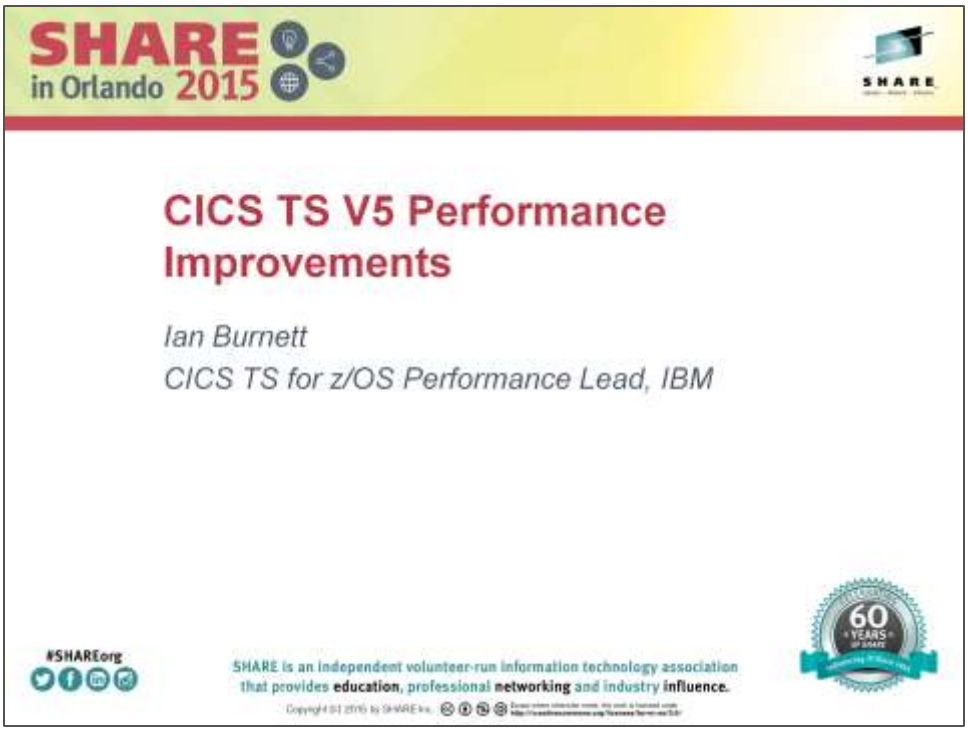


SHARE Orlando 2015

Tuesday, 11th August

Ian Burnett
IBM CICS TS for z/OS Performance Lead
ian.burnett@uk.ibm.com
@IanBurnett

Legal Disclaimer

- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including 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 results similar to those stated here.
- IBM, the IBM logo, and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

2

Measurement Process

- Overnight automation on dedicated LPAR
  - Dedicated CPUs, CHPIDs, DASD
- 5 RMF intervals recorded
  - Various transaction rates
- Total CICS address space accumulated
  - Divided by transaction rate to give CPU/tran
- Average CPU/transaction over 5 intervals compared
- Any difference analysed using Hardware Instrumentation (HIS)

Environment

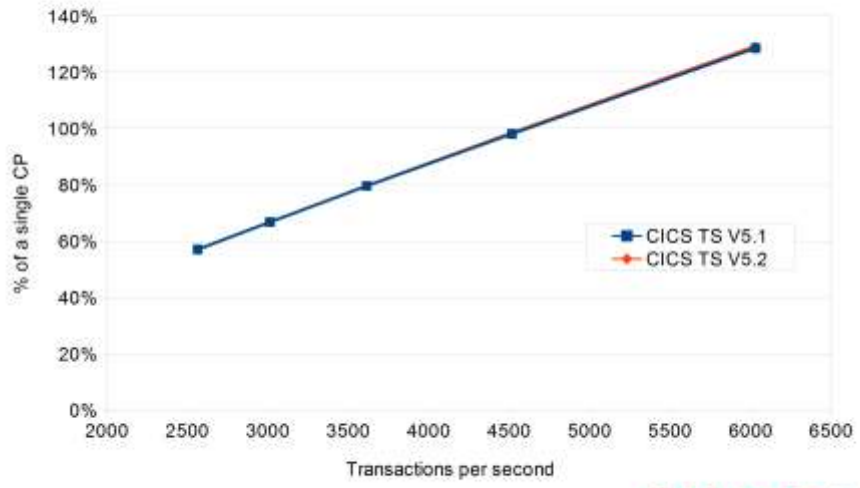
- Hardware
  - zEC12 2827-799 model HA1
    - LPAR with up to 32 dedicated CPs
    - Separate LPAR with 4 dedicated CPs for network driver
  - DASD DS8800
  - Internal Coupling Facility with ICP links
- Software
  - z/OS 2.1
  - CICS TS V5.1 refresh 18 Feb 2014
  - CICS TS V5.2

This section covers some of the workloads which are executed during the development phase to ensure that upgrading to a release does not incur a performance overhead.

DSW (Static Routing)

- COBOL/VSAM
- All transactions routed from 2 TORs to 2 AORs
- All FILE requests are Function Shipped to 1 FOR
- 50% of transactions issue FC requests
- All FC requests are VSAM LSR
  - Average of 6 requests per transaction (all transactions)
  - 69% Read, 10% Read for Update, 9% Update, 11% Add, 1% Delete
- 16 CPs – 5 CICS regions

DSW (Static Routing)



DSW (Static Routing)

ETR	CICS %	ms/tran
2563.06	57.03%	0.223
3011.97	66.75%	0.222
3613.27	79.61%	0.220
4515.94	98.11%	0.217
6029.03	128.57%	0.213

CICS TS V5.1
Average CPU / tran = 0.219ms

ETR	CICS %	ms/tran
2562.81	57.00%	0.222
3011.61	66.74%	0.222
3613.01	79.61%	0.220
4515.30	98.47%	0.218
6028.32	129.29%	0.214

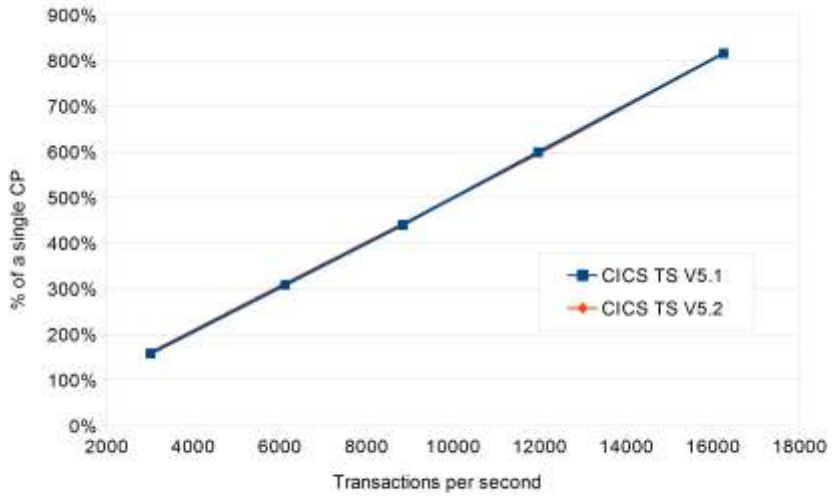
CICS TS V5.2
Average CPU / tran = 0.219ms

< 1% difference

DSW (CPSM Dynamic Routing)

- COBOL/VSAM
- All transactions routed from 4 TORs to 30 AORs via CPSM
- 50% of transactions issue FC requests
- All TS requests are TS Shared
- All FC requests are VSAM RLS
  - Average of 6 requests per transaction (all transactions)
  - 69% Read, 10% Read for Update, 9% Update, 11% Add, 1% Delete
- 16 CPs - 34 CICS regions + CMAS + WUI

DSW (CPSM Dynamic Routing)



DSW (CPSM Dynamic Routing)

ETR	CICS %	ms/tran
3006.60	158.00%	0.526
6118.61	308.48%	0.504
8830.54	440.00%	0.498
11962.02	599.67%	0.501
16238.38	815.93%	0.502

CICS TS V5.1
Average CPU / tran = 0.506ms

ETR	CICS %	ms/tran
3005.68	159.81%	0.532
6111.82	311.00%	0.509
8827.54	441.50%	0.500
11963.57	596.41%	0.499
16252.29	817.04%	0.503

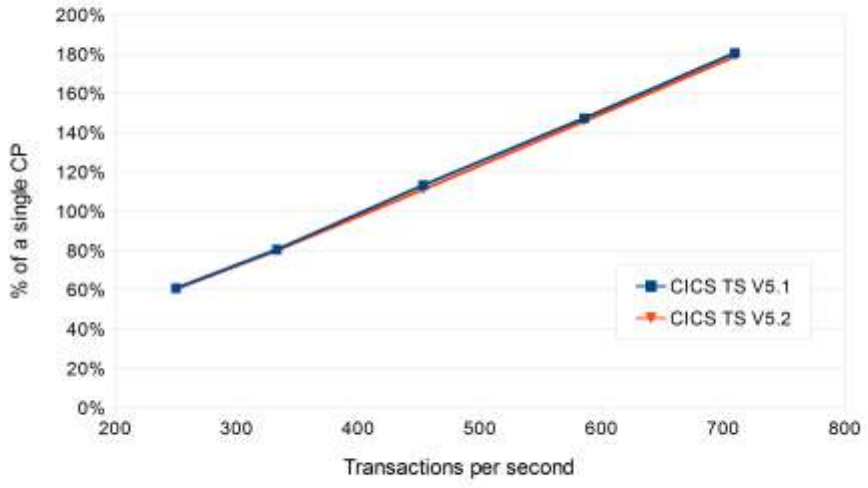
CICS TS V5.2
Average CPU / tran = 0.508ms

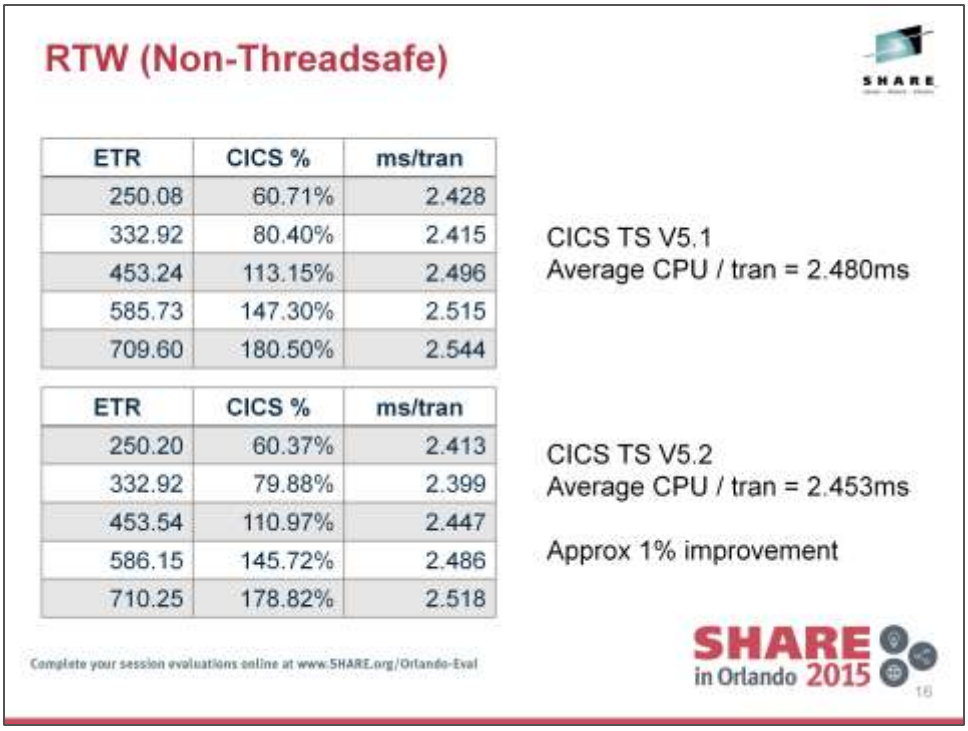
< 1% difference

RTW Single Region

- COBOL/DB2
- 7 transaction types
- 20 Database tables
- Average 200 DB2 calls per transaction
  - 54% select, 1% insert, 1% update, 1% delete
  - 8% open cursor, 27% fetch cursor, 8% close cursor

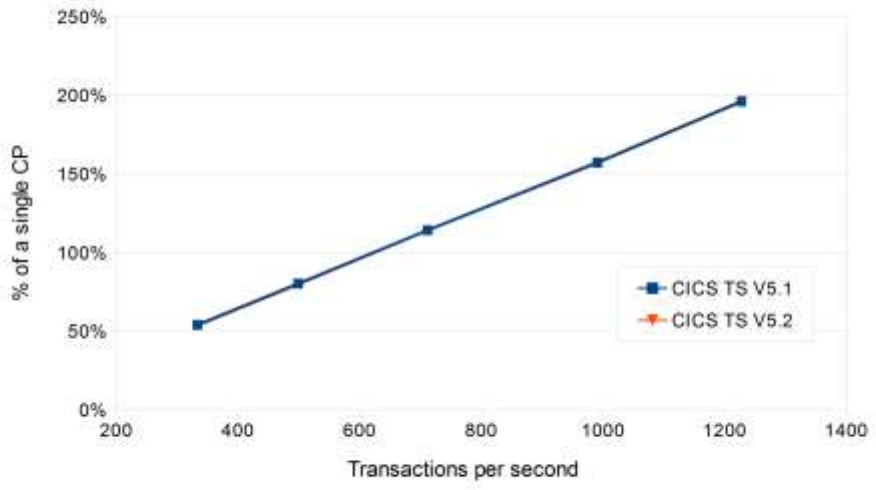
RTW (Non-Threadsafe)

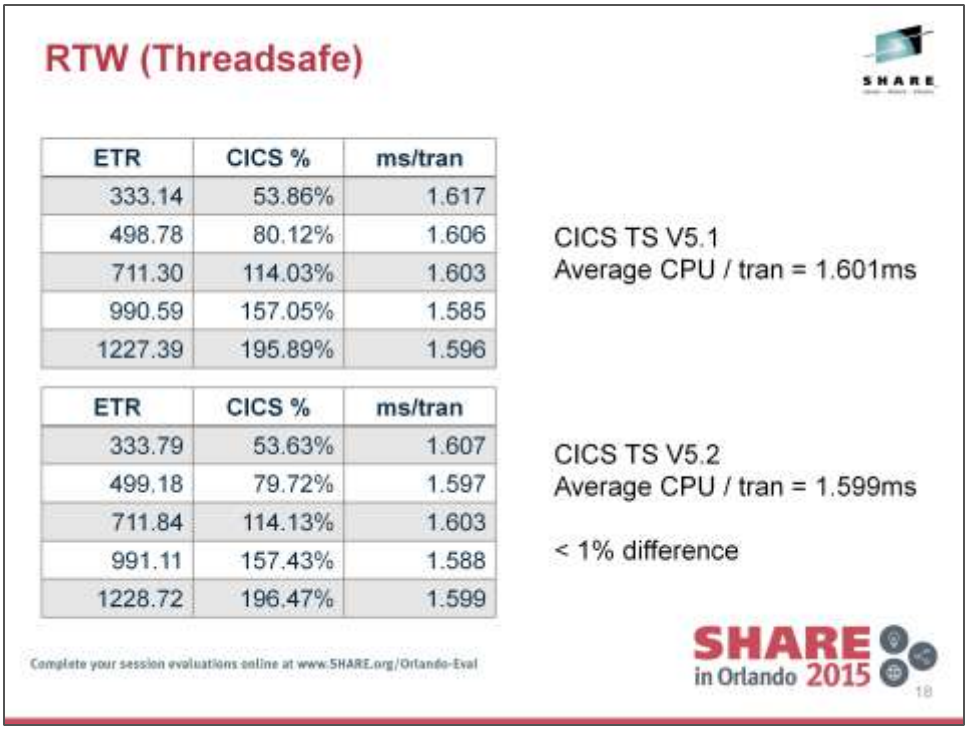




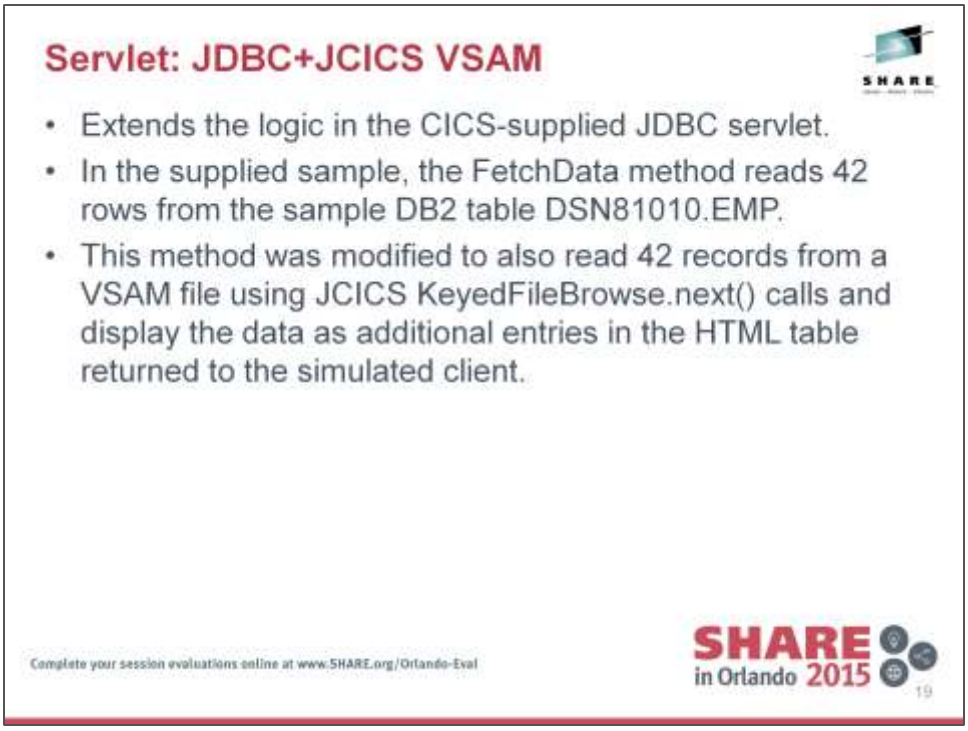
The tables show RMF data extracted from 5 different transactions rates for both CICS TS V5.1 and CICS TS V5.2.

RTW (Threadsafe)



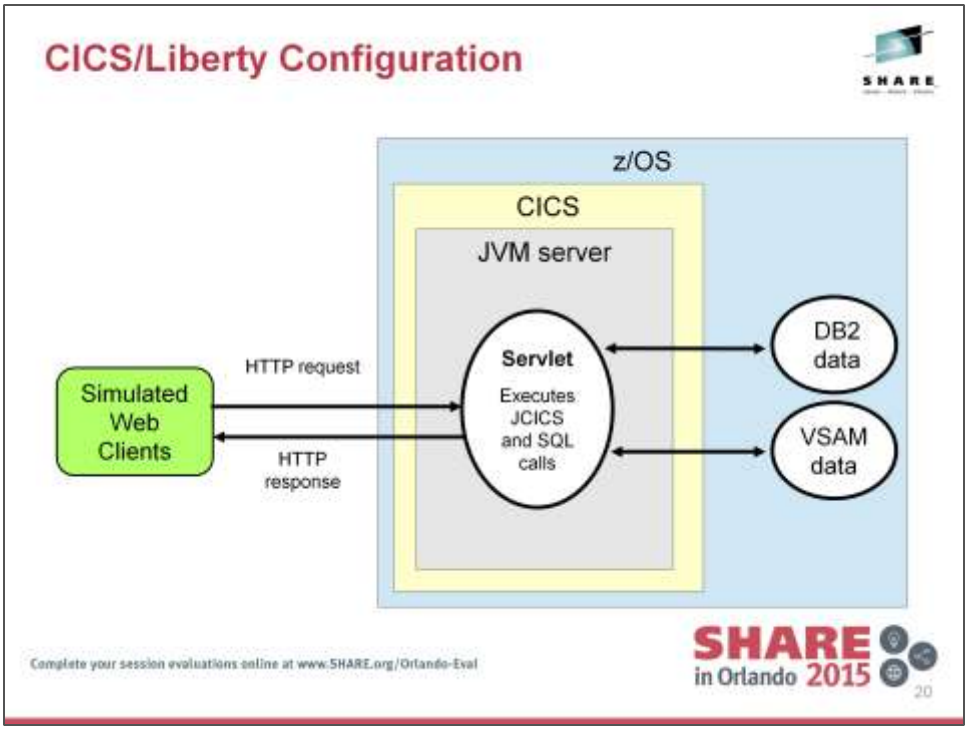


The tables show RMF data extracted from 5 different transactions rates for both CICS TS V5.1 and CICS TS V5.2.



Overview of the JDBC+JCICS servlet

The VSAM file contains a copy data held in the DB2 sample database used by the JDBC servlet.

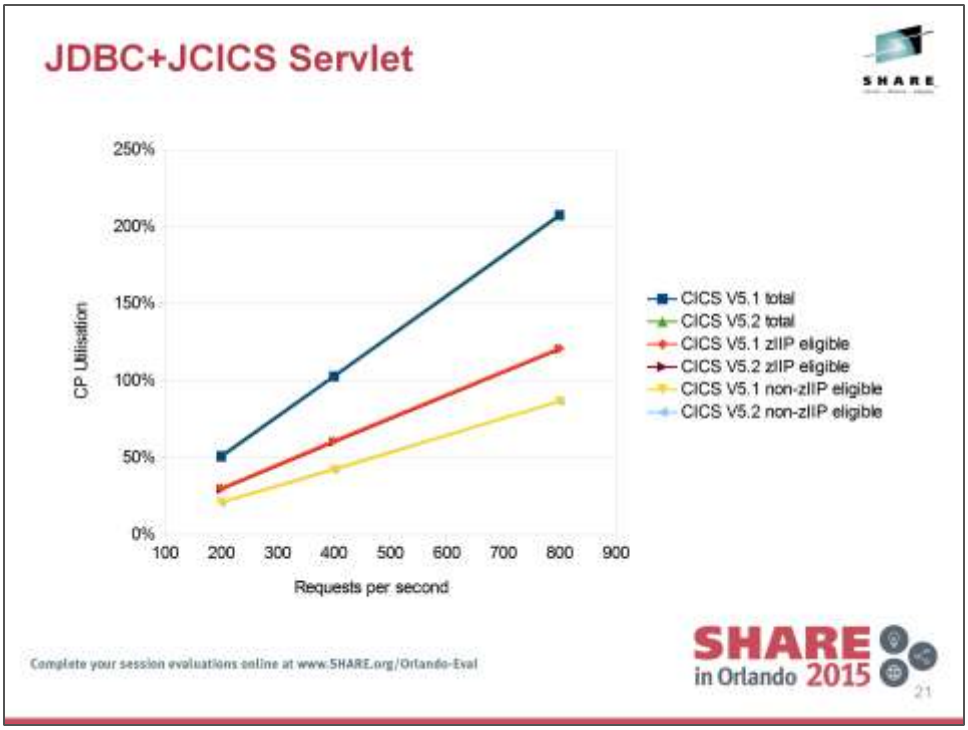


Representation of CICS/Liberty configuration.

Liberty runs within a JVM server within a CICS region.

