WAVV 2012 Solving CICS Performance Problems

Mike Poil

CICS Level 3 Service

IBM Hursley

poilmike@uk.ibm.com

April 2012

ON DEMAND BUSINESS™

Important 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. 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.

When the word "CICS" is used, it refers to CICS TS for VSE/ESA 1.1.1 unless specified otherwise.

Agenda

- Introduction.
- Basics of Performance Monitoring.
- Free tools that I have used.
- What about OEM performance tools?
- Hardware and Software operation.
- What does Resource Usage analysis give you?
- What does Degradation Analysis give you?
- Exception and Trend reporting.
- How do you interpret and validate the data?
- Case studies.
- Q & A.
- Additional reference material.

Introduction

- This is based on experience of looking at *actual* CICS performance problems and helping customers to solve them.
- I do not claim to know everything nor am I infallible.
- If you have questions, feel free to email me.
- The general techniques discussed in this presentation can be applied to many types of performance problems within z/VSE.

Basics of Performance Monitoring

- Monitoring is a requirement at every level of your system.
- Don't assume that something cannot be a problem because it has never been one before, always check everything.
- Identify the "normal" profile at each level, when performance is bad you may see why from the difference or get a clue, and keep historical data.
- When a problem happens, track down *all* changes.
- Components may be close to their limits and even a small change can produce a critical situation.
- In the complex systems that are used today, the problem may be due to something completely unrelated that is behaving differently, i.e. the component seeing the problem may be the victim.

Basics of Performance Monitoring

- For development and migration projects, use representative load tests and monitor them to assess the potential impact.
- Do not assume that the production behaviour will always be the same as it was in test!
- If you open PMR, give IBM as much background/history information as possible.

Free Tools

- **This only covers those that I have used.**
- **CPUMON**
  - A batch program showing *total* z/VSE cpu usage at intervals in units of seconds.
  - A very low overhead.
  - It must run at a high PRTY.
  - Output to SYSLST in CSV or XML format.

Free Tools

§ MONPTN

- A program showing the % of time that selected CICS or non-CICS partitions are ready-to-run (dispatchable by z/VSE), and their cpu utilisation % at intervals in units of seconds; it is a download from the WAVV 2012 site.
- If the reported %ready-to-run is much greater then the reported %cpu, the partition is being starved of cpu time.
- A very low overhead despite having a very high sampling frequency for accuracy.
- It must run at a very high PRTY.
- Output to SYSLST in CSV format, e.g.

```
2012/02/16-11:11:52,C1,015.07,00012.05
2012/02/16-11:12:52,C1,014.93,00014.28
2012/02/16-11:13:52,C1,020.87,00019.74
2012/02/16-11:14:52,C1,011.93,00011.51
2012/02/16-11:15:52,C1,011.67,00011.06
```

Columns 2 to 4 repeat for each partition
Column 3 is %ready-to-run
Column 4 is %cpu
This shows *very* good cpu availability as %cpu is just below %ready-to-run.

Free Tools

§ CICS Monitor Data

- § Very detailed CICS task-level resource usage and resource wait data.
- § Modest overhead with output of task and exception records to DMF.
- § IBM DFH$MOLS analysis output to SYSLST with one page per task! **L L**
- § I modified DFH$MOLS to produce CSV-format data and analyzed it using a relational database.

Free Tools

§ CICS Statistics Data and DFHSTUP

- § Statistics are always collected, you choose whether or not to write data records.
- § SIT STATRCD=ON tells CICS to write to DMF (if active) at the end of every interval (default 3 hours) and then causes a "reset" of many of the data values.
- § A "reset" always occurs at the CICS end-of-day; the default is midnight and you need a PLTPI program to change this.
- § DFHSTUP output to SYSLST or to // DLBL DFHPRNT, it can produce interval reports and/or a summary report.
- § Useful for overall CICS analysis.
- § The reset at every interval may help with drilling down to times of bad performance, set the interval based on how long you get bad response for bearing in mind that the minimum interval is 1 minute.

Free Tools

§ DFH0STAT (STAT transaction)

- § Overall CICS activity since CICS was started, but beware of resets.
- § Output to SYSLST and requires SIT SPOOL=.
- § Run manually or in the CICS PLTSD.
- § COBOL and Assembler source code in PRD1.BASE.
- § Member DFH0STAT in ICCF library 59 to linkedit the object code.
- § I have a Rexx analyzer and cumulative CSV data generator as a WAVV 2012 download.
- § Another WAVV 2012 download has the latest version of DFH0STAT with all APARs included *plus* an extra change to show partition-level cpu time.

