17035 - MQ for z/OS

Using and Abusing New Hardware and the New V8 Features

Wednesday March 4th 2015
10:00 - 11:00 AM
Willow B, Sheraton Seattle

Mayur Raja (mayur_raja@uk.ibm.com)

#SHAREorg

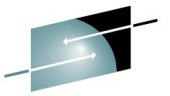
SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.

MQ for z/OS V8 New Features Deep Dive

NOTES

- In this presentation we will take a deep dive in to the new features of MQ for z/OS V8.

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.
- If the text contains performance statistics or references to benchmarks, insert the following language; otherwise delete:
  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.
- If the text includes any customer examples, please confirm we have prior written approval from such customer and insert the following language; otherwise delete:
  All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.
- Please review text for proper trademark attribution of IBM products. At first use, each product name must be the full name and include appropriate trademark symbols (e.g., IBM Lotus® Sametime® Unyte™). Subsequent references can drop "IBM" but should include the proper branding (e.g., Lotus Sametime Gateway, or WebSphere Application Server). Please refer to http://www.ibm.com/legal/copytrade.shtml for guidance on which trademarks require the ® or ™ symbol. Do not use abbreviations for IBM product names in your presentation. All product names must be used as adjectives rather than nouns. Please list all of the trademarks that you use in your presentation as follows; delete any not included in your presentation. IBM, the IBM logo, Lotus, Lotus Notes, Notes, Domino, Quickr, Sametime, WebSphere, UC2, PartnerWorld and Lotusphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. Unyte is a trademark of WebDialogs, Inc., in the United States, other countries, or both.
- © IBM Corporation 2014. All Rights Reserved.
- If you reference Adobe® in the text, please mark the first use and include the following; otherwise delete:
  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.
- If you reference Java™ in the text, please mark the first use and include the following; otherwise delete:
  Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- If you reference Microsoft® and/or Windows® in the text, please mark the first use and include the following, as applicable; otherwise delete:
  Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- If you reference Intel® and/or any of the following Intel products in the text, please mark the first use and include those that you use as follows; otherwise delete:
  Intel, Intel Centrino, 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.
- If you reference UNIX® in the text, please mark the first use and include the following; otherwise delete:
  UNIX is a registered trademark of The Open Group in the United States and other countries.
- If you reference Linux® in your presentation, please mark the first use and include the following; otherwise delete:
  Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.
- If the text/graphics include screenshots, no actual IBM employee names may be used (even your own), if your screenshots include fictitious company names (e.g., Renovations, Zeta Bank, Acme) please update and insert the following; otherwise delete: All references to [insert fictitious company name] refer to a fictitious company and are used for illustration purposes only.

Please Note

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.

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.

Agenda

- 64 Bit Buffer Pools
- 8 Byte Log Relative Byte Address (RBA)
- Channel Initiator (CHINIT) Statistics and Channel Accounting Data
- Storage Class Memory (SCM) (Flash Memory)
- Other Enhancements

Agenda

NOTES

- The following topics are covered in this presentation:
  - 64 Bit Buffer Pools
  - 8 Byte Log Relative Byte Address (RBA)
  - Channel Initiator (Chinit) Statistics and Channel Accounting Data as written out to System Management Facilities (SMF) and used to monitor activity in the Channel Initiator address space
  - Storage Class Memory (SCM) (Flash memory)
  - Other Enhancements

