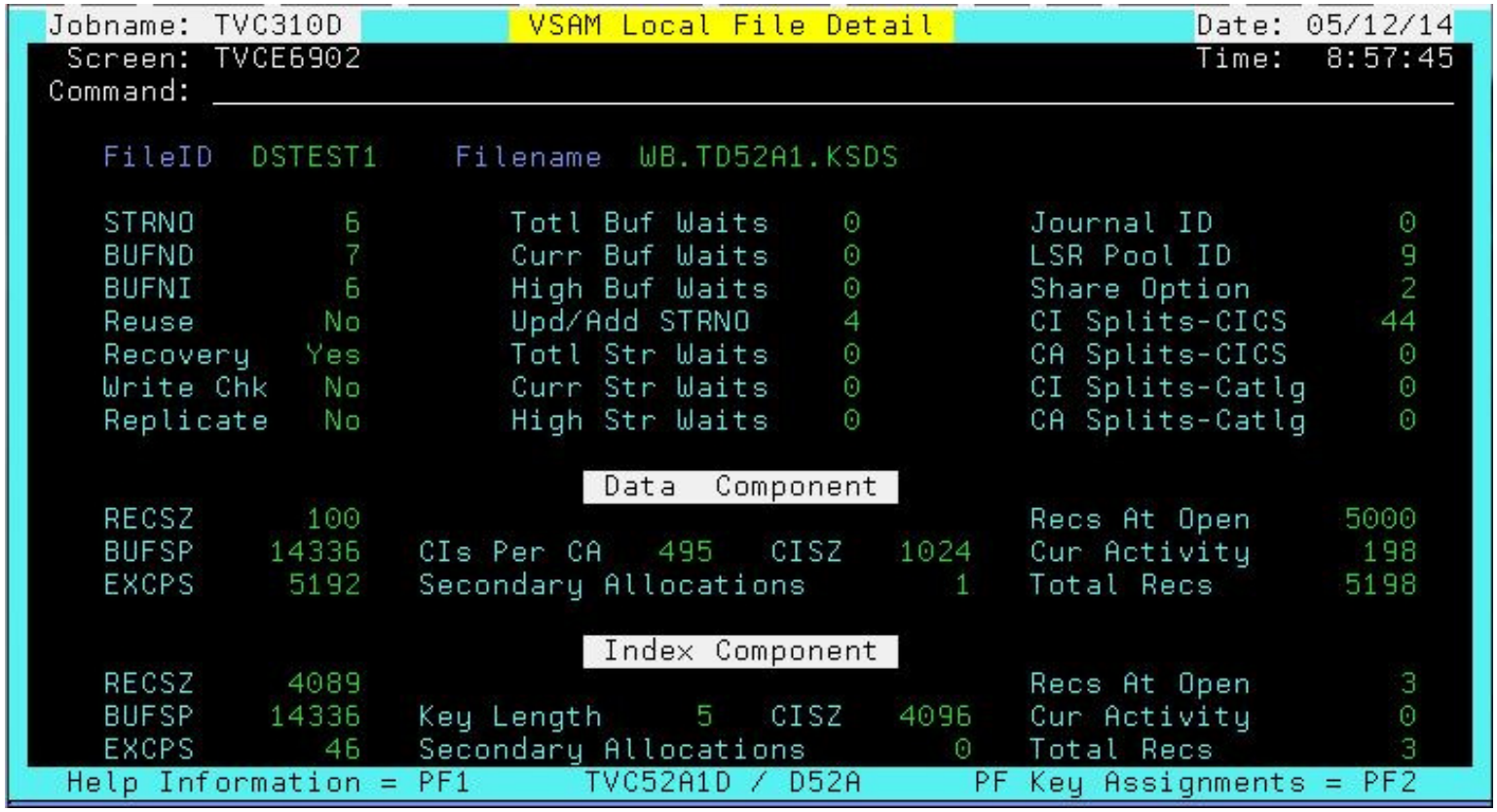
ASG-TMON has both online and batch reporting capabilities. The slides contain sample screen and report output.



