

## Care and Feeding of Monitor APPLDATA

SESSION 0761

Bill Bitner  
IBM Corp.  
1701 North St.  
Endicott, NY 13760  
(607) 752-6412  
bitner@gdlvm7.vnet.ibm.com

SHARE 78.0 - March 1992

(c) Copyright IBM Corporation 1991,1992 - All Rights Reserved

### Disclaimer

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "As is" basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environments may vary significantly.

Users of this document should verify the applicable data for their specific environment.

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Permission is hereby granted to SHARE to publish an exact copy of this paper in the SHARE proceedings. IBM retains the title to the copyright in this paper, as well as the copyright in all underlying works. IBM retains the right to make derivative works and to republish and distribute this paper to whomever it chooses in any way it chooses.

1

This presentation was put together in order to take the mystery out of APPLDATA. It has been presented at the following places:

- October 1991 - members of OV/VM Performance area visiting EPL from Dallas.
- November 1991 - SHARE 77.5 Pittsburgh PA.
- March 1992 - SHARE 78.0 in Anaheim, CA.

Feel free to re-use this presentation. Please let me (Bill Bitner) know of any major group or area (inside or outside of IBM) that you present it to and any comments that resulted. I plan on getting a lot of mileage out of this, and will spend time making improvements. I enjoyed pulling this pitch together. I hope it's of value to you.

## Trademarks

The following are trademarks of the IBM Corporation

- IBM
- VM/ESA

2

## Introduction

- In order to do performance management and analysis, it is becoming more and more important to understand what is going on inside the virtual machines.
- Use of CP Monitor APPLDATA can help.
- Agenda
  - Why do we need APPLDATA?
  - What is APPLDATA?
  - How does it work?
  - How is it currently being used?
  - Next Steps
- Presentation will center around APPLDATA in VM/ESA 1.0 ESA Feature.

3

VM Development has been saying APPLDATA is the cure for a lot of performance headaches. With the size and complexity of performance analysis growing, these headaches are becoming very painful. APPLDATA is a relatively simple concept and that's part of what this presentation is meant to show. At the same time, it isn't perfect and we'll cover that as well. I'll put my money where my mouth is and go thru a real example of using APPLDATA.

A little bit of APPLDATA in other releases will be covered. Unless otherwise noted, the material is based on VM/ESA 1.0 ESA feature as well as VM/ESA 1.1 which is now generally available. It wasn't at the time this pitch was first put together.

#### Why is APPLDATA Needed?

- To CP a virtual machine is just a virtual machine. It processes requests for various system services, but often has no idea what they are really for.
- One can associate resources consumed with a virtual machine, but not for actual tasks going on in virtual machine.
- Server virtual machines magnify problem because they are often just satisfying requests from end users.
- Server virtual machines have grown in number and amount of resources they consume.
- The performance analyst needs to know what's going on up there.

4

Growing up, I shared a room with an older brother. It was on the second floor. There were a number of times we would disagree and next thing you know we were fighting. Dad would yell up from the first floor, "what are you boys doing up there?". It was a good thing he didn't know.

To CP, the happenings of a virtual machine are the same way. CP can see there is a lot of activity (I/Os, paging, privops, etc.), but has no way to associate them with "what's going on up there". If the performance analyst is to understand the system performance, they need to be able to see it all.

As the use of server machines grows, this becomes more and more important.

## VM Monitor Review

VM/ESA (ESA Feature)

Domain	Name	Event	Sample	HiFreq
0	System		Sample	Hi Freq
1	Monitor	Event	Sample	
2	Scheduler	Event		
3	Storage	Event	Sample	
4	User	Event	Sample	Hi Freq
5	Processor	Event	Sample	
6	I/O	Event	Sample	Hi Freq
7	Seek	Event		
10	Appldata	Event	Sample	

MONDCSS  
monitor  
saved  
segment

v  
Monwrite CMS Program  
Collection Machine  
IUCV  
CP \*MONITOR system service

5

This is a brief overview of what the VM Monitor is and how it works. The table shows the various Domains and the type of data found in them. Event means we cut records when certain events occur. Sample means we gather the data at fixed intervals. HiFreq means some form of state sampling is used.