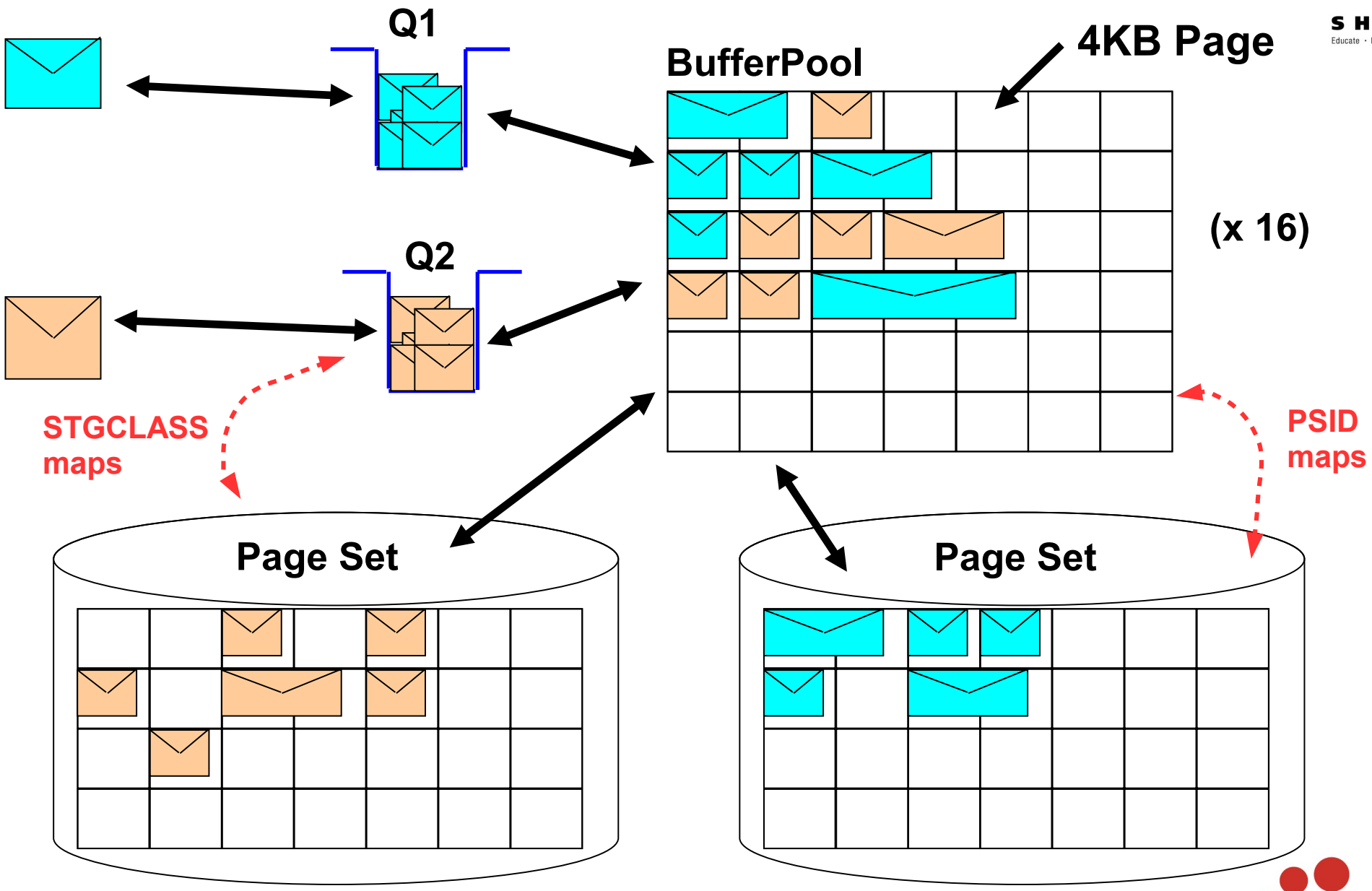
64 Bit Buffer Pools

64 Bit Buffer Pools

NOTES

- First, we'll take a look at 64 bit buffer pool support.