Free Tools

§ CICS Auxtrace

- § Requested by L3 in obscure cases where we need to see exactly what CICS is doing as there may be a bug.
- § We will typically ask for non-standard component-level trace to be enabled by CETR, e.g. DS=1-2 and FC=1-2 to look at VSAM I/O-related problems (DS=1-2 to see every CICS partition wait, and FC=1-2 to see EXCPAD events to determine when VSAM is rejecting requests or actually starting I/O operations.).
- § Analysis is very difficult.

§ SDAID

- § Requested by L3 in obscure cases where we suspect that the problem is SVC-related.

OEM Monitoring Tools

- Very powerful.
- Analyzing CICS performance problems without them can be a real challenge.
- Produce detailed data with online and batch reporting.
- Make sure that you can still access the detailed data for a while after it was created.

Hardware Operation - Central Processor

- z/VSE does not normally have exclusive rights to a cpu.
- There is a small LPAR overhead, and actual cpu utilisation is measured at the LPAR level.
- z/VM and z/VSE add overhead.
- z/VSE multi-processing increases the overhead and has limitations.
- At a certain level of cpu utilisation, performance will begin to degrade.
- As it grows further, the degradation will tend to increase exponentially, which is the effect of queuing.

Hardware Operation - Central Processor

- As you move towards the saturation point of the available cpu time, you end up with a "latent" demand.
- That is the amount that would be consumed if it was available *and* there were no other inhibitors.
- Latent demand is difficult to measure, and if the only way out of the situation is to have a faster cpu, how much faster should it be?
- Answer: Probably more than you think!

Hardware Operation - Dasd I/O

- **Unless you have PAV, each volume only allows one active I/O at a time.**
- **Whatever, the physical I/O operation:**
  1. Starts when the operating system issues the SSCH instruction.
  2. Waits in the I/O subsystem for a path to the device - PENDING time.
  3. Waits for the disk to position to the correct place - DISCONNECT time.
  4. Waits for data transfer - CONNECT time.
  5. Is complete when the I/O interrupt occurs.
- **Pending should be small.**
- **Disconnect time will be zero for a cache hit.**
- **Connect time depends on the amount of data and the hardware used.**

Hardware Operation - Dasd I/O

- **At a certain level of utilisation, I/O performance will begin to degrade due to queuing, and as it grows further, the degradation will tend to increase exponentially.**
- **On z/VM without PAV:**
  - When using z/VM with dedicated dasd and non-shared full-pack minidisks, the real queuing will be seen only in z/VSE.
  - Otherwise there can be z/VM-level queuing as well.
- **See the reference material at the end of the presentation that shows you how to use the AR SIR SMF command to obtain dasd I/O response and queuing times.**

Hardware Operation - Storage

- Insufficient real storage causes paging.
- When a Virtual Address is used that it is not allocated to Real Storage, a Program Check occurs; interrupt code X'11' is a Page Fault.
- For a valid address, z/VSE will allocate a real page and schedule a page-in if the original content of the frame was paged-out.
- Page I/O is high priority but is subject to dasd I/O performance.
- Page-in I/O activity is the one to concentrate on as page-out is normally asynchronous with execution.
- The z/VM paging design is very sophisticated and is *very* different to z/VSE; this is a useful URL: http://www.vm.ibm.com/perf/tips/prgpage.html

Hardware Operation - Storage

- At a certain level of paging, performance will begin to degrade due to queuing, and as it grows further, the degradation will tend to increase exponentially.
- Don't forget to monitor Virtual Storage usage to see how close you are to the limits!

Software operation

- **The more you understand about the software, the better you will be able to monitor and tune it, for example:**
  - **Consider z/VM and z/VSE are reporting a dasd page-in rate of 40 pages per second, so they both show the same impact don't they?**
  - **Actually, NO!**
  - **z/VSE is reporting 40 task interrupts per second at one page-in per request.**
  - **z/VM is reporting ? interrupts per second as it reads several pages per request as it can use block paging.**
- **You need to interpret the reported values in the context of the way the software works.**

z/VSE Multi-Processor Operation

§ Multiple CPU limit - QUERY TD:

```
AR 0015  CPU    STATUS     SPIN_TIME     NP_TIME TOTAL_TIME NP/TOT
AR 0015   00    ACTIVE         18852     6100562   13713442  0.444
AR 0015   01    ACTIVE         17906     5424786   12294566  0.441
AR 0015   02    ACTIVE         13405     4685876   10013585  0.467
AR 0015                  ----------------------------------------
AR 0015 TOTAL                  50163    16211224   36021593  0.450
AR 0015
AR 0015                NP/TOT: 0.450      SPIN/(SPIN+TOT): 0.001
AR 0015  OVERALL UTILIZATION: 120%         NP UTILIZATION:  54%
```