The figure illustrates how an application (in this case the IBM supplied MONWRITE) connects to the CP System Service \*MONITOR with IUCV. \*MONITOR and MONWRITE handshake in order to know when and what data is placed in the special DCSS for monitor data.

---

**What is APPLDATA?**

- It was first introduced in VM/SP Rel 6, followed by HPO 6.
- It is available in VM/ESA (both features).
- APPLDATA provides a method for putting application data into the CP Monitor.
- Data collected by "Sample" as opposed to "Event".
- Designed with server virtual machines in mind.

6

---

APPLDATA was introduced in VM/SP 6 for support of SFS and is also used in HPO 6. With VM/ESA, it is available in both features. There are some differences which will be pointed out.

The main purpose is to allow an application to add data to the CP Monitor. In a common place, it is easier to analyze and combine with other performance data. Nothing is worse than having so many sources of performance data you don't have room on the desk to take notes.

The current implementation is based on sampling data (SAMPLE), not tracing (EVENT). Sampling keeps the amount of data to a manageable size and provides the necessary info for the majority of cases.

As mentioned, it was introduced with SFS (Shared File System) and therefore designed with the server virtual machine in mind. Since the growth and importance of server virtual machines in VM has grown, this is golden opportunity to manage them.

---

**VM/ESA 1.0 370 Feature**

- APPLMON must be in Directory OPTION statement of virtual machine providing data.
- CP Command - MONitor ENable APPLdata
- When enabled collects data for all virtual machines with data
- Class 10 (X'A')
- Single Record - Code 00

7

This foil gives the basics of APPLDATA in 370 feature (also SP and HPO 6). To be allowed to produce APPLDATA, a virtual machine must have the APPLMON operand in the directory OPTION statement. The CP Monitor command was extended to include the new class of APPLDATA. In this class there is a single record.

---

**VM/ESA 1.0 ESA Feature**

---

- APPLMON must be in Directory OPTION statement of virtual machine providing data.
- CP Command - MONitor EVent ENable APPLdata
- CP Command - MONitor SAMPlE ENable APPLdata
- Above commands have option USERID userid-list
- Domain 10 (X'A')
- Record 1 = Application Data Event Record (APPLDATA collection has ended).
- Record 2 = Application Data Sample Record

8

---

In the ESA feature, again APPLMON must appear in the OPTION statement. Both the Monitor Event and Sample commands were extended to include the new domain (ESA monitor uses domains, not classes). There are two records in this domain.

Wait a second, I said APPLDATA was for Sampling only, what's this Event thingy?? That is merely a record to signal that APPLDATA collection has ended. This is important during reduction, since knowing whether there was a problem with data collection is valuable.

Two differences from 370 feature to note:

- The event data for collection ending.
- The ability to select individual userids of APPLDATA in ESA and not in 370. In 370, it is an all or nothing scenario. This isn't bad since the number of APPLDATA applications in those environments is low. However, that selection can be important in large systems to control amount of data collected.

#### Process in Virtual Machine

1. Create data structure to serve as buffer for application data.
2. Initialize data values and application collection.
3. Issue Diagnose X'DC' to notify CP that buffer exists.
4. Continue processing as normal.
5. If application is ending, issue Diagnose X'DC' to notify CP that buffer is to be deleted.

9

We'll look at first at the process in a virtual machine (application) and then at the process in the system as a whole. There are basically five steps:

1. you need to have some storage structure for the "performance data" you are interested in. This can be hardcoded or something created dynamically.
2. You'll want to initialize the structure (counters, timers, whatever) and the collection process.
3. A Diagnose x'DC' is used to signal CP that you are interested in having APPLDATA collected. This will be covered in much more detail later.
4. Processing continues as normal with the "performance data" being updated by the application as appropriate.
5. The Diagnose x'DC' can be reissued when application terminates. This signals CP that you are no longer interested in data collection.



#### Diagnose Code X'DC

- Control is maintained by "APPLMON" in virtual machine's directory OPTION statement.
- Entry value of Rx is guest real address of parameter list.
- Parameter list contains (\* - more detail to follow)
  - DIAG - X'DC'
  - Function Code (\*)
  - Length of parameter list
  - Product ID Address (\*)
  - Buffer Length (\*)
  - Buffer Address (\*)

10

The APPLMON entry in directory OPTION statement is what allows you to issue Diagnose x'DC'. Diag DC is structured like most diagnoses, with the Rx register pointing to parameter list with above items.