Simulated browser requests are made to Liberty listener port specified in the JVM profile parameter -Dcom.ibm.cics.jvmserver.wlp.server.http.port=nnnnn

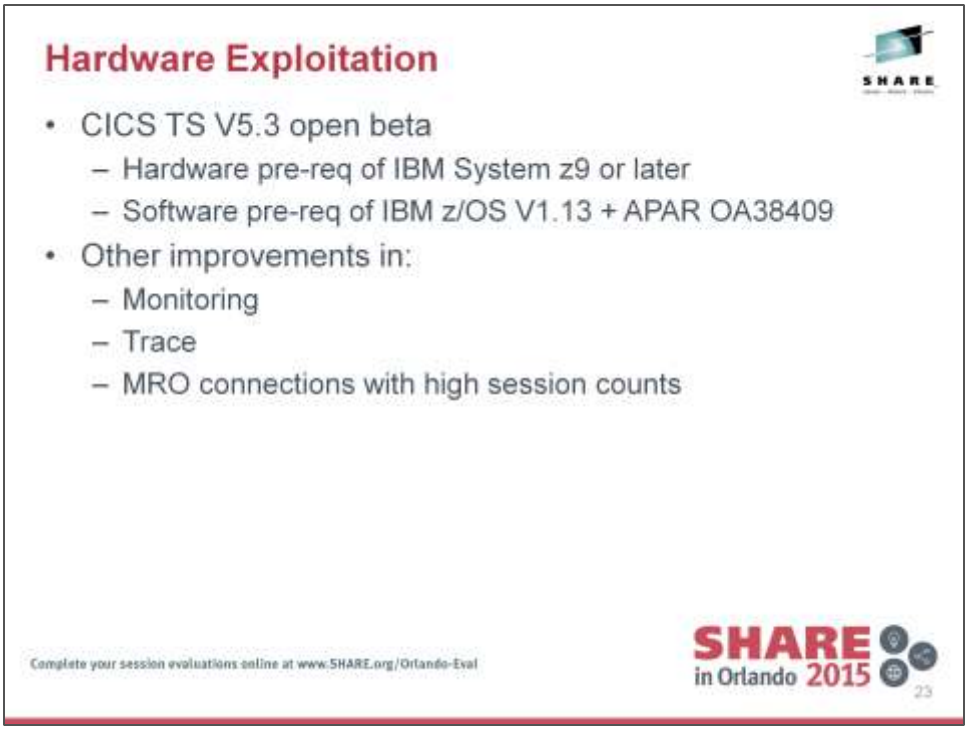
The servlet runs on a T8 TCB within CICS and is capable of JCICS and SQL calls.



Both using DB2 V10
Single JVMserver with maximum 100 threads

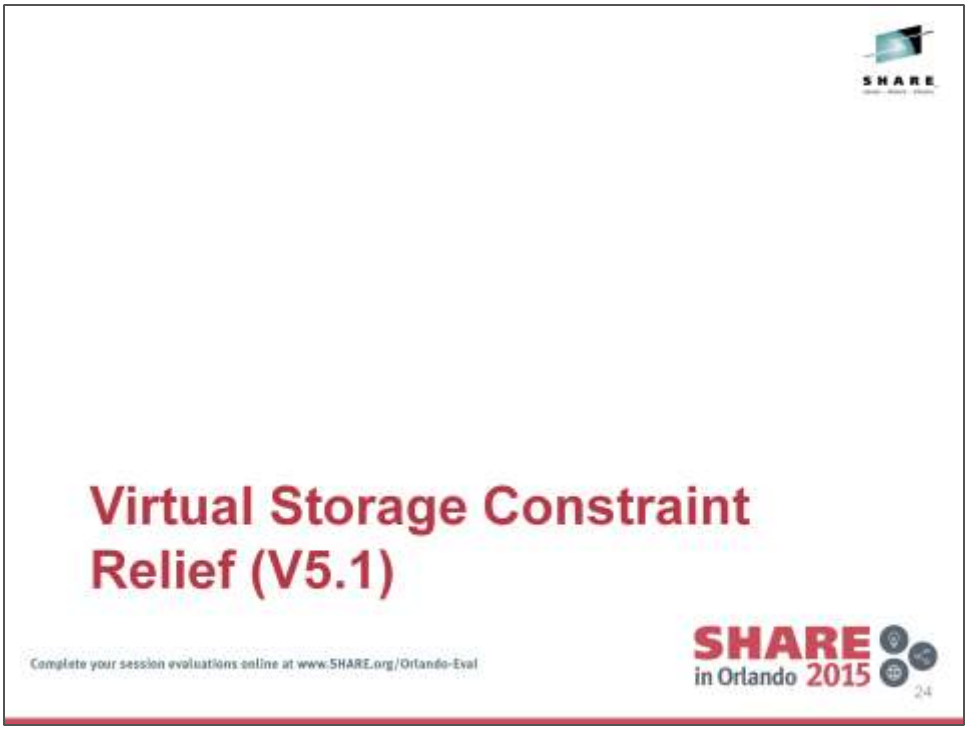
Java environments:
Both use Java7-64bit SR7
Both used fixed generational heaps
-Xgcpolicy:gencon
–Xnompressedheap
-XXnosuballoc32bitmem
–Xmx200M –Xms200M –Xmnx60M –Xmns60M -Xmox140M -Xmos140M

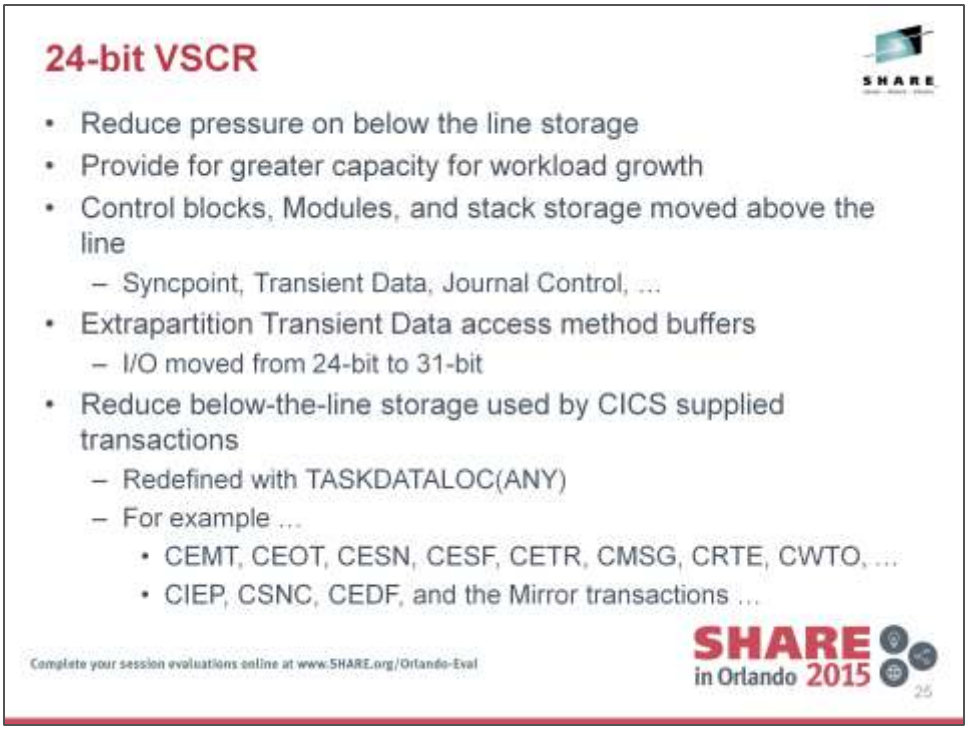
Data collected from RMF report
Workload driven by WSIM.
20 minutes warmup period
Injection rate increased every 10 minutes.
Mean CPU usage per request reported for last 5 minutes of 10 minute interval.



Internal performance improvements are made in many areas of the CICS TS V5.3 open beta offering to help reduce CPU overhead. These include the exploitation of a number of the new hardware instructions introduced with the IBM z9, cache alignment of some key CICS control blocks, the use of prefetch, reduced lock contention within monitoring algorithms, improvements to the MRO session management algorithms, and further tuning of internal procedures.

Improvements in efficiency have noticeable improvements in the CICS trace facility, the CICS monitoring facility, and for MRO connections with high session counts.





Stack storage for Syncpoint, Transient Data and Journal Control moved to 31-bit from 24-bit

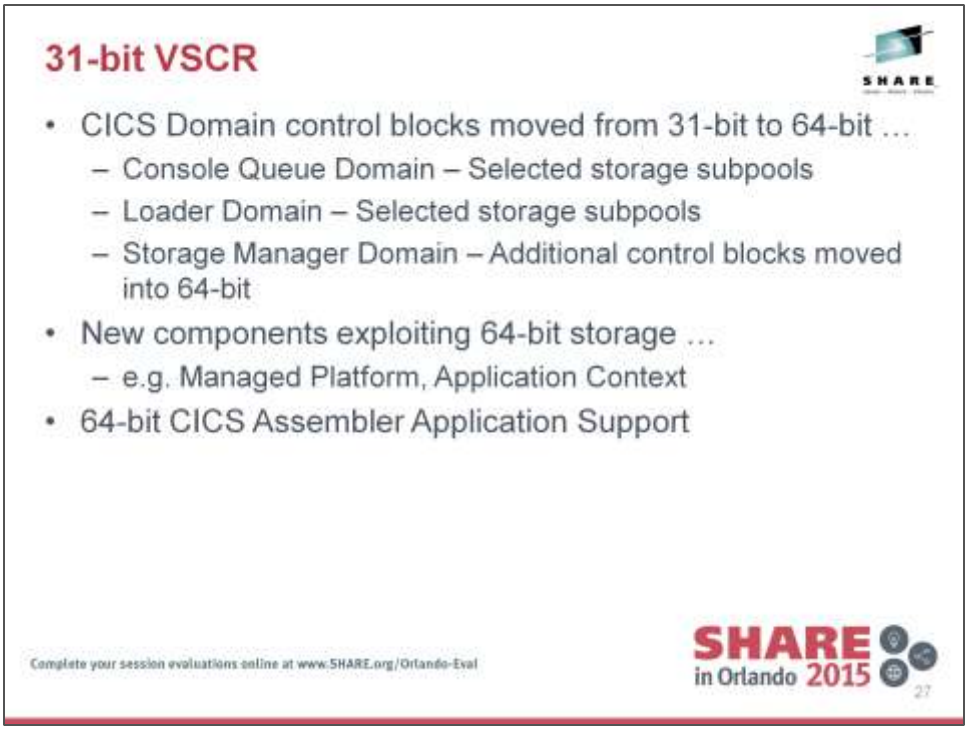
Modules for Journal Control and Transient data move to 31-bit from 24-bit

Extrapartition Transient data buffers and control blocks moves to 31-bit from 24-bit

All CICS transactions now have TASKDATALOC(ANY) as default

24-bit VSCR

- User Exit Global Work Area
  - New GALOCATION parameter on the ENABLE PROGRAM command
    - LOC24 – The global work area is in 24-bit storage (default)
    - LOC31 – The global work area is in 31-bit storage
- COMMAREA on XCTL now in 31-bit
  - Only copied to 24-bit if needed by target program
- Language Environment APAR PM57053 (z/OS V1R13)
  - Reduces LE's use of 24-bit CICS storage in the SDSA



Console queue processing

CICS now uses 64-bit storage for the console queue processing trace table and the console queue transaction entry table. These tables were previously in 31-bit storage taken from the ECDSA.

Storage allocation control blocks

CICS now uses 64-bit storage for the storage element descriptor (SCE) and free storage descriptor (SCF) control blocks, which control storage allocation. Use of 24-bit and 31-bit storage is reduced, especially in systems with a lot of storage allocation activity, for example, systems with subpools that keep an element chain and that have many small records.

Loader control blocks

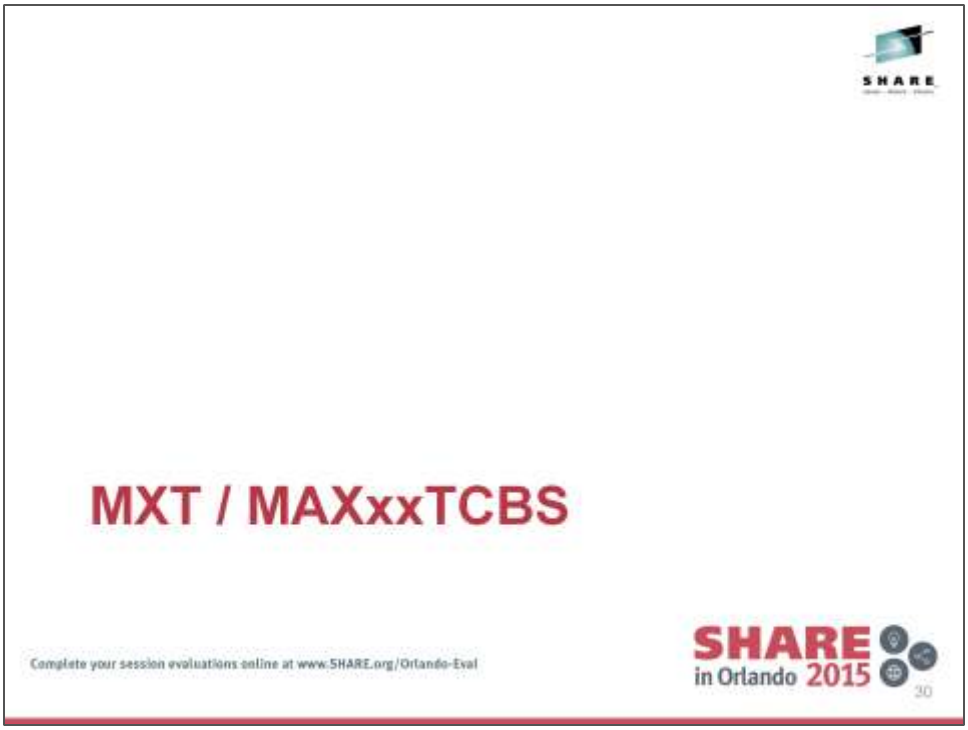
CICS now uses 64-bit storage for the Active Program Element (APE), Current Program Element (CPE), and CSECT descriptor control blocks in the loader domain. These control blocks were previously in 31-bit storage, and could occupy a significant amount of storage. To provide access to the 64-bit storage, the size of the tokens used on the PROGRAM_TOKEN and NEW_PROGRAM_TOKEN options on the XPI calls ACQUIRE_PROGRAM, DEFINE_PROGRAM, and RELEASE_PROGRAM has increased from 4 bytes to 8 bytes. You must change and recompile global user exit programs that use these options. Exit programs that do not use the PROGRAM_TOKEN or NEW_PROGRAM_TOKEN option are not affected.

AMODE(64) Application Support

- 64-bit CICS Assembler Application Support – AMODE(64)
  - Non-LE assembler only
- Provides application support to access large data objects
- Cache large amounts of data above the bar
  - EXEC CICS GETMAIN64 / FREEMAIN64
- Applications can pass data in 64-bit storage using channels
  - EXEC CICS PUT64 CONTAINER / GET64 CONTAINER
  - CICS keeps the container data in 64-bit storage
- EXEC CICS LINK / LOAD / XCTL / RETURN
  - AMODE(64) ↔ AMODE(31) ↔ AMODE(64) ↔ AMODE(24)

AMODE(64) Application Support

- AMODE(64) Assembler Programs are NOT supported as …
  - Global or Task User Exit Programs (GLUEs or TRUEs)
  - User Replaceable Programs (URMs)
- Only the CICS Command Level Programming Interface is supported
  - No support for CICS Resource Manager APIs
    - e.g. DB2, WebSphere MQ, IMS DBCTL, etc, …

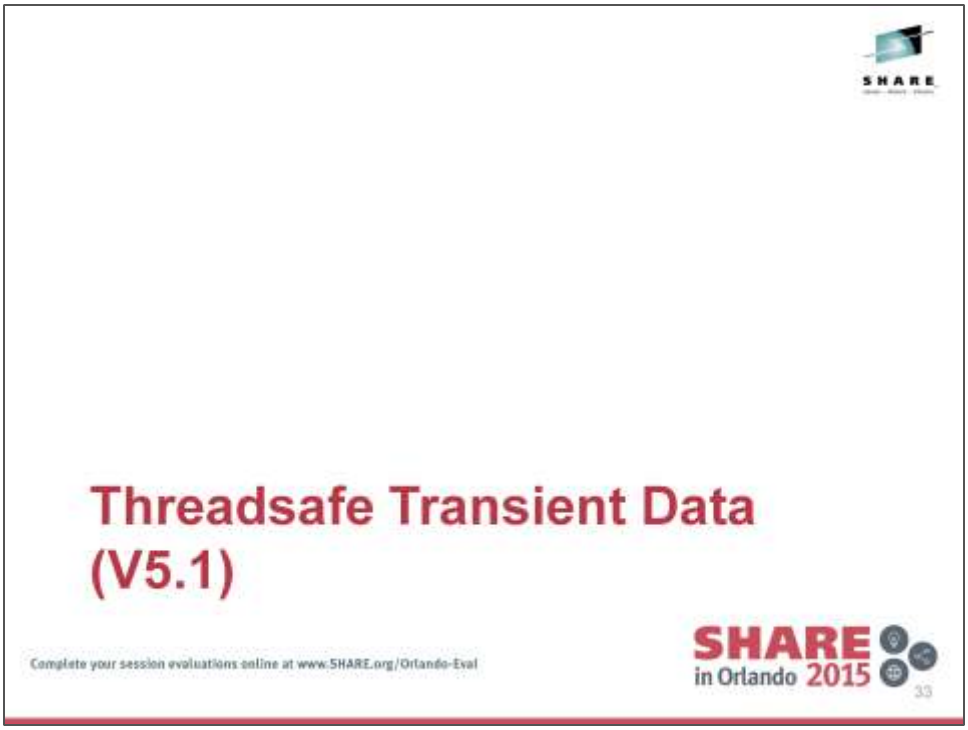


MXT

- MXT limit increased to 2000
- Now defaults to 250
  - Was 500 in V5.1
- Not advisable to run with default MXT value
  - Should be tuned for your environment
- Excessive MXT values can:
  - Waste LSQA storage for MVS performance blocks
  - Consume CPU cycles during MVS WLM scans

MAXOPENTCBS / MAXXPTCBS

- SIT parameter removed in TS V5.1
  - Automatically calculated based on MXT
  - MAXOPENTCBS = (2 * MXT) + 32
  - MAXXPTCBS = MXT
- Reinstated for V5.2
  - If not specified, calculated as per V5.1



PROGRAM CONCURRENCY Recap

- We run CICS with STGPROT=YES
- My application ...
  - … runs USER key
  - … is threadsafe
  - … makes DB2 calls
- How do I maximise time spent on an Open TCB?

CICS TS V4.1 TCB Switching

STGPROT	Exec key	CONCURRENCY	API	Initial TCB	DB2 or MQ command	Threadsafe command	Non-threadsafe command
Yes/No	(any)	QUASIRENT	CICS	QR	QR → L8 → QR	no change	no change
		THREADSAFE		QR	L8	no change	QR
No	(any)	THREADSAFE	OPEN	L8	no change	no change	L8 → QR → L8
Yes	CICS	THREADSAFE	OPEN	L8	no change	no change	L8 → QR → L8
Yes	USER	THREADSAFE	OPEN	L9	L9 → L8 → L9	no change	L9 → QR → L9

CICS TS V4.2+ TCB Switching

STGPROT	Exec key	CONCURRENCY	API	Initial TCB	DB2 or MQ command	Threadsafe command	Non-threadsafe command
Yes/No	(any)	QUASIRENT	CICS	QR	QR → L8 → QR	no change	no change
		THREADSAFE		QR	L8	no change	QR
		REQUIRED		L8	no change	no change	L8 → QR → L8
No	(any)	THREADSAFE	OPEN	L8	no change	no change	L8 → QR → L8
		REQUIRED		L8	no change	no change	L8 → QR → L8
Yes	CICS	THREADSAFE	OPEN	L8	no change	no change	L8 → QR → L8
		REQUIRED		L8	no change	no change	L8 → QR → L8
Yes	USER	THREADSAFE	OPEN	L9	L9 → L8 → L9	no change	L9 → QR → L9
		REQUIRED		L9	L9 → L8 → L9	no change	L9 → QR → L9

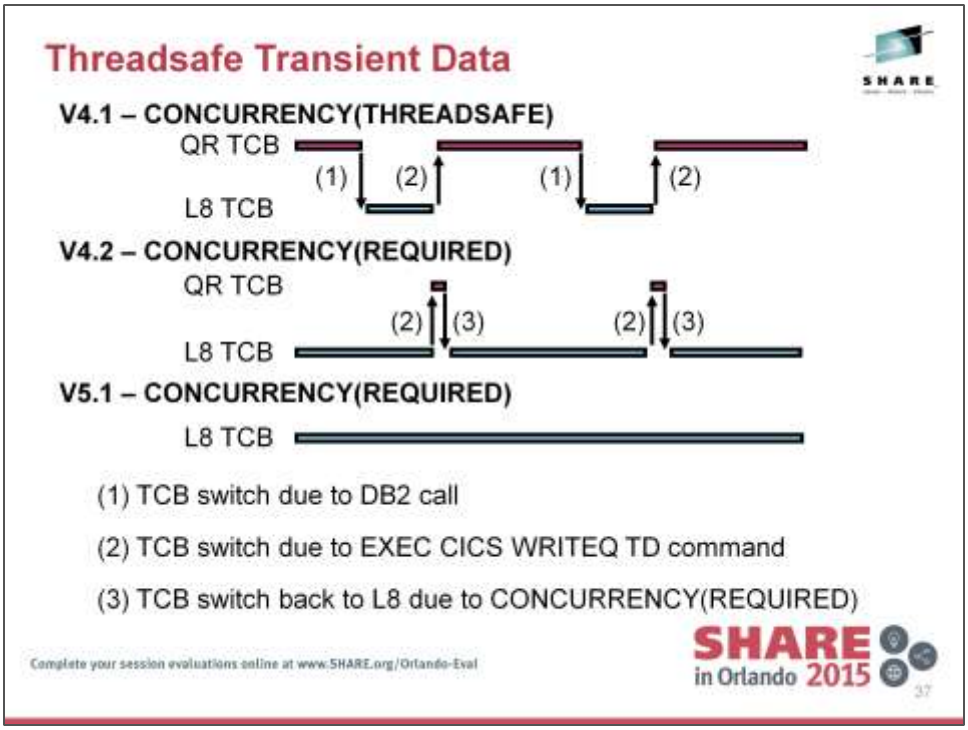


Chart shows an application which alternately executes DB2 SQL calls and then WRITEQ TD commands.