- § Collect the data over a representative period using this command or CPUMON.
- § NP task code cannot run on more than one cpu simultaneously and a single partition cannot run on more than one cpu simultaneously .
- § The NPS (Non-Parallel Share) "NP/TOT" says we can expect to exploit a maximum of about 2 of the 3 cpus (i.e. 200% total cpu) based on the z/VSE-provided formula 0.9/NPS:

  Max CPUs = 0.9 / 0.450 = 2

z/VSE Multi-Processor Operation

- NP usage cannot exceed 100%, we see 54% for this data sample.
- What affects NPS?
  - Normally, the relative use of key zero code and SVCs compared to application cpu.
  - If it decreases dramatically, much more application cpu is being used.
  - If it increases dramatically, more SVCs for z/VSE services or privileged code is being executed.
- I have used NPS to confirm that what other data was showing me was true.
- Use z/VSE cpu balancing to allow z/VSE to dynamically switch the cpus off when they are not required for the workload.
- In this 3-cpu case, z/VSE will dynamically switch cpus 02 and 01 off and on; the IPL cpu 00 will never be switched off.

z/VSE Paging

- **z/VSE does not page if the IPL option NOPDS is active.**
- **While page-in is active, the whole task has to wait (status x'86'), this may be CICS QR that runs almost all of the CICS work.**
- **z/VSE normally runs with PAGEX ON under z/VM:**
  - For normal page faults, z/VM causes a Program Check interrupt that allows z/VSE to place the task in a pseudo-page fault wait (status x'55').
  - Another Program Check occurs when the page fault is resolved, and z/VSE makes the task dispatchable (status X'83').
  - But the whole z/VSE virtual machine *can* enter a page wait state until the page fault is resolved.
  - z/VSE does not count pseudo-page faults.

Resource Usage Analysis

- CPUMON is an example of a monitor that will provide this.
- Usage analysis is a simplistic way to determine that you may have a problem based on the *amount of usage*, and can show that you have one in the right circumstances.
- You must make sure that you collect data that relates to the time when you are having problems.
- For example, it is no good looking at average usage over a a long time for a problem that lasts for a few minutes, the problem usage will be hidden in the smoothing of the data over time.
- You may need to know at what level of usage *you* get problems, there are some generally accepted values, but they may need to be adapted to your environment.

Resource Usage Analysis

- **For example:**
  - In an unconstrained environment, 60% or higher cpu utilisation in a single CICS partition can impact response times.
  - In a cpu-constrained environment you may not get that far, so you could be using less than 60% and have a problem, e.g. multiple z/VSE guest Virtual Machines contending for cpu time in an LPAR, which is itself contending for cpu time.
  - A Key Performance Indicator is the QR TCB Cpu:Dispatch ratio, which will reduce as the cpu becomes constrained, use DFH0STAT or DFHSTUP.
  - But there is no single value that is "good" (unless you are getting 70% or higher) as it is very workload-dependent; discover what it is for *every* CICS partition when response is good and use that as the baseline for *that* partition.
  - Reference material at the end of the presentation discusses this issue.

Resource Usage Analysis

§ CICS QR CPU:Dispatch ratio calculation:

- § This is only valid for the QR subtask (QR_SUBD), which does most of the work.
- § ((TCB CPU+DS TCB CPU)*100)/TCB DISPATCH TIME
- § (02:41*100)/03:26 = 78% **J J J**
- § DFHSTUP does not show DS TCB CPU.

Accumulated cpu time . . . : 03:55:36.99905 ç my change to DFH0STAT shows total partition cpu

. . .

TCB Name	TCB Status	TCB Start Time	Op. System Waits	Op. System Wait Time	TCB Dispatch Time	TCB CPU Time	DS TCB CPU Time
QR_SUBD	Active	08:20:14.88922	10,569,482	16:34:18.01804	03:26:00.58476	02:38:10.47016	00:02:46.96747
RO_SUBD	Active	08:20:15.25301	139,110	19:57:12.33731	00:03:06.26546	00:00:05.47575	00:00:00.46200
Totals					03:29:06.85023	02:38:15.94591	00:02:47.42948

Degradation Analysis

- Degradation analysis helps to explain why things did not run faster by showing the time spent in waits as well as using resources, this helps you to focus on where to apply the tuning effort.
- Simple degradation analysis can come from a monitor such as DFH0STAT, which shows MXT delays, VSAM string waits etc.
- You know that there is an impact from all of them, but perhaps not by how much, although it can identify what to tune.
- But not always, for example, increasing MXT to resolve MXT waits may not be the solution and can result in a worse problem such as SOS.
- OEM software normally provides good degradation analysis, with the amount and/or the percentage of the time spent in all states.

Do I use Resource Usage or Degradation Analysis?

- I often need to use both.