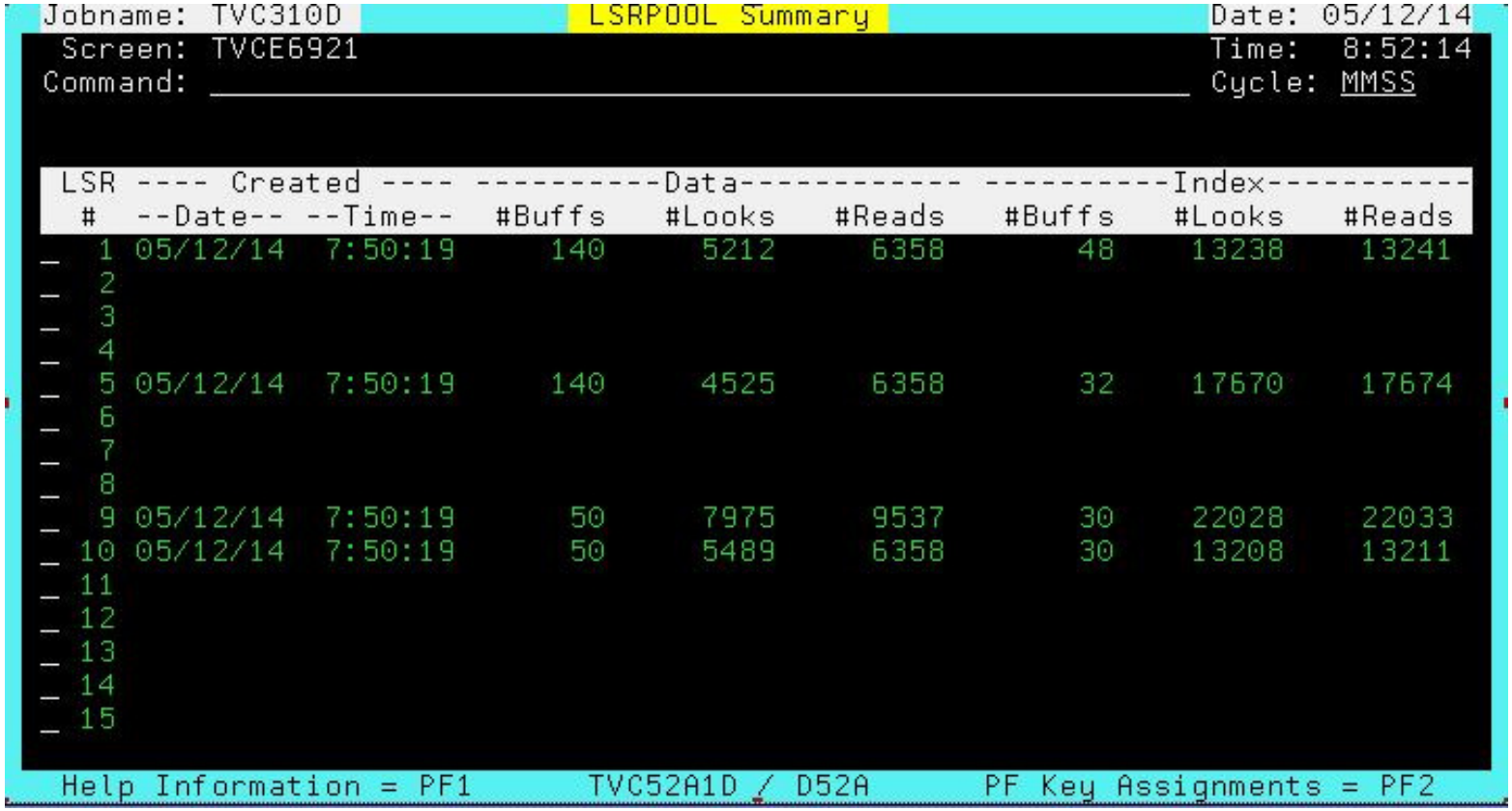
```
Jobname: TVC310D              VSAM File Activity                Date: 05/12/14
 Screen: TVCE6901                                               Time:  8:56:43
Command: _____________________________________________________________________

   Search: _______                                          Entry   18 of   59
   FileID    Type  Status  Acc    Read    Add   Updt Gupdt Delet Brwse   EXCP
_  DSTESTO   VSAM  CLO,ENA RUADB
_  DSTESTP   VSAM  CLO,ENA RUADB
_  DSTESTQ   VSAM  CLO,ENA RUADB
_  DSTESTR   VSAM  CLO,ENA RUADB
_  DSTESTS   VSAM  CLO,ENA RUADB
_  DSTESTT   VSAM  CLO,ENA RUADB
_  DSTESTU   VSAM  CLO,ENA RUADB
_  DSTESTV   VSAM  CLO,ENA RUADB
_  DSTESTW   VSAM  CLO,ENA RUADB
_  DSTESTX   VSAM  CLO,ENA RUADB
_  DSTESTY   VSAM  CLO,ENA RUADB
_  DSTESTZ   VSAM  CLO,ENA RUADB
_  DSTEST1   KSDS  OPE,ENA RUADB       0  1188  2189  3179   990     0   5238
_  DSTEST2   KSDS  OPE,ENA RUADB       0  1188  2189  3179   990     0   5238
_  DSTEST3   KSDS  OPE,ENA RUADB       0  1188  2189  3179   990     0   4418
_  DSTEST4   KSDS  OPE,ENA RUADB       0  1188  2189  3179   990     0   5745
_  DSTEST5   KSDS  OPE,ENA RUADB       0  1188  2189  3179   990     0   5238
  Help Information = PF1      TVC52A1D / D52A      PF Key Assignments = PF2
```