---

**Diagnose Code X'DC ...**

- Function Code
  - X'00' = Notify CP of APPLDATA area
  - X'01' = Notify CP to discontinue use
- Product Id Address
  - points to 16 byte field containing identifier
  - This is key to separating APPLDATA from various applications
  - Suggested that it includes product number and info to determine level of product or application.
- Buffer Length - between 1 and 4024 bytes inclusive.
- Buffer Address - virtual address of data buffer.

11

---

The function code is the flag for CP of whether you (the application) are interested in data collection. The wording in some manuals is a little confusing. It says Start is "Start Interval Recording", this doesn't mean CP will at that very moment cut a sample record. It only means CP will include this APPLDATA at the next sample interval. More on this later.

The Product Id is the method we're going to use to tell all this data apart. We could have 100s of virtual machines providing APPLDATA with different meanings.

The buffer length is such that it will fit in 4K frame with monitor record header.

The storage is just your application's virtual storage.

---

**Diagnose Code X'DC ...**

- More than one buffer can be declared for a single virtual machine.
- The application can place data into the buffer at any time.
- Return Codes in Y register of Diagnose.

12

You may need more than one buffer and this is allowed. However, you'll need to do a Diagnose DC for each. There is no need to be concerned with CP blocking you from updating the buffers.

Ry of Diag DC is return code. Possibilities include (not all):

- 0 = okie dokie
- 2 = invalid function code
- 5 = not authorized to use Diag DC
- 6 = buffer length not in allowable range.

### System Process

- MONITOR Commands to Enable and Start Monitor Collection
- CP Monitor collects data at 1 minute (default) interval
- Application defines APPLDATA area and issues DIAG x'DC'
- CP processes Diagnose
- End of next 1 minute interval, CP collects data from APPLDATA area.
- Application continues to run and update APPLDATA fields
- End of next 1 minute interval, CP collects data from APPLDATA
- ...and so on...
- Application issues Diagnose x'DC' with function code to end collection.
- CP APPLDATA Event record (D10 R1) created.
- End of next 1 minute interval, CP no longer collects from specific buffer.

13

Now that we know what the application must do, we'll fit it into the system as a whole. Monitor domains and intervals would be set up as normal. Also Monwrite or other \*MONITOR application could be running. Monitor would be collecting sample data at a preset interval. After the application issues Diag DC to start, CP will take a snapshot of the APPLDATA buffer at the next sample interval. This continues until the Application issues Diag DC to stop or the monitor itself is stopped. If the Application stops and APPLDATA Event is enabled, CP will cut a record to indicate APPLDATA collection for that buffer has terminated.

---

**Monitor Data Contents - Domain 10 Record 2**

- Record Header Info
  - record length
  - domain/record
  - TOD
- CP added info
  - Offset to Application Buffer
  - Length of Application Buffer
  - Userid
  - Product Identifier
  - Flag Byte
- Data from Application

14

There are three parts to Sample Record for APPLDATA. The first is just the normal record header info. Note the TOD, we'll use that later.

CP also adds some info that is useful. The offset should be used to get to the data, since over time CP may change size of the CP added data. The userid is userid that issued Diag DC. The prod id is what was passed with the diag. The flag byte contains things like "this is the very first sample interval for this buffer".

Finally, there is a copy or snapshot of the user supplied data.

#### How Currently Used

- SFS (Shared File System) and CRR (Coordinated Resource Recovery) use APPLDATA in the server virtual machines.
- SFS and CRR are closely tied in structure and use same Product Identifier.
- Data Buffer contains a series of counters and timings associated with key structures of the SFS server.
- IBM Products with support include:
  - VMPRF
  - VMPPF
  - VMPAF

15

As great as APPLDATA is; it hasn't caught on very fast. Hopefully, this pitch will help turn that around. Two components of VM currently use APPLDATA (SFS and CRR). They use APPLDATA to hold a series of counters and timings. There are a number of IBM PPs that support this data. VMPRF uses a subset of data to produce SFS specific reports, and includes all values in the SUMMARY/TREND files. With these files, VMPAF can surface any of the variables and also includes a large number of pre-defined equations based on them.