Exception Reporting

- Useful for day-to-day reporting to reduce the amount of output to what is a problem (Limit) or could be one soon (Threshold).
- For example, a limit condition is CICS MXT, and a threshold could be 80% of MXT.
- Enable console messages to alert operations for problems that are invisible or difficult to spot like CICS MXT, ALLOCATE waits etc.
- Report on transactions with a response time > n seconds, although you will need to filter out those that always behave that way (but shouldn't you be tuning them?).
- Make sure that you can still get at the detailed performance data when you get the exception report!

Trend Reporting

- Watch your system over time and keep the data.
- Placing the data in a simple Relational Database gives you a fast way to analyze it, keeping a lot of data as printout is generally of no use.
- Are things much the same, better or worse, and how quickly is it changing?

Understanding and Validating Monitor Output

- **Ensure that you understand the meaning of each data value.**
- **Documentation may be factually correct but completely meaningless.**
- **CICS/VSE and CICS TS are often different, for example, a SUSPEND in CICS TS is a normal wait, but this is not true for CICS/VSE.**
- **Try to find out how the values are obtained:**
  - Is a given value obtained by an event, which is normally accurate, or sampled and at what sampling rate?
  - If states are sampled only every second, how much can you trust the reported state% values for a problem that lasts for a short time?
  - If summary data is collected over a minimum interval of 15 minutes, how will that help you with a problem that lasts for a very short time?

Understanding and Validating Monitor Output

- Here is an example of a screenshot for task response time data, what does it tell us? See the next slides . . .

```
Dispatch          0.0159  % of Response Dispatched                  5.81
Wait              0.2598  % of Response Waiting                    94.28
Response          0.2757
CPU               0.0029  % of Dispatch Waiting for CPU            81.81
Wait DispQ        0.0043  % of Wait in Dispatch Queue               1.70


 Wait Type          Count       Tot Time       Avg Time    Percent of Response
WAIT 1ST DISP:          1         0.0044         0.0044                   1.64
FILE CONTROL:           3         0.0344         0.0114                  12.52
JOURNAL WAIT:           8         0.0498         0.0062                  18.11
TS AUX+MAIN:            5         0.0000         0.0000                    .00
```

Understanding and Validating Monitor Output

```
Dispatch          0.0159  % of Response Dispatched              5.81
Wait              0.2598  % of Response Waiting                94.28
Response          0.2757
CPU               0.0029  % of Dispatch Waiting for CPU        81.81  ç 5%
Wait DispQ        0.0043  % of Wait in Dispatch Queue           1.70
```

- The *internal* response time is 0.2757 seconds.
- The task was "active" ***5.81%*** of the time, and was in a CICS wait (CICS TS SUSPENDed state) ***94.28%*** of the time.
- It used **0.0029** seconds cpu.
- 0.0159 - 0.0029 = 0.0130 seconds was spent waiting for the cpu, which means that 0.0130/0.2757 = ***5%*** of the total response time is due to waiting for the cpu, this is *not* a CICS wait, an internal DFHDSSRM SUSPEND macro is used to enforce a CICS wait.

Understanding and Validating Monitor Output

Dispatch	0.0159	% of Response Dispatched	5.81
Wait	0.2598	% of Response Waiting	94.28
Response	0.2757		
CPU	0.0029	% of Dispatch Waiting for CPU	81.81
Wait DispQ	0.0043	% of Wait in Dispatch Queue	1.70

- Wait DispQ looks like it should be the time spent in waiting for the first dispatch, it appears to be a trivial amount.
- Once you believe that you understand the meaning of each value, watch out for anomalies and ask the Vendor about them.

Understanding and Validating Monitor Output

Wait Type	Count	Tot Time	Avg Time	Percent of Response
WAIT 1ST DISP:	1	0.0044	0.0044	1.64
FILE CONTROL:	3	0.0344	0.0114	12.52
JOURNAL WAIT:	8	0.0498	0.0062	18.11
TS AUX+MAIN:	5	0.0000	0.0000	.00

- The sum of wait times here = 0.0044 + 0.0344 + 0.0498 + 0 = 0.0886
- But the reported "Wait" is 0.2598 seconds!
- Using this data we can only account for ***34%*** of the reported wait.