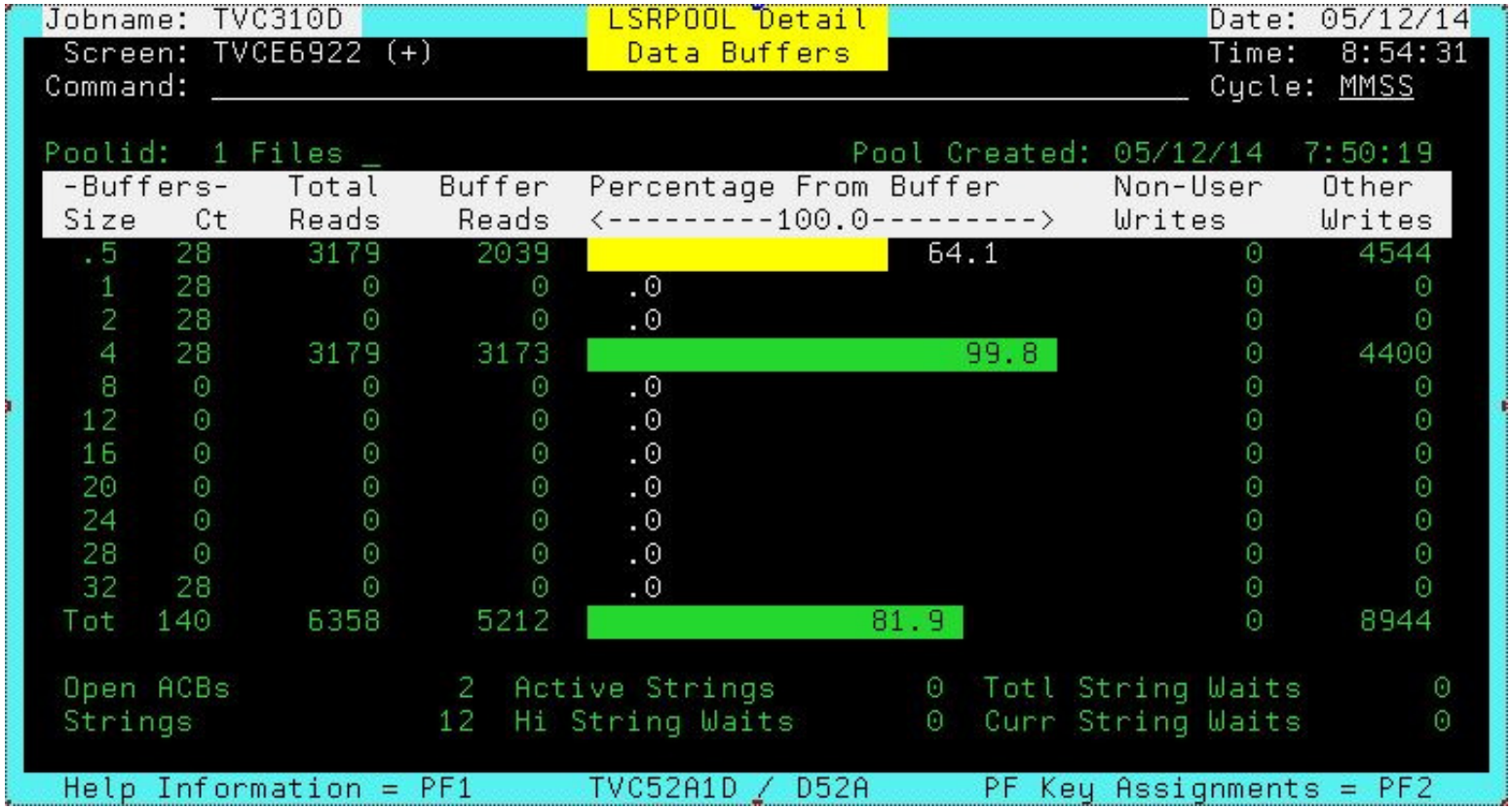
Vendor Tuning Products



Vendor Tuning Products



Vendor Tuning Products



Vendor Tuning Products

DATE: 05/06/14 T M O N R E P O R T W R I T E R PAGE: 1
TIME: 06:31:49 T R A N S A C T I O N S W I T H C A S P L I T S
DATA IS FROM 05/06/14

TRANS ID	TERM ID	CICS TASK NO	END TIME	RESP TIME TOT	DISP TIME TOT	CPU TIME TOT	WAIT TIME TOT	FILE I/O TIME	FILE COUNT TOT	FLAG BYTE 6
IESN		32	5:26:16.2280	.0788	.0195	.0015	.0593	.0492	1	03
IEGM	R000	53	5:26:37.4992	.0931	.0657	.0049	.0274	.0377	7	03
WBFC		60	5:30:22.6157	3.0382	.1415	.0154	2.8967	2.9344	44	03
WBFX		156	5:30:25.5149	5.9236	.0066	.0019	5.9170	5.2777	11	03
WBFX		90	5:30:25.5241	5.9420	.0022	.0017	5.9398	5.2612	11	03
WBFX		68	5:30:25.5545	5.9758	.0344	.0033	5.9414	5.3308	11	03
WBFX		112	5:30:25.5780	5.9945	.0039	.0018	5.9906	5.3140	11	03
WBFX		134	5:30:25.5850	5.9998	.0091	.0021	5.9907	5.3183	11	03
WBFY		116	5:30:33.4120	13.8283	.2618	.1123	13.5665	13.1322	1,134	03
WBFY		78	5:30:33.4169	13.8356	.2378	.115β	13.5978	12.9739	1,134	03
WBFY		122	5:30:33.4209	13.8367	.2444	.1119	13.5923	13.2015	1,134	03
WBFY		144	5:30:33.4251	13.8348	.2976	.1087	13.5372	13.0843	1,134	03
WBFY		100	5:30:33.4298	13.8470	.2230	.1148	13.6240	13.2365	1,134	03
WBFY		94	5:30:33.4343	13.8520	.2787	.1116	13.5733	13.2057	1,134	03
WBFY		138	5:30:33.4390	13.8534	.1594	.1095	13.6940	13.1963	1,134	03
WBFY		72	5:30:33.4435	13.8628	.2512	.1146	13.6116	13.2707	1,134	03
WBFB		64	5:30:39.3139	19.7355	7.3540	4.5420	12.3815	14.2775	53,946	03