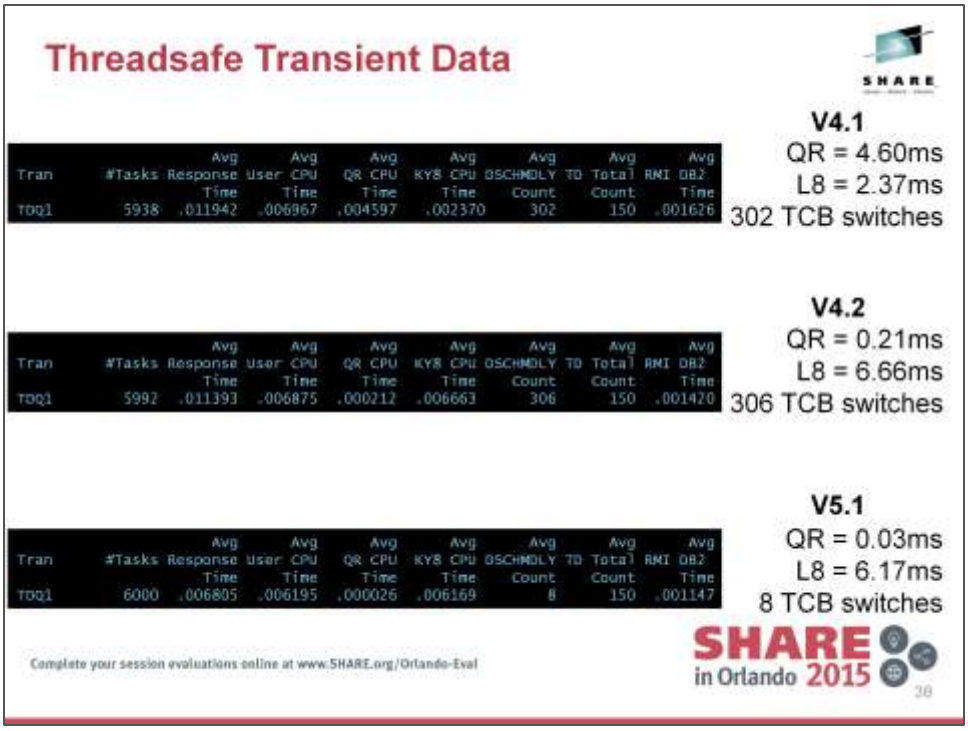
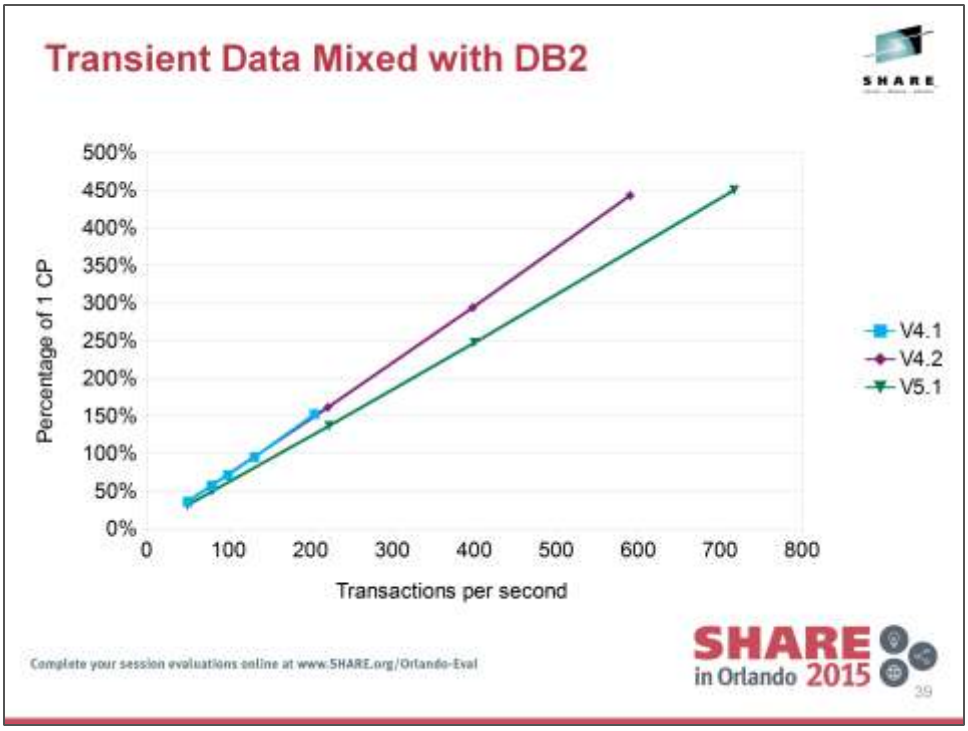


Chart shows a extracts from CICS Performance Analyzer reports for each of the various CICS TS levels.

V4.1 shows a significant number of TCB switches, with a large fraction of CPU consumed on the QR TCB.

V4.2 introduced CONCURRENCY(REQUIRED), which does not reduce the TCB switches, but reduces significantly the amount of CPU time the application spends executing on the QR TCB.

V5.1 introduces threadsafe transient data, which removes the need to switch the the QR TCB for the WRITEQ TD command.



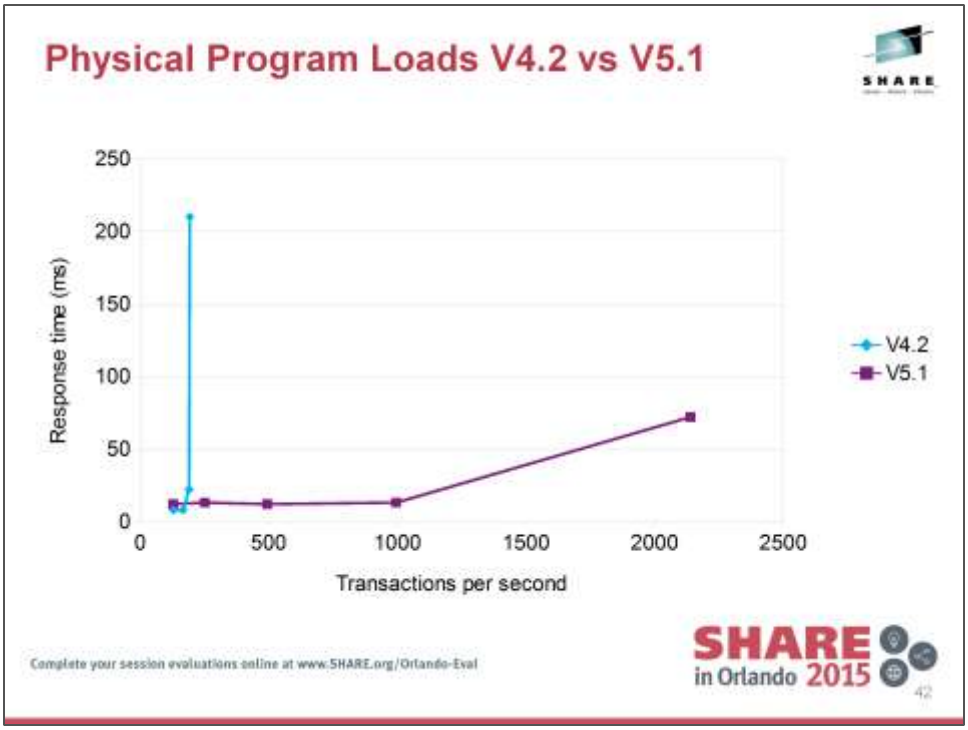
Note that the V4.1 line hits a limit around the 210 transactions per second mark. This is because (as shown on the previous slide), each transaction costs around 4.60ms of CPU time on the QR TCB. Therefore, the maximum throughput for this transaction will be 1000 ms / 4.60 ms/tran = 217 transactions per second.

The V4.2 and V5.1 lines do not see this limit as there is significantly less CPU time spent on the QR TCB.

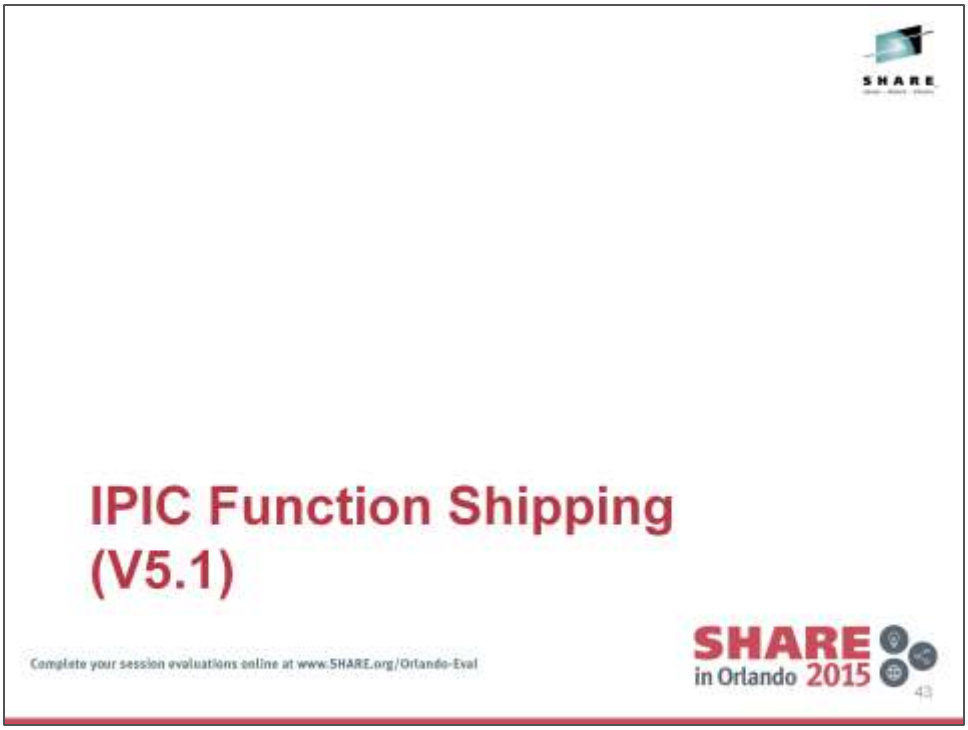
The V5.1 line is slightly lower than the V4.2 line due to the reduction in CPU cost of the incurred TCB switches.

Threadsafe Program Load

- When running on an open TCB and a CICS program load is requested there is no longer a TCB switch to the RO TCB
  - EXEC CICS LINK, LOAD, XCTL, …
- CICS RO TCB will still be used for …
  - CICS program LOADs when NOT running on an Open TCB
  - DFHRPL and LIBRARY Dataset Management
- Updated Loader global statistics
  - New statistics on RO TCB program load requests
  - Load time recorded by module
- Benefits …
  - Reduced contention for the single CICS RO TCB
  - Reduced path length – RO TCB switch eliminated
  - Significantly increased potential CICS program LOAD capacity

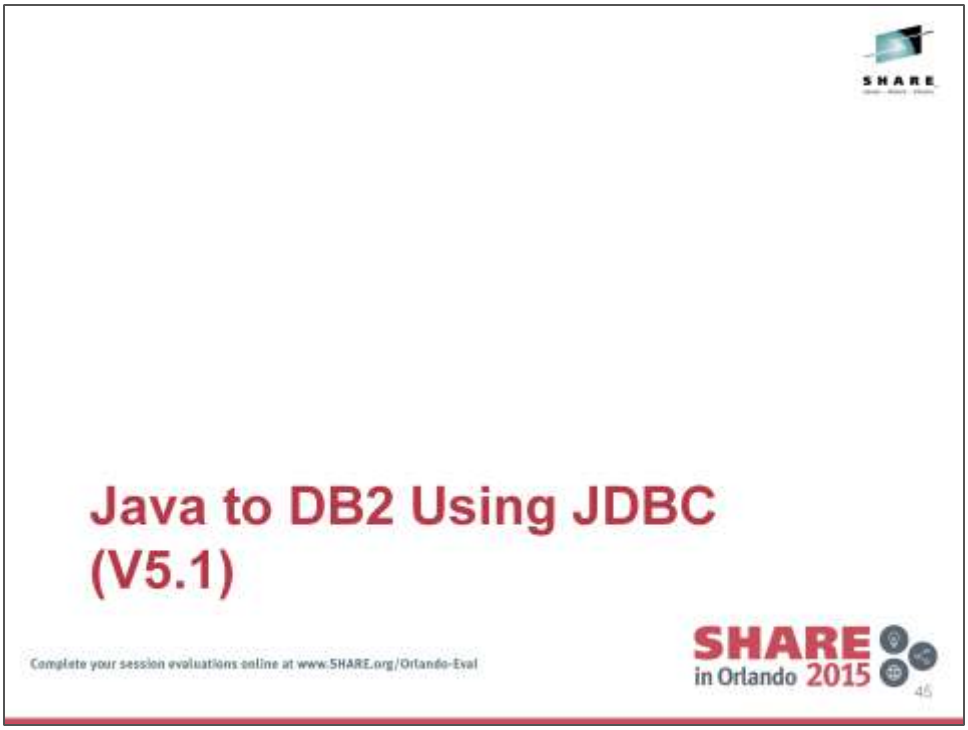


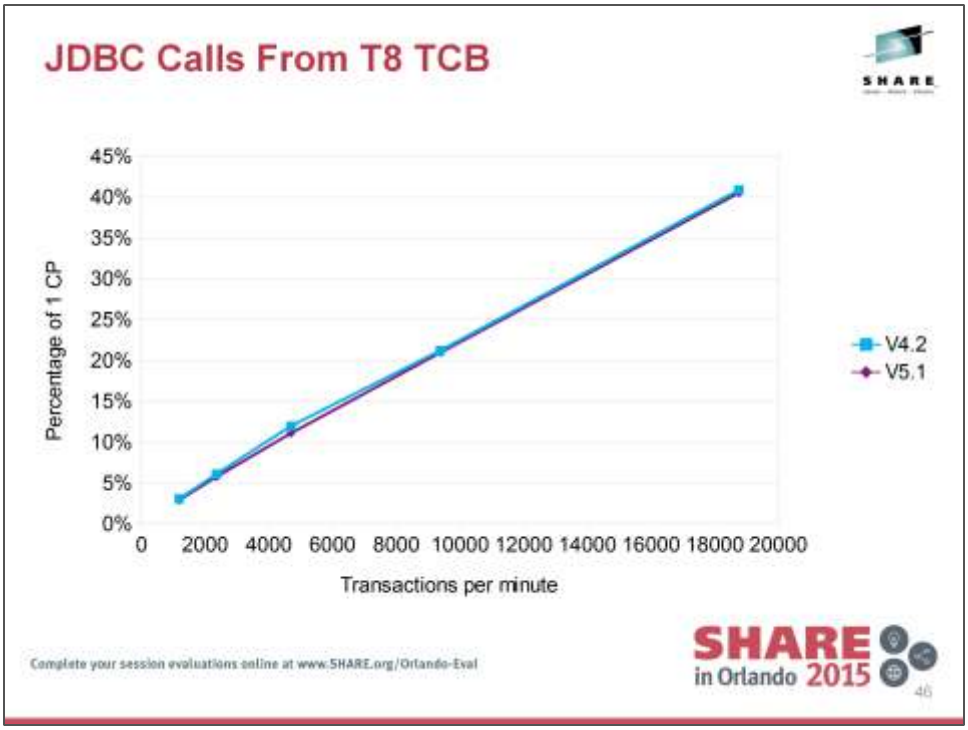
With CICS TS V4.2, the RO TCB quickly reaches capacity, while V5.1 shows loading on an open TCB, which scales significantly better.



IPIC Function-Shipping

- V4.2 – Mirror task uses Open TCB
- V5.1 – Originating task uses Open TCB
- Function-ship performance
  - Response times comparable to XCF
  - Response times better than LU6.2
  - Better throughput achievable than LU6.2



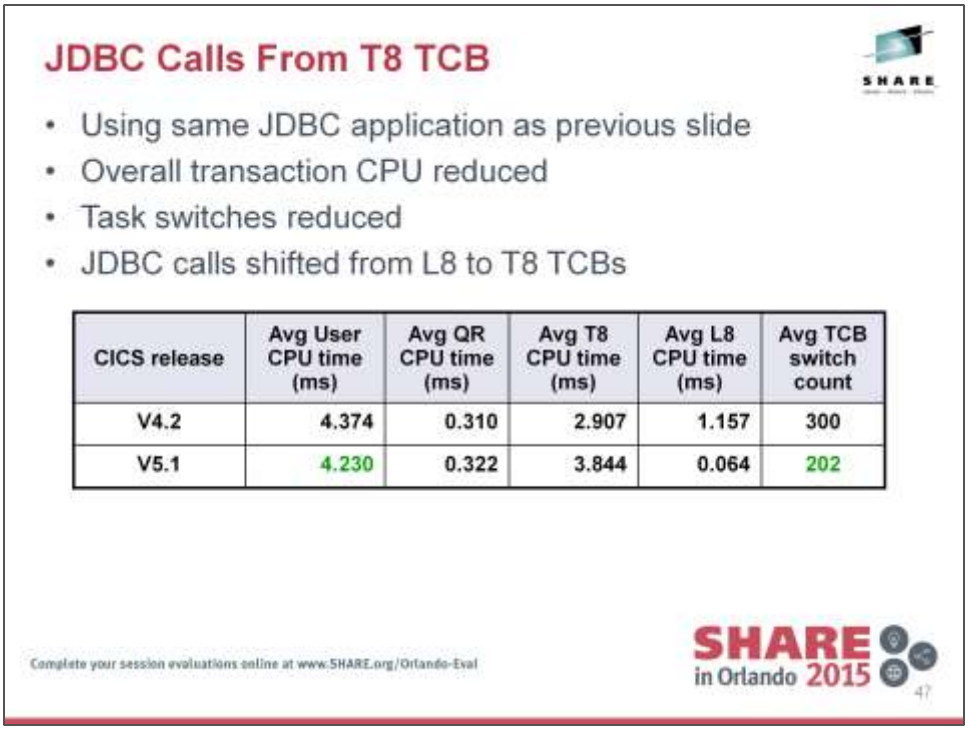


Benchmark used a modified version of the CICS-supplied JDBC sample application CICSDB2DynamicSQLExample.

Application reads 43 rows from DB2 table and writes results to CICS terminal to give a mix of JDBC and JCICS calls.

Both measurements use DB2 V10.

Both configurations scale well, while CICS V5.1 gets a small benefit from reduced TCB switching.



Modified CICSDB2DynamicSQLExample to write results to CICS terminal rather than HFS file.

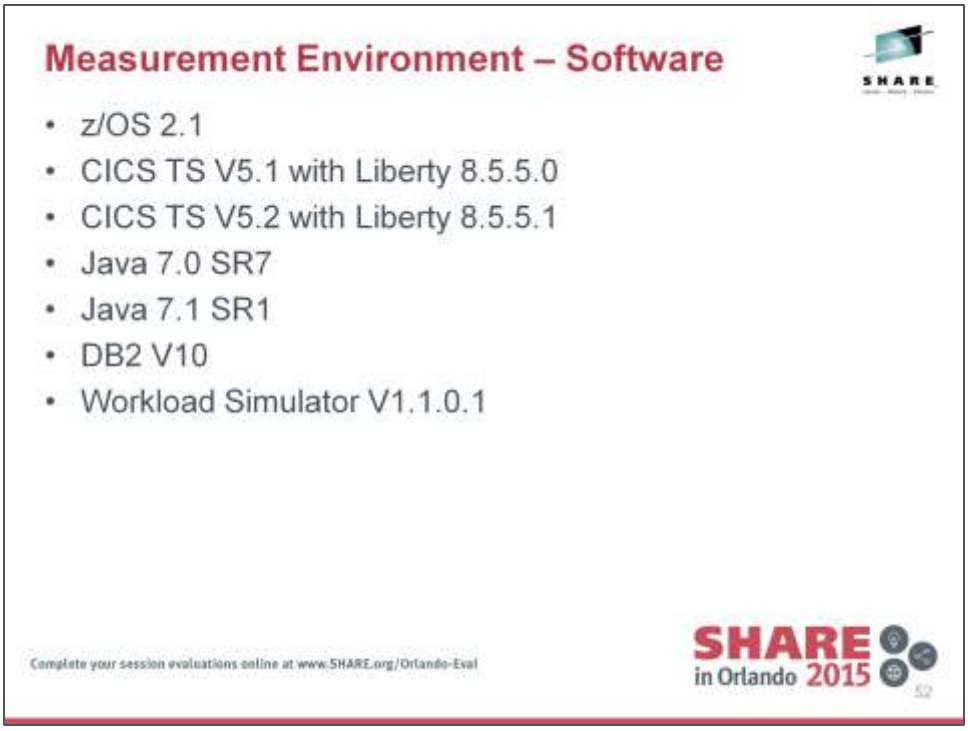
Small amount of L8 time used for SYNCPOINT at transaction completion.

Threadsafe-enabled SPI

- V5.1
  - TASK (SET)
  - TRACEDEST (INQUIRE / SET)
  - TRACEFLAG (INQUIRE / SET)
  - TRACETYPE (INQUIRE / SET)
- V5.2
  - PROGRAM (INQUIRE / SET / DISCARD)
  - TRANSACTION (INQUIRE / SET / DISCARD)
  - SYSTEM (INQUIRE / SET)
  - DISPATCHER (INQUIRE / SET)
  - MVSTCB (INQUIRE)
  - MONITOR (INQUIRE / SET)
  - STATISTICS (EXTRACT / INQUIRE / SET)

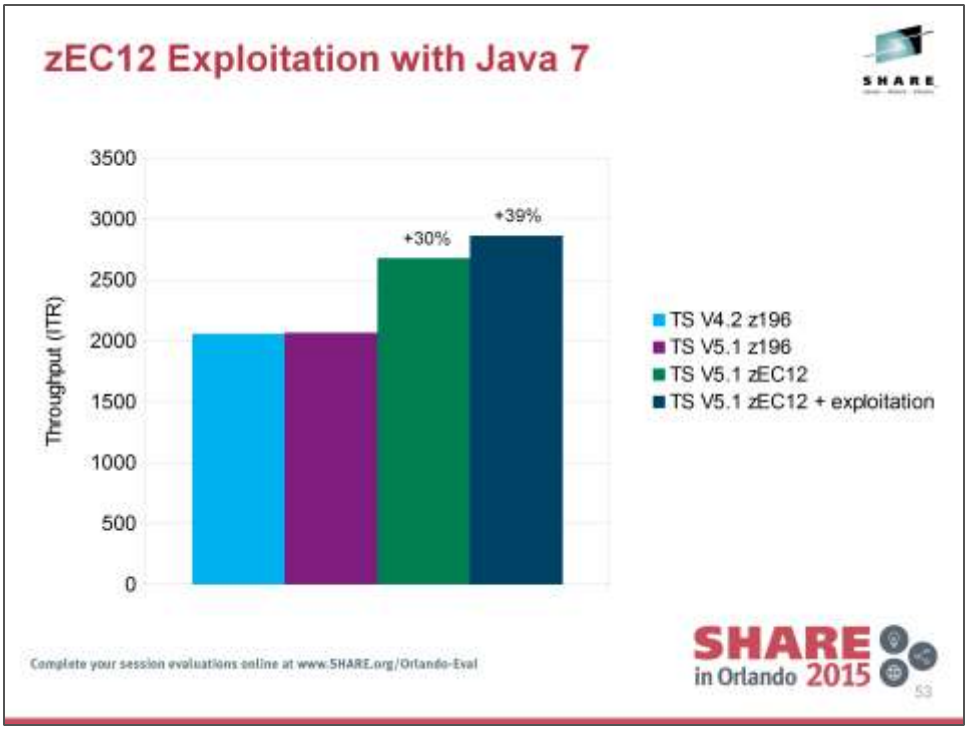
Measurement Environment – Hardware

- zEC12 2827-779 model HA1
  - Target LPAR with 3 dedicated CPs and 1 dedicated zIIP
  - Driver LPAR with 3 dedicated CPs
- DASD DS8800
- Internal Coupling Facility with ICP links



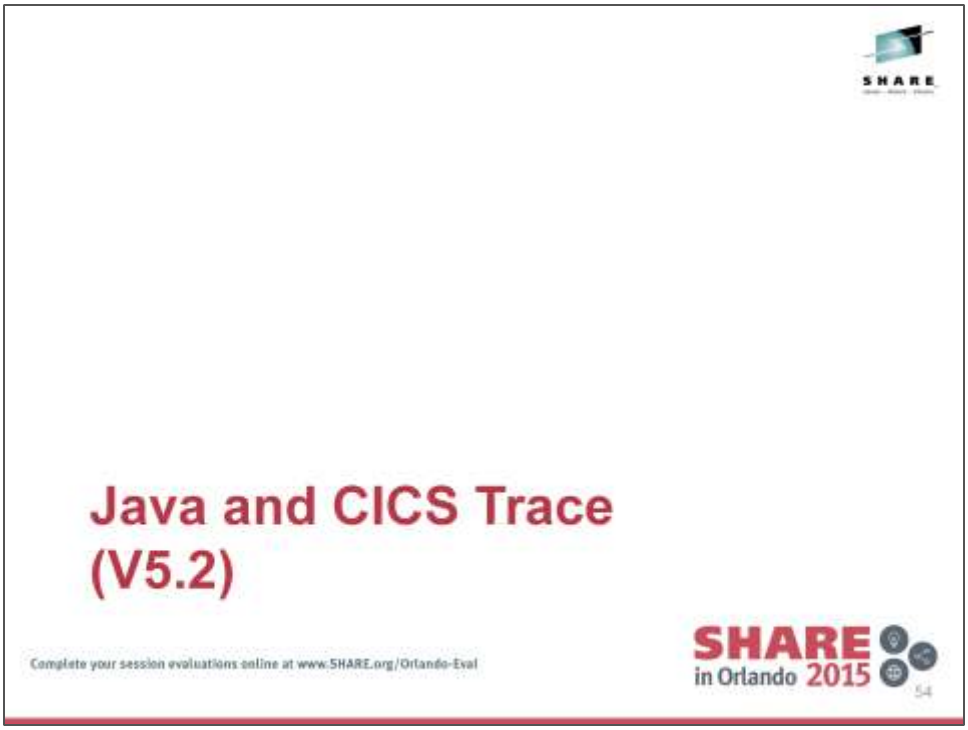
CICS V5.1 GA included Liberty 8.5.0.0.

