What Mother Never Told You About CICS Performance

Presented 20th February 2018
michaelalanpoil@gmail.com
Last update: March 2022

http://www.ibm.com/zVSE

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.

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 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.

Note: The written content has been updated since it was first presented.

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 CICS TS for z/OS and z/OS itself have more performance features that can be exploited.
- Its performance characteristics look to be different to z/VSE - a recent z/OS migration that I experienced suggested that CICS TS for z/OS was noticeably more efficient in its use of processor capacity, and it handled "problem" transactions (and batch work) much better than was the case on z/VSE (and z/OS multi-CPU support is *much* better than z/VSE).
- 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 a 10% or more CPU delta (i.e. 50% within z/VSE is at least 55% total), and I have known z/VM bugs to cause problems at times (the bugs that I found have been fixed for some time).
- Using the default Vertical Polarization Mode, AKA HiperDispatch, might, or might not, 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 6.2 is the only supported release.
- 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.
- Beware of batch Assembler causing an effect known as SIIS or SNIS that can cause a significant increase in CPU usage compared to "normal“.
- SIIS might be an issue when upgrading from a BC12 or EC12.
- Install the PTFs for z/VSE 6.2 APARs DY47814, DY47815, DY47824 and DY47847 to (potentially noticeably) reduce CPU consumption within the z/VSE operating system itself.
- Customer or Vendor batch that causes it must be corrected.
- Native z/VSE does not have the tools to show it, but you can collect z/VM Monitor Data to show that it is occurring, although it can be difficult to apportion blame because the data is collected at the Logical CP (CPU) level. IBM’s analysis will require Hardware, z/VM and z/VSE SMEs and by it’s nature, can be very slow to perform.

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 do you know how to correctly interpret the data produced by the monitor?
- Some issues that I know about in CICS Statistics and Monitoring data.
- CICS QR TCB CPU time - the QR subtask (the 4th task from the top of a STATUS xx command output) 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, and the more z/VSE activity that is required by CICS, the less accurate it becomes.
  - QR TCB CPU/Partition CPU is a better idea of the CPU Capture Ratio, for example, customer data for very CPU-intensive applications showed a capture ratio of about 0.6, but more typical customers would show a lower value.
  - It affects CPU time in the Monitoring Task Performance record (and ASG TMON transaction CPU), so you are never sure how much CPU any given transaction actually uses, however, on average it will be higher by calculating reported CPU/capture ratio.

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 long; 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 severe 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 likely 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. (Maybe 6x 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 transaction/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 - but beware of the issues detailed a few slides back!
- It is not a perfect solution as it may not have all of the data that you 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 (e.g. ASG TMON) 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 what the overhead is.
- 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 MONVSE that reports every "n" seconds in CSV format.
- It contains useful z/VSE-level data (VSECPU%) from the same source as CPUMON and:
  - Native z/VSE LPAR CPU (LPARCPU%) or z/VM Total Time (VMCPU%), which includes the z/VM CP overhead.
  - Total CPU for every LPAR based on only CP CPU types, or the total z/VM LPAR CPU utilisation including ICFs and IFLs.
  - T/V ratio for VM/VSE or (LPAR CPU+PR/SM Overhead)/LPAR CPU.
- The following for up to 11 partitions:
  - CPUA% - the likely percentage of samples active on a CPU.
  - CPUW% - - the likely percentage of samples waiting for a CPU.
  - NP% - the percentage of samples when ready and NP code was in use (see later).
  - CPU% - the partition's CPU time as returned by the z/VSE.
  - I/O operations per second.
- It has a low CPU and storage overhead, but is able to sample task states very frequently in order to get good sampled data values, but it MUST run very high PRTY.