Understanding and Validating Monitor Output

```
Task nnnnn Dispatched Elapsed:  0.0035998445 Start: =032960= 08:52:04.8073354377 End: =033355= 08:52:04.8109352822
Task nnnnn Inactive   Elapsed:  0.0067285000 ç this is the time in this wait state                                JCP
Task nnnnn Dispatched Elapsed:  0.0000938430 Start: =033544= 08:52:04.8176637822 End: =033549= 08:52:04.8177576252
Task nnnnn Inactive   Elapsed:  0.0007748125                                                                    JCP
Task nnnnn Dispatched Elapsed:  0.0014919375 Start: =033578= 08:52:04.8185324377 End: =033654= 08:52:04.8200243752
Task nnnnn Inactive   Elapsed:  0.0165056570                                                                    FCIOWAIT
Task nnnnn Dispatched Elapsed:  0.0021633437 Start: =034791= 08:52:04.8365300322 End: =034852= 08:52:04.8386933759
Task nnnnn Inactive   Elapsed:  0.0000047500                                                                    JCP
Task nnnnn Dispatched Elapsed:  0.0000055938 Start: =034854= 08:52:04.8386981259 End: =034861= 08:52:04.8387037197
Task nnnnn Inactive   Elapsed:  0.1712594062                                                                    ISC
Task nnnnn Dispatched Elapsed:  0.0035288750 Start: =042117= 08:52:05.0099631259 End: =042398= 08:52:05.0134920009
Task nnnnn Inactive   Elapsed:  0.0220010313                                                                    ISC
Task nnnnn Dispatched Elapsed:  0.0000633750 Start: =044536= 08:52:05.0354930322 End: =044541= 08:52:05.0355564072
Task nnnnn Inactive   Elapsed:  0.0304503117                                                                    JCP
Task nnnnn Dispatched Elapsed:  0.0000517188 Start: =046488= 08:52:05.0660067189 End: =046491= 08:52:05.0660584377
Task nnnnn Inactive   Elapsed:  0.0035750937                                                                    FCIOWAIT
Task nnnnn Dispatched Elapsed:  0.0005190313 Start: =046783= 08:52:05.0696335314 End: =046850= 08:52:05.0701525627
Task nnnnn Inactive   Elapsed:  0.0045506570                                                                    JCP
Task nnnnn Dispatched Elapsed:  0.0039169680 Start: =047211= 08:52:05.0747032197 End: =047726= 08:52:05.0786201877
Task nnnnn *** Total Dispatched 0.0154345306
Task nnnnn *** Total Inactive   0.2558502194
Task nnnnn *** Total Response   0.2712847500
```

- An analysis of AUXTRACE for that task shows the *actual* distribution of activity, with all of the CICS waits except the one for first dispatch.

Case Study 1

- The CICS application spawns a batch job to run a *very large* program to perform many calculations based on VSAM data.
- Response time is "acceptable" when there is a maximum of 3 concurrent jobs, after that they slow down exponentially; the customer needs to run 9 concurrent jobs with a response time of seconds.
- The OEM z/VSE monitor resource usage output shows cpu time, EXCPs, storage usage and elapsed time.
- z/VSE is set up correctly and is not delayed by z/VM.
- DLBL BUFNI and BUFND values to tune VSAM "mixed" processing.
- The jobs run faster, but not fast enough to meet the requirements, what should you do next?

Case Study 1

- I asked the customer to configure the OEM monitor for task wait states.
- Running one job showed that a noticeable percentage of time was in the SUP system task.
- The Supervisor DRM says that SUP is for "Program Fetch Processing", i.e. loading the program into storage.
- When two or more jobs run, the monitor now shows time waiting for *access* to the SUP system task, and this grows exponentially according to the number of jobs that run in parallel.
- The SUP system task can only handle one request at a time, it if is busy, other requestors in z/VSE must wait.
- Any suggestions for a solution?

Case Study 1

- **My solution was to copy the phase to a VDISK sublibrary.**
- **Why?**
  - Because the biggest inhibitor to performance was the time required to load the program, you need to make that faster, and making that faster then allows the other jobs to get their LOAD requests processed faster.
  - Remember the 80:20 rule when deciding what to tune.
- **9 jobs now run in parallel with an appropriate response time, and even a single job on its own runs significantly faster than before.**
- **I used resource usage analysis to see what could be tuned with some effect, but needed Degradation Analysis to identify and fix the problem.**

Case Study 2

- A native z/VSE system in the main LPAR of a CEC with one cpu has a single production CICS partition, which is experiencing poor response times for 10 to 15 minutes at certain times of the day.
- OEM CICS monitoring software is available but they are not licensed for the z/VSE component.
- However, the customer is not familiar with it and needs advice on what output is required.
- The customer is happy to use any of the free tools mentioned in this presentation.
- How would you approach resolving this problem?
- When we have discussed this, we can look at some sample data on the next few slides.

