IBM

What Mother Never Told You About CICS Performance

February 2018 - poilmike@uk.ibm.com

z/VSE 6.2

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.

* 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.

References

- CICS TS for VSE/ESA Performance Guide.

- CICS TS for VSE/ESA Problem Determination Guide.

- CICS TS for VSE/ESA Shared Data Tables Guide.

- z/VSE LVC July 2014 An introduction to tuning VSAM file performance under CICS TS in z/VSE.

- Other z/VSE Presentations:

  https://www-03.ibm.com/systems/z/os/zvse/documentation/presentations.html

Abstract

This presentation is based on what I have learnt while working in CICS Level 3 Service at Hursley as a result of my own performance evaluations, resolving customer performance PMRs and working onsite with customers on performance issues.

While some of the information can be found in previous LVC and WAVV presentations, it has been brought up-to-date and enhanced, but there is some duplication where it felt it was useful in context. It may add to or correct information in the CICS TS for VSE/ESA Performance Guide, and possibly what you see on CICS TS for z/VSE performance the Internet.

Real world examples and my own performance evaluation results are included. Because all CICS systems are not the same, you may see different results. YMMV.

I am still learning!

With apologies to Melinda Varian for borrowing part of the title of one of the best VM Technical Support articles that I ever read.

Agenda

- Just because it is called "CICS" doesn't mean that you can use CICS TS for z/OS tuning recommendations

- z/VM - the good, the bad and the ugly

- z/VSE is not perfect either

- Performance monitor output is always correct - right?

- Response time analysis

- Using DMF for performance data

- The cost of CICS monitoring

- How I measure cpu utilisation

- Using one cpu versus multiple cpus

- The potential impact of CICS being too busy or having internal constraints

Agenda

- Using DFH0STAT

- How to interpret the QR Cpu:Dispatch ratio on z/VSE

- VSAM Lookaside

- VSAM hints and tips

- The cost of function shipping

- A significant MRO limit

- Should you only use main temporary storage?

- The cost of dumps and extrapartition datasets

- AOB

- Questions

Just because it is called "CICS" doesn't mean that you can use CICS TS for z/OS tuning recommendations

- There is a lot of knowledge out there about CICS performance, but much of it is for z/OS.

- CICS TS for z/OS is a very different product that runs under a very different Operating System and both have many more performance features that can be exploited.

- z/VSE products can be similar, but often do not work and hence cannot be tuned in the same way as they are in z/OS, for example, VSAM can be very different.

- CICS TS for z/OS can exploit multiple cpus and cpu types simultaneously for running transactions, but z/VSE does *not* allow a single CICS to exploit more than one cpu even though the CICS code could.

z/VM - the good, the bad and the ugly

- Using z/VM has advantages, for example, the use of Minidisk Cache to reduce dasd I/O service times, and the ability to run z/VSE with z/Linux in the same LPAR.

- Using z/VM has a cpu cost, which I have seen to be about a 10% cpu delta (i.e. 50% within z/VSE is about 55% actual), and I have known z/VM bugs to cause problems at times.

- Using the default Vertical Polarization Mode, AKA HiperDispatch, *might* cause a problem:

  - A customer that uses two Virtual cpus migrated to z/VM 6.3 and the CICS QR Cpu:Dispatch ratio (see later) reduced noticeably - a potential sign of a cpu constraint.

  - When the LPARs were dynamically changed to run in Horizontal Polarization Mode (i.e. no HiperDispatch), the QR Cpu:Dispatch ratio was very similar to z/VM 6.2.

  - Refer to z/VM documentation for details on z/VM HiperDispatch.

  - z/VSE does not understand HiperDispatch.

  - Be aware that z/VSE does not appear to have been included in any of the official z/VM performance evaluations for some time.

z/VSE is not perfect either

- z/VSE has limitations in terms of what it its design can handle, one of them being outright processor capacity.

- You do not have the comprehensive selection of performance monitors that are available on z/OS, and that can have significant implications for any type of performance analysis.

Performance monitor output is always correct - right?

- *I hope that you don't believe that!*

- The problem is this - how do you know what is accurate and what is not?

- *And equally importantly, do you know how to correctly interpret the data produced by the monitor?*

- Here are some issues that I know about in CICS Statistics and Monitoring data.

- CICS QR TCB cpu time - the QR z/VSE subtask (the 4th task from the top of a STATUS xx command output and the second DFHEVID1) handles almost all of the CICS instruction processing, and the CICS tasks hop on and off other CICS-owned z/VSE subtasks when performing activities that would block QR, for example, loading a program or performing TCPIPSERVICE-related Socket I/O.

  - The API used does not capture all of the cpu utilisation.

  - The more z/VSE activity that is required by CICS, the less accurate it seems to become.

  - You can compare the QR TCB cpu against Partition cpu to get better idea of the real cost.

Performance monitor output is always correct - right?

- VSAM I/O wait reported by CICS Monitoring (FCIOWTT) is the time that the task was waiting for normal and split I/O completion (FCIOWAIT), plus the time waiting for another task's CI split to complete (FCCIWAIT), but it does **not** include other wait states that can contribute significantly to it, for example:

  - FCXCWAIT due to waiting for an Exclusive Control Conflict; this can be a substantial value due to the number of retries that can be made by CICS before VSAM accepts the request; this is not a bug.

  - FCPSWAIT due to waiting for a FILE string, which can be log; this is not a bug.

  - The above are "invisible" by being accounted for in the overall Suspend time.

- Where does your CICS monitor get the data from?

- Does it use CICS to provide some of it? In which case it can suffer from the same flaws.

- Or does it do all of it itself? In which case you are reliant on it being correct unless you have a way to independently verify it.

Response Time Analysis

- *If you don't have a way to look in detail at transaction response times, you are at a disadvantage in terms of performance analysis and tuning; there is scope for general tuning without it, but you won't really be able to see how effective the tuning has been, nor where the real pain points are and when during the day.*

- CICS internal response times are built from the following components:

  - Dispatch time - the elapsed time that CICS was running the task's code, and should be accurate in CICS Statistics and Monitor data; cpu time is a subset as a result of CICS losing control to higher priority work and/or cpu time being reported as lower than actual.