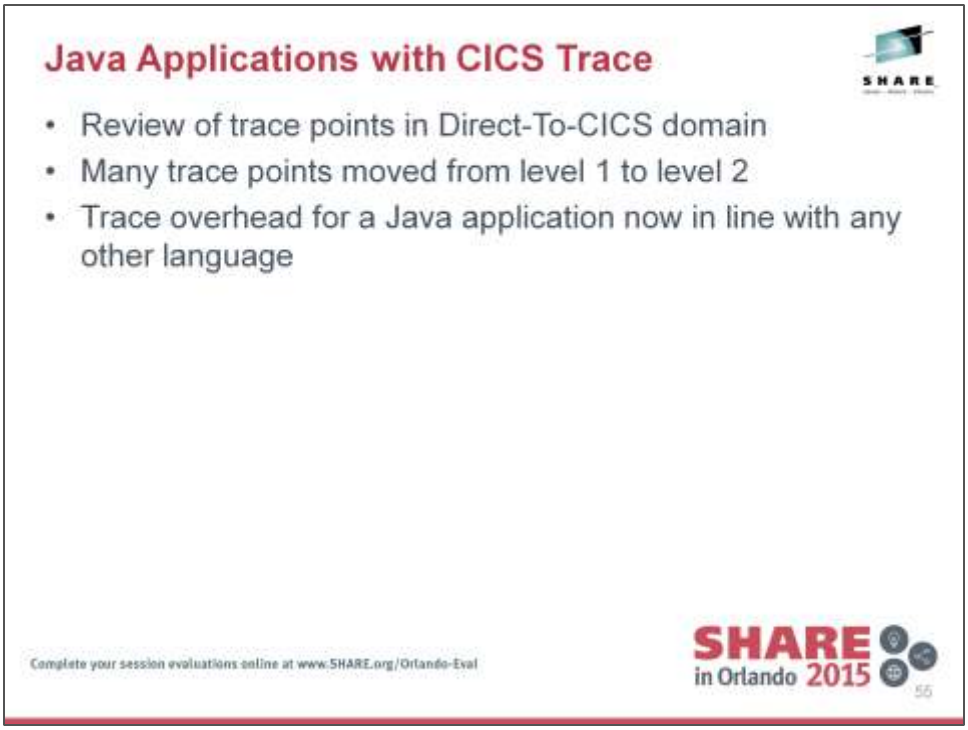
Tests were run at V5.2 GA time. Several APARs to update the level of Liberty on both V5.1 and V5.2 since these measurements were run.



z196 configuration: an LPAR on a 2817-779 machine with 4 dedicated CPs - considered to be equivalent to a 2817-704. Running z/OS V1R13
zEC12 configuration: an LPAR on a 2827-778 machine with 4 dedicated CPs - considered to be equivalent to a 2827-704. Running z/OS V1R13
CICS V5.1 used Java 7 SR3
CICS V4.2 used Java 6.0.1 SR3
Data collected from RMF report
zEC12 expoitation enabled with:
–Xaggressive and -Xjit:noResumableTrapHandler
Simple Java workload shows 24% improvement on zEC12 and 25% with –Xaggressive option, in line with LSPR expectations
https://www.ibm.com/servers/resourcelink/lib03060.nsf/pages/lsprITRzOSv1r13

Using complex Java workload – Axis2 webservice
Equivalent throughput using CICS V5.1 on z196 compared to CICS V4.2
30% improvement in throughput using CICS V5.1 on zEC12 compared to CICS V4.2 on z196
39% improvement in throughput using CICS V5.1 with Java 7 zEC12 exploitation compared to CICS V4.2 on z196





A thorough review of the trace points took place and many were changed to only be emitted with L2 trace enabled.

The result is that enabling trace in a region running Java costs approximately the same as would enabling trace for an equivalent application in a non-Java environment.

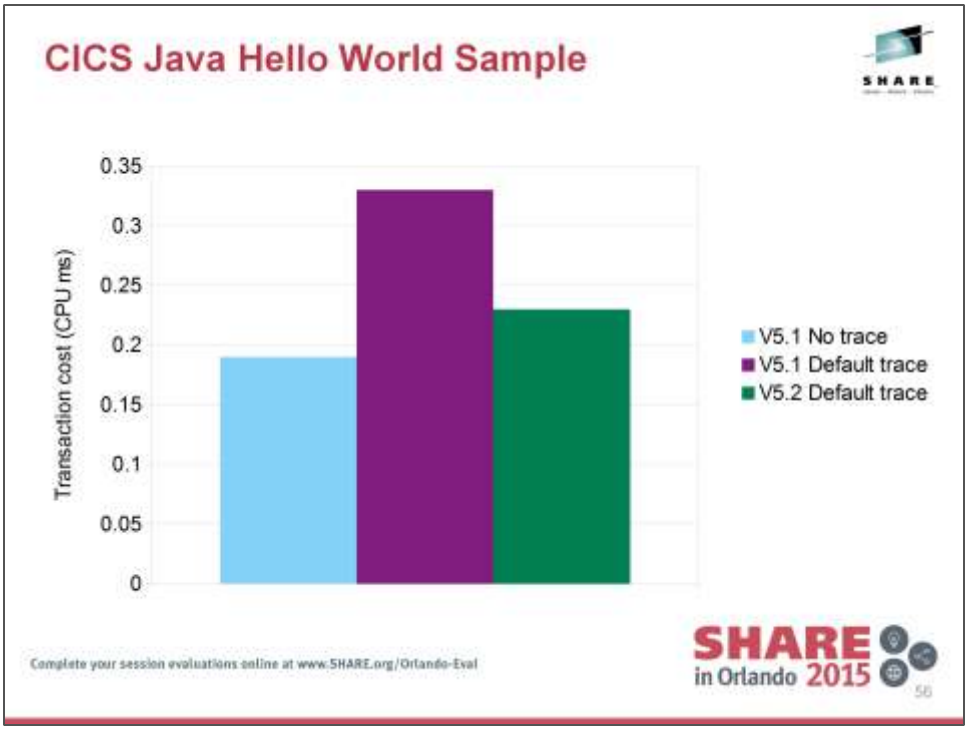
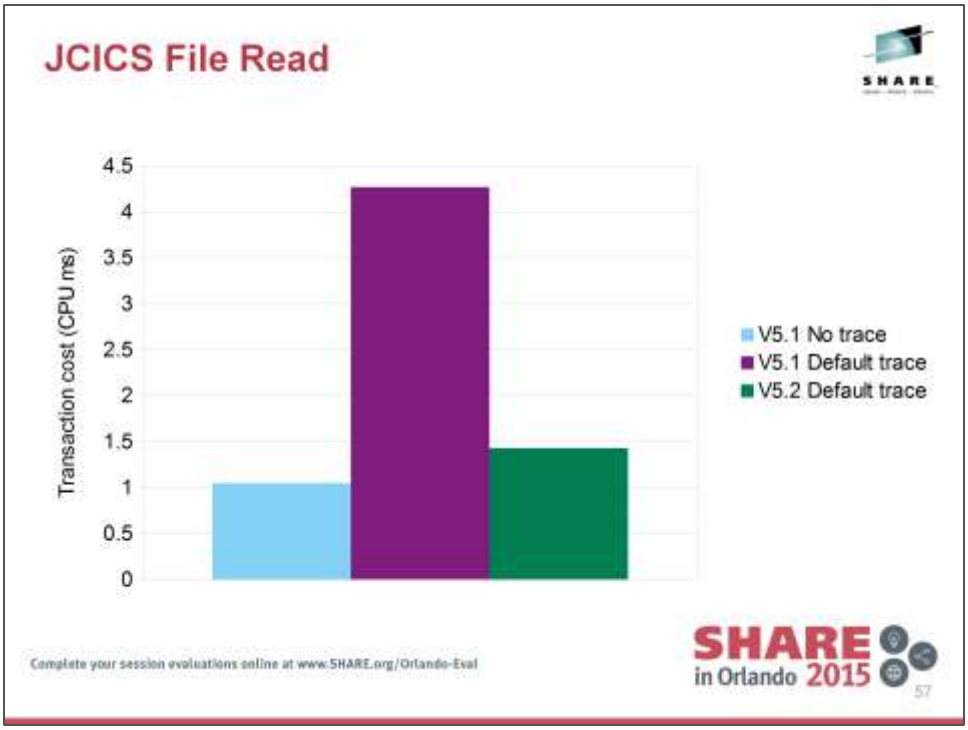


Chart shows that enabling default trace in V5.1 adds a large overhead to the region.

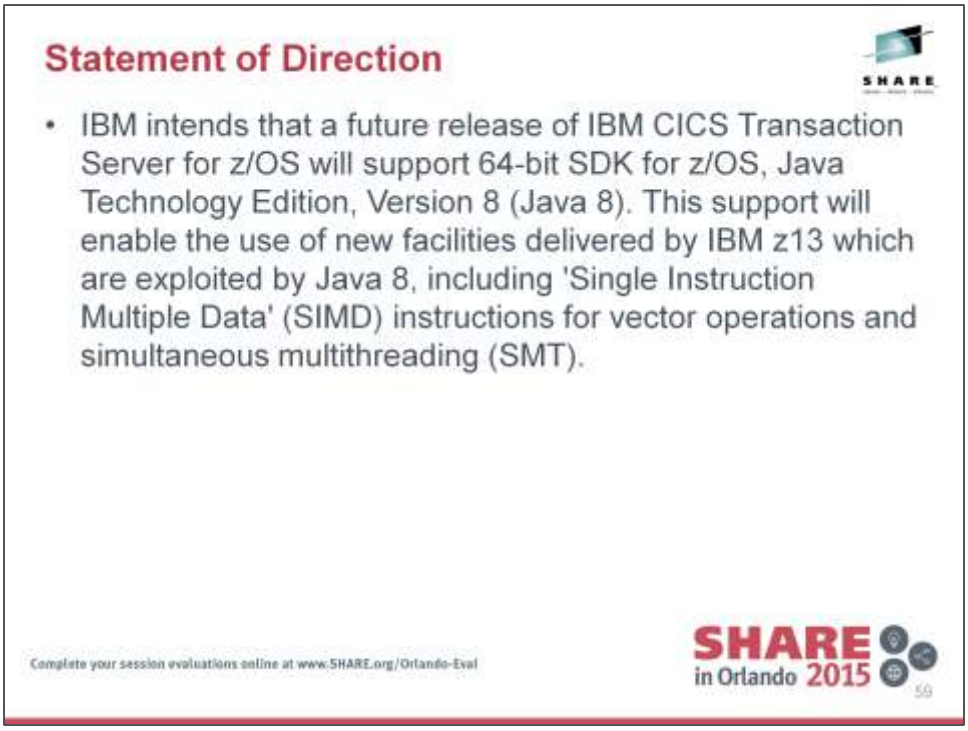
Enabling the same level of trace in V5.2 significantly reduces this overhead.



Application performs 120 JCICS FILE READ operations.

Chart shows that enabling default trace in V5.1 adds a large overhead to the region.

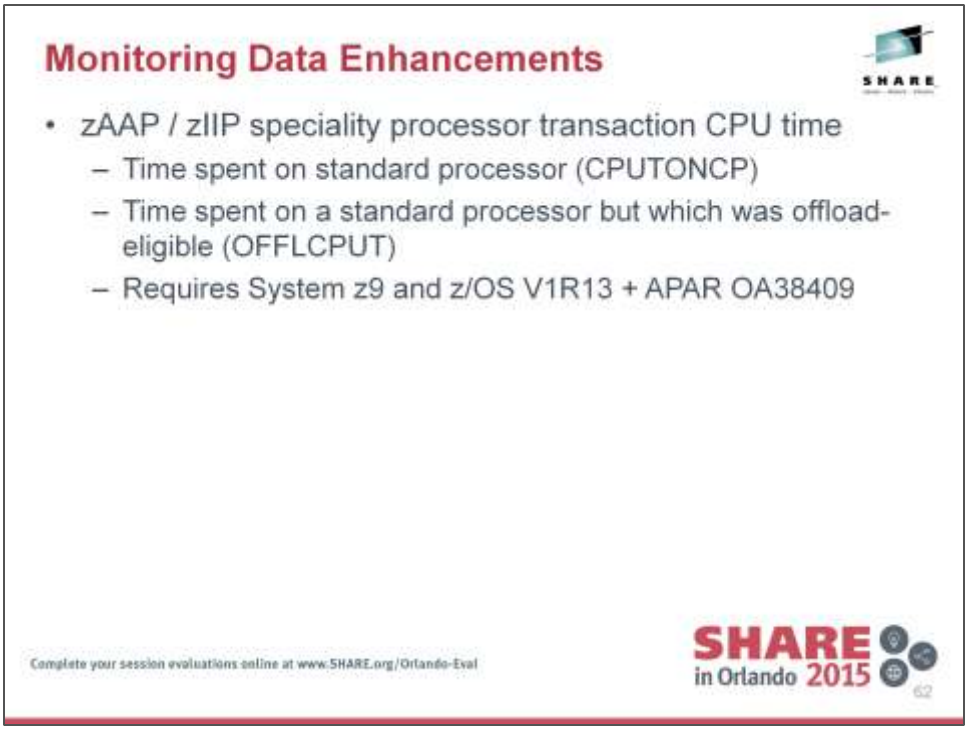
Enabling the same level of trace in V5.2 significantly reduces this overhead.



IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Monitoring Data Enhancements

- Transaction wait times
  - Intra/extra-partition TD queue lock waits (TDILWTT / TDELWTT)
  - Exclusive control of VSAM CI wait time (FCXCWTT)
  - VSAM string wait time (FCVSWTT)
  - IPIC session allocate wait time (ISALWTT)
  - RO and SO TCB delay (ROMODDLY / SOMODDLY)
  - MRO / LU6.1 / LU6.2 session allocate wait time (TCALWTT)
- Transaction performance related to region load
  - Current active task count and MXT setting (CURTASKS / MAXTASKS)
- Inbound SSL cipher code (SOCIPHER)



The new CPUTONCP and OFFLCPUT fields in the DFHTASK performance class group for a transaction can be used to calculate the processor time that a task spends on a zIIP or zAAP specialty processor, and also show you the processor time that the task could have spent on a specialty processor.

Field 436, CPUTONCP, shows the total task processor time spent on a standard processor. To calculate the task processor time spent on a specialty processor, subtract the time recorded in this field from the time recorded in field 008, USRCPUT.

Field 437, OFFLCPUT, shows the total task processor time that was eligible for offload to a specialty processor, but actually ran on a standard processor. To calculate the total task processor time that was not eligible for offload, subtract the time recorded in this field from the time recorded in field 436, CPUTONCP.

To calculate the total task processor time that was either actually spent on a specialty processor, or eligible to be spent on a specialty processor, use the following equation:
(OFFLCPUT + (USRCPUT - CPUTONCP))

Note: The times shown in the CPUTONCP and OFFLCPUT fields are only available when running on a system that supports the Extract CPU Time instruction service that is available on IBM System z9® or later hardware. For z/OS, Version 1 Release 13, the PTF for APAR OA38409 must also be applied.

Monitoring Data Enhancements

- Physical hardware environment
  - CEC Machine Type and Model ID (CECMCHTP / CECMDLID)
- Application task and shared storage usage and waits
  - Fields updated to include 64-bit storage areas
- Channels and containers
  - Fields now include PUT64 / GET64 CONTAINER calls
- Number of exceeded policy rule thresholds (MPPRTXCD)
- Application context information
  - Platform, application, operation name
  - Major, minor, micro version numbers
- Default value of RMI data collection option changed to YES

Enhanced Statistics Information

- Dispatcher Statistics
  - Global, TCB mode, TCB pool
- Monitoring Statistics
- Transaction Statistics
- JVMPROGRAM, LIBRARY, PROGRAM, URIMAP
  - Enhanced to include Application, Platform, version, and entry point information
  - Enhanced to include private variants of resource

Dispatcher Statistics – Global

- Last Excess TCB Scan (DSGLXSCN)
  - The date and time of the last CICS dispatcher excess MVS TCB scan
- Last Excess TCB Scan–No TCB Detached (DSGLXSND)
  - The date and time of the last CICS dispatcher excess MVS TCB scan that did not detach any TCBs

Dispatcher Global Report


```
Dispatcher Start Date and Time. . . . . : 05/16/2014  04:04:34.9633
Address Space CPU Time. . . . . :     00:00:29.882586
Address Space SRB Time. . . . . :     00:00:16.516442
Current number of dispatcher tasks. . . . . :                30
Peak number of dispatcher tasks . . . . . :                  75
Current ICV time (msec) . . . . . :                        1000
Current ICVR time (msec). . . . . :                        5000
Current ICVTSD time (msec). . . . . :                       100
Current PRTYAGE time (msec) . . . . . :                    1000
Current MRO (QR) Batching (MROBTCH) value . :                 1
Last Excess TCB Scan. . . . . : 05/16/2014  05:28:10.1478
Number of Excess TCB Scans. . . . . :                         1
Last Excess TCB Scan - No TCB Detached. . . : 05/16/2014  05:28:10.1478
Excess TCB Scans - No TCB Detached. . . . . :                 1
Number of Excess TCBs Detached. . . . . :                     0
Average Excess TCBs Detached per Scan . . . :                 0
Number of CICS TCB MODEs. . . . . :                          18
Number of CICS TCB POOLs. . . . . :                           4
```

Dispatcher Statistics – TCB Mode

- Dispatchable Queue – Current (DSGTMCDQ)
  - The current number of dispatchable tasks queued for the TCB.
- Dispatchable Queue – Peak (DSGTMPDQ)
  - The peak number of dispatchable tasks that have been queued for the TCB.
- Dispatchable Queue – Average (DSGTMADQ)
  - The average number of dispatchable tasks that have been queued for the TCB.

Dispatcher TCB Mode Report

TCB Mode	Open	TCB Pool	< TCBs Attached > Current	Peak	<- TCBs In Use -> Current	Peak	TCB Attaches	<- Dispatchable Queue -> Current	Peak	Average
QR	No	N/A	1	1	1	1	0	1	27	1.12
RO	No	N/A	1	1	1	1	0	1	1	1.00
CO	Unk	N/A	0	0	0	0	0	0	0	0.00
SZ	Unk	N/A	0	0	0	0	0	0	0	0.00
RP	Unk	N/A	0	0	0	0	0	0	0	0.00
FO	No	N/A	1	1	1	1	0	0	0	0.00
SL	No	N/A	1	1	1	1	0	0	0	0.00
SO	No	N/A	1	1	1	1	0	0	0	0.00
SP	No	N/A	1	1	1	1	0	0	0	0.00
EP	No	N/A	2	2	2	2	0			
TP	Unk	N/A	0	0	0	0	0			
D2	Unk	N/A	0	0	0	0	0			
S8	Unk	N/A	0	0	0	0	0			
L8	Yes	Open	1	1	0	1	0			
L9	Unk	N/A	0	0	0	0	0			
X8	Unk	N/A	0	0	0	0	0			
X9	Unk	N/A	0	0	0	0	0			
T8	Unk	N/A	0	0	0	0	0			
Totals			9		8		0			

Dispatcher Statistics – TCB Pool

- Time Max TCB Pool Limit last reached (DSGLTCBL)
  - The time at which the pool reached the maximum TCB limit.

Dispatcher TCB Pool Report


```
TCB Pool. . . . . . . . . . . . . . . . . . . . . :          OPEN
Current TCBs attached in this TCB Pool. . . . . :           170  ...
Peak TCBs attached in this TCB Pool . . . . . . :           170  ...
Max TCB Pool limit (MAXOPENTCBS). . . . . . . . :           170  ...
Time Max TCB Pool Limit last reached. . . . . . : 15:47:39.2782  ...
Total Requests delayed by Max TCB Pool Limit. . :           819  ...
Total Max TCB Pool Limit delay time . . . . . . : 00:01:57.2105  ...
Current Requests delayed by Max TCB Pool Limit. :             0  ...
Current Max TCB Pool Limit delay time . . . . . : 00:00:00.0000  ...
Peak Requests delayed by Max TCB Pool Limit . . :            67  ...
                                                                 ...
                                                                 ...
    ...  Current TCBs in use in this TCB Pool. . . . . . :             7
    ...  Peak TCBs in use in this TCB Pool . . . . . . . :           170
    ...  Times at Max TCB Pool Limit (MAXOPENTCBS) . . . :           198

    ...  Total Number of TCB Mismatch waits. . . . . . . :          5092
    ...  Total TCB Mismatch wait time. . . . . . . . . . : 00:13:26.4493
    ...  Current TCB Mismatch waits. . . . . . . . . . . :             0
    ...  Current TCB Mismatch wait time. . . . . . . . . : 00:00:00.0000
    ...  Peak TCB Mismatch waits . . . . . . . . . . . . :            78
    ...  Requests Delayed by MVS storage constraint. . . :             0
    ...  Total MVS storage constraint delay time . . . . : 00:00:00.0000
```


Monitoring Statistics

- User transactions ended (MNGUTNUM)
  - The number of user transactions that have ended.
- System transactions ended (MNGSTNUM)
  - The number of system transactions that have ended.
- Time last user transaction attached (MNGLUTAT)
  - The date and time of the last transaction attach processed by the monitoring domain.
- Time last user transaction ended (MNGLUTCL)
  - The date and time at which the last transaction ended.

Monitoring Statistics

- MXT at last user transaction attach (MNGMXUTA)
  - The current MXT value at the time of the last transaction attached.
- Current tasks at last attach (MNGCAUTA)
  - The current number of user transactions attached in the region at the time of the last transaction attached.
- Average user transaction resp time (MNGAUTRT)
  - The rolling average user transaction response time.
- Peak user transaction resp time (MNGPUTRT)
  - The maximum user transaction response time.
- Peak user transaction resp time at (MNGLUTRT)
  - The timestamp of the maximum user transaction response time.