### Example VMPRF SFS APPLDATA Support

PRF083 Run 02/14/1992 08:49:13 SFS\_BY\_TIME  
From 09/26/1991 21:49:02 SFS Activity by time  
To 09/26/1991 22:31:02  
For 2520 Secs 00:42:00 Example SFS APPLDATA

< Time Per File Pool Request >										
From Time	To Time	Userid	FPR Count	FPR Rate	Total	CPU	Lock	Block I/O	ESM	Other
21:49	22:31	CRR	893	0.354	0.000	0.002	0	0.000	0	0.001
21:49	22:31	SFS1	91865	36.453	1.629	0.003	0.355	0.583	0	0.689
21:49	22:31	SFS2	86963	34.507	1.750	0.003	0.388	0.646	0	0.713

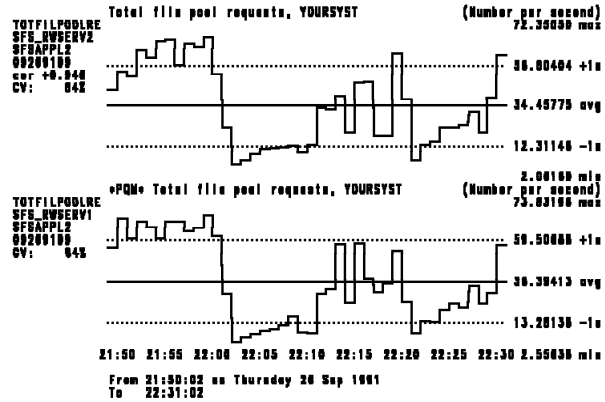
< Server Utilization >											< Agents >	
From Time	To Time	Userid	Total	CPU	Page Read	Check point	QSAM	Active	Held	Dead locks w/ RB		
21:49	22:31	CRR	1.4	0.1	1.3	0	0	0.0	0.0	0		
21:49	22:31	SFS1	51.0	10.8	32.3	7.9	0	59.4	271.4	0		
21:49	22:31	SFS2	51.0	10.5	32.8	7.8	0	60.4	267.3	0		

16

This is an example VMPRF report. It has been edited to fit onto a single page. There are two other VMPRF reports for SFS, but I chose this one since it does something unique. It combines APPLDATA from SFS with other monitor data for the server machines.

In this report, three server machines are shown. The first, CRR, is the CRR Recovery Server. The next two are SFS File Pool servers. The "Time per File Pool Request" area shows CPU time normalized to filepool requests. CPU time comes from normal monitor records for virtual machines and the file pool request rate comes from APPLDATA.

### Example VMPAF SFS APPLDATA Support



17

This is an example of VMPAF support for SFS APPLDATA. In addition to all the counters and timings put in APPLDATA that VMPAF exposes, there are several pre-defined equations that make analysis easier. This example shows the file pool request rates for each of the two file pool servers. VMPAF could be used to compare any of the APPLDATA statistics with any other monitor (or other source) available.



#### What to include in APPLDATA?

- Two things -
  - Running Count of requests
  - Timing of requests (stop watch)
- Third item = time stamp when Sample is taken which is included in Monitor data.
- Deltas of three items from sample to sample give:
  - Count - delta of count of requests
  - Usage - delta of timings
  - Interval - delta of time stamps
- From these we can calculate:
  - Rate = count / interval
  - Response Time = usage / count
  - Utilization = usage / interval

18

This foil was interesting to put together. It tries to answer the question of "what is good performance data?". Anyone knowing the VM monitor knows we don't always answer that question correctly.

For the majority of the cases, it is three simple things. Two we'll ask the application to provide; and the third is provided by CP. We need to know how many requests are made (simple clicker as people go thru turnstile) and how long it takes (start stopwatch on way in and stop on way out). Take those two and include TOD from CP.

We'll periodically sample the current values for those three items. Taking the deltas of each from two adjacent samples gives us the Count (how many people), Usage (how much time for handling them), and Interval (how long between samples). Then simple division gives us important data: Rate (how many people per minute), Response Time (how long to handle each request), and Utilization (how busy are we).