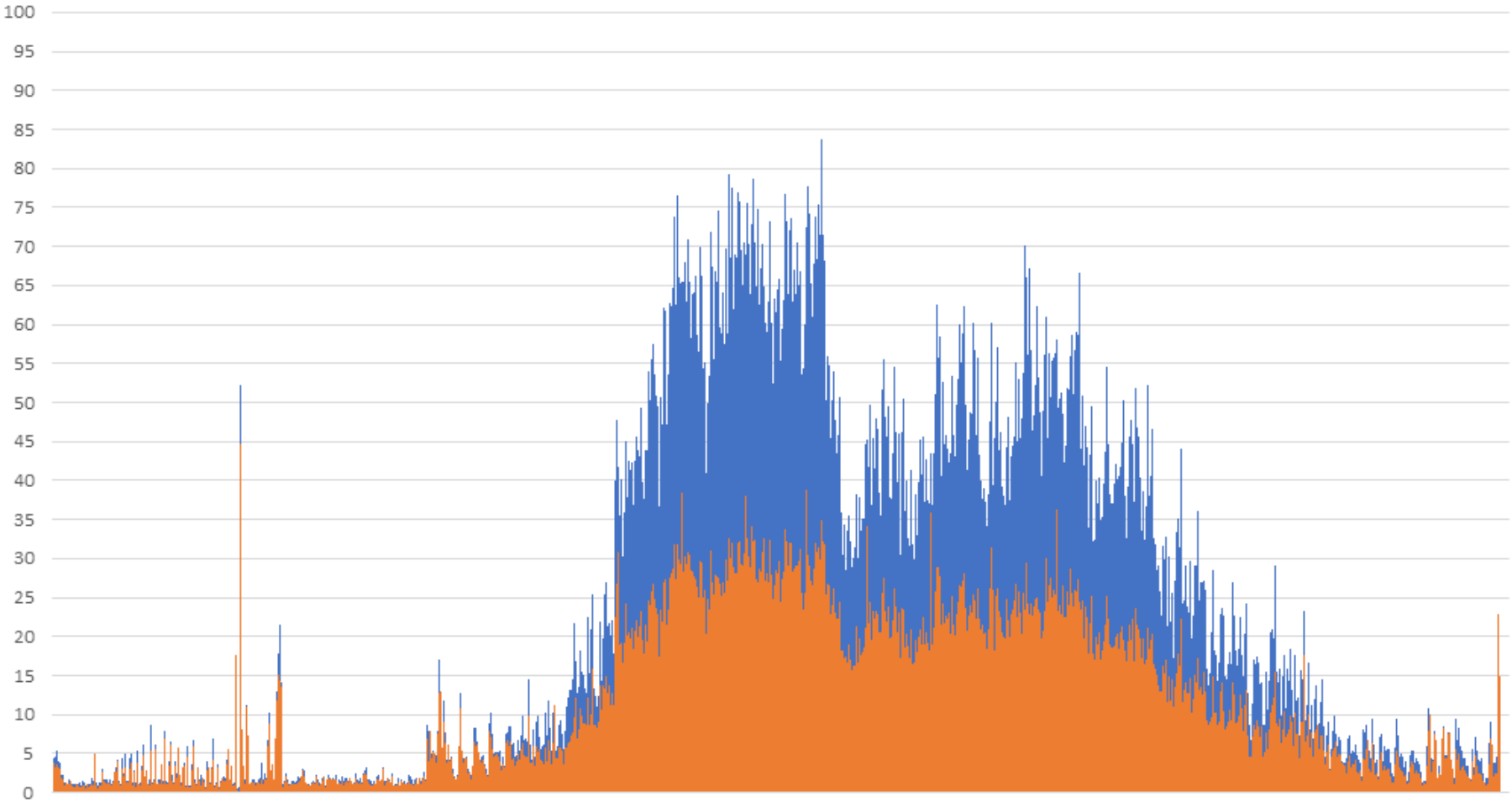
How I Measure Cpu Utilisation

- The number of samples in an interval after the first can act as an indicator of a z/VSE-level CPU constraint (the first interval is short to get to the requested interval boundary) and they should be reasonably consistent or z/VSE might be having a CPU constraint.
- 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 MONVSE samples!
- CPUW% 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.
- 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 - MONVSE 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, the customer needs the MIPS of two large CPUs because there was 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 MONVSE data for 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