Monitoring Statistics Report


```
User transactions ended . . . . . :   905698
System transactions ended . . . . . :      11
Time last user transaction attached . : 05/16/2014  05:28:43.5198    ...
Time last user transaction ended. . . : 05/16/2014  05:28:43.5215    ...
Average user transaction resp time. . :             00:00:00.001168
Peak user transaction resp time . . . :             00:00:00.104882
Peak user transaction resp time at. . : 05/16/2014  05:26:55.8512

                        ... MXT at last user transaction attach . :      650
                        ... Current tasks at last attach. . . . . :        8
```

rolling_avg_resp_time:

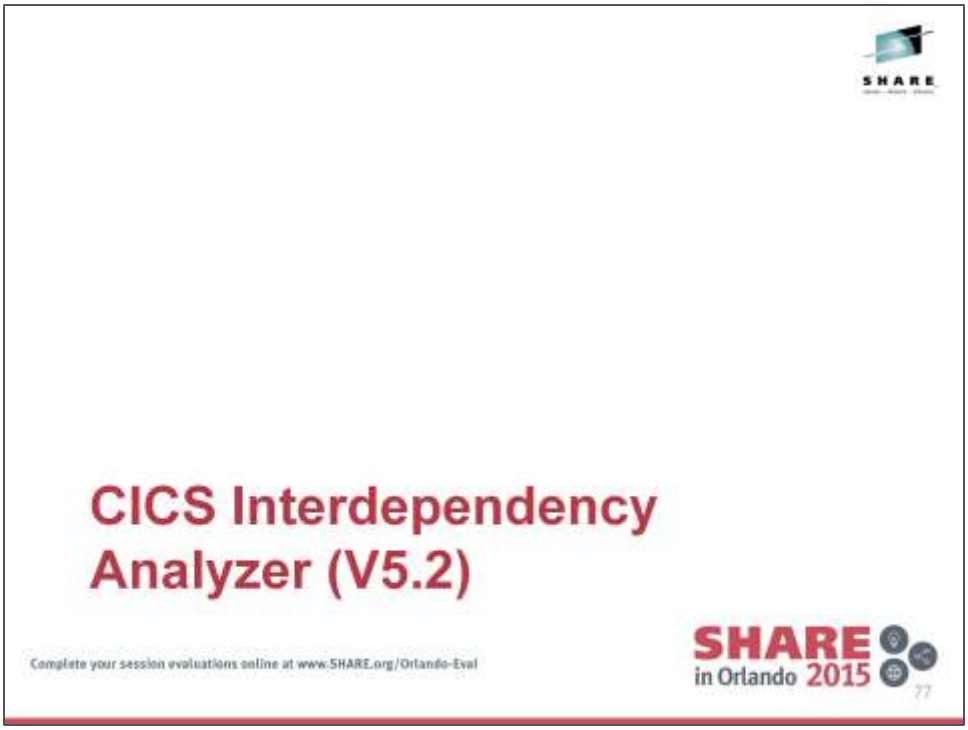
$$\frac{(\text{curr\_avg\_user\_resp\_time} \times \text{num\_completions}) + \text{this\_resp\_time}}{\text{num\_completions} + 1}$$

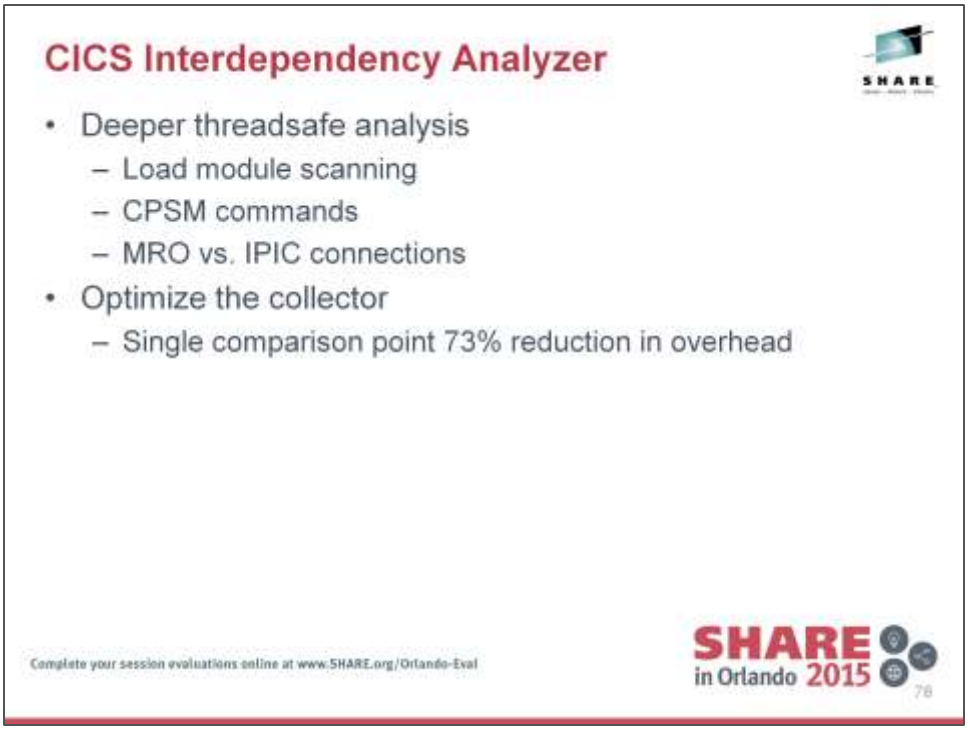
Transaction Manager Statistics

- Time last transaction attached (XMGLTAT)
  - The date and time when the last user transaction was attached
- Time MAXTASKS last changed (XMGLSMXT)
  - The date and time when MXT was last set or changed dynamically
- Time the MAXTASKS limit last reached (XMGLAMXT)
  - The date and time when the number of active user transactions last equalled MXT
- Currently at MAXTASKS limit (XMGATMXT)
  - Indicates whether the CICS region is currently at MXT

Transaction Manager Statistics

Total number of transactions (user + system)		19,274
Current MAXTASKS limit		650
Time MAXTASKS last changed	05/15/2014	12:20:16.9640
Current number of active user transactions		1
Time last transaction attached	05/15/2014	12:40:24.6730
Current number of MAXTASK queued user transactions		0
Times the MAXTASKS limit reached		7
Time the MAXTASKS limit last reached	05/15/2014	12:34:21.7237
Currently at MAXTASKS limit		No
Peak number of MAXTASK queued user transactions		164
Peak number of active user transactions		650
Total number of active user transactions		19232
Total number of MAXTASK delayed user transactions		456
Total MAXTASK queuing time		000-00:00:13
Total MAXTASK queuing time of currently queued user transactions		00:00:00





CICS IA V5.2 can scan load modules to identify CICS commands which are non-threadsafe.

Collector can also understand EXEC CPSM commands, along with being able to differentiate between MRO and IPIC connections.

Following charts cover the optimise the collector item.

Configuration

- DSW Workload
  - BMS / COBOL / VSAM
  - 2 TOR → 2 AOR → 1 FOR topology
  - Constant transaction rate of 3,800 transactions/second
- Hardware
  - zEC12 HA1 – equivalent to 2827-716
- Software
  - z/OS V2.1
  - CICS TS V5.2
  - CICS IA V5.2

CICS IA Parameters

- CICS IA collection file shared via RLS
- Interdependency data collected
- Usage counts maintained
- Dynamic calls monitored
- All APIs and SPIs monitored

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

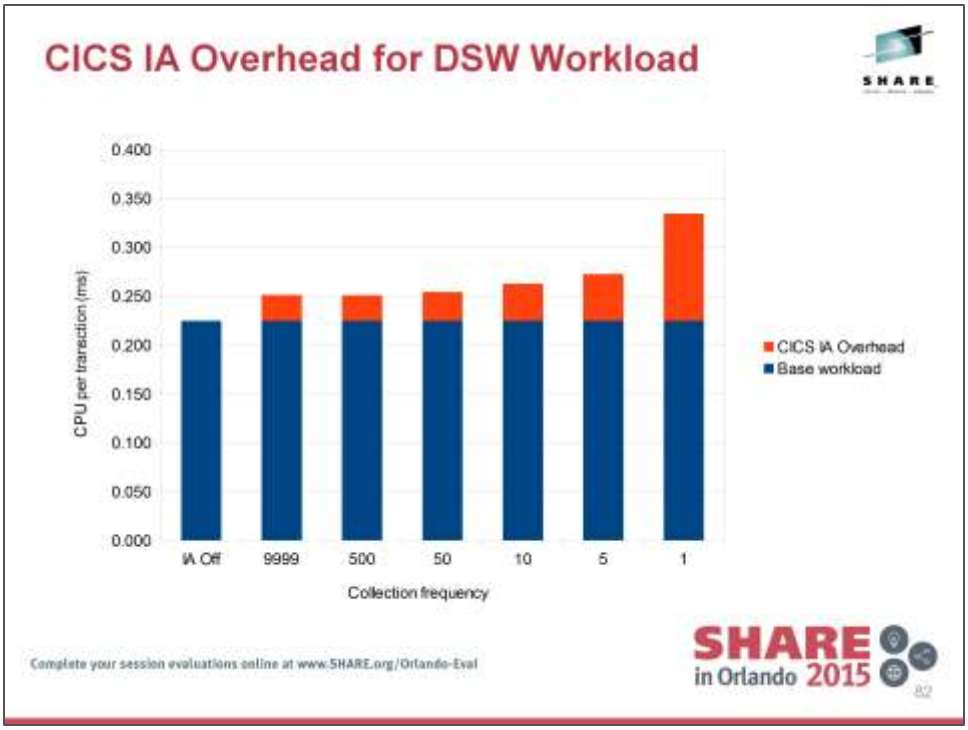
SHARE in Orlando 2015

Full parameters:

```
CINT Collector runtime options for IYCUZC31:
  GLOBAL OPTIONS:
    VSAM_FILE_SHARE=Y HL_TRACE=N RESTORE= N LANGUAGE=E
    DATE_FORMAT='YYYY/MM/DD' TIME_FORMAT='HH:MM:SS 24 hrs'
  GENERAL OPTIONS:
    DATA_TO_COLLECT=I PERIODIC_SAVES=Y CINB_TRIGGER=5
     RESTORE_DATA=N MULTIPLE_SIGNON=N
    MAINTAIN_USAGE_COUNTS=Y DS_SIZE=16   TRANSID_PREFIX=''
    PROGRAM_EXCLUDE_LIST='CIUXPROG' RESOURCE_PREFIX_LIST='CIUPFXTB'
    TRANSACTION_EXCLUDE_LIST='CIUXTRAN'
    DUMP_HLQ='DUMP    ' DYNAMIC_CALL=Y
    TRIGGER FOR TASK COLLECTION=9999 COLLECT LONG RUNNING TASKS=N
    Collect Application Data = 'N'; Application data is not collected
  DATE/TIME OPTIONS:
    HOUR_OF_DAY='YYYYYYYYYYYYYYYYYYYYYYYY' DAY_OF_WEEK='YYYYYYY'
    DAY_OF_MOUNTH='YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY'
    MOUNTH_OF_YEAR='YYYYYYYYYYYY'
  DEPENDENCY OPTIONS.
    CICS API OPTIONS:
      PROGRAMS=Y FILES=Y TRANSACTIONS=Y TASK_CONTROL=Y PRESENTATION=Y
      TS_QUEUES=Y TD_QUEUES=Y JOURNALS=Y DTP=Y COUNTERS=Y FEPI=Y
      WEB_SERVICES=Y EXITS=Y OTHERS=Y EVENT_PROC=Y ATOM_SERVICES=Y
      XML_TRANSFORM=Y WSA_ADDRESSING=Y
    CICS SPI OPTIONS:
      PROGRAMS=Y FILES=Y TRANSACTIONS=Y TEMP_STORAGE=Y
      TRANSIENT_DATA=Y DB2=Y DJAR=Y BR_FACILITY=Y CORBASERVER=Y
      TCPIP_SERVICE=Y FEPI=Y JOURNALS=Y LIBRARY=Y CONNECTIONS=Y
      BTS_PROC=Y BUNDLES=Y ATOM_SERVICES=Y CSD=Y XML_TRANSFORM=Y
      MQ_CONN=Y JVM_SERVICES=Y TERMINALS=Y CICS_SYSTEM=Y TASKS=Y
      DUMPS=Y VTAM_CONN=Y STATISTICS=Y TRACING=Y SHUTDOWN=Y
    DB2/IMS/MQ/CPSM OPTIONS: DB2=N INQUIRE_DB2=Y MQ=N IMS=N CPSM=N
    NATURAL OPTIONS: PROGRAM_CALLS=N ADABAS_CALLS=N
  AFFINITY OPTIONS.
    INTER-TRANSACTION:
      ENQ/DEQ=Y TS_QUEUE=Y ADDRESS_CWA=Y RETRIEVE_WAIT=Y LOAD=Y
      GETMAIN_SHARED=Y CANCEL=Y
    TRANSACTION-SYSTEM:
      INQUIRE/SET=Y ENABLE/DISABLE=Y EXTRACT=Y COLLECT_STATS=Y
      PERFORM=Y RESYNC=Y WAIT=Y DISCARD=Y CREATE=Y CSD=Y
```

Test Measurements

- Use RMF to measure overall CPU and transaction rate
- 5 minute measurement interval
- Vary CICS IA collection frequency
- CICS IA enabled for all five regions



Left column shows the cost of the workload with IA completely disabled.

Every other column shows where the collection frequency has been set to every 1 in *n* tasks. Increasing the frequency of the collection (i.e. towards the right of the chart), increases the associated overhead.

CICS IA Overhead for DSW Workload

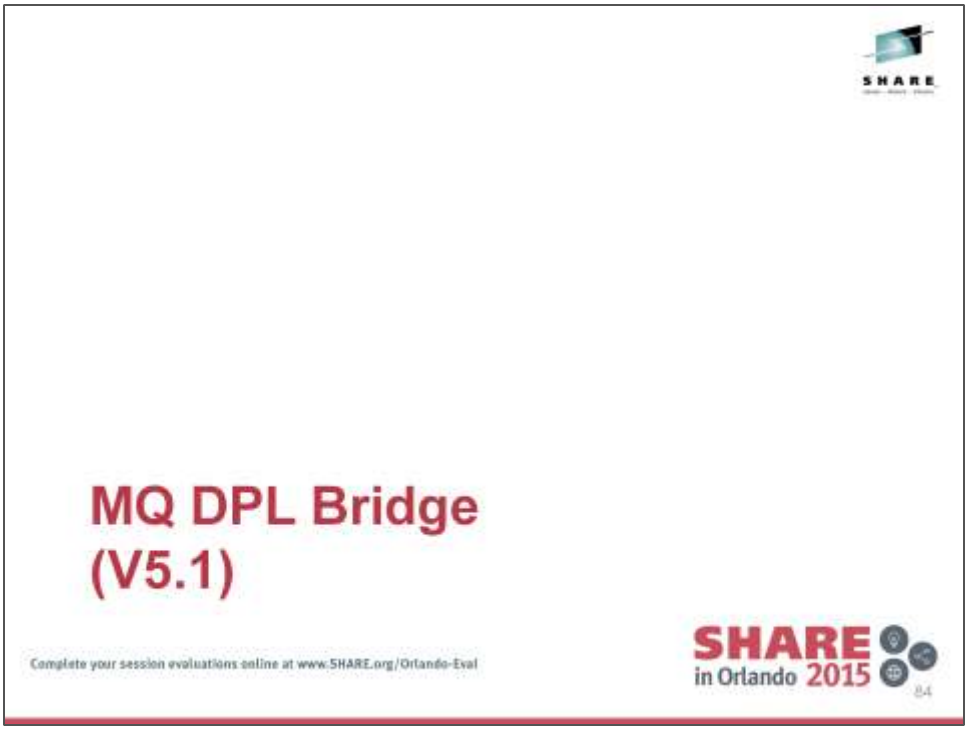
Collect every *n* tasks	Base workload (CPU ms)	CICS IA overhead (CPU ms)	Saving in CPU
IA Off	0.225	0.000	-
1	0.225	0.110	-
5	0.225	0.048	56%
10	0.225	0.038	65%
50	0.225	0.030	73%
500	0.225	0.026	76%
9999	0.225	0.027	76%

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

SHARE in Orlando 2015

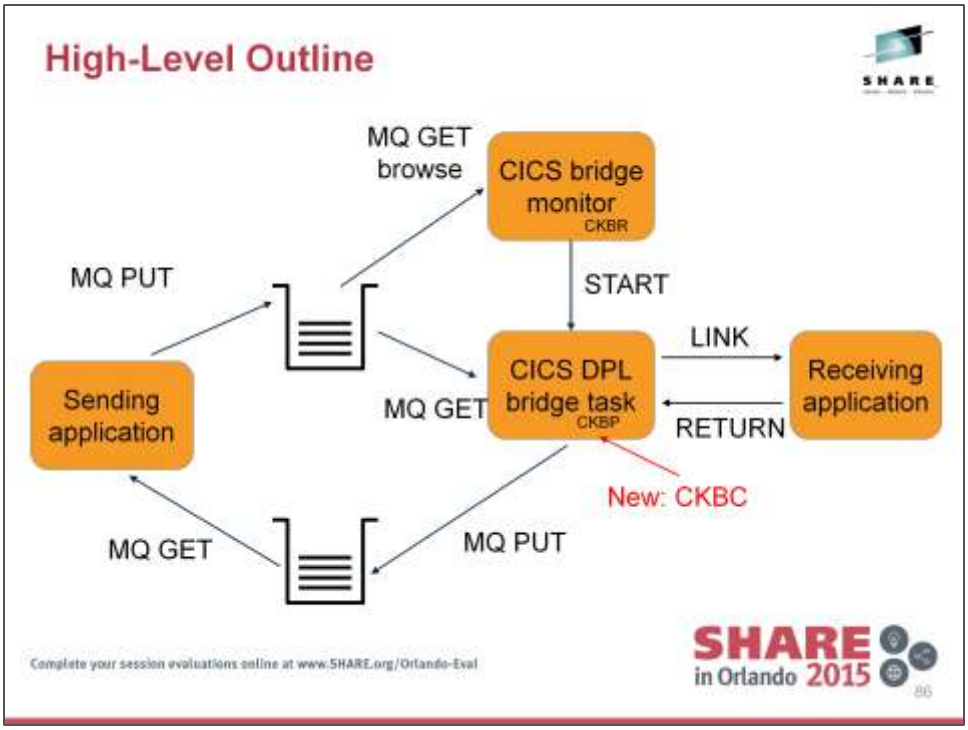
83

"% Saving in CPU" is calculated when comparing with CICS IA collecting data for every task (i.e. second row).



CICS MQ DPL Bridge

- Enables a client application:
  - to invoke a server application running under CICS
  - by sending an MQ message
- Data passed to server app on a LINK API command:
  - in a COMMAREA
    - restricted to 32k of data
    - to send > 32k requires multi-sends and LINKs per UOW
  - in a Container
    - new feature
    - 32k data restriction removed



The EXEC CICS LINK between the CICS DPL bridge task and the receiving application passes a COMMAREA, which is limited to 32kB.

To use this functionality, change the CKBP transaction to the new CKBC transaction and code your applications to accept data on the DFHMQBR_CHANNEL channel, using the DFHREQUEST and DFHRESPONSE containers.

Transactions

•CKBR

- CICS Bridge Monitor transaction – long running task

•CKBP (for Commareas) or CKBC (for Containers)

- CICS DPL Bridge task

MQ calls issued

•CKBR – MQGET BROWSE WAIT

•CKBP/CKBC – MQOPEN and MQGETs

- 1 MQGET per inbound message within the UOW

•CKBP/CKBC – MQOPEN and MQPUTs (when reply-to-queue specified)

- 1 MQPUT per outbound message sent within the UOW
- MQPUT1 used when only 1 outbound message per UOW

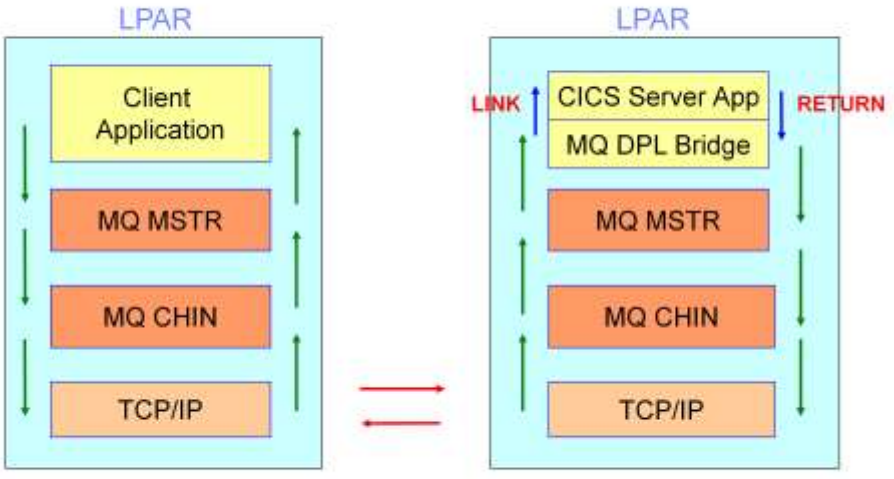
CICS MQ DPL Bridge

- Message sizes used:
  - 32 kB
  - 256 kB
  - 1 MB
- Using containers a single message is sent and received
- Using COMMAREAs:
  - a single 32 kB message is sent and received for the 32 kB scenario
  - multiple 32 kB messages are sent and received for the 256 kB and 1 MB scenarios
    - note that the server app will be linked to multiple times for these 2 scenarios

CICS MQ DPL Bridge

- Performance environment
  - TPNS used to drive clients
  - clients run in separate LPAR from CICS server app
  - MQ DPL request and data sent:
    - from an MQ subsystem on the client LPAR
    - to an MQ subsystem on the CICS server app LPAR
    - using TCP/IP
  - data returned to client of same size as data sent
  - CPU usage on CICS Server LPAR is measured

Performance Environment



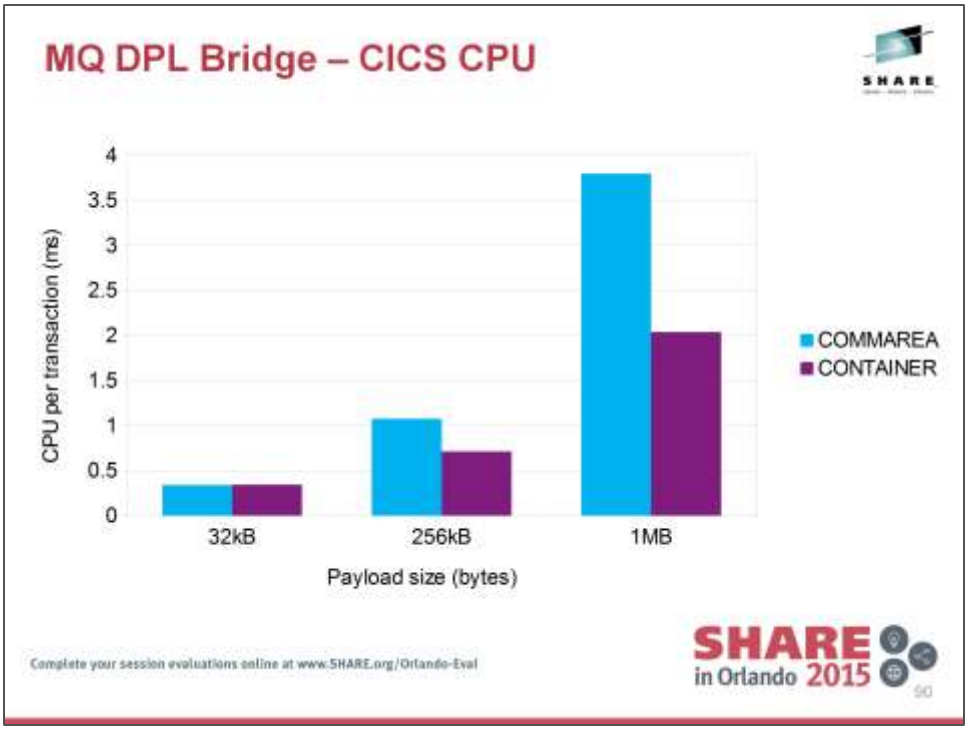


Chart shows CPU costs for the CICS region only.