  - Suspend time - the elapsed time when it is waiting to be dispatched; it is the result of a number of wait states such as FCIOWAIT, MXT wait, Dispatcher selection wait etc. which you may or may not be able to tune (reduce).

- *It is probable that you cannot reasonably account for the wait activity correctly under all circumstances when using **any** available CICS monitor!*

- I have used Internal CICS trace to correctly account for wait times using an internal analysis tool, and I have needed to use AR DUMPs and filter out bad trace data.

- Using Auxtrace skews trace timestamps badly, and can hurt response times in a production CICS system enough for a customer to say that it mustn't be used. (5x slower!?)

Response Time Analysis

- Response time analysis for all transactions writing to a KSDS log file when running a resource hog that issues a huge number of READNEXT for a different file.

- Required a very large Internal trace table with only EI=1 and DS=1 active and an AR DUMP.

- When the resource hog was behaving reasonably, it showed this for the log file:

  - Total FCIOWAIT   0.19 seconds and counted as FCIOWTT time from 172 waits
  - Total FCXCWAIT   0.02   SUSPEND time from 12 waits
  - Total FCCIWAIT   0.00   FCIOWTT time from 1 wait
  - *The 162 WRITE (Add) requests took 0.2 seconds elapsed or an average of 0.001.*

When the resource hog was not behaving reasonably:

  - Total FCIOWAIT   4.96 seconds and counted as FCIOWTT time from 2,377 waits
  - Total FCXCWAIT   10.08   SUSPEND time from 1,371 waits
  - Total FCCIWAIT   1.12   FCIOWTT time from 101 waits
  - *The 205 WRITE (Add) requests took 16 seconds elapsed or an average of 0.080!*

- A lot of CA splits occurred according to the number of FCIOWAITs!

Response Time Analysis

- I have seen the non-specific USERWAIT state be the main reason for response time issues, which could be TCP/IP Socket I/O wait when the product performs its own Socket I/O (e.g. MQ/VSE), but you may need to ask whoever owns the product.

- The CICS Problem Determination Guide has details of the types of wait.

- The CICS Customisation Guide has details of the CICS Monitoring counters such as FCIOWTT, but don't expect it to always tell you in detail what CICS includes in them!

- *If it appears that a CICS performance monitor can't help you with all CICS performance problems - welcome to my world!*

Using DMF for Task Performance Data

- If you don't have a Vendor CICS monitor, you can potentially get useful task-level data by enabling CICS Monitoring and collecting it with DMF - it is free software!

- It contains simple counts such the total number of File Control requests, but also contains task cpu time, with elapsed time for task dispatch and waits for events such as VSAM file I/O, and for these values it tells you how many times it waited, which can be *very* useful.

- This is by no means a perfect solution as it may not have all of the detailed data that you can get with Vendor products, but I have used it to solve customer performance problems.

- Rather than use DFH$MOLS to produce huge amounts of printed data, try my DFH$MCSV Assembler code, which converts the task performance data records into a sequential file with one CSV format record per task containing character data.

- You get access to *all* of the standard data variables in a single line (but not additional values added by a DFHMCT, for which you will need to modify DFH$MCSV), and being CSV format it can be processed easily in a spreadsheets etc.

- *Report Writer and other printed output is sometimes just not helpful, and can need a huge amount of time to analyse, with the potential of making mistakes from manual calculations*.

The Cost of CICS Monitoring

- If you or the Vendor monitor use CICS Monitoring, it could add a 10% or higher cpu delta to CICS, plus cpu time for possibly a lot of I/O in the data collection partition; the DMF overhead should be quite low.

- *Never run data collection at a lower PRTY than CICS!*

- If the CICS monitor does everything itself, I can't help you with the overhead.

- *A global monitoring interface in its own partition can sometimes be a performance killer when used inappropriately!*

- **A thought** - how much better would CICS run and how much more cpu capacity would you have if you were not monitoring it? I actually advised one customer to do that it they ran out of cpu capacity during the yearly peak time!

How I Measure Cpu Utilisation

- I have a monitor called PTNMON, and like CPUMON, it reports every "n" seconds to SYSLST in CSV format, but it has no XML output format option (yet) and hence you can't use the data for the IBM Capacity Planning tool.

- It contains useful z/VSE-level data from the same source as CPUMON, but adds:

    - Native z/VSE LPAR cpu (LPARCPU%) or z/VM Total Time (VMCPU%), which includes the z/VM CP overhead that shows the true cost of running z/VSE.

    - Cpu for every LPAR (CPCCPU%) or the z/VM LPAR cpu utilisation including ICFs and IFLs (CPCCPU% or LPARCPU% depending on the version being used).

- The following for up to 15 partitions:

    - DISP% - the percentage of samples that showed ready-to-run by z/VSE.

    - NP% - the percentage of samples when NP code was in use (see later).

    - CPU% - the partition's cpu time as returned by the z/VSE GETFLD macro.

- It has a *very* low cpu and storage overhead but is able to sample task states at up to 300 times per second in order to get good sampled data values, but it **MUST** run at or close to the highest PRTY to get accurate sampled values, typically *above* POWER.