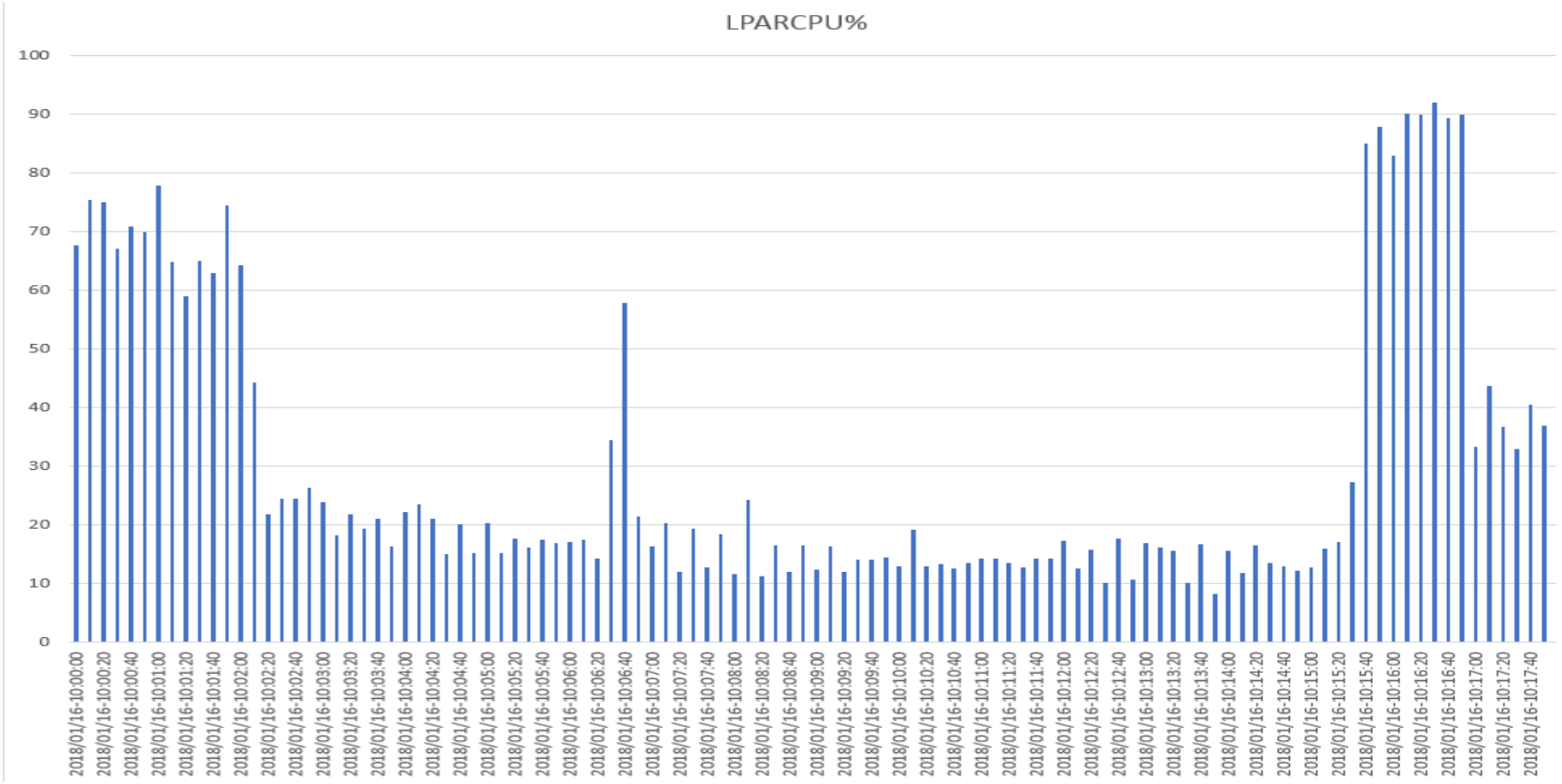
- A VM/VSE customer used two CPUs unnecessarily, and we measured a 10% overall CPU reduction when using one, but the average dispatch time per CICS task reduced by 30%!
- Dynamic Cpu Balancing may or may not help in all circumstances.
- z/VSE needs to serialise certain types of processing; the ones that I know about are SVCs and using Key zero, and it labels the task's code path as Non-Parallel (NP).
- Only one task that needs 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.
- MONVSE shows z/VSE NP% and NP/TOT and estimates NP% by partition.
- I have seen average z/VSE NP% values from 15% (amazing!) to 70% (bad!), and values like 99% during parts of overnight batch.
- z/VSE suggests that the maximum number of CPUs that can be used on average is 0.9/NPS (NPS=NP/TOT as shown by QUERY TD etc.).