CICS MQ DPL Bridge

- Total CPU costs
  - additional CPU costs are shown on the next slide
    - CICS CPU
    - MQ Master address space CPU
    - MQ Channel Initiator address space CPU
    - TCP/IP CPU

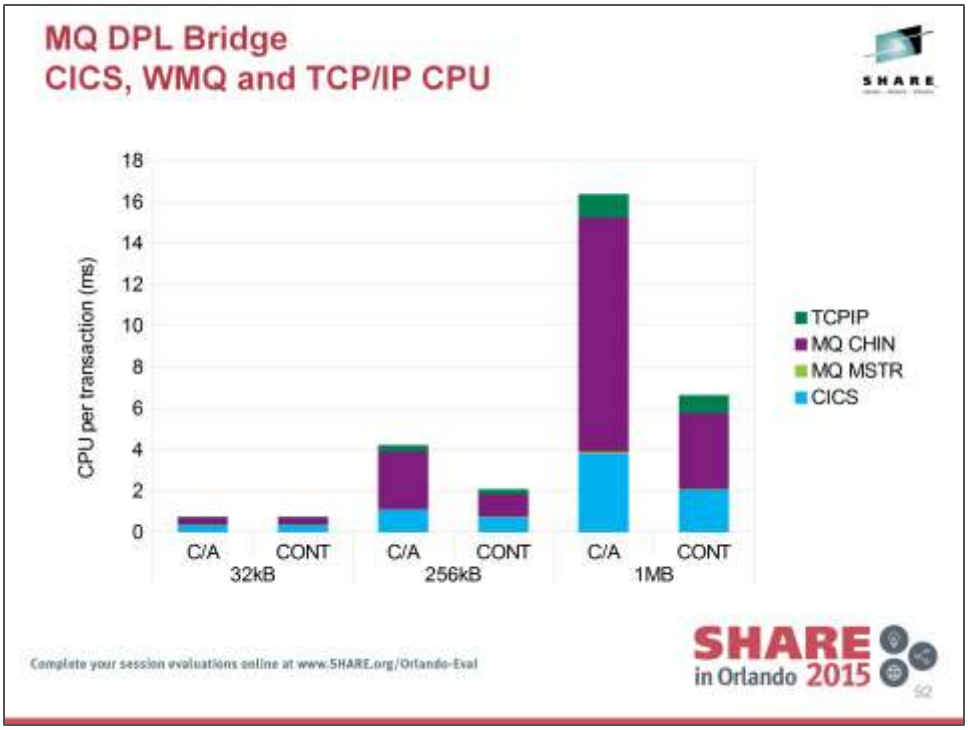
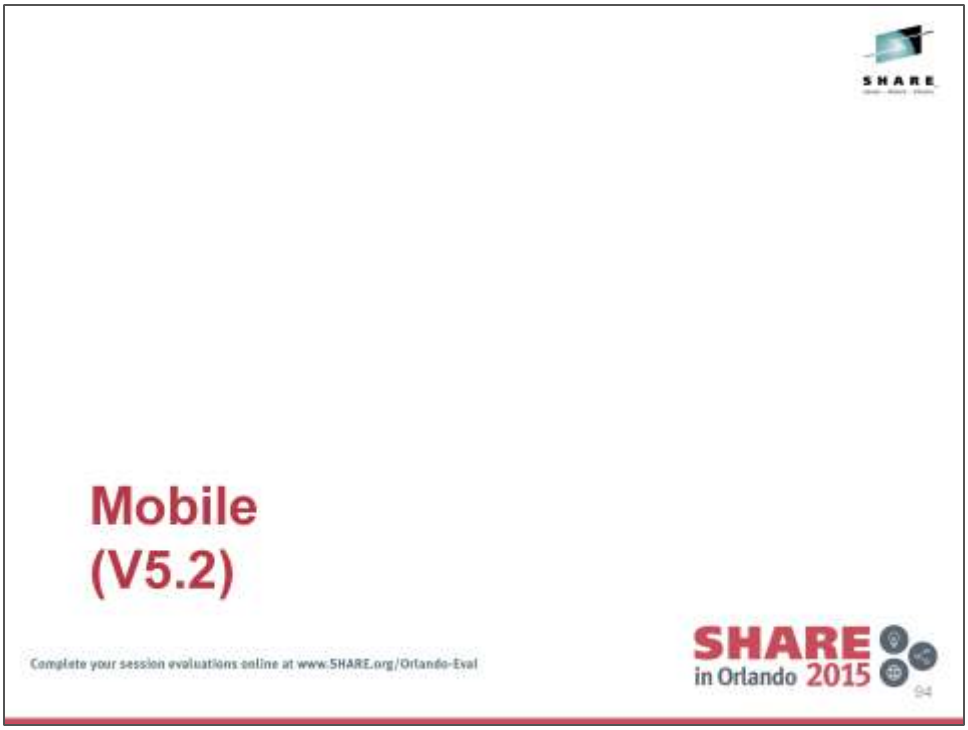


Chart shows total CPU cost for all relevant address spaces on the LPAR.

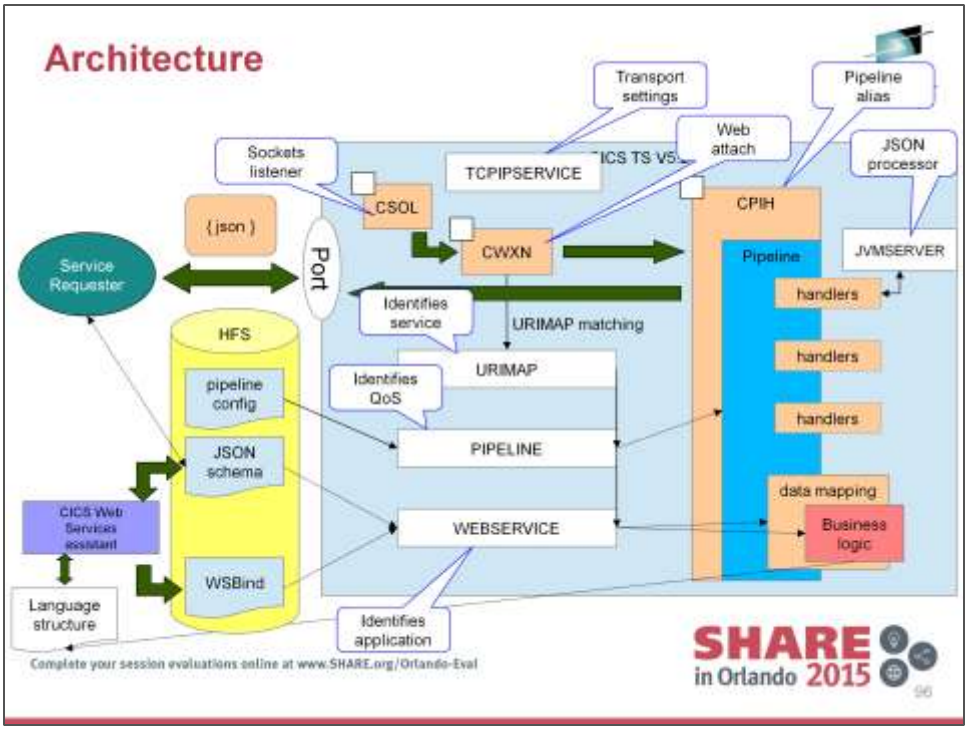
Summary

- MQ DPL Bridge – COMMAREAs versus containers
  - message sizes < 32 kB
    - CPU and response times similar
  - message sizes > 32 kB
  - due to multiple messages required to be sent for COMMAREAs:
    - significant CPU reduction using containers
      - 46% CICS CPU reduction for 1 MB messages
      - 60% total CPU reduction for 1 MB messages
    - substantial response time improvements using Containers
      - e.g. 233ms versus 25ms for the 1 MB scenario



CICS JSON Support

- Acts as a traditional CICS pipeline handler
  - Very similar implementation to Axis2 XML Web Services processing
  - Specifically NOT Liberty JSON handling
- Mobile Feature Pack now integrated into base CICS TS product



JVMSERVER

- Referenced by PIPELINE XML file
- Use supplied DFHJVMAX.jvmprofile file
- SIT parm JVMPROFILEDIR specifies location
- Specified fixed heap size of 400MB

```
DEFINE JVMSERVER(JSONJVM)    GROUP(GJSON)
       JVMPROFILE(DFHJVMAX) THREADLIMIT(50)
```

TCPIPSERVICE

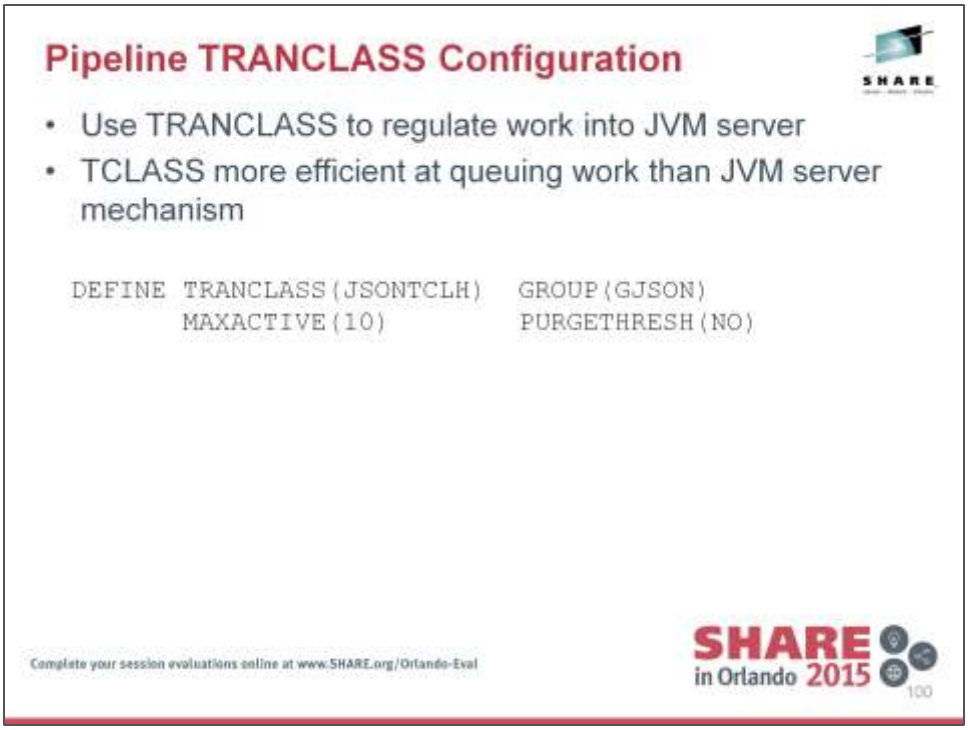
- CWXN is the web attach transaction
- DFHWBAAX is the default HTTP analyzer program

```
DEFINE TCPIPSERVICE(JSONTCP1) GROUP(GJSON)
       PORTNUMBER(6000)       TRANSACTION(CWXN)
       PROTOCOL(HTTP)         URM(DFHWBAAX)
       IP(1.2.3.4)            BACKLOG(250)
```

Pipeline TRANSACTION Configuration

- Define a CPIH alias transaction for TCLASS use
- Uses standard inbound pipeline router DFHPIDSH

```
DEFINE TRANSACTION(JPIH)       GROUP(GJSON)
       PROGRAM(DFHPIDSH)       TRANCLASS(JSONTCLH)
       SPURGE(YES)             TASKDATALOC(ANY)
       DESCRIPTION(JSON HTTP Inbound Router)
```



During testing, we found that using a TRANCLASS to throttle work through the JVM was more efficient than the JVMSERVER THREADLIMIT attribute.

PIPELINE Configuration

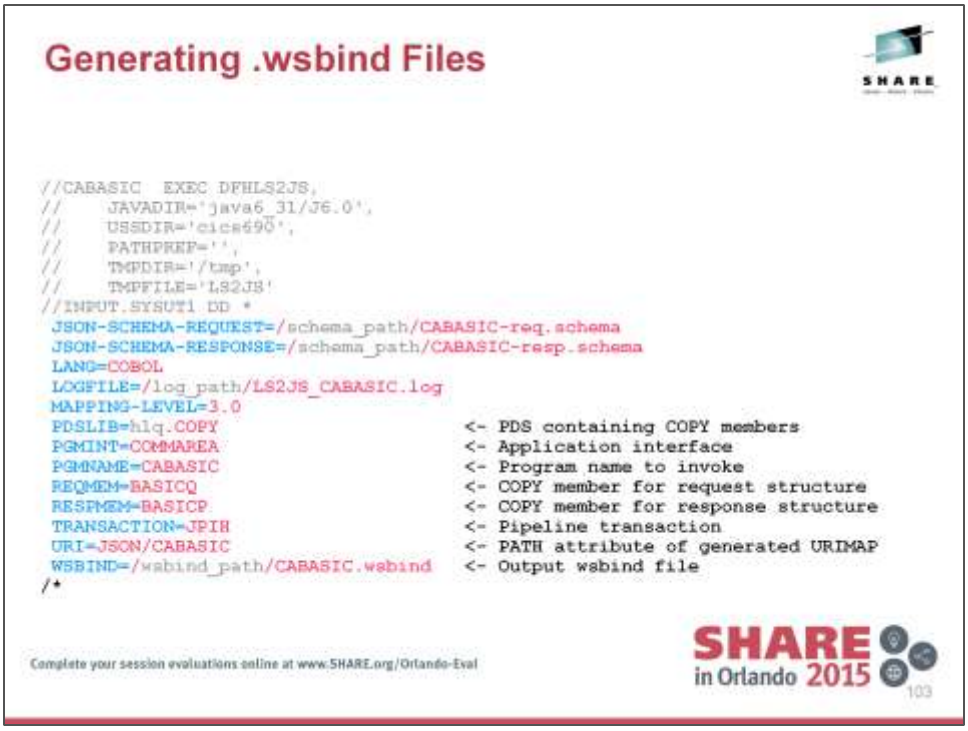
- CONFIGFILE = Location of XML pipeline file
- SHELF = Directory to contain in-use wsbind files
- WSDIR = Location of source wsbind files

```
DEFINE PIPELINE(JSONPIP1)      GROUP(GJSON)
       CONFIGFILE(/prefix/jsonprovider.xml)
       SHELF(/var/cicsts/myapplid/)
       WSDIR(/wsdir_prefix/wsbind)
```

PIPELINE XML Configuration

- JSONJVM = Name of JVMSERVER resource

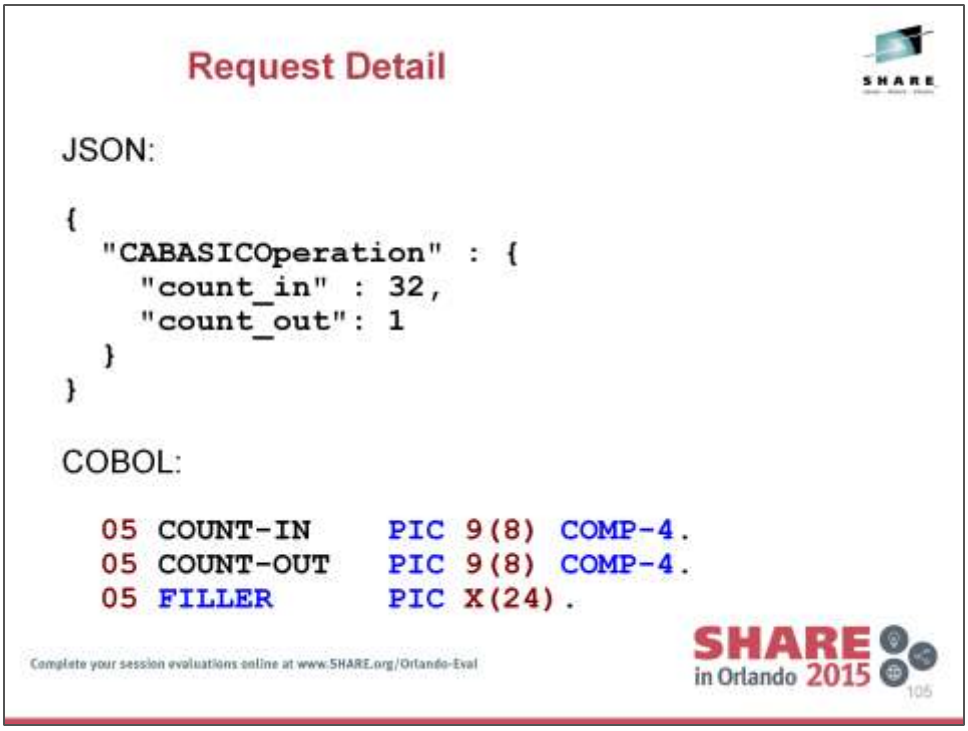
```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <service>
    <terminal_handler>
      <cics_json_handler_java>
        <jvmserver>JSONJVM</jvmserver>
      </cics_json_handler_java>
    </terminal_handler>
  </service>
  <apphandler_class>com.ibm.cicats.axis2.CICSAxis2ApplicationHandler</apphandler_class>
</provider_pipeline>
```



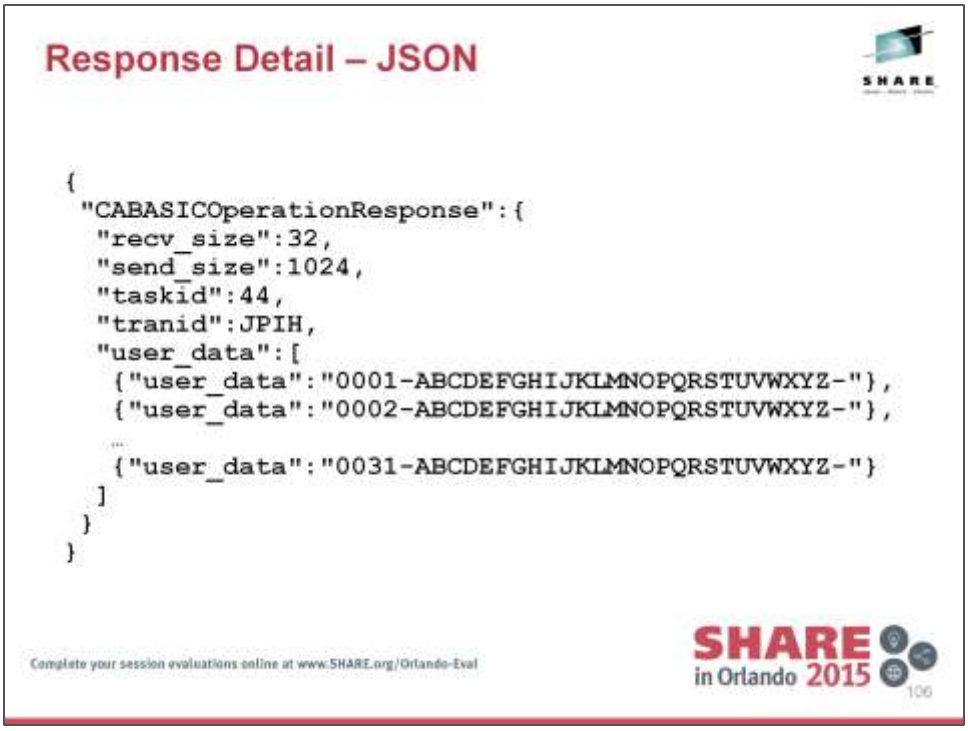
A sample of the JCL used to produce the *.wsbind files for the JSON pipeline parsing application.

Application Outline

- Request contains 32 bytes of application data
  - 180 bytes of JSON
- Response size
  - 32 bytes user data (103 bytes JSON)
  - 1,024 bytes user data (1,638 bytes JSON)
  - 4,096 bytes user data (6,342 bytes JSON)
  - 16,384 bytes user data (25,159 bytes JSON)
- COBOL backend application
  - Uses CHANNEL interface
  - No business logic



JSON formatting has been added for clarity. The on-the-wire format removed all unnecessary whitespace as would be found in a typical production environment.

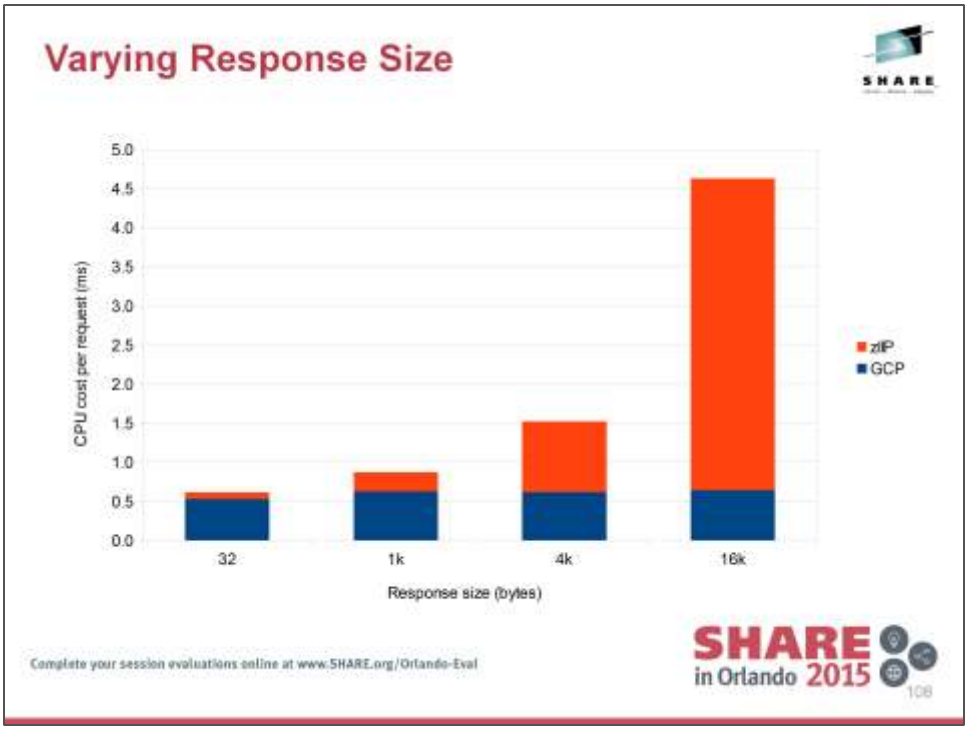


JSON formatting has been added for clarity.