Thank You

Please forward your questions or remarks to

poilmike@uk.ibm.com

z/VSE Live Virtual Classes

z/VSE @ http://www.ibm.com/zvse/education/

LINUX + z/VM + z/VSE @ http://www.vm.ibm.com/education/lvc/

Read about upcoming LVCs on @ http://twitter.com/IBMzVSE

Join the LVC distribution list by sending a short mail to alina.glodowski@de.ibm.com

References

CICS Library	http://www-03.ibm.com/systems/z/os/zvse/documentation/#cics
WAVV Presentations	http://wavv.org/
Ingolf's Blog	https://www.ibm.com/developerworks/mydeveloperworks/blogs/vse/?lang=en

VSAM APARs

Ensure that your VSAM Service Level includes all performance-related APARs. You can search the IBM Support Portal for your VSAM release's APARs.

Search on 5686CF905 51C for VSAM in z/VSE 5.1 (the "compid" and CLC), then use the "Content type" tick box to filter out everything except APARs. Use 5686CF905 52C for z/VSE 5.2 and 5686CF805 02C for z/VSE 4.3.

Early z/VSE 4.3 and 5.1 releases had a bug that stopped the EXCPAD being used in standard CICS File Control VSAM requests and the types of request used by DL/I and for DFHTEMP (and possibly DB2/VSE). This has been fixed for some time and is not present in z/VSE 5.2. Not having EXCPAD active means that CICS cannot dispatch other tasks until *all* VSAM I/O is complete for that one EXEC CICS request - imagine the impact of a CA split!

Some APARs have had changed performance while fixing a bug, all the more reason to know the performance profile of your environments to spot the difference after any changes!

The effect of Paging on Performance

Tuning normally considers the role of storage in performance as there may be delays from paging that are a result of a poor Virtual/Real ratio. They are not normally VSAM's fault unless you have allocated a ridiculously large amount of buffers. Paging delays will also affect the whole CICS partition, not just VSAM.

Paging should be monitored at the z/VSE level *and* at the partition level if possible, and at the z/VM level.

At the z/VM level, ensure that the z/VSE Guest VM's working set is not impacted by stealing due to other VMs' real storage demands, e.g. use CP SET RESERVE on the z/VSE Guest.

Paging at the z/VM level will not be obvious within a z/VSE system when using standard performance monitor outputs, and can cause random erratic response times that appear to have absolutely no explanation when looking at CICS and even z/VSE performance data.

NSR normally requires much more GETVIS storage than LSR because the storage is not shared - imagine how much storage results from defining 500 files each with 1,000 buffers!

The more LSR pools you have, the more GETVIS storage is normally required in total to get the same look aside compared to fewer larger LSRPOOLs.

Index CISZ Considerations

The required Index CISZ is related to two values.

The Data CISZ - the bigger this is, the smaller the Index CISZ tends to be because VSAM needs to keep fewer CI high keys in each Sequence Set CI, which is the lowest level of the Index; e.g. a 4K CISZ needs to map CI/CA=180 high keys per cylinder but an 18K CISZ only needs to map CI/CA=45.

The size of the keys - the bigger the keys, the more space is needed to map the keys.

VSAM compresses keys in the Index, and IDCAMS assumes a certain amount of compression when it checks that your Index CISZ is not too small or defaults the Index CISZ. Keys less than about 10 bytes often do not compress well as each compressed key needs 3 bytes of control information, and VSAM adds additional bytes at regular intervals within the CI. See the VSE/VSAM User's Guide and Application Programming Chapter 7.

If the Index CISZ is not big enough, some of the CA's will not be fully populated with Data CI's because there will be no room for the last "n" CI high keys in the matching Sequence Set CI. This will affect the amount of dasd space that is used and may even increase the occurrence of CA splits, although it will depend on how many CA's are affected. The Index Set comprises all Index CI's above the Sequence Set and is not affected by a CI size that is too small.

Index CISZ Considerations

If this occurs, it is difficult to identify as there are no tools to show it that I am aware of. You could use DITTO to browse or IDCAMS to print the Index component and have a look when you have a lot of time to spare. The Index component can be read as a dataset as each CI contains one logical record. LISTCAT tells you how many records and hence how many CI's are in the Index - the size of the Sequence Set is the number of CA's, normally cylinders, so the difference must be the size of the Index Set.