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 partition exceeds something like 70% dispatchable despite having excellent CPU availability.
- As CICS activity is mostly single-threaded, the busier it gets, the more likely it is that internals delays (increased suspend time) will increase response time.
- 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 when CICS partitions are hit by a significant MXT problem, so CPU drops, and then recover, with CICS in catch-up mode causing a CPU spike.



Using DFH0STAT

- I have an enhanced version of DFH0STAT, and it is continually evolving.
- 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 z/VSE 6.2:
  - Transaction Classes.
  - VSAM files: Distinguishes between a VSAM base (KSDS/ESDS) from a path (KSDSP/ESDSP), distinguishes between a REMOTE file and a REMTBL remote Data Table; for an open file - LSRPOOL data (if LSR is used), DEFINE CISZ, SHAREOPTIONS, CI and CA splits since OPEN.
  - LSRPOOL lookaside % overall and by buffer size; estimated LSRPOOL storage usage.
  - System and Transaction Dump totals, and counts by dump code.
  - ISC/IRC, including MRO SVA current and HWM transfer buffer usage (see later).
  - Terminals.
  - VTAM RAPOOL value etc.
  - Journaling.

Using DFH0STAT

- Program "Type" column showing 'ASM', 'COB' etc.
- "Program Totals" contains a "Programs in Use" section with types and sizes.
- "Program Totals" shows estimated program storage use in the CDSA, SDSA etc.
- Sample enhanced VSAM file data.

```
          Access         LSR  Str Waits    Read     Get Update   Browse       Add      Update     Delete      Data       Index
Filename  Method  Type   Pool Max Total  Requests   Requests    Requests   Requests   Requests   Requests     EXCPs      EXCPs

FILEB      VSAM    KSDS    2     0    0          0          0          0          0          0          0          0          0
FILEB     BUFFER WAIT HWM      0 TOTAL             0 LSRPOOL BUFFER SIZE DATA  4096 INDEX  2048
FILEB     SPLITS CI            0 CA                0 DEFINED CISZ        DATA  4096 INDEX  2048 EXCC           0 SHAREOPTIONS 2
FILEB     DATASET NAME    TCOM.CICS2.IYBTZCCA.FILEB                    DSNSHARING Y BASE R/W RECSZ     80 RECORDS     500,000
FILEB     STRINGS   150 STRINGS4UPDATE   120 KEYLEN   6 CI/CA 180 COMPRESSED N XXL N INDEX-RECORDS       1,396
```

Using DFH0STAT

- Sample enhanced Program data:

```
Program Totals
_______________
  Programs. . . . . . . :       1,301
    Assembler . . . . . :       1,245
. . .
Programs in use . . . :            72
  Assembler in use. . :            71          5,028K
  Assembler-24 in use :            11            204K
  Assembler-31 in use :            60          4,825K
  C in use. . . . . . :             0              0K
  C-24 in use . . . . :             0              0K
  C-31 in use . . . . :             0              0K
  COBOL in use. . . . :             1            151K
  COBOL-24 in use . . :             0              0K
  COBOL-31 in use . . :             1            151K
. . .
Maps in use . . . . . :             1              2K
Partitionsets in use. :             0              0K
_____________________________________   _____________
Total . . . . . . . . :            73          5,181K

CDSA Programs . . . . :             5             36K
SDSA Programs . . . . :             2             13K
RDSA Programs . . . . :             5            157K
ECDSA Programs. . . . :            23            722K
ESDSA Programs. . . . :             1              1K
ERDSA Programs. . . . :            37          4,253K
SVA Programs. . . . . :             0              0K
ESVA Programs . . . . :             0              0K
Unused Programs . . . :             1              5K
Not Located Programs. :       1,291
_____________________________________   _____________
Total . . . . . . . . :       1,365          5,187K
```