Case Study 2 Data

- **Typical DFH0STAT output:**

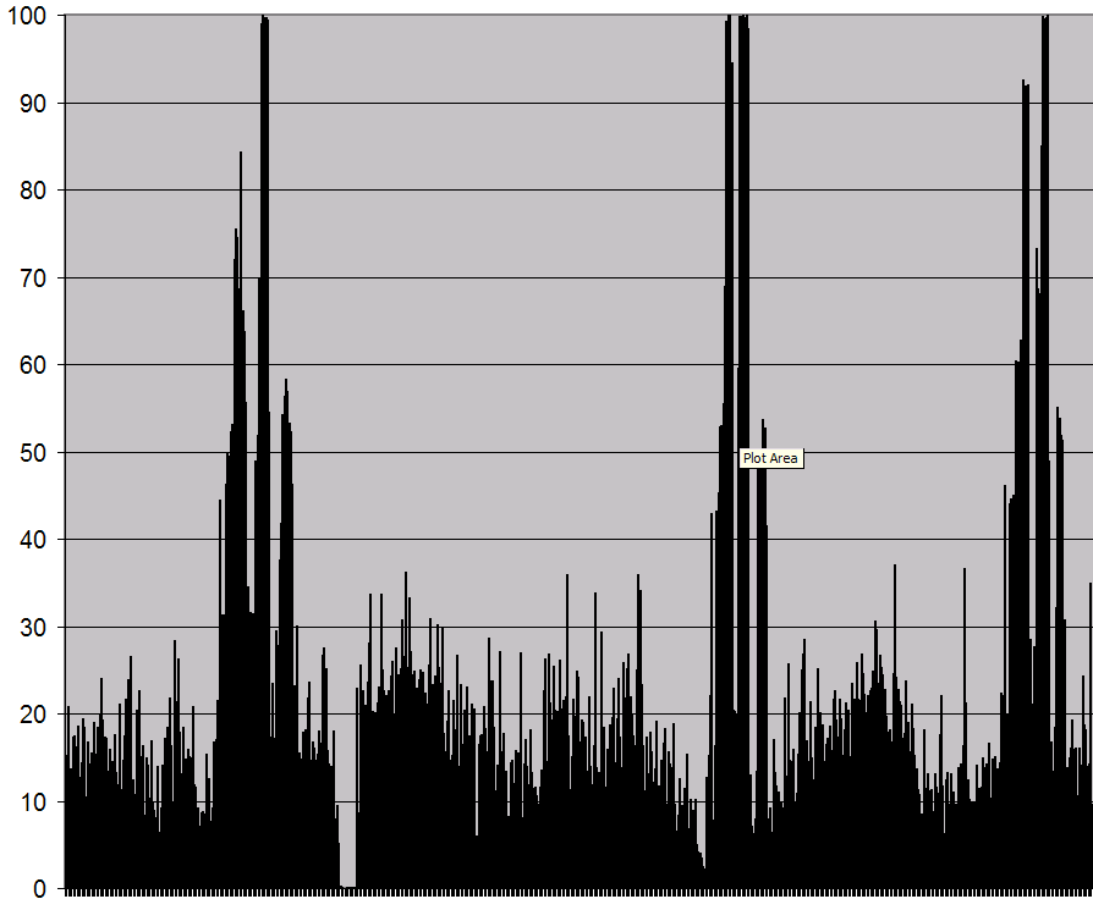
```
Maximum transactions allowed (MXT) . . . :              45
Times at MXT . . . . . . . . . . . . . . :             141
. . .
Transactions Delayed by MXT. . . . . . . :           1,362
Total MXT queueing time. . . . . . . . . :  03:42:13.88408
```

TCB Name	TCB Status	TCB Start Time	Op. System Waits	Op. System Wait Time	TCB Dispatch Time	TCB CPU Time	DS TCB CPU Time
QR_SUBD	Active	04:56:27.59985	34,177,906	10:05:14.15444	02:25:04.18305	00:48:11.26400	00:00:00.00000
RO_SUBD	Active	04:56:27.74450	3,469	12:30:11.71719	00:00:06.62030	00:00:00.00000	00:00:00.00000
Totals					02:25:10.80336	00:48:11.26400	00:00:00.00000

- **MXT issues, but QR Cpu:Dispatch is not high.**
- **But the data is for 12.5 hours!!!!!!**
- **Try looking at resource usage for CICS and other high PRTY partitions using MONPTN . . .**

Case Study 2 Data

- MONPTN cpu data for just the CICS partition shows the problem:



Case Study 2 Data

- An example of the OEM monitor data for a slow task duing one of the times of high cpu usage.
- Compare "TRAN CPU" to "TRAN LIFE"!!!!