For safety, define the Index component CISZ by using this formula when keys are 64 bytes long or bigger.

Index CISZ = Data CI/CA * (Key length/3) and round *up* to the next LSR boundary.

For keys of length 10 bytes to 63, you could substitute (Key length/2) for safety.

LSR boundaries are 0.5K, 1K, 2K, 4K and multiples of 4K to 32K. I would use 4K unless it is grossly inefficient or you wanted to be able to select the CISZ for special handling.

Using a value that is larger than required should reduce the total number of CI's in the Index Set and improve the performance slightly. The whole Index will use more dasd space in this case, but this is normally small compared to the amount of dasd space that is required for the Data component.

How to Calculate Available Getvis Storage

In this case, GETVIS F2,RESET was done after CICS started:

```
GETVIS F2
AR 0015 GETVIS USAGE    F2-24     F2-ANY                      F2-24     F2-ANY
AR 0015  AREA SIZE:   11,260K   122,876K  (122,876K = ALLOC 120MB - 4K for SIZE=DFHSIP)
AR 0015  USED AREA:    9,828K   104,068K MAX. EVER USED:     9,828K   104,068K
AR 0015  FREE AREA:    1,432K    18,808K LARGEST FREE:       1,432K    18,808K
```

If no RESET was done, the F2-24 MAX. EVER USED = F2-24 AREA SIZE.

F2-ANY includes F2-24, but DFH0STAT output shows 24-bit and pure 31-bit.

Only 4K pages are counted. Actual USED will be less than shown as you cannot see free storage within the used pages.

MAX. EVER USED is the High-Water-Mark (AREA SIZE - LARGEST FREE is an approximate 24-bit HWM if no RESET was done).

LARGEST FREE is contiguous and may be less than FREE AREA.

"Available" contiguous storage is the smaller of (AREA SIZE - MAX. EVER USED) and LARGEST FREE.

Assuming that the data is representative, 24-bit usage could be increased by a maximum of about 1,024K and 31-bit usage by about 16MB (18,808K - 1,432K).

How to Calculate Available Getvis Storage

An LVC presentation in June 2013 has a lot of detail about z/VSE and CICS storage concepts.

Simplified CICS, VSAM, EXCP and Dasd Processing

The EXEC CICS request to a local file results in the execution of one or more VSAM macros, each of which calls VSAM and normally results in one or more VSAM I/O requests:

- If it is a read, the I/O wait time is at cpu speeds if the record is in an appropriate buffer, this is a "LOOK ASIDE". (SHR(4) look aside is *very* restricted.)
- Otherwise VSAM issues an EXCP to read or write and, in relative terms, a *lot* of cpu time is required *and* the wait time is longer:
  - EXCP causes z/VSE to build an I/O request and adds it to the device queue, it is started by SSCH if the device is not busy and VSAM gets control back. (The request is started later if the device is busy, which adds more wait time.)
  - VSAM passes control to the CICS EXCPAD exit, which issues a SUSPEND for FCIOWAIT on an ECB and the CICS task waits for I/O completion; other tasks can be dispatched while the I/O is active.
  - Meanwhile, the I/O operation is running and ends with an I/O interrupt; a read presents I/O interrupt quickly for a cache hit, otherwise the wait time increases while the dasd reads the data; a write is normally written to cache with an immediate I/O interrupt, however, a PPRC/Metro Mirror device will delay the I/O interrupt until the data is in the remote cache thus adding to the wait time.

Simplified CICS, VSAM, EXCP and Dasd Processing

- The I/O interrupt stops z/VSE dispatching the current task and the waiting CICS task's ECB is POSTed; if another EXCP request is queued for the device, it is started.
- CICS is re-dispatched by z/VSE.
- CICS RESUMEs the task when it next looks for POSTed I/O requests, and dispatches it when all higher priority CICS tasks are not active; the EXCPAD exit completes and returns to VSAM.