Using DFH0STAT

- You can get useful performance and profiling information from the STAT REXX program:

```
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 ran 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 Time/TCB Dispatch Time (DS TCB CPU Time is a subset of TCB CPU Time).
- With the data below, it is 00:00:50.96825/00:01:59.57785 - 0.43 or 43%.

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 application CPU usage relative to the use of z/VSE services such as I/O, so it is really a measure of application (and potentially VSAM buffering) CPU intensity.
- I watch for it reducing because that can be a sign of a CPU constraint, although some variance 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.

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 - the Law of Diminishing Returns.
- Allocating more than about 1,000 buffers of the same type (Data/Index) and size may start to become expensive in terms of CPU time for buffer scanning, but YMMV applies.
- 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, and so does the enhanced DFH0STAT.
- 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; you can get a lot of these for ESDS log files.
- 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 if and when fix is made available.
- Exclusive Control conflicts can cause a lot of work in CICS and VSAM due to the potential number of retries involved before VSAM will handle the request; this is not a bug.

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=ALLREQS 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).
- The next slide shows before and after using DSNSHARING for a very busy KSDS BASE and its PATH and moving them to a different LSRPOOL, saving about 40M EXCPs per day. This level of saving is extremely unusual, but it was good news for the customer!
- DSNSHARING might be required to avoid the Base and AIX not being in sync; this is not a 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. You will see this on the next slide.
- The enhanced DFH0STAT will find the same number of CI and CA splits for the base and path when DSNSHARING is active; 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

```
File          LSRPOOL  1 KSDS  BASE EXCPs      699085 DEXCPs      590764 IEXCPs      108321 EXCPs/second    10.51
Index/Data EXCP ratio  0.18 EXCPs/Request    0.89 EXCPs/task      0.59 File requests      781341 Requests/task      0.66
Read/Write ratio        0.50 Read      729    0.09% Read Update     260204   33.30% Browse           0    0.00%
Rewrite     260204   33.30% Add      260204   33.30% Delete          0    0.00%
CI split/Add   7131   0.03 CA split/Add    185   0.00 Exclusive Control/Request   7056   0.01 SHR(2)

File          LSRPOOL  6 KSDS  BASE EXCPs      586488 DEXCPs      562436 IEXCPs       24052 EXCPs/second     8.62
Index/Data EXCP ratio  0.04 EXCPs/Request    0.79 EXCPs/task      0.51 File requests      744691 Requests/task      0.65
Read/Write ratio        0.50 Read      595    0.08% Read Update     248032   33.31% Browse           0    0.00%
Rewrite     248032   33.31% Add      248032   33.31% Delete          0    0.00%
CI split/Add   6762   0.03 CA split/Add    188   0.00 Exclusive Control/Request   8535   0.01 SHR(2)

File          LSRPOOL  1 KSDSP PATH EXCPs    36836080 DEXCPs       15351 IEXCPs    36820729 EXCPs/second   553.55
Index/Data EXCP ratio 99.99 EXCPs/Request   68.84 EXCPs/task     31.12 File requests      535119 Requests/task      0.45
Read/Write ratio 9999999.99 Read        0    0.00% Read Update          0    0.00% Browse      535119  100.00%
Rewrite           0    0.00% Add           0    0.00% Delete          0    0.00%
CI split/Add      0   0.00 CA split/Add      0   0.00 Exclusive Control/Request    819   0.00 SHR(2)