How I Measure Cpu Utilisation

- The number of samples in an interval after the first can act as an indicator of a z/VSE-level cpu constraint (the first is short in order to get to the requested interval boundary).
- They should be reasonably consistent or z/VSE *might* be having a cpu constraint; the maximum samples below is 6,000 per interval, and you can see a cpu constraint.

| SAMPL	VSECPU	VMCPU	CPCCPU	SVCK/S	F2DISF	F2NF	F2CPU
5912	9.03	6.73	39.87	9.5	6.92	0.52	7.52
5907	14.05	10.38	47.88	13.7	11.83	0.59	12.16
5916	9.76	7.16	42.05	10.4	7.76	0.61	8.1
5918	11.54	8.65	47.98	12.4	8.87	0.64	9.51
5922	9.74	7.32	42.92	10	7.62	0.44	8.12
5916	10.74	8.11	47.26	11	8.86	0.52	8.93
5884	15.73	11.76	67.98	17.2	13.63	0.75	12.82
5915	12.01	8.84	50.07	12.2	9.97	0.63	10.18
5915	13.47	10.05	52.38	14.3	10.57	0.63	11.02
4573	39.2	43.7	158.47	47.7	25.1	1.33	11.58
3827	30.04	33.8	196.85	34.9	34.57	1.67	9.03
4732	22.76	22.95	196.89	25.2	28.7	1.1	10.57
4750	13.91	10.38	196.99	13.6	31.28	1.05	11.69
5835	34.31	31.22	119.25	26.1	15.01	0.94	13.26
5913	24.24	23.88	64	30.3	11.8	0.81	11.94
5920	23.39	23.17	69.79	29.5	11.22	0.71	11.23

How I Measure Cpu Utilisation

- On the customer system with the HiperDispatch issue, the number of samples varied a lot between about 130 and 149 even though z/VM had an entitlement of two real cpus; after HiperDispatch was switched off the number settled to 149 or even the maximum of 150.

- *Warning - capping can have a big impact on the number of PTNMON samples!*

- *DISP%, and often NP%, grow relative to CPU% when CICS is delayed, the bigger the difference, they greater the potential impact, and CPU% may drop at the same time.*

- *When there is a cpu constraint, DISP% does **not** tell you how much cpu CICS would have used had there been no constraint.*

- You can handle an amount of cpu constraint before it causes problems, how much you can handle is dependent on what you decide are acceptable response times, however, as delays increase, the impact on response times can become exponential.

- Task level performance data will show dispatch time, cpu time and dispatch delays so that you can measure the impact - PTNMON just shows that there *could* be a problem.

Using One Cpu versus Multiple Cpus

- It is generally known that z/VSE works most efficiently with one cpu, and I would not normally advise using more than two.

- If you need more than one cpu, you have no choice, for example, one customer needed the MIPS of two large cpus because there was and is no single cpu that was fast enough.

- The down sides of using one cpu are:

  - The possibility of a high PRTY loop causing a complete lockout in CICS - use PRTY SHARE intelligently.

  - Having more than a small number of busy production CICS partitions increases the possibility of cpu contention due to the number of partitions that may want to be dispatched simultaneously.

- The next slide shows one of a number of busy CICS partitions on a z/VSE system with one cpu that is suffering a cpu constraint due to other CICS at a higher PRTY SHARE. Orange is CPU% and the blue is DISP%.

Using One Cpu versus Multiple Cpus
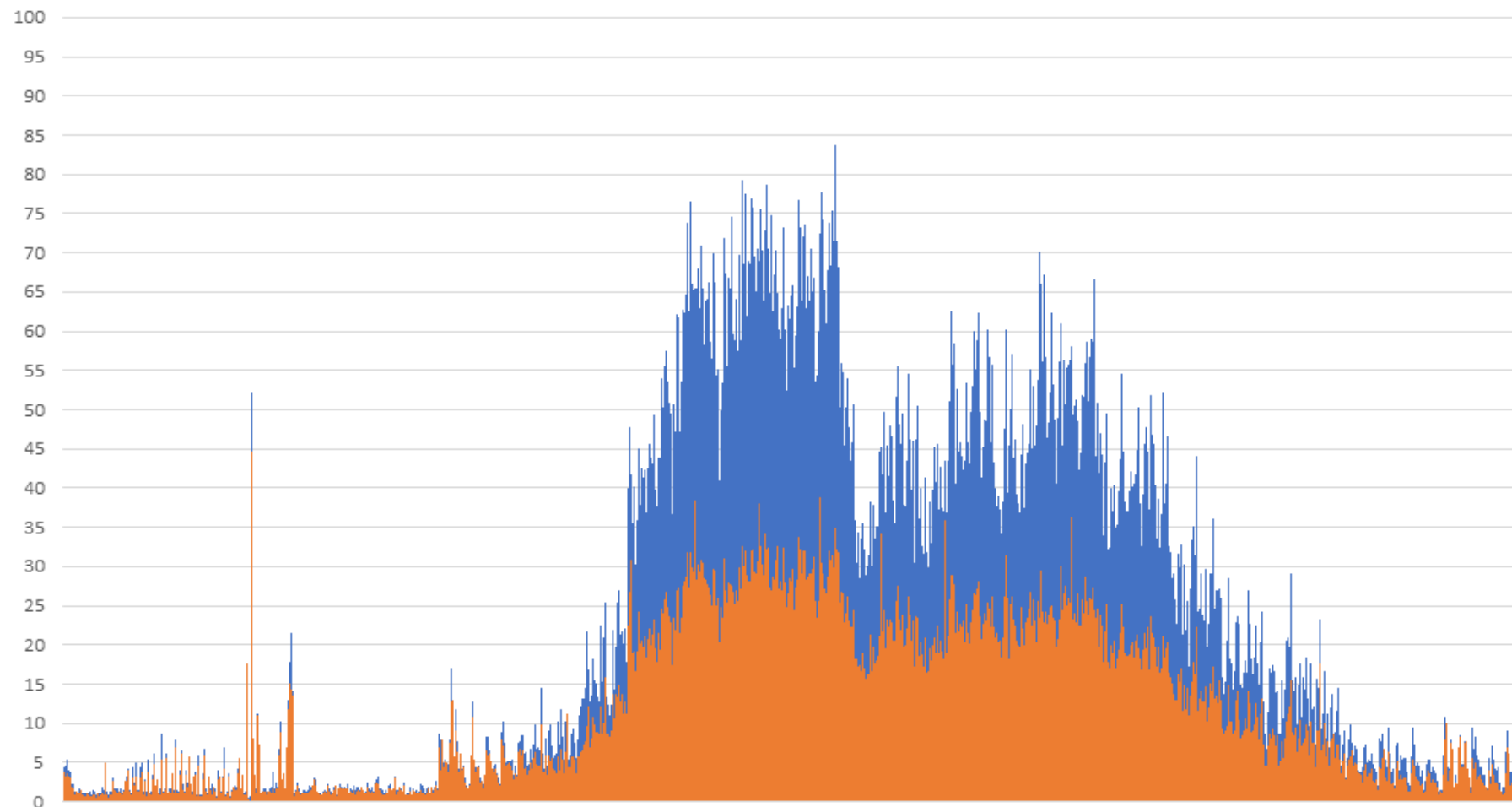
Using One Cpu versus Multiple Cpus