VSAM issues more I/O requests as required. For example, an EXEC CICS READ for a KSDS uses a VSAM I/O request to read a CI from every level in the Index component and then to read a CI from the Data component. (Now imagine 150 VSAM I/Os from a CA split!)

Control returns after the VSAM macro in CICS, which will eventually return to the application after its EXEC CICS request.

z/VM uses cpu time when it intercepts the z/VSE SSCH so that it can build and issue the real SSCH. But Mini Disk Cache (MDC) may be able to read the data from its cache and avoid the real I/O, thus making some I/O faster than is possible for Native z/VSE!

z/VM cpu time is used to handle the I/O interrupt before it passes it to z/VSE.

The cpu time is accounted for as CP overhead.

CICS Monitor Data

It is not really the correct type of data for tuning VSAM, but I have included some information for reference.

See the CICS Performance Guide chapter 6 and the Customization Guide for full details.

When enabled, data records are written to DMF. However, the data may be intercepted by Vendor software and written elsewhere. It can require up to a 10% cpu *delta* to collect it, e.g. 50% average over 15 minutes when disabled means up to 55% when enabled.

IBM provides DFH$MOLS to format it, and it produces one page per task by default!

DFH$MOLS output shows GMT, which is the assumed time zone of the real or virtual TOD clock from STCK.

CICS Monitor Data Task Performance records contain request counts and wait times, but *not by file,* which is where Vendor products can help. FCIOWAIT and FCCIWAIT times are summed in FCIOWTT. Other FCxxyyyy waits are added to SUSPTIME (total task wait time).

You see the number of EXEC CICS requests and the number of VSAM API requests, so contention can be deduced.

CICS Monitor Data Task Exception records provide string and buffer wait counts.

CICS Monitor Data

This task had a *lot* of Exclusive Control FCXCWAITs, which is seen in the ratio of VSAM requests to EXEC CICS requests, and is a big component of the SUSPTIME field since FCXCWAIT is not reported separately.

You don't get activity by file, so which one(s) is a guess!

```
-----FIELD-NAME-------------------------UNINTERPRETED-------------------------------INTERPRETED---------------

     DFHTASK C001      TRAN            E3D9C1F1                                  TRA1
     ...
     DFHCICS T005      START           CAE81843D15C819A                13/02/11 11:30:13.1378  G.M.T.
     DFHCICS T006      STOP            CAE8184A92BA8000                13/02/11 11:30:20.2213  G.M.T. Response 7.08
     ...
     DFHFILE A036      FCGETCT         000003E8                                  1000  READ or READ UPDATE
     DFHFILE A037      FCPUTCT         000003E8                                  1000  REWRITE
     ...
     DFHFILE A093      FCTOTCT         000007D0                                  2000 EXEC CICS file requests
     DFHFILE A070      FCAMCT          00000FA3                                  4003 Actual VSAM requests are 2x bigger!
     ...
     DFHJOUR A058      JCPUWRCT        000007D6                                  2006
     ...
     DFHTASK S007      USRDISPT        0000247B00000FA3                00:00:00.14942      4003  Dispatch time and count
     DFHTASK S008      USRCPUT         0000034B00000FA3                00:00:00.01348      4003  Cpu time
     DFHTASK S014      SUSPTIME        00069CE200000FA3                00:00:06.93404      4003  Total wait time
     DFHTASK S102      DISPWTT         0000368700000FA2                00:00:00.22334      4002  Wait component times here
     ...
     DFHFILE S063      FCIOWTT         0000AE0E000003E4                00:00:00.71292       996  Includes any CI/CA split
     DFHJOUR S010      JCIOWTT         0001A4E9000007D1                00:00:01.72404      2001  Journal I/O
     ...
     DFHTASK S125      DSPDELAY        000002AA00000001                00:00:00.01091         1  Wait for dispatch
     ...
     DFHTASK S129      ENQDELAY        0000002D00000001                00:00:00.00072         1  Wait for ENQ
```

CICS ESDS Notes

A standard WRITE uses RPL option DIR.

WRITE MASSINSERT uses RPL option SEQ, which uses less EXCPs if you are writing many records at a time. An UNLOCK is required to terminate the MASSINSERT.