File          LSRPOOL  6 KSDSP PATH EXCPs      586488 DEXCPs      562436 IEXCPs       24052 EXCPs/second     8.62
Index/Data EXCP ratio  0.04 EXCPs/Request    1.15 EXCPs/task      0.51 File requests      510847 Requests/task      0.44
Read/Write ratio 9999999.99 Read        0    0.00% Read Update          0    0.00% Browse      510847  100.00%
Rewrite           0    0.00% Add           0    0.00% Delete          0    0.00%
CI split/Add      0    0.00 CA split/Add  0       0.00 Exclusive Control/Request    916   0.00 SHR(2)

Profile       EXCPS due to LSR reads                  91050465
Profile       EXCPS due to LSR reads                  48384486 (-47%)
Profile       EXCPS from LSR reads per second          1368.25
Profile       EXCPS from LSR reads per second           711.47 (-48%)

Profile       DISPATCHER cpu      time per transaction 0.002008 seconds based on TCB cpu time
Profile       DISPATCHER cpu      time per transaction 0.001666 seconds based on TCB cpu time (-17%)
Profile       DISPATCHER dispatch time per transaction 0.005583 seconds
Profile       DISPATCHER dispatch time per transaction 0.004262 seconds (-24%)
```

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.
- Use 16Gbps channels.
- Use z/VSE 6.2 ZHPF support.

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 might 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 might 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.
- A CICS Shared Data Table (SDT) is typically used for a small, busy, read-only (reference) file and its use saves a very significant amount of CPU time as shown on the previous slide, something that has been proven in many customer production CICS environments where SDTs transformed CICS performance and even avoided the need for a Processor upgrade.