- *A VM/VSE customer used two cpus unnecessarily, and we measured a 10% overall cpu reduction when using one, but the average dispatch time per CICS task reduced by 30%!*

- Dynamic Cpu Balancing may or may not help in all circumstances.

- z/VSE needs to serialise certain types of processing; the ones that I know about are SVCs and using Key zero, and it labels that task's code path as Non-Parallel (NP).

- *Only one partition needing to run NP code can be dispatched at a time no matter how many cpus are available, limiting the potential to exploit more than one cpu.*

- PTNMON estimates NP% by partition, and QUERY TD shows z/VSE-level NP%.

- I have seen z/VSE-level NP% values from 15% (amazing!) to 70% (really bad!).

- The 70% was when using only one cpu - attempting to use two cpus in this case would not be advisable, and the maximum possible cpu capacity for z/VSE will be limited.

The Potential Impact of CICS Being Too Busy

- *I have seen response time, VSAM, SOS and even MRO issues when a z/VSE system with a single CICS system exceeds 80% to 90% dispatchable despite having excellent cpu availability.*

- If you get a period of time where CICS is much busier than normal, it may take a while to recover, and you can get the same effect if there is something like an AICA abend, or . . .

The Potential Impact of CICS Internal Constraints

- What happens in z/VSE when CICS partitions are hit by a significant MXT problem and then recover, with CICS in catch-up mode causing a cpu spike.

LPARCPU%

© 2018 IBM Corporation
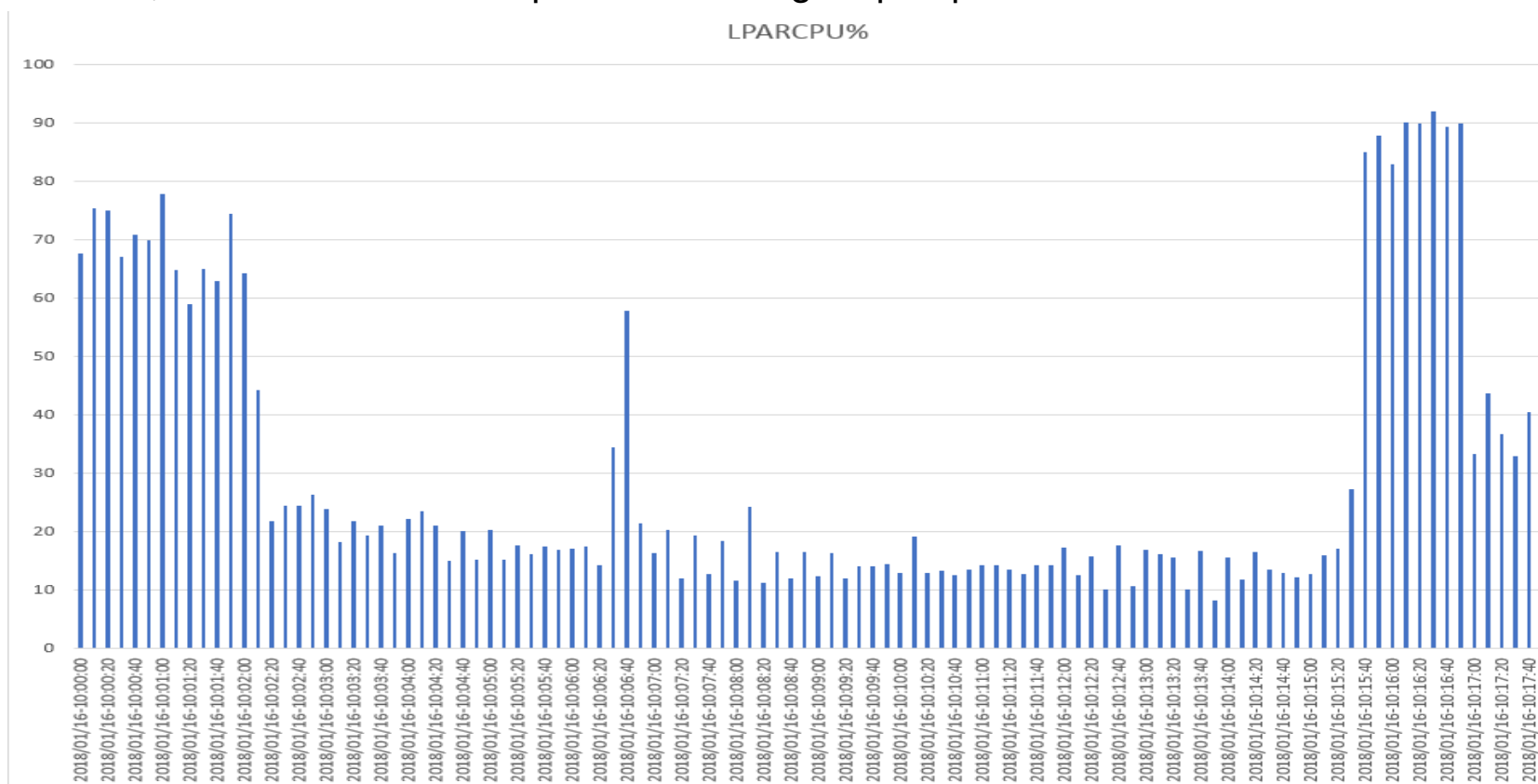
Using DFH0STAT

- I have an enhanced version of DFH0STAT that produces almost the same output as DFHSTUP, however, it includes data that DFHSTUP does not and cannot report on.