#### What to include in APPLDATA? ...

- If the requests can be served by multiple servers, include count of servers or individual counters for each server.
- Every option should be included somewhere.  
Example -
  - There are five types of requests.
  - Do not just include counters for 3 of them.
  - Unless you include a "TOTAL" or "OTHER" category.
- If there are other methods of getting "performance" data, be sure to use the same values and meanings for all methods.
- Be careful of wrapping values.
- May be worthwhile to create set of macros for "counting" and "timing".

19

A few more things to consider in data collection. Things do get more complicated with multiple servers, but can be managed if you think about it. Please don't leave things out that you think are unimportant. The one time they become important may be when your system is running on its knees. (VM has some spots like this).

Be consistent. If you have a QUERY PEOPLE command, make sure the data returned has the same meaning and value as the PEOPLE variable in monitor.

You can only count so high with 2 bytes. Anticipate ranges needed to avoid unnecessary wrapping. A single wrap per interval is nasty, but containable. Multiple wraps per interval leaves useless data.

Since counting and timing is most of what you need, macros might make life easier.

---

**What to include in APPLDATA? ...**

- Any constants in APPLDATA should be initialized prior to issuing Diagnose x'DC'.
- Update performance fields in APPLDATA area. Do not manage else where and move. This helps keep data current when it goes into monitor.
- Various timing mechanisms exist -
  - STCK instruction
  - DIAG x'0C' - Pseudo Timer

20

---

Some APPLDATA specific thoughts. Constants should be set in APPLDATA buffer before Diag DC since CP could sneak in between Diag and initialization. Update the values right in the buffer. If you keep them else where and move them, there can be anomalies caused by timing between CP sampling and your moving of the data.

Two ways to get at time are STCK and DIAG C. The Store Clock instruction is nonprivileged and gets TOD. DIAG C (Pseudo Timer) gets some different data, but is much more expensive. The data from DIAG C may be obtainable from other CP Monitor records.

### Example Implementation

- Modify stress tool for paging called THRASHER.
- THRASHER inputs
  - Number of pages to reference
  - Number of times thru reference loop
  - Number of seconds to sleep between loops
- Application Data Collected
  - number of pages being referenced
  - number of seconds to sleep
  - Accumulator of times thru loop
  - Accumulator time to reference pages (milliseconds)

21

Talk is cheap, so lets try a real example. We (performance evaluation) have a program to help stress the paging system. I've modified it to use APPLDATA. It will collect 4 items. The first two are just initialization or config data. The second two are the two items we talked about earlier: Accumulator of times thru loop (Running count of requests) and Accumulator of time to reference pages (timing of requests). Later we'll throw in TOD to get the other things I talked about.

### Example Implementation ...

Create data structure to serve as buffer for application data.

```
*      Definitions for Diagnose x'0C' and the APPLDATA buffer
DCPARML DS      0F
DIAGDC   DC      X'00DC'
FUNCCODE DC      X'00'
LENPARML DC      X'10'
PRODLAD  DC      A(PRODID)
          DS      CL2
BUFLLEN  DC      H'20'
BUFADDR  DC      A(APPLBUF)
          DS      0F
STRTINT  EQU      X'00'      Func code to indicate APPLDATA use
STOPINT  EQU      X'01'      Func code to indicate stopping
PRODID   DC      C'PPPPPPPPFNVVRMM' Product Identifier
APPLBUF  DS      0D          Start of actual APPLDATA Buffer
ADSLEEP  DS      CL8          Appldata value for seconds in sleep
ADPAGES  DS      F'00'       Appldata pages referenced in loop
ADTIMER  DS      F'00'       Appldata total time to process loop
ADLOOPS  DS      F'00'       Running count of times thru loop
APPLELEN EQU      * APPLBUF   Length of Buffer
```

22

The entire program is included in the APPLDATA package, and is usually handed out with the presentation.

This shows data structures for Diag DC parm list and the actual APPLDATA buffer the program uses.