- However, there is still a case for using an SDT when adds, deletes and updates are also performed, and my tests show that the overhead of the AOR swapping from an SDT access to Function Shipping with SDT data updates is small, however, a file with a large percentage of update activity is obviously not a good candidate for an SDT.

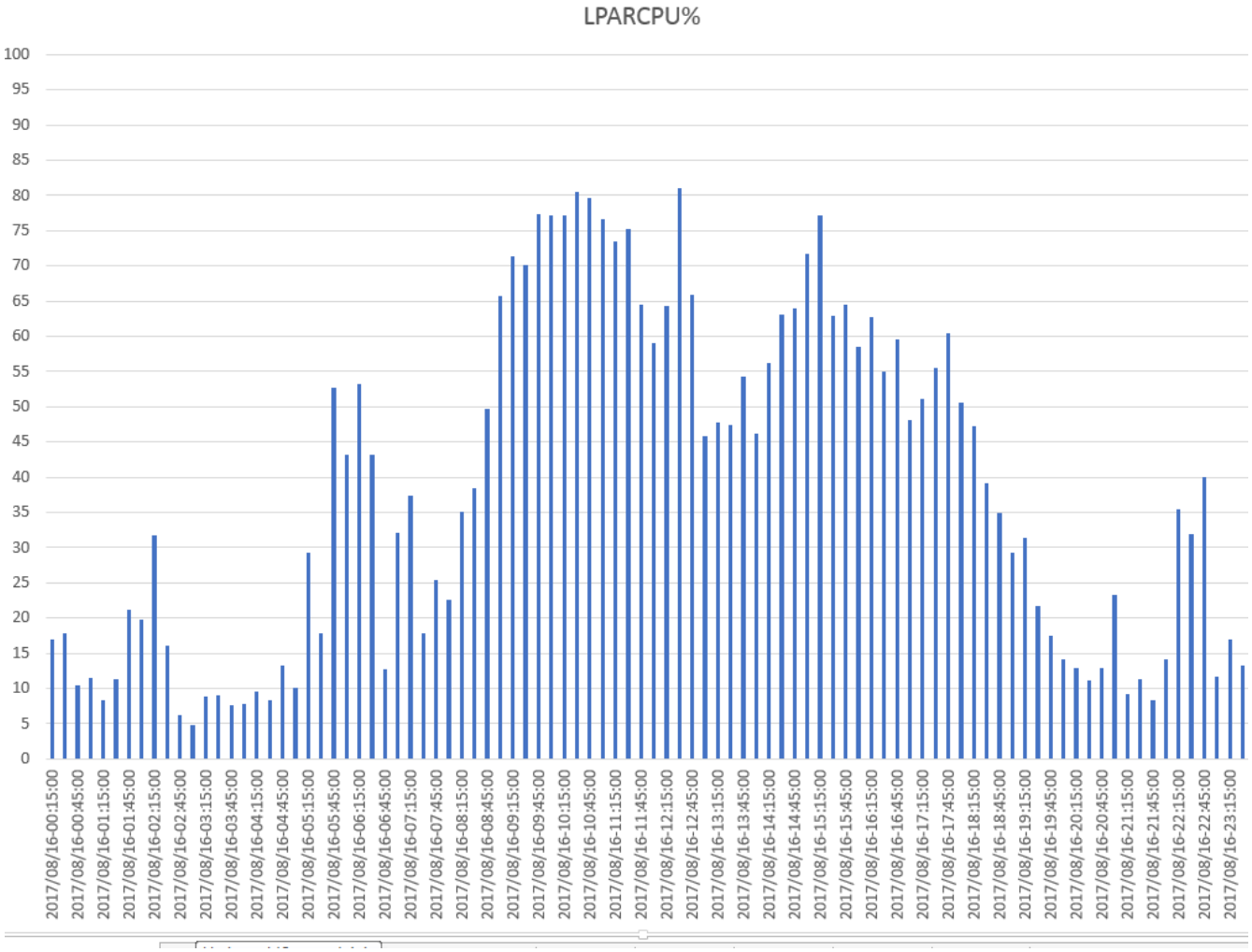
The Cost of Function Shipping

- The limitation of 2GB of SDT per CICS Partition is annoying and is not the case for the z/OS product, however, there is nothing to stop you using multiple FORs.
- If an SDT is not an option, consider moving 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. However, the overall environment will now be less resilient to failure because a problem in a file-owning AOR could now affect all connected AORs.
- Yet another option, if that is technically possible, it to give each AOR its own copy of a very busy file.
- The AOR (the Client) typically uses only MRO Send sessions, and the FOR (the Server) only MRO Receive sessions, so you do not need to define the same number of Send and Receive sessions for a Connection; that applies to other CICS to CICS MRO connections.
- Don't use MROFSE=YES in an AOR because it can occasionally cause problems.

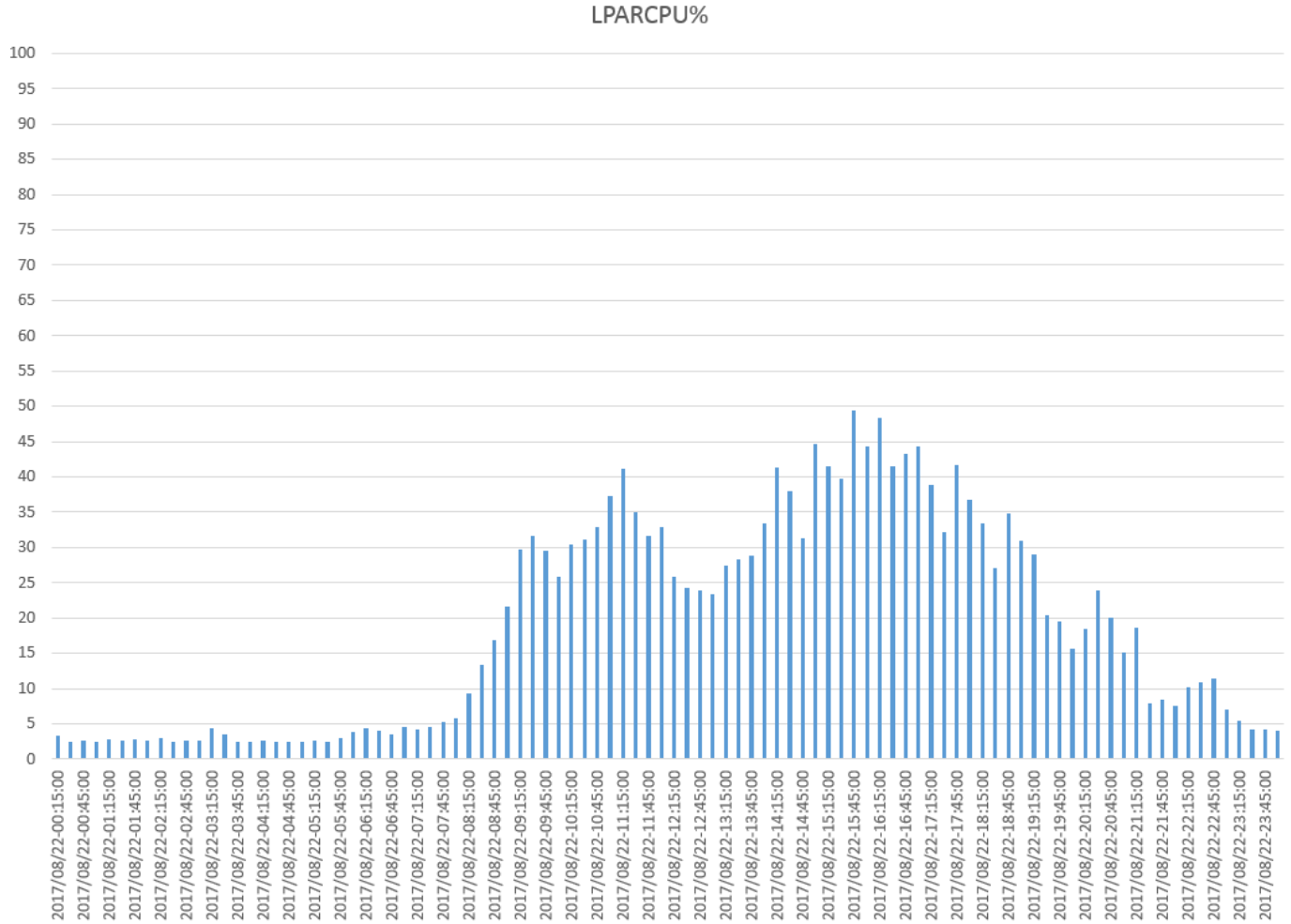
The Cost of Function Shipping

- The next two slides show MONVSE 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 another slowdown could cause it.
- 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 and SOS Above; this is not a bug. The 31-bit EDSA subpool TSMAIN contains the data.
- 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 while CICS is active.