- Where you see "enhanced DFH0STAT", DFHSTUP output might provide the same data.

- The new data, some of which is already in the DFH0STAT supplied with CICS TS for z/VSE 2.1 and 2.2 in ICCF Library 59, is as follows:

  - Transaction Classes.

  - VSAM file: LSRPOOL buffer sizes, DEFINE CISZ and SHAREOPTIONS, CI and CA splits since OPEN, LSRPOOL buffer waits, plus a counts of VSAM Exclusive Control Conflicts if you install a fix to CICS.

  - System and Transaction Dump totals, and counts by dump code.

  - ISC/IRC, including current and HWM MRO SVA transfer buffer usage (see later).

  - Terminals.

  - VTAM.

  - Journaling.

Using DFH0STAT

- You can get useful performance and profiling information if you post-process the output:

```
DISPATCHER cpu      time per transaction 0.002236 seconds based on TCB cpu time
DISPATCHER cpu      time per transaction 0.005539 seconds based on accumulated partition cpu time
DISPATCHER dispatch time per transaction 0.006274 seconds


LSRPOOL  1 data  hit ratio  54%
LSRPOOL  1 index hit ratio  64%
LSRPOOL  1 is responsible for  44.10% of all LSR Read EXCPs and  49.25% of all LSR activity


LSRPOOL  2 data  hit ratio  82%
LSRPOOL  2 index hit ratio  98%
LSRPOOL  2 is responsible for   6.19% of all LSR Read EXCPs and  20.16% of all LSR activity


FILE FILEF   Number of CA splits since OPEN        178
FILE FILEG   Number of Exclusive Control Conflicts      88689


LSRPOOL  2 KSDSP FILEF
EXCPs          2789435  Data EXCPs    2671721 Index EXCPs     117714 EXCPs/second    42.22
Index/Data EXCP ratio 0.04  EXCPs/Request    0.24  EXCPs/task       2.40
File requests  11470635    Requests/task    9.88  Read/Write ratio  9.74
Read Update       669741    5.84%  Browse   9732507   84.85%
Rewrite           669741    5.84%  Add       398646    3.48%  Delete    0    0.00%


LSR file map  LSRPOOL  1 Index buffer size  2048 File FILEA   EXCPs    9335580
                                                 File FILEB   EXCPs       6086
                                                 File FILEC   EXCPs       4968
                                                 File FILED   EXCPs        347
                                                 File FILEE   EXCPs          5
```

Using DFH0STAT

- With CICS Statistics, some data values are subject to regular Interval and Midnight resets, and DFH0STAT loses data when a file or an LSRPOOL is closed etc.

- Use SIT STATRCD=OFF to avoid Interval resets, but that does not stop the Midnight reset, which can be delayed with a PLTPI program to change the time.

- *Run DFH0STAT at a time when you have as much information as possible in storage.*

- You might run it after CICS activity has peaked and again later to get total task activity.

- In one case where CICS was 24x7, the customer used CEMT P STAT RESETNOW at 0900 and run DFH0STAT at 1800 so that the data could be used to profile the busiest time.

- If you can handle the amount of output, you might find it useful to perform regular resets and run DFH0STAT just before each one to see more accurately when issues occur.

- When you have DMF active, the Interval/Midnight and USS data records from events such as a file being closed are sent to DMF and can be used by the DFHSTUP Summary report, and avoid data loss; the Summary report will lose a certain amount of detail though.

How to Interpret the QR Cpu:Dispatch Ratio on z/VSE

- The QR TCB Cpu:Dispatch Ratio is supposed to show you if there is a cpu constraint that affects CICS, however, the TCB cpu time is not accurate on z/VSE as we discussed earlier.

- QR TCB Cpu:Dispatch Ratio = (TCB Cpu/TCB Dispatch).

- With the data below, it is (00:00:50.96825+0:00:02.32522)/00:01:59.57785 - 0.45 or 45%.

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	05:00:27.09690	158,563	07:02:49.69133	00:01:59.57785	00:00:50.96825	00:00:02.32522

- For z/OS, the recommendation is at least 80% or you have a cpu constraint, but I have only seen this only a few times on z/VSE, and was because the applications had high cpu usage relative to the use of z/VSE services such as I/O; this could indicate either cpu-intensive or very efficient applications!

- Once I am happy with it, I watch for it reducing because that can be a sign of a cpu constraint, although some reduction on a day-by-day basis is to be expected.

VSAM Lookaside

- *Lookaside is where VSAM-generated **read requests** are satisfied from data that is cached in buffers, which could be reads that are required for "write" operations.*

- It can make a huge difference in the time that is required to process an EXEC CICS request.

- VSAM searches buffers of the appropriate size to see if it can find the CI, and if it does, it will return the record immediately and avoid a costly EXCP (a z/VSE I/O operation request).

- *In theory, more buffers = more Lookaside = better performance.*

- However, VSAM Lookaside does not appear to work consistently from my observations:

    - NSR is mostly less effective than LSR.

    - By necessity, SHR(4) performs very little Lookaside.

    - The AIX processing used by VSAM to access the Base Cluster seems to have very limited Lookaside, and hence an AIX can be costly to use in terms of EXCPs.

- Lookaside is normally more effective on the Index component.

- Browse is less likely to get good Data component Lookaside than random access.

- **YMMV**.

VSAM Lookaside

- CSD LSRPOOL resource definitions with Index **and** Data buffer counts normally works best.

- Continually increasing buffer counts may not improve Lookaside by a worthwhile amount.

- Allocating more than about 500 to 1,000 buffers of the same size may start to become expensive in terms of cpu time for buffer scanning.