TRAN ID	TRAN NUMBER	TRAN USE	TRAN LIFE	TRAN PGM	TRAN I/O	TRAN WAIT	TRAN SUSP	TRAN WTR	TRAN RSCS	TRAN EXPC	TRAN TMIO	TRAN CPU
Z001	0022345	1	115.4	0.055		115.3		0.121		0.002		0.043

TRAN ID	TRAN NUMBER	TRAN USE	TRAN LIFE	WAIT COUNT	WAIT RESP	WAIT TIME	WAIT MAX RESP	WAIT ID	WAIT COUNT	WAIT RESP	WAIT TIME	WAIT MAX RESP
Z001	0022345	1	115.4	64	1.803	115.3	115.1	USER	57	0.001	0.081	0.032
								KCP	7	16.47	115.3	115.1
								SUMMARY	32	1.803	57.69	115.1

Case Study 2 Data

§ Worst cases in an 11-minute data sample when response was bad:

TIME	ID	NUMBER	LIFE	CPU
11.24.01	XXXX	22901	408.7	244
11.26.45	XXXX	25689	89.01	28.62
11.23.16	X365	24418	76.2	20.68
11.23.36	X315	24458	80.43	19.98
11.30.08	TCP		304.2	19.33
11.20.10	Z001	22982	171.2	17.37
11.20.10	Z001	22975	171.4	17.31
11.20.09	Z001	22962	171.5	17.23
11.20.11	Z001	22919	176.8	17.17
11.20.10	X315	23027	135.1	17.16
11.22.56	X341	24453	40.53	16.61
11.22.56	X341	24454	40.5	16.6
11.22.49	X387	24455	33.7	16.59
11.22.49	X315	24457	33.69	16.59
11.22.56	X387	24466	20.56	16.54
11.22.56	X100	24464	20.63	16.53
11.22.56	X100	24465	20.63	16.53
11.23.54	Z001	24331	133.1	13.27
11.20.12	Z001	23051	136.6	9.877
11.20.11	Z001	23049	135.7	7.997
11.23.55	Z001	24143	147.4	7.077
11.23.55	Z001	24319	134.4	5.737
11.23.55	Z001	24243	138.1	5.695
11.25.04	TCP		303.7	5.666
11.23.55	X315	24438	105.4	5.644
11.23.55	X315	24417	115.3	5.639
11.23.54	Z001	24322	133	5.578
11.23.54	Z001	24367	129.1	5.569
11.23.55	X315	24494	39.88	5.539
11.23.26	X414	24499	9.944	5.452
11.23.26	X349	24502	8.153	5.449

Case Study 2 Data

- Transaction "XXXX" is consuming huge amounts of cpu time.
- Either tune or buy a faster cpu - do NOT buy more cpus, that will NOT make CICS run faster as it can only exploit one cpu.

Case Study 3

- "Multiple z/VSE guests under z/VM have begun to experience CICS performance problems, and this includes many task time-outs".
- This is all the description you are going to get, it is time to pretend you are CICS L3 with a new PMR to look at with not a lot of background.
- No OEM monitoring software for z/VSE and CICS.
- What do you need to know and how would you approach this problem?
- If we don't have time now, consider going away and thinking about it and then come back to me with your ideas.

Any Questions ?

Reference Material

- Otherwise known as "outtakes".

Free Tools

§ SIR SMF

- § Useful for an analysis of general dasd I/O response times based on hardware and software measurement.
- § It is documented in the z/VSE Hints and Tips PDF.
- § SIR SMF=ON
- § SIR SMF . . . to display on SYSLOG
- § SIR SMF=OFF
- § Unless "VSE" is used, the I/O counts wrap to zero every 65,536 requests!
- § I am not sure about the accuracy for z/VM minidisks that are not a full volume.

Free Tools

§ SIR SMF

- § "QUEUED" is the time in z/VSE and in the I/O subsystem.
- § Connect is data transfer.
- § Disconnect is anything else, and is dependent on cache performance.

```
SIR SMF
AR 0015 DEVICE  I/O-CNT     QUEUED      CONNECT     DISCONN     TOTAL
AR 0015                   MSEC/SSCH   MSEC/SSCH   MSEC/SSCH   MSEC/SSCH
AR 0015
AR 0015  120          691       0.114       0.098       0.000       0.213
AR 0015  121            4       0.170       0.170       0.000       0.341
AR 0015  122           23       0.111       0.133       0.000       0.244
AR 0015  326            3       0.085       0.170       0.000       0.256
AR 0015  328          328       0.116       0.135       0.000       0.252
AR 0015  32B        46386       0.119       0.347       0.000       0.466
AR 0015  331        14101       0.275       0.124       0.000       0.400
AR 0015 1I40I  READY
```

Free Tools

§ SIR SMF

- § "QUEUED" is the time in z/VSE, "PENDING" is in the I/O subsystem.

```
sir smf=331
AR 0015 TIMINGS FOR 331 BASED ON      14114 I/O INSTRUCTION
AR 0015
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       0.162        0.113        0.124        0.000        0.000        0.400
AR 0015 1I40I   READY
```