- CICS divides the number of TS buffers in the SIT by 2 and assigns them to input and output usage, so you may need to allocate more than 2 times what you think you need to avoid I/O.
- 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 issue and not a CICS problem.
- 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 or might need adjusting, but you could save perhaps a 5% or bigger 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.
- Using DFHDCTSP potentially causes many IESX transactions to be run in order to capture CICS SYSLST messages for the "Inspect Message Log" IUI function, and is likely to add overhead for no good reason; review DFHDCTSP in ICCF library 59 and consider using the CICS-supplied definitions instead.

AOB

- A customer turned both CICS internal trace and a Vendor Macro Level Interpreter product trace off (MLI allows CICS TS to run Macro Level code), and this is the before and after from an analysis by DFH0STAT; 0.000736/0.001701 = 43%, 0.000971/0.001969 = 49%, An overall reduction of more than 50%!

```
DISPATCHER cpu time per transaction 0.001701 seconds based on TCB cpu time
DISPATCHER cpu time per transaction 0.001969 seconds based on accumulated partition cpu time

DISPATCHER cpu time per transaction 0.000736 seconds based on TCB cpu time
DISPATCHER cpu time per transaction 0.000971 seconds based on accumulated partition cpu time
```

- Using IESZNEP can cause a CICS EDSA control block leak for program IESSVL that is associated with the long-running CSNE task; you will see this in DFHPD430 PG=1 output.

Thank You

More Information

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

- Jen'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