- Lookaside % = (Lookasides * 100)/(Lookasides + Reads)

- *Your Vendor monitor may calculate it differently.*

- Aim for 90%+ for the Index.

- 80% would be good for the Data, but may be completely unrealistic in practice.

- I calculate Lookaside for the LSRPOOL as a whole and for individual buffer sizes.

- *Just because Lookaside is working well does **not** mean that there is no opportunity for tuning the busiest files in the LSRPOOL.*

VSAM Hints and Tips

- Consider not using VSAM Compression unless you need to - VSAM uses more cpu time per request and I have been seen a measureable difference on customer systems.

- A large Data CISZ can:

  - Reduce the impact of CA splits.

  - Improve Browse performance.

  - Cause more Exclusive Control conflicts and produce worse performance.

- *Sometimes you have to trade worse performance in one area for better performance in another, but more important, area.*

- For a 3390, a Data CISZ(2048) CA split requires about 800 EXCPs and possibly more than 0.5 second elapsed, whereas 16K requires about 130 EXCPs and less than 0.1 second; all other CICS requests for the file are blocked until the split completes.

- The number of Exclusive Control conflicts for a file is shown by the enhanced DFH0STAT when you also apply a fix to CICS.

VSAM Hints and Tips

- Don't use FILE STRINGS(1) to reduce Exclusive Control FCXCWAIT time, because the re-dispatch delay (FCPSWAIT) can be worse than the effect of FCXCWAIT; this is not a bug.

- FILE string waits are reported in the "Files" section of DFH0STAT or DFHSTUP, and are caused by CICS limiting access to the file and *not* VSAM.

- Avoid LSR string waits, which are reported on in the "LSR" section of the reports.

- Don't define a small number of LSR buffers as that can cause VSAM X'90' errors and AEIU abends, or even pass DUPREC back to the program when VSAM has an internal problem with buffers, or you might just get LSR buffer waits.

- *NSR buffering has performed worse and used more EXCPs and cpu time than LSR in almost every one of my performance evaluations, and customer VSAM performance has improved in every case where I recommended that they be converted to LSR.*

- Closing and opening NSR files can cause Partition GETVIS fragmentation, which can go unnoticed when you start and shut down CICS every day.

VSAM Hints and Tips

- Use DSNSHARING=ALLREQS when the you map two or more FILE definitions to the same dataset because you *might* get better performance.

- Use DSNSHARING when you use a Base and Path(s) as each Path being opened causes the Base to be opened again, which can avoid the need for VSAM SHR(4).

- DSNSHARING might be **required** to avoid the Base and AIX not being in sync; this is not a CICS or VSAM bug.

- *Using DSNSHARING reports the same number of EXCPs for each FILE in CICS Statistics, so don't count them more than once; this is not a bug.*

- *The number of EXCPs for a Path that you see in CICS Statistics does **not** include EXCPs for the AIX, they are just for the Base Cluster, however, they are counted in the LSR statistics; this is not a bug.*

.

VSAM Hints and Tips

- **The last, and potentially very important tips . . .**

- Don't forget the 80/20 rule - tune the busiest files!

- CICS Statistics typically show you what has happened over a long period of time, and there could be hotspots that are lost in the data even when it appears to be working well.

- VSAM performance can be affected by other problems or by the level of activity in CICS, and tuning VSAM might **not** be the answer! (You saw that earlier.)

- VSAM tuning may not be able to compensate for bad application use of VSAM.

- Sometimes you just cannot tune VSAM to make it work as fast as you would like!

- I once tuned a VSAM file's browse performance, and that allowed a resource hog to impact other transactions even more than it did before! I backed the change out.

The Cost of Function Shipping

- Using MRO or ISC is always expensive in terms of cpu time on z/VSE and this is not a bug, however, when used for DPL or Transaction Routing, the cost is not normally a problem.

- *However, Function Shipping is in a league of its own and I would **not** recommend it being used to handle a large percentage of the CICS environment's request activity, that is unless you have the cpu capacity that allows you not to care about it!*

- I ran a test of 100,000 tasks, using a mixture of Assembler and COBOL with a lot of EXEC CICS LINKs (a known cpu overhead), in order to show an extreme comparison of the cost of MRO. CICS was configured as a 1-tier, and then as a 3-tier with a TOR, an AOR and a FOR that used MROLRM=YES; Function Shipping included VSAM and both Main and Auxiliary Temporary Storage requests, and both included a significant amount of VSAM, DFHTEMP and journal I/O to generate a good amount of cpu time from EXCP activity to avoid skewing the cpu time too much.

- *The 3-tier environment required more than 6 times the amount of cpu time, took 2.5 times the elapsed time, and there was 10 times the amount of SVC activity!*

- Using MROLRM=NO, ISC and/or 2 cpus would have caused even more overhead.

- *The results were not enormously different to what I saw for a customer!*

The Cost of Function Shipping

- *MROLRM=YES can produce a reduction of up to a 30% delta in cpu time in the FOR compared with MROLRM=NO, but can cause MRO Sessions to remain busy for longer and may require more to be defined.*

- The enhanced DFH0STAT output will show if you do not have enough sessions, for example, if the Peak contentions winners is the same as the Send count or you see non-zero values for Peak outstanding allocates or Queued allocates.

- Use CICS Shared Data Tables for small, busy, primarily read-only files.

- Move files to the AOR that either uses them exclusively, or to the AOR that uses them the most to make it an AOR/FOR, connecting it to all AORs that use the files.

- The AOR (the Client) typically uses only MRO **Send** sessions, and the FOR (the Server) only MRO **Receive** sessions, hence you do not need to define the same number of Send and Receive sessions for a Connection; the same applies to other CICS to CICS MRO connections.

- Don't use MROFSE=YES in an AOR because it can occasionally cause problems.

The Cost of Function Shipping

- The next two slides show PTNMON LPARCPU% before and after correcting an application bug that caused a huge increase in Function Shipping.