### Example Implementation ...

Initialize data values and application collection.

```
*
* Set up APPLDATA area and issue Diagnose x'0DC' to tell CP
* 1. We initialize some values in our APPLDATA buffer area
* 2. Set up parm list for Diagnose
* 3. Issue Diagnose and check for errors
*
MVC ADSLEEP,SLEEPVAL      Initialize Sleep Time in APPLDATA
MVC ADPAGES,NUMPAGES      Initialize Page counts in APPLDATA
MVI FUNCODE,STRINT        Set function code to start
LA  R4,DCPARML            Set R4 to address of parm list
SLR R5,R5                Clear R5 for return code
DC  X'83',X'45',X'00DC'   Issue Diag DC
LTR R5,R5                Check for 0 Return Code
BNE DCFAILED              Branch if Diag failed
```

23

This sample of code shows the setup and issuing of the Diag DC. First we initialize the constants in the buffer, set the function code to start, and then issue the Diagnose. My example program does do nominal error checking for Diag DC.

### Example Implementation ...

Stop watch example.

```
*
* Compute time difference
*
      STCK  TODEND      Store time of day clock
      LM   R8,R9,TODEND get ending TOD
      SLR  R9,R3        Subtract low words
      BC   11,NOBORROW  Branch if not borrowed
      S    R8,-F010     Subtract for borrow
NOBORROW DS 0H
      SLR  R8,R2        Subtract high words
      SRDL R8,12        Shift to get microseconds
      D    R8,-F010000  Divide to get milliseconds into R9
      A    R9,ADTIMER   Add to running timing
      ST   R9,ADTIMER   Store running timer in APPLDATA buff
      L    R9,ADLOOPS   Get running count of passes
      LA   R9,1(R9)     Bump running counter up one
      ST   R9,ADLOOPS   Store in APPLDATA Buffer
```

24

Registers 2 and 3 contain the TOD from when we started the request. In this code section, we first get the ending TOD via STCK into Register pair 8 and 9. This code shows how to subtract the two. It then goes on to get the delta into more manageable units (microseconds to start, and then milliseconds). Value is stored into APPLDATA buffer (ADTIMER) along with the loop counter (ADLOOPS).

### Example Implementation ...

- Home grown exec used to extract info from Monitor data. (250 lines)
- Home grown exec to report info. (125 lines)
- Home grown execs to input to VMPAF. (150 lines)

### Example output -

```
THRASHER Monitor Report          13 Feb 1992 19:36:37
Sleep value (seconds) : 3
Pages in reference loop : 3000

Time      Interval  Loop  Usage  Rate  Response  pct.
              Count   (ms)                Time (ms)  Util.
18:37:05    30       10   5191   0.333   519.1    17.3
18:37:35    30       10   3355   0.333   335.5    11.2
18:38:05    30       10   3377   0.333   337.7    11.3
18:38:35    30       10   1024   0.333   102.4     3.4
18:39:05    30       10   1034   0.333   103.4     3.4
...
19:03:35    30        9    448   0.300    49.8     1.5
19:04:05    30       10    635   0.333    63.5     2.1
19:04:35    30       10    723   0.334    72.3     2.4
19:05:05    30       10    474   0.333    47.4     1.6
19:05:36    30       10    503   0.331    50.3     1.7
19:06:06    30       10    704   0.333    70.4     2.3
Total/Average      588  50202   0.332   85.4     2.8
```

25

Collecting the data is the first step. To be of value we'll want to reduce the data and report some how. I've put together some execs for my example. First I use an EXEC to pull out all the APPLDATA records from the monitor data. Then a second EXEC is run to produce the report shown on this foil. I hope to add an exec in future to pass data on to VMPAF.

I admit these are quick and dirty to some extent. I don't do a lot of error checking and I currently don't take advantage of other related monitor data for reports. However, as an example, these EXECs show that reduction is capable with a little effort. In addition, for PPs that are common, I believe the owners of reduction packages will take care of the reporting. Just tell them what is there.

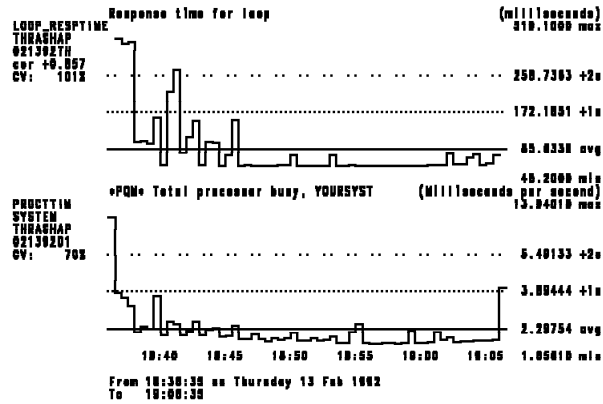
My execs are available upon request.