Response Detail – COBOL

- Change OCCURS clause for varying response size

```
05 RECV-SIZE    PIC 9(8) COMP-4.
05 SEND-SIZE    PIC 9(8) COMP-4.
05 TASKID       PIC 9(8) COMP-4.
05 TRANID       PIC X(4).
05 FILLER       PIC X(16).
05 USER-DATA    PIC X(32) OCCURS 31 TIMES.
```

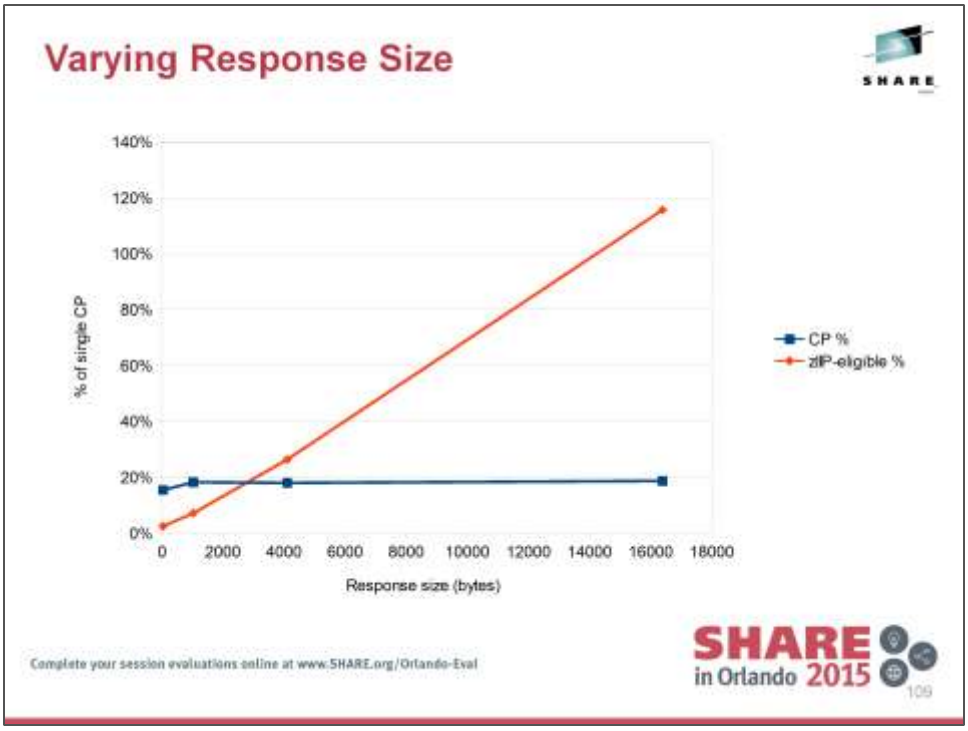


Data is calculated from RMF data which encapsulates the whole of the CICS address space.

The IEAOPTxx parmlib member option PROJECTCPU and and IEASYSxx parmlib member option ZAAPZIIP were both configured to YES to enable recording of offload-eligible CPU time, and zAAP-on-zIIP execution respectively.

zIIP figures represent the IIPCP field in the RMF report, while GCP figures represent the CP fields minus the IIPCP field.

Note that the amount of non-zIIP eligible CPU cost remains approximately constant, regardless of the response size. The serialisation of the JSON response occurs in the CICS Java implementation, which is where the variation in CPU cost originates.

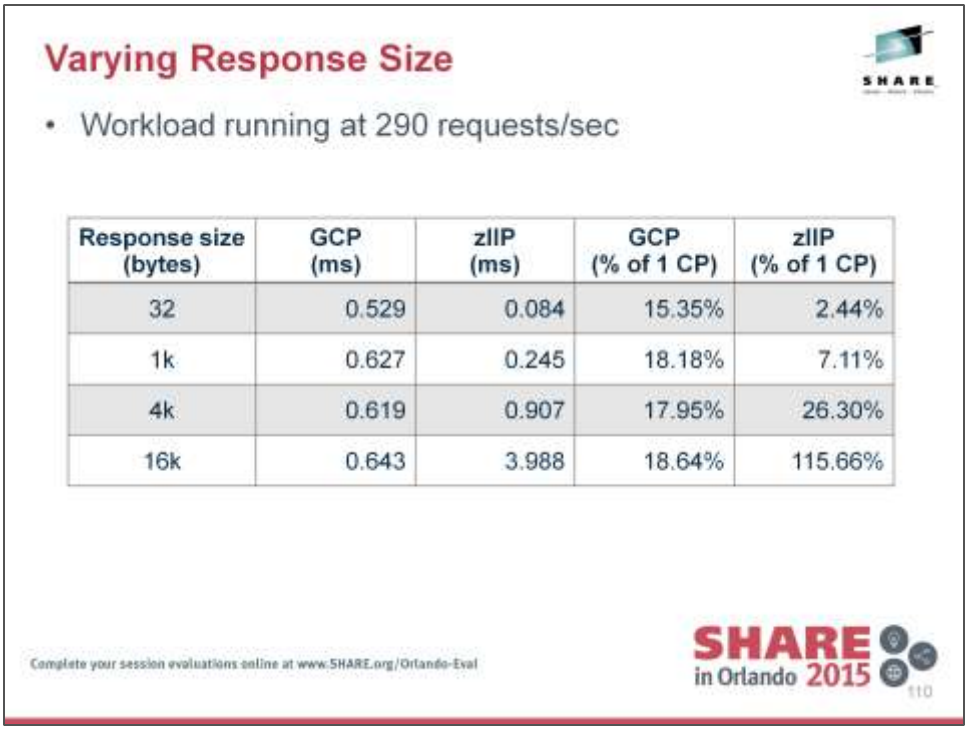


Data is calculated from RMF data which encapsulates the whole of the CICS address space.

The IEAOPTxx parmlib member option PROJECTCPU and and IEASYSxx parmlib member option ZAAPZIIP were both configured to YES to enable recording of offload-eligible CPU time, and zAAP-on-zIIP execution respectively.

zIIP figures represent the IIPCP field in the RMF report, while GCP figures represent the CP fields minus the IIPCP field.

Note that the amount of non-zIIP eligible CPU cost remains approximately constant, regardless of the response size. The serialisation of the JSON response occurs in the CICS Java implementation, which is where the variation in CPU cost originates.

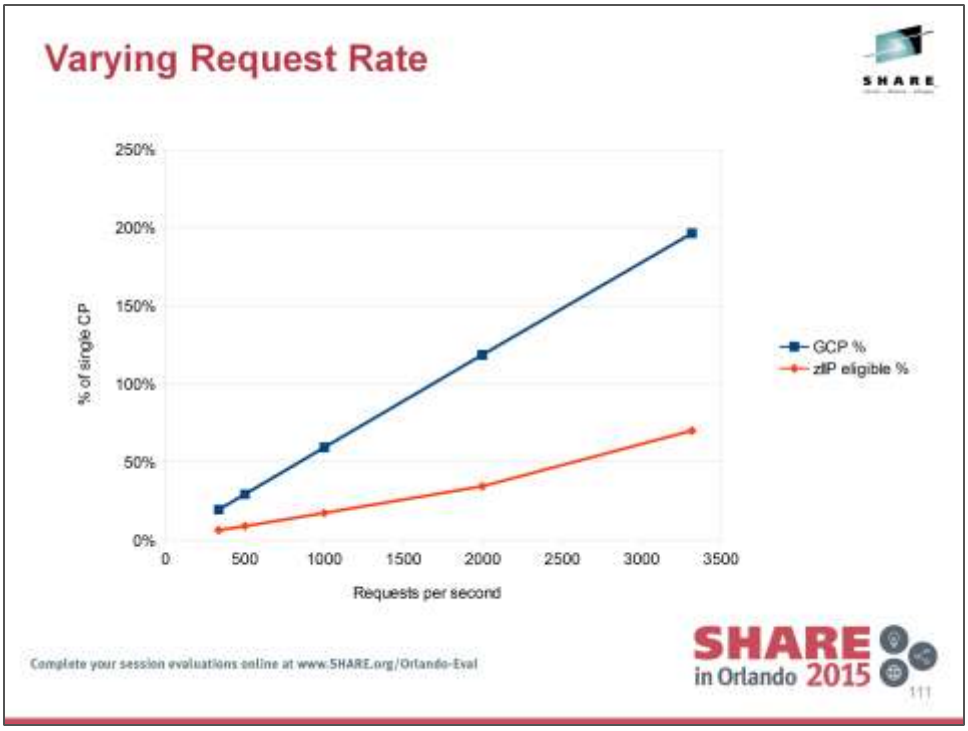


Data is calculated from RMF data which encapsulates the whole of the CICS address space.

The IEAOPTxx parmlib member option PROJECTCPU and and IEASYSxx parmlib member option ZAAPZIIP were both configured to YES to enable recording of offload-eligible CPU time, and zAAP-on-zIIP execution respectively.

zIIP figures represent the IIPCP field in the RMF report, while GCP figures represent the CP fields minus the IIPCP field.

Note that the amount of non-zIIP eligible CPU cost remains approximately constant, regardless of the response size. The serialisation of the JSON response occurs in the CICS Java implementation, which is where the variation in CPU cost originates.

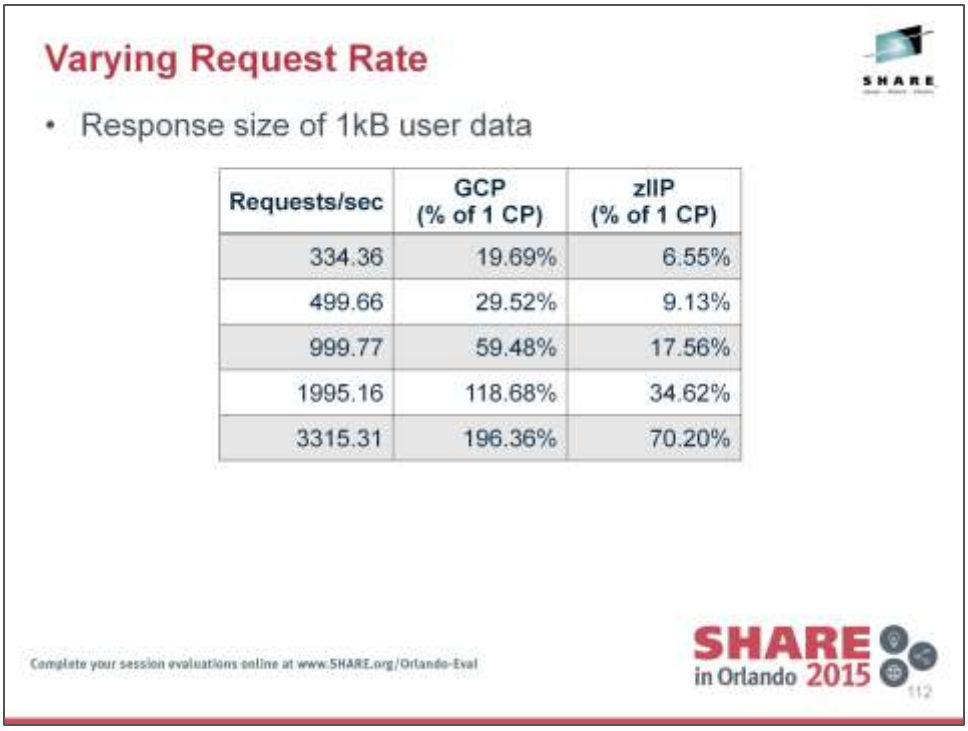


Data is calculated from RMF data which encapsulates the whole of the CICS address space.

The IEAOPTxx parmlib member option PROJECTCPU and and IEASYSxx parmlib member option ZAAPZIIP were both configured to YES to enable recording of offload-eligible CPU time, and zAAP-on-zIIP execution respectively.

zIIP figures represent the IIPCP field in the RMF report, while GCP figures represent the CP fields minus the IIPCP field.

Note that the amount of non-zIIP eligible CPU cost remains approximately constant, regardless of the response size. The serialisation of the JSON response occurs in the CICS Java implementation, which is where the variation in CPU cost originates.

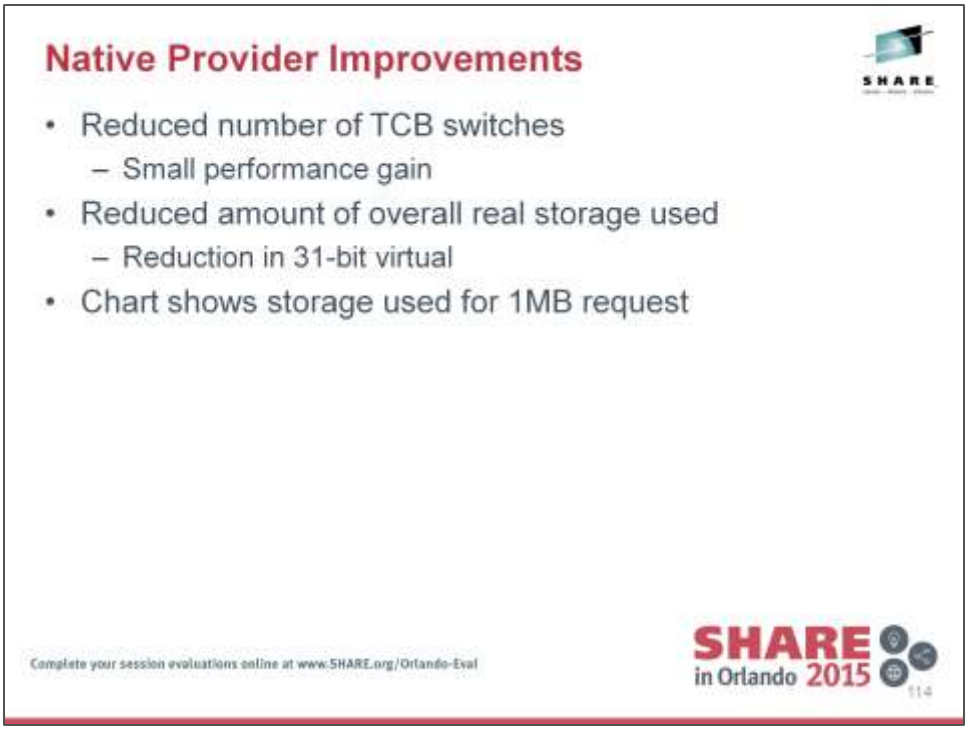


Data is calculated from RMF data which encapsulates the whole of the CICS address space.

The IEAOPTxx parmlib member option PROJECTCPU and and IEASYSxx parmlib member option ZAAPZIIP were both configured to YES to enable recording of offload-eligible CPU time, and zAAP-on-zIIP execution respectively.

zIIP figures represent the IIPCP field in the RMF report, while GCP figures represent the CP fields minus the IIPCP field.

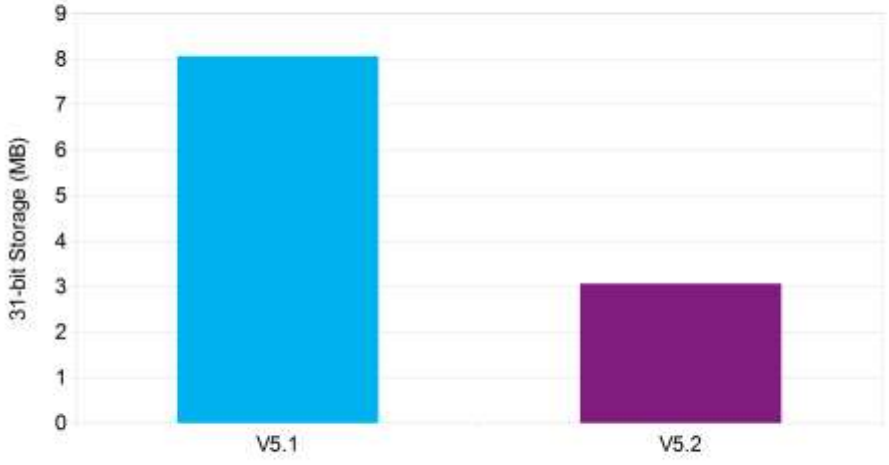
Note that the amount of non-zIIP eligible CPU cost remains approximately constant, regardless of the response size. The serialisation of the JSON response occurs in the CICS Java implementation, which is where the variation in CPU cost originates.

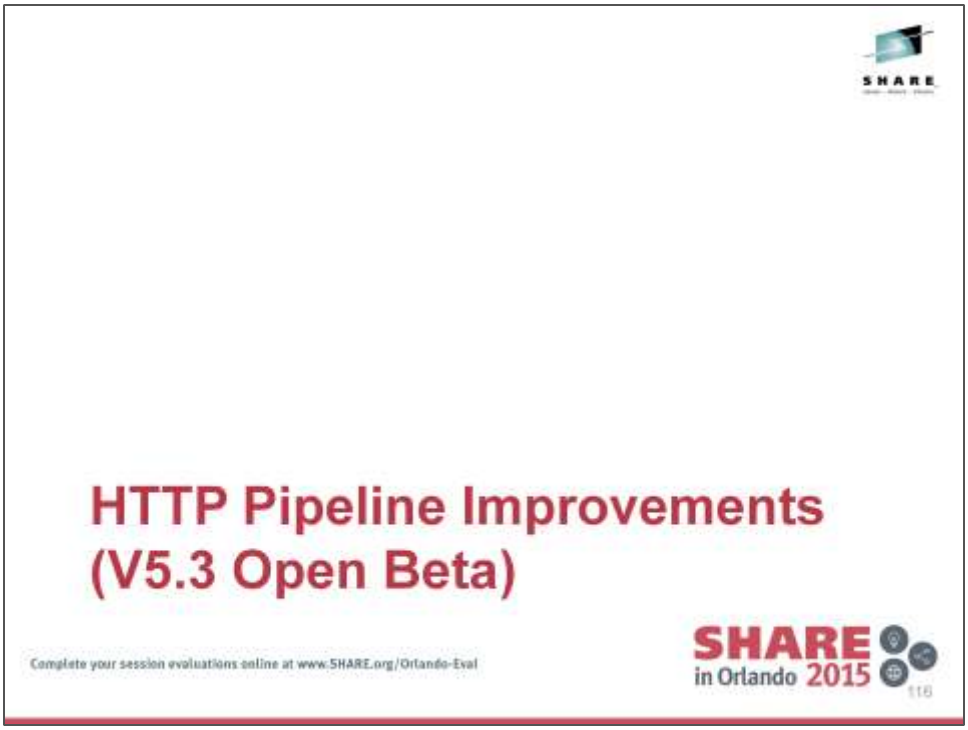


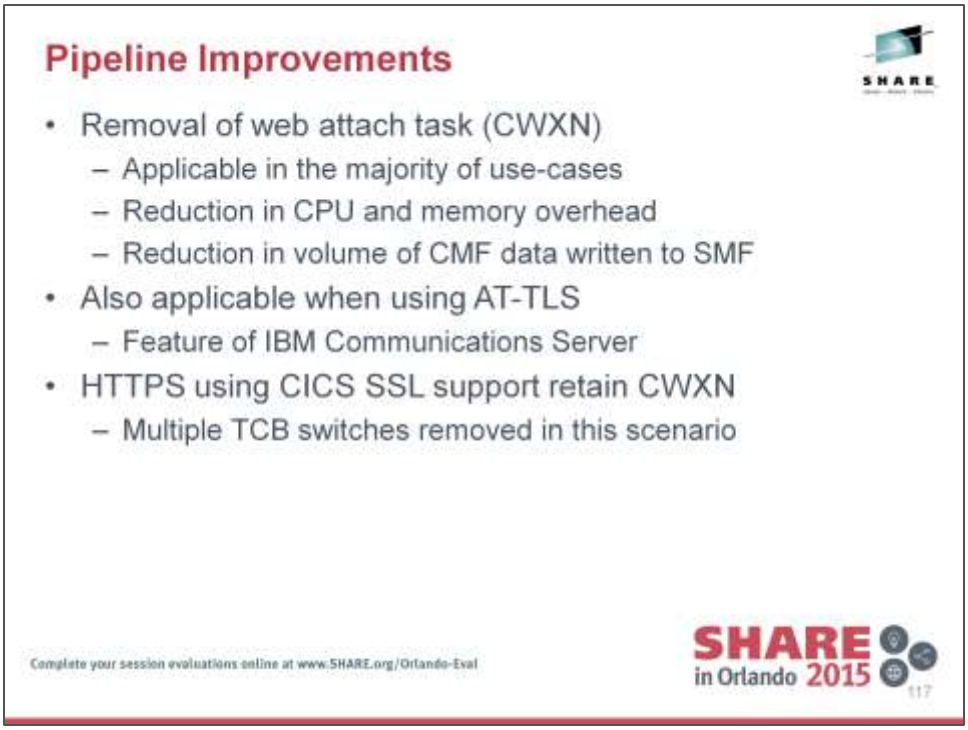
Note that overall storage usage has been reduced.

Reduction has been achieved from the 31-bit storage areas.

Web Service Provider 31-bit Storage





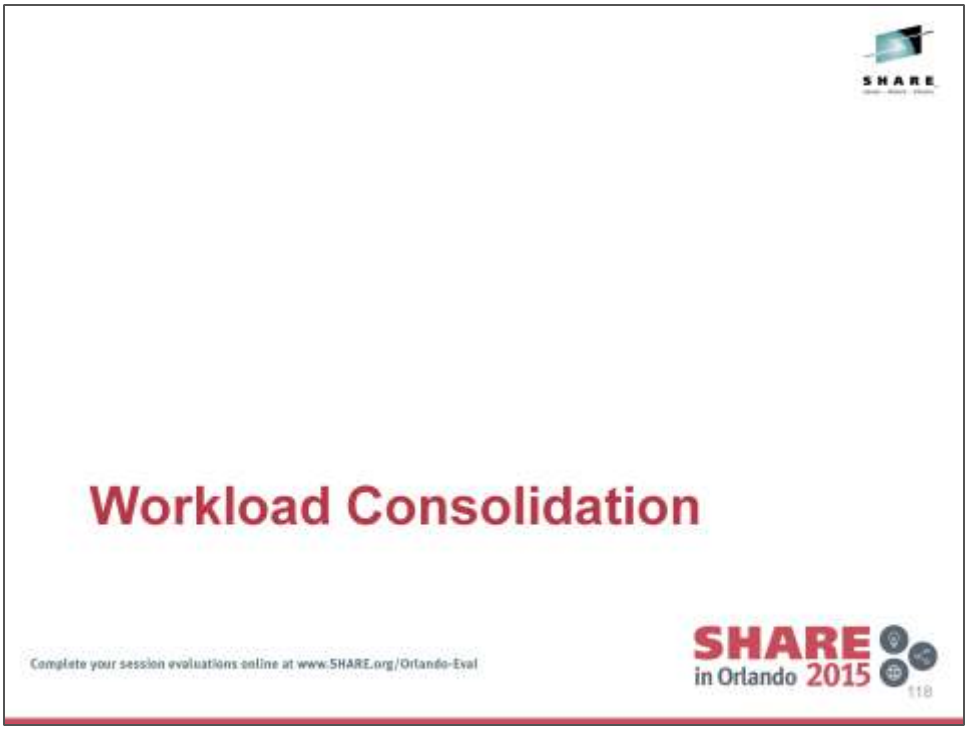


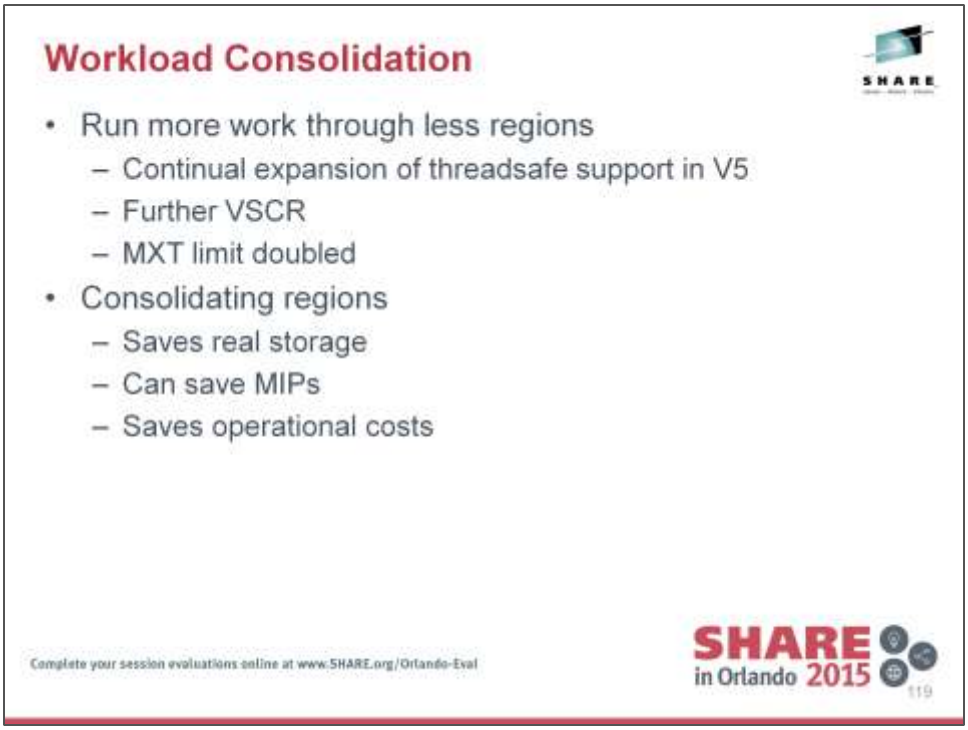
For each HTTP request into CICS, whether as a web, a web service, or a JSON request, there are two tasks associated with the request. This results in two SMF records being emitted from CICS if MN=ON and MNPER=ON.