- ***Get to know what the typical CICS usage profiles look like, then you can see when something is wrong!***

- Consider using DTIMOUT(mmss) and SPURGE(YES) to avoid CSMI hangs causing Sympathy Sickness in connected CICS partitions.

The Cost of Function Shipping
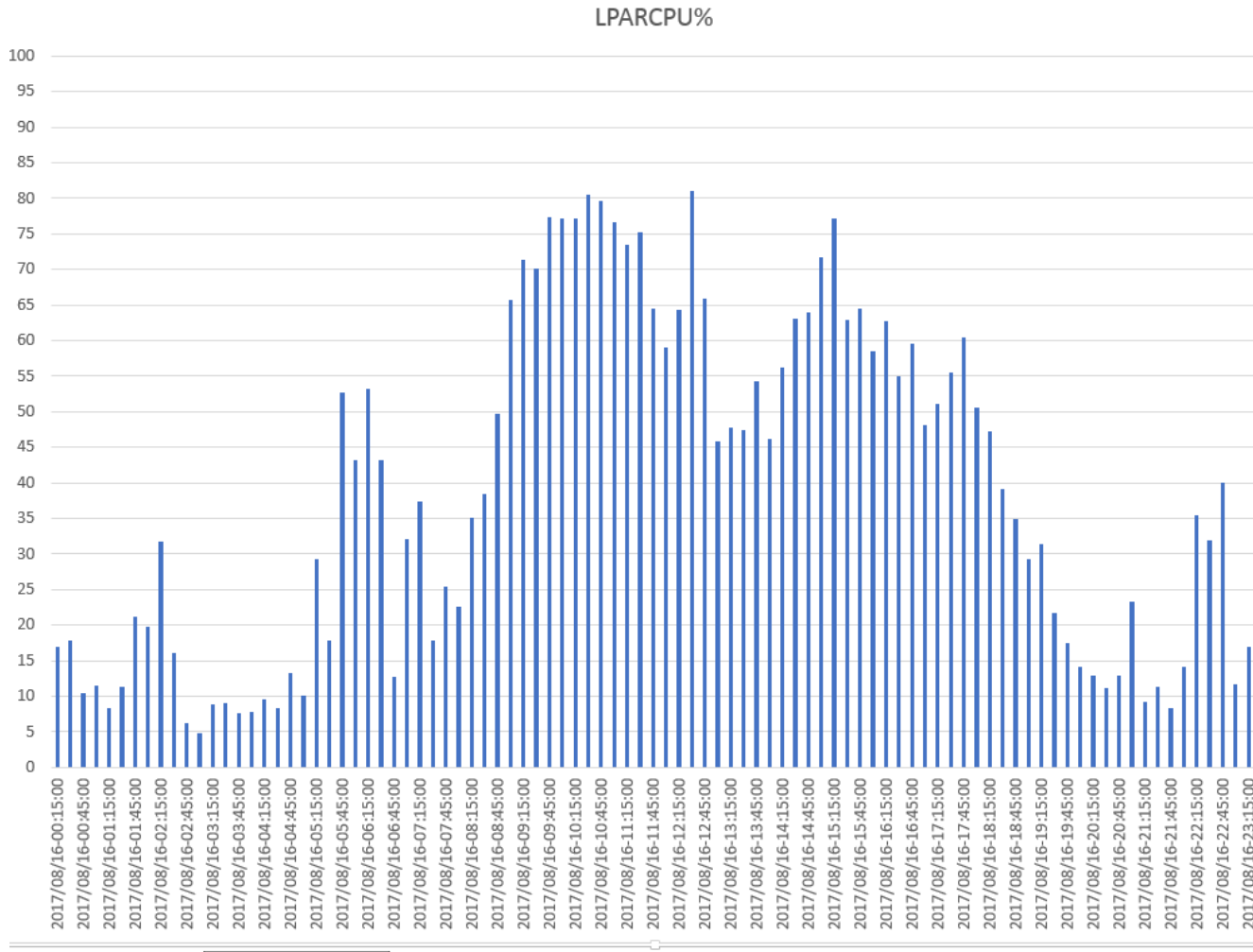
The Cost of Function Shipping
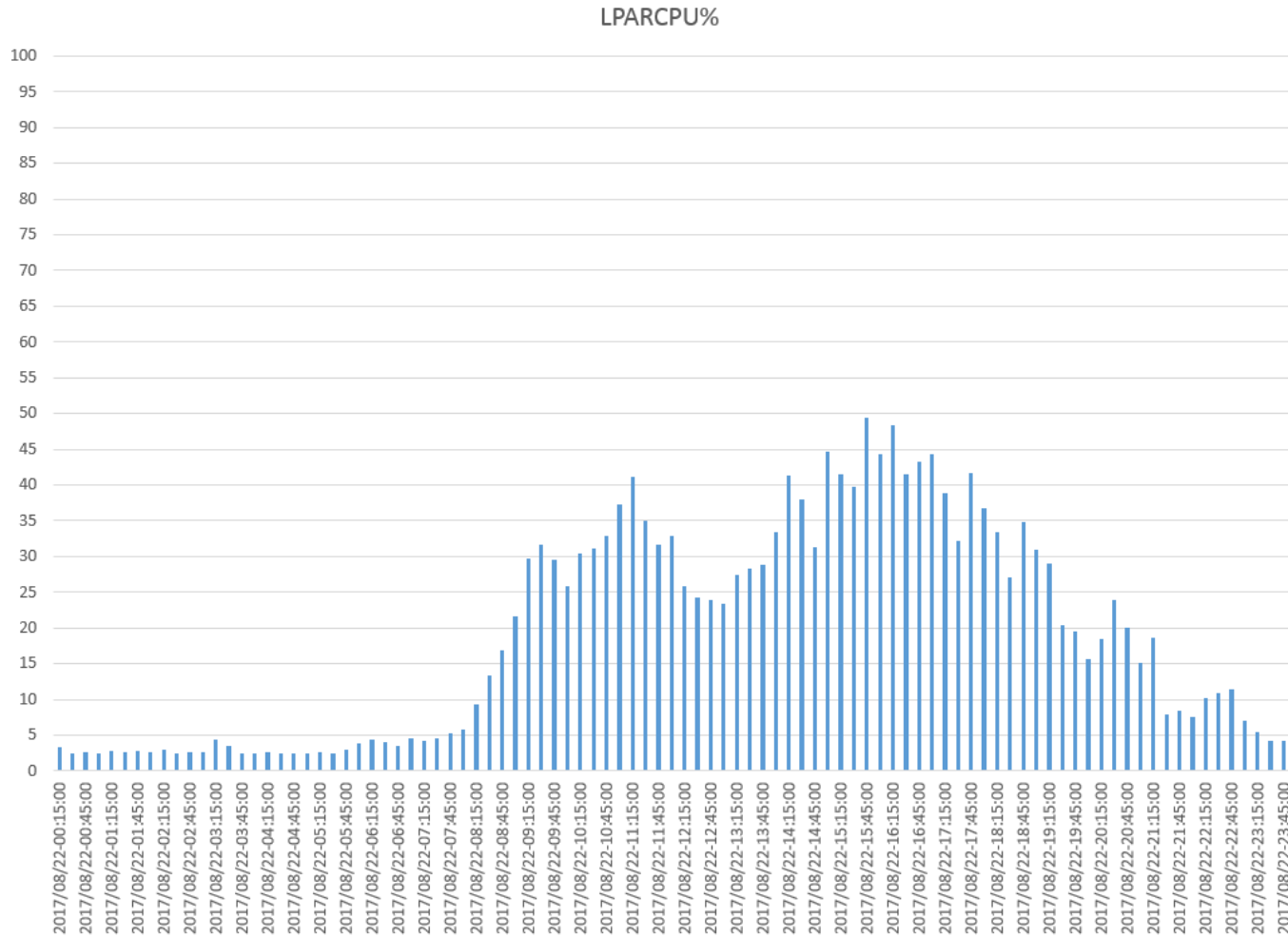
A Significant MRO Limit