The Report example shows the time stamp, the three deltas for the interval, and the three simple calculations. In addition, the configuration information is included up top.



Example Implementation ...

VMPAF example results



26

For those of you who don't like looking at rows or numbers (and I don't know too many who do), VMPAF can be used to show graphics of the data and do correlation. The example here shows two graphs. The top is Response time for a loop (time it took to reference all 3000 pages in the reference loop). This comes from the APPLDATA info the program provided. The bottom graph is Total processor busy time. This comes from the VMPRF summary file. When THRASHER first starts up, it takes a little more to get started. Therefore both Response time for Loop and Processor busy time are higher at the start. Some of the other spikes are caused by background jobs that were running.

### Next Steps

- Heighten awareness of APPLDATA use and benefits.
  - It is ideal for many server type applications.
  - As server virtual machine usage grows, so does the need for comprehensive performance information.
  - Many opportunities exist inside and outside of IBM.
- Extend ability to analyze APPLDATA
  - Most capabilities currently exist only for known users (SFS and CRR).
  - User exits for monitor reduction packages to deal with APPLDATA?
  - Capability to separate APPLDATA from rest of Monitor data to avoid a second pass thru the data?

27

The VM community has work to do in order to realize the full potential of APPLDATA. This includes getting more products to use it and the reduction packages to report what is collected. In addition, for APPLDATA to be worthwhile for installation specific use, the reduction packages need to be extended.

---

### Summary

- APPLDATA is the prescribed method for collecting performance data of an application.
- Most common use is with server virtual machines.
- Next Steps -
  - Increase use and acceptance of APPLDATA.
  - Extend ability to analyze APPLDATA.

28

I didn't spend a lot of time showing how APPLDATA can be useful. I would hope anyone that has tried to do performance analysis on a system with complex applications and/or servers realizes this. Performance analysis has become more complex. In order to avoid it becoming impossible, we need to address performance management across the board.

I plan to continue working to increase the awareness of APPLDATA and its use. If anyone knows of a product that should use it, let me know. If anyone has a product where they think it might help, let me know. I'm willing to work with them to make it a reality.

## Acronyms

**APPLDATA** Application Data

**CMS** Conversational Monitor System

**CP** Control Program

**ESA** Enterprise Systems Architecture

**HPO** High Performance Option

**SFS** Shared File System

29

## Bibliography

1. "CP Planning and Administration for 370" VM/ESA 1.0, SC24-5430
2. "CP System Command Reference for 370" VM/ESA 1.0 SC24-5434
3. "CP Command and Utility Reference" VM/ESA 1.0 SC24-5519
4. "CP Programming Services" VM/ESA 1.0 SC24-5520

30

## APAFOIL Processing Options

APAFOIL

August 31, 1990

Release 3.0

### Runtime values:

DEVICE	3820A
BIND (Odd, Even)	1.00i, 1.00i
TWOPASS	YES
INDEX	NO

### Foil Set: 1

Input File (Current)	APPLDATA
----------------------	----------

### Layout of Heading (FOILHD Tag or Default)

FOILHD	NULL	NULL	NULL
FRAME			NONE
FOILHD1	HEAD	NULL	NULL

### Layout of Body (LAYOUT Tag or Default)

FRAME	BOX
FRAMEWT	LIGHT
RULE	SOLID
BORDER	NONE
RUBRICWT	LIGHT

### Layout of Footing (FOILFT Tag or Default)

FOILFT	NULL	NULL	NUMBER
FRAME			NONE

### Statistics:

Title Page	1
Contents	0
Parts	0
Foils	30
Notes	0
Overflow	0
Total Pages	31

### APAFOIL Messages:

Information	0
Warning	0
Error	0

### System Variables:

SYSVAR D	YES
SYSVAR F	MIN4
SYSVAR G	NO
SYSVAR H	NO
SYSVAR I	NO
SYSVAR M	&zsvmsv(1).
SYSVAR N	INCLUDE