With the removal of the CWXN transaction, less CPU and storage is required to process the request, and there will only be one resultant SMF record emitted from the CICS region.

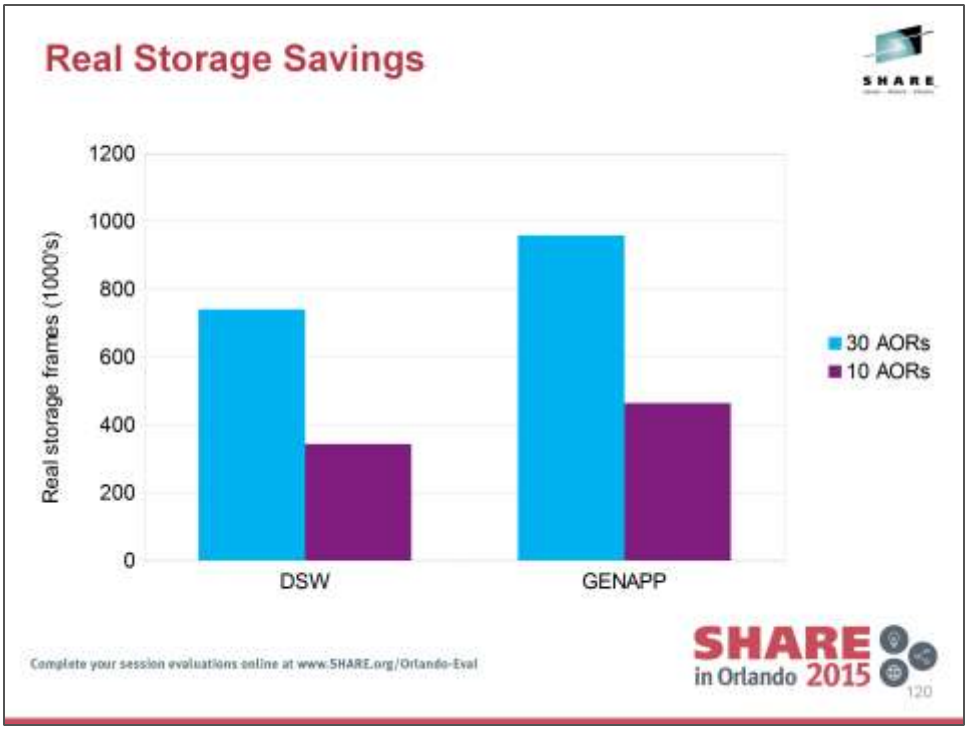
Application-Transparent Transport Layer Security (AT-TLS) is a feature of IBM Communications Server, where TLS encryption is handled by the TCPIP address space, rather than the receiving address space.

CICS TS V5.3 open beta permits TCPIPSERVICE resources to be configured as AT-TLS aware, enabling CICS to obtain security information from the network stack.

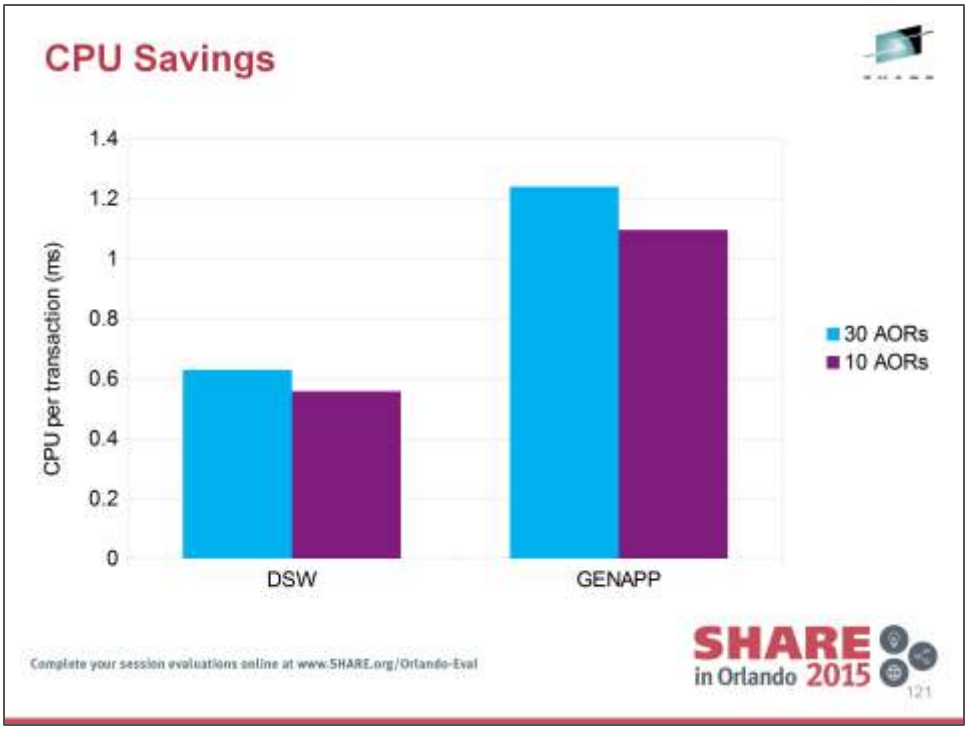




With CICS TS V5.1, the MXT value can now be set to 2000, rather than 999.

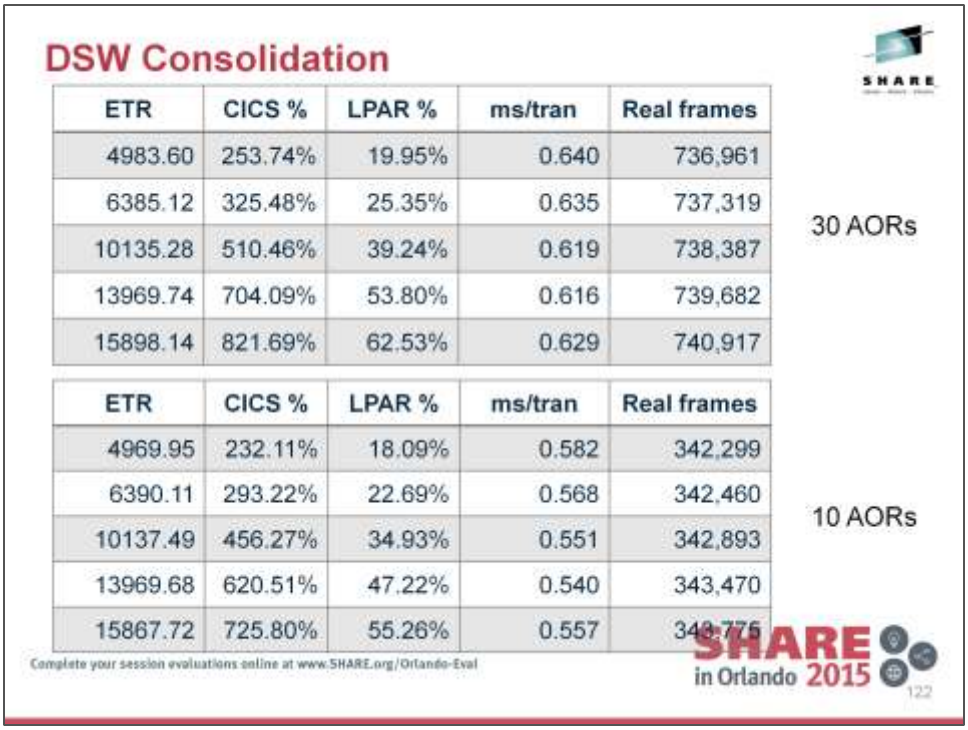


Real storage usage covers all CICS regions in the consolidation configuration.



CPU savings can be achieved by consolidating CICS regions.

For both workloads, the same transaction rate was achieved using fewer Application Owning Regions (AORs).



Hardware Instrumentation Services data was collected for the final measurement interval.

ETR is External Throughput Rate (transaction rate).

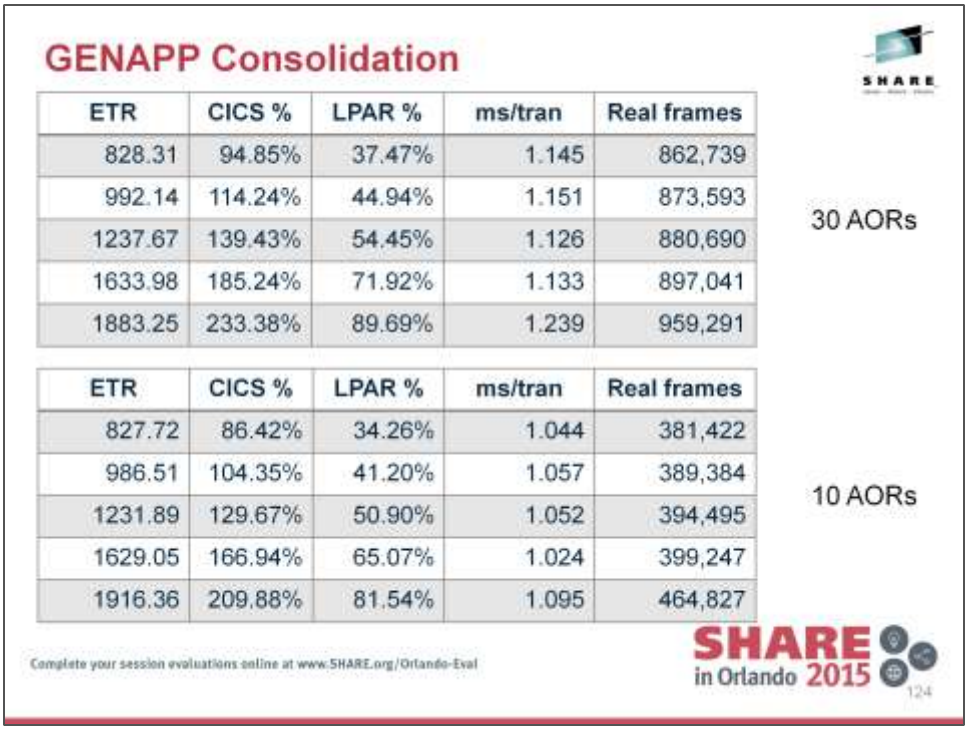
CICS % refers to the CP field on an RMF report class report.

LPAR % refers to the RMF workload activity report.

Hardware Data (DSW)

	30 AORs	10 AORs	Delta
Execution Samples	2487298	2201099	-11%
Instruction First Cycle (IFC)	379000	371470	-2%
Micro Seconds per transaction	628.34	556.43	-11%
Cycles per instruction	6.53	5.90	-10%
MIPS per CP	797	882	+10%
Data cache misses (samples)	744894	608550	-18%
Instruction cache miss includes TLB miss	90483	66626	-26%
% Cycles used by TLB misses	6.82	5.94	-13%
Relative Nest Intensity (RNI)	0.48	0.34	

CPU savings are achieved because there are significantly fewer CPU cycles spent waiting for data-cache and TLB misses.



Hardware Instrumentation Services data was collected for the final measurement interval.

ETR is External Throughput Rate (transaction rate).

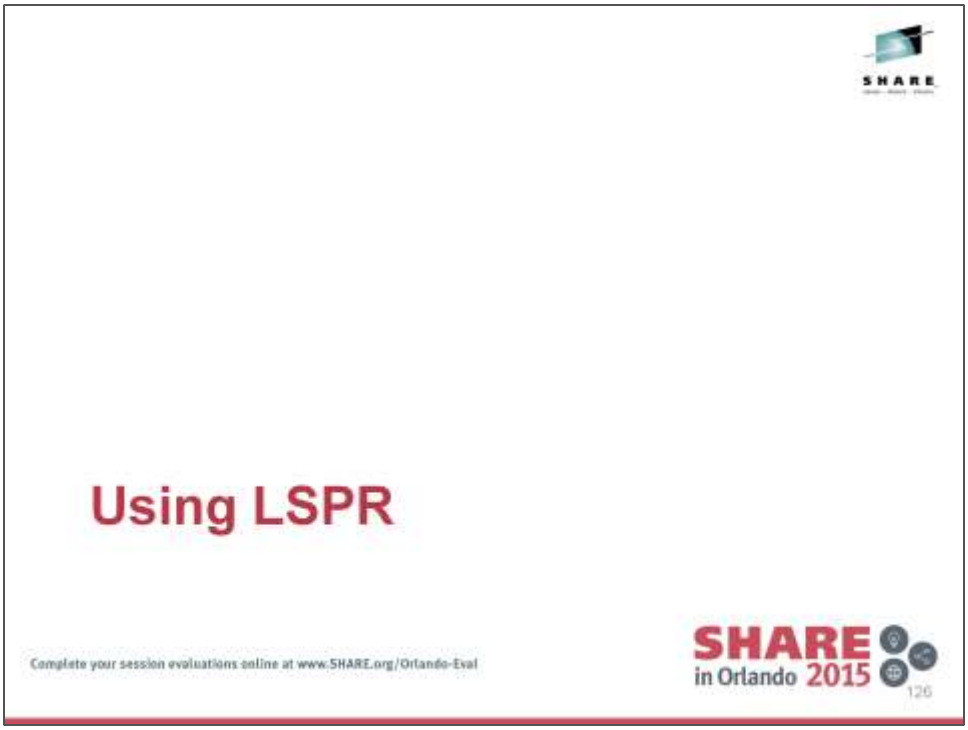
CICS % refers to the CP field on an RMF report class report.

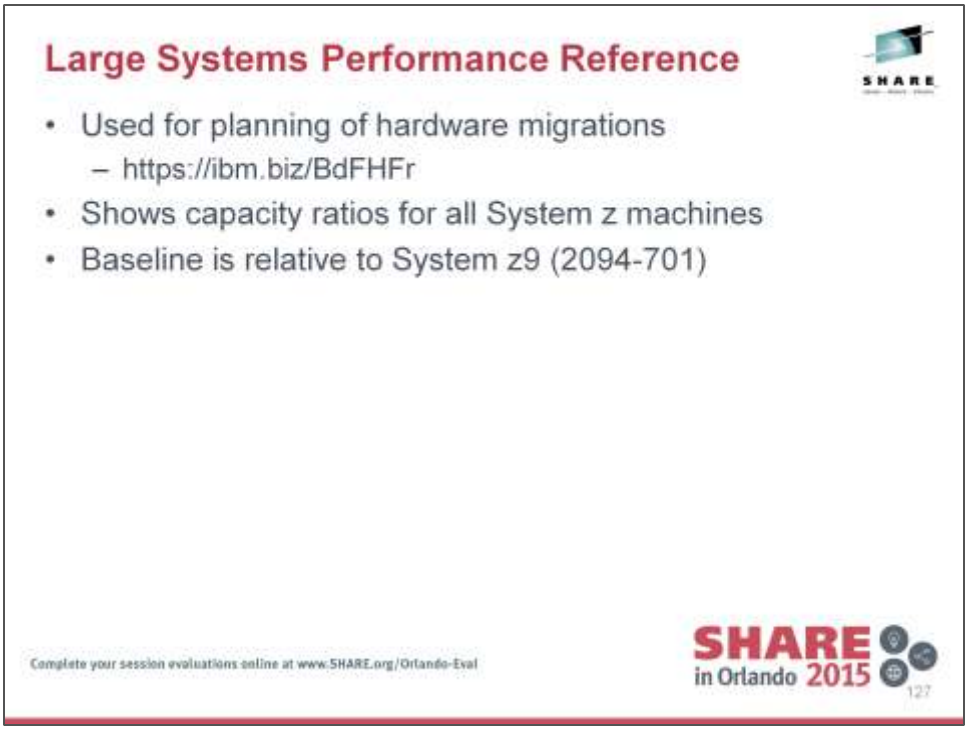
LPAR % refers to the RMF workload activity report.

Hardware Data (GENAPP)

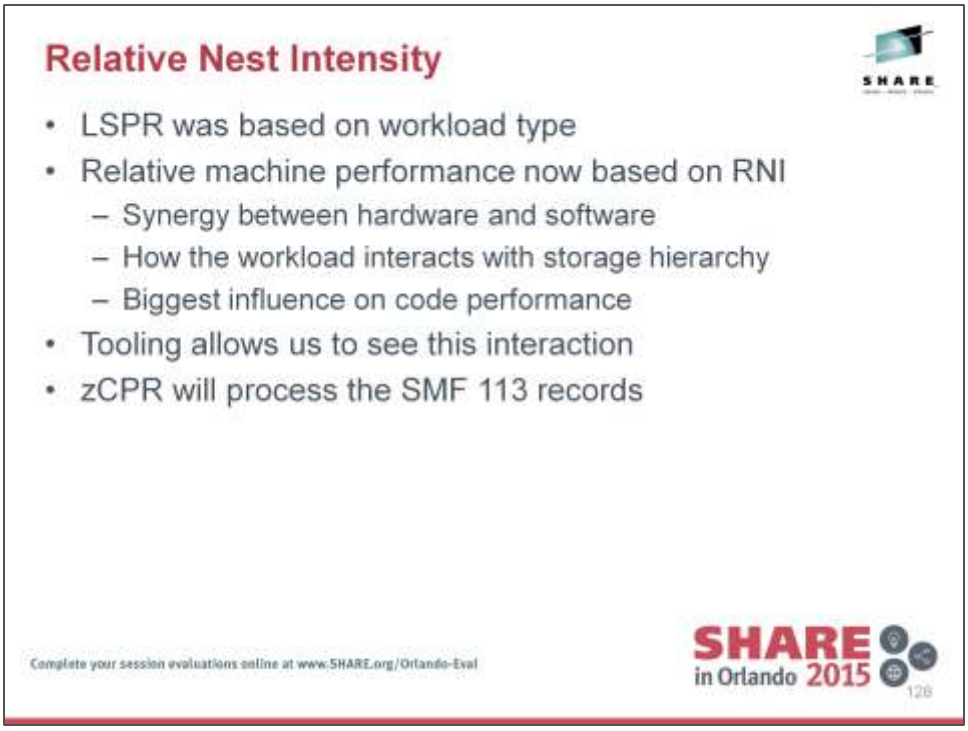
	30 AORs	10 AORS	Delta
Execution Samples	3517830	3188565	-9%
Instruction First Cycle (IFC)	589236	590667	+2%
Micro Seconds per transaction	1240	1095	-11%
Cycles per instruction	5.97	5.39	-10%
MIPS per CP	898	1003	+11%
Data cache misses (samples)	1145876	932896	-18%
Instruction cache miss includes TLB miss	149468	115015	-23%
% Cycles used by TLB misses	9.95	9.23	-7%
Relative Nest Intensity (RNI)	0.75	0.51	

CPU savings are achieved because there are significantly fewer CPU cycles spent waiting for data-cache and TLB misses.





https://ibm.biz/BdFHFr



See Hardware Instrumentation Services documentation for the production of SMF 113 records.

The use of SMF 113 records is now recommended by IBM when considering hardware upgrades.

Internal Throughput Rate

- ITR is defined as number of transactions per CPU second
  - So if you know how many per CPU sec, you also know how much each transaction would cost
- #CPs / ITR is the relative CPU used by 1 transaction in the LSPR tables
- ITR / #CPs is the relative speed of one CPU

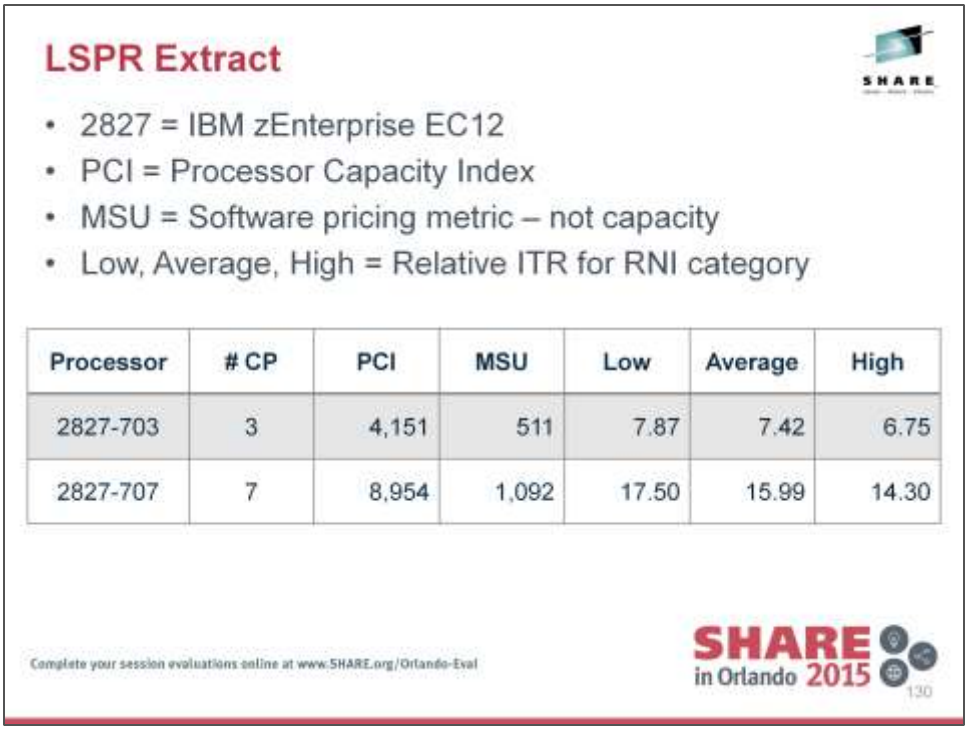


Table is an extract of the LSPR found here:

https://www.ibm.com/servers/resourcelink/lib03060.nsf/pages/lsprITRzOSv1r13?OpenDocument

Example from previous table

- Assuming your workload has an "Average" RNI
  - 2827-703 : 3 CPs / 7.42 ITR = 0.404 secs per tran
  - 2827-707 : 7 CPs / 15.99 ITR = 0.438 secs per tran
- Therefore:
  - Throughput can scale 2.2 times horizontally
    - 7.42 → 15.99
  - CPU per CICS transaction will increase by 8%
    - 0.404s → 0.438s
- Important for non-threadsafe applications