- MRO ships data between CICS partitions using 24-bit System GETVIS Transfer Buffers.

- There is a 256K limit on Transfer Buffers, and when reached, can cause AZI2 abends or even close a Connection - this is a tuning issue and is not a bug.

- It is normally related to CICS being slowed enough to stop data shipping requests (MRO "SWITCH" functions) taking place quickly enough for buffers to be freed for reuse; a CICS System dump or any major slowdown could cause it when you are close to the limit.

- When a large number of Sessions have been defined for Connections, you can also get a failure when connecting CICS partitions.

- You will see DFHIRP return code 0208 in the trace, and this is **not** due to a lack of System GETVIS storage as the manual says it is!

Should You Only Use Main Temporary Storage?

- Using Main Temporary Storage *is* more efficient than using Auxiliary Temporary Storage.

- However, large amounts of Main can result in a very large amount of the CICS 31-bit EDSALIM being allocated, with the potential for severe storage fragmentation in 31-bit EDSA subpool TSMAIN, and could cause SOS Above; this is not a bug.

- *There is no formula for converting DFHTEMP disk space to EDSALIM storage requirements.*

- Defining an appropriate number of TS buffers in the SIT can reduce the overhead, and the 31-bit storage requirement in subpool TSBUFFRS is a constant.

- Unless a TSQ is recoverable, CICS only writes data to DFHTEMP when it needs to steal an active buffer, and hence it is possible to perform no or minimal DFHTEMP I/O.

- DFH0STAT and DFHSTUP show tuneable DFHTEMP buffer read and write counts, and non-tuneable forced buffer writes for recovery.

- DFHSTUP shows CICS subpool usage, otherwise format a CICS dump with DATA SM=1.

- See the CICS Performance Guide for subpool names.

The Cost of CICS Dumps and Extrapartition Datasets

- *To keep it simple - suppress **ALL** dumps that you don't need to look at, and fix application programs that produce dumps!*

- *Most CICS System Dumps (SDUMPs) occur on QR, and QR stops doing **all** other useful transaction-related work until the dump has finished.*

  - I have seen an application program check SR0001 SDUMP cause SOS and sympathy sickness in connected CICS partitions resulting in a CICS outage!

  - Dumping to SYSDUMP is very slow, and if you don't use // OPTION SYSDUMPC and SYSDUMP fills while the dump is being taken, the impact is even worse!

  - Slow SDUMP performance is a z/VSE and not a CICS issue!

- *Transaction dumps have a noticeable impact on QR because of the amount of internal CICS activity that is required to obtain information for the dump, and because all DFHDMP I/O is synchronous, that is, QR is blocked while I/O is active!* This is not a bug.

- Extrapartition I/O is also synchronous; this is also not a bug.

AOB

- ICVR is TCB cpu time, and is rounded to multiples of 500 milliseconds, plus it is always 500 milliseconds more than you asked for; hence it starts at 1000 milliseconds and is then increments of 500 milliseconds; the DFHSIT macro default of 5000 or 20000 is very high, and a runaway task could have a huge impact on performance and take CICS some time to recover from.

- ICV=250 might improve responsiveness to certain types of wait state when CICS is idle; ICV is set to multiples of 250 milliseconds, starting at 250.

- Your Vendor software may not like it, but you could save perhaps a 5% cpu delta by turning CICS Internal Trace off, however, we might not be able to solve your CICS problems!

- MXT is a hidden problem, but I have a long-running CICS program that can typically report on the console when it occurs.

Thank You

Please forward your questions or remarks to
zvse@de.ibm.com
poilmike@uk.ibm.com

More Information

… on VSE home page: http://ibm.com/vse

- **Ingolf's z/VSE blog: https://www.ibm.com/developerworks/mydeveloperworks/blogs/vse**

- **Requirements: https://www-03.ibm.com/systems/z/os/zvse/contact/requirement.html**

- **z/VSE service & support: http://www-03.ibm.com/systems/z/os/zvse/support/**

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/

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