- § z/VSE's perspective.

```
sir smf,vse,331
AR 0015 TIMINGS FOR 331 BASED ON     276194 I/O INSTRUCTION
AR 0015 MAXIMUM I/O QUEUE   3
AR 0015
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       0.162        0.000        0.304        0.000        0.000        0.467
AR 0015 1I40I   READY
```

z/VSE I/O Request

- A simple asynchronous I/O request in Assembler:

```
         EXCP    CCB1     Start request
+        SVC     0
         WAIT    CCB1     Wait for I/O completion and the ECB being posted
+        SVC     7
CCB1     CCB     SYSLOG,CCW1      I/O request block, which contains an ECB
CCW1     CCW     9,DATA,0,12      Command code, buffer address, flags, count
DATA     DC      CL12 'Hello World!'
```

- z/VSE system tasks use SYSIO SVC 15, which has the highest priority.
- WAITM SVC 29/X'1D' is used to wait for the first of n ECBs to be posted, CICS normally uses z/OS WAIT SVC 132/X'84' code 01.

z/VSE Dasd I/O

- **The elapsed time of a simple application I/O is a sum of:**

1. Processing the request before the Access Method can issue EXCP.
2. z/VSE processing the EXCP request, queuing it to the device and waiting until it is at the top of the queue so that z/VSE can execute the SSCH instruction to start the I/O.
3. Processing and queuing in z/VM for the real SSCH.
4. The SSCH operation.
5. z/VSE handling the I/O interrupt and posting the CCB (or IORB) used by the EXCP.
6. Waiting for the task that requested the I/O to be re-dispatched, this can be a significant delay when CICS is used.

- **Watch out for VSAM exclusive control errors, which can result in CICS *and* VSAM code being repeated several times, not only is there delay, but also more cpu time is used, e.g. output ESDS in CICS should have STRNO=1.**

z/VSE Task Management Internals

- Once z/VSE gives control of a cpu to a task, and except for interrupt activity on that cpu, it will effectively retain control of a cpu until it goes into a wait state, it abends or a higher priority task becomes dispatchable and no other cpu is available.
- z/VSE assigns each task a dispatchability status, which is externalised in the STATUS command.
- The PCB and PCE control blocks have information about the partition.
- The TIB, TCB and TCBX control blocks have task information.
- Although they are not "official" interfaces, the formats have been reasonably consistent for many years, they have DSECT mapping macros in PRD2.GEN1, and can be used (with care) if you have no other way to get information that you need.

z/VSE Task Management Internals - Partition

- Each partition has a PIK value, e.g. BG is always X'0010'.
- Each task has a Task ID, which may be 1 or 2-bytes long, e.g. BG is always X'0021' (you see a Task ID in STATUS and in the console message reply id).
- Before 4.2, the PCB always has 1-byte task ids in field TIDSTR, otherwise GETFLD is used to provide 2-byte task ids.
- GETFLD returns z/VSE information, and may require a PIK value; you can use GETFLD to convert a partition id such as BG to the PIK.
- The PCB has a z/VSE page-in count in PCBPICNT.
- The PCB has partition and overhead cpu in PCPUTIME and POVHTIME, but you need to use GETJA PART= to cause z/VSE to update the values before you access them in the PCB, or use GETFLD FIELD=CPUTIME.

z/VSE Task Management Internals - Task

- The TCB has useful information about the last task interrupt, and the address of the TIB; GETFLD requires the task id to find the TCB.
- The TIB has the task status in TIBRQID, X'83' or 'X'5C' means 'READY TO RUN', i.e. the task is eligible for dispatch, anything else means a wait state.
- Use the IBM z/VSE Central Functions Supervisor Calls and Internal Macros manual to understand macros such as GETFLD.

CICS CPU Time Accuracy

- CICS uses an emulated z/OS STIMER macro to capture cpu time.
- z/VSE records TASK (CICS and user) instruction processing but does not capture much of the z/VSE overhead, which can be a significant percentage depending on the amount of SVC activity compared to pure instruction execution in CICS.
- So the reported cpu time is subject to a degree of inaccuracy, this will be true for any product that relies on CICS-maintained cpu times.
- CICS statistics only reports CICS Dispatcher-managed TCB cpu time (emulated z/OS subtasks for QR, SO etc.), it does not report on z/VSE subtasks such as DFHIRPST that are not managed by the dispatcher, nor cpu time for OEM z/VSE subtasks.

CICS CPU Time Accuracy

- I have a modified version of DFH0STAT and DFH$STAS that captures partition cpu time and shows it as "Accumulated cpu time".
- Use this or z/VSE performance data at the partition level to work out the capture ratio to see what the real cpu cost is.
- Ensure that your z/VSE monitor includes overhead and partition time.
- The capture ratio could be something like only 60%.
- Keep tracking it over time as it may change, this is could be important for capacity planning.
- Remember to factor in the overhead of higher levels of software and hardware such as z/VM and LPAR management.