Buffer Pools: What we have pre-V8 - 1

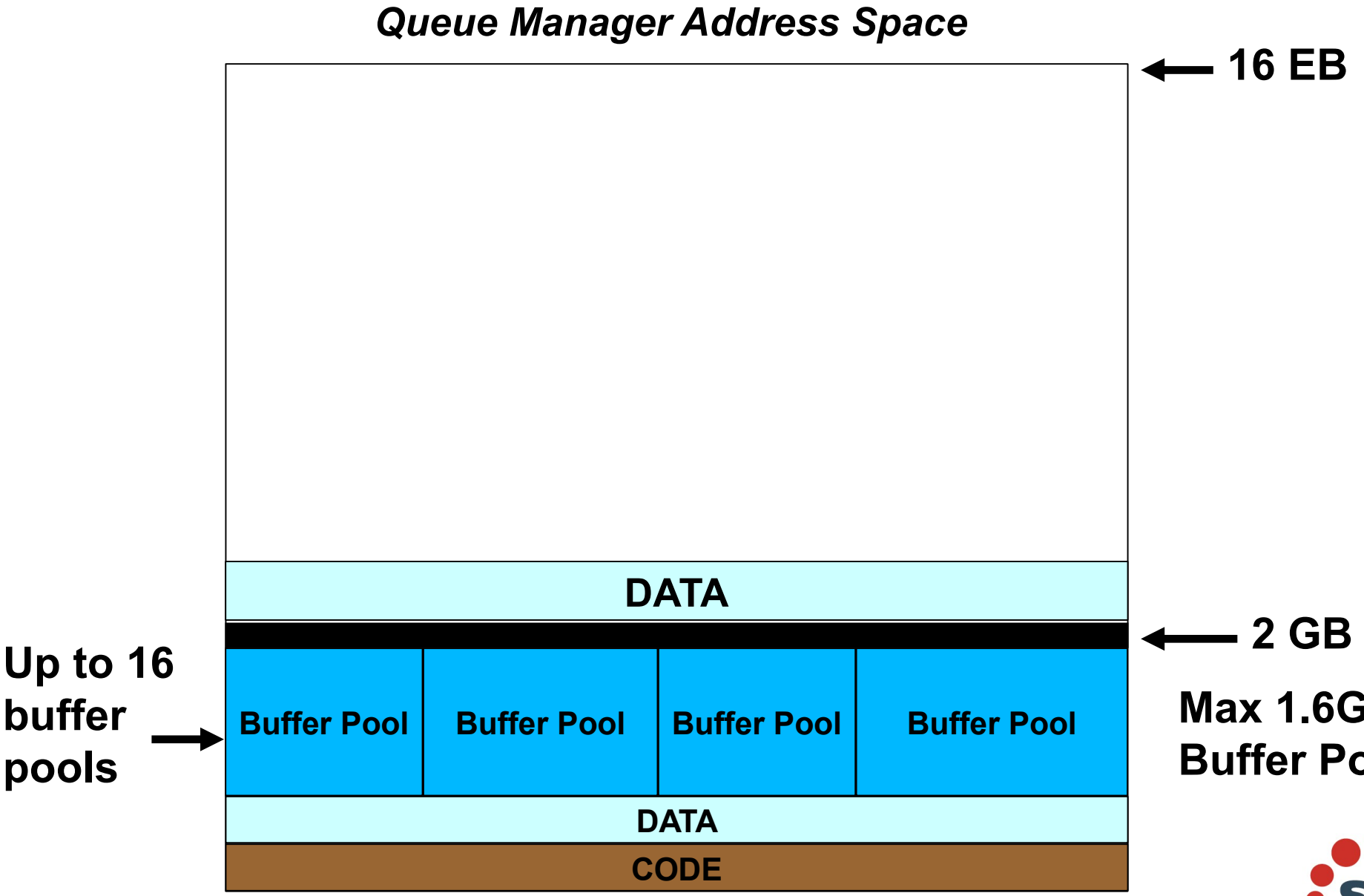


Buffer Pools: What we have pre-V8 - 1

NOTES

- This slide shows a representation of the current use of buffer pools with queues and page sets.
- A queue is configured, via a STGCLASS mapping, to use a specific page set for the storage of messages. One or more page sets can be configured (via a PSID mapping) to use a particular buffer pool to “buffer” the messages.
- When a message is written to a queue, it is stored as one or more 4K pages. The data is initially written to the buffer pool, and may at some later stage be written out to the page set. The pages may be written out to the page set if there are not enough pages in the buffer pool to contain all the messages on the queues. When a message is got, the data is retrieved from the buffer pool, if the necessary pages are not in the buffer pool, then they must first be retrieved from the page set, before the message can be returned.
- On this slide, there are two queues containing messages, which are using two different page sets, however, these two page sets are both using the same buffer pool.

Buffer Pools: What we have pre-V8 - 2



Buffer Pools: What we have today - 2

NOTES

- This slide shows a representation of the queue manager address space.
- All of the queue manager code resides below the bar, in 31 bit storage.
- Some queue manager data resides below the bar however, in earlier releases, some queue manager data (e.g. locks, security data, the IGQ buffer etc.) was moved into 64-bit storage above the bar. Also, as new features like PubSub were implemented, 64-bit storage was exploited.
- Prior to MQ version 8, buffer pools were defined in 31 bit storage. So, taking into account the code and data requirements mentioned above and the common storage usage on the system, there was a maximum of approximately 1.6 gigabytes of storage available for use by buffer pools.

Buffer Pools: The Problems

- Not much space below the bar for buffer pools
  - Maximum 1.6GB, depending on common storage usage
- Put/Get to/from:
  - Buffer pool = 'memory' speed (fast)
  - Page set = 'disk' speed (slow)
- With less/small buffer pools, can spend a lot of time:
  - Putting pages from buffer pool into page set (to free buffers)
  - Getting pages from page set into buffer pool (to satisfy get requests)
  - This is detrimental to performance
- A maximum of 16 buffer pools
  - But, up to 100 page sets .. hence page sets must share buffer pools
- System programmers can spend a lot of time tuning:
  - Buffer pool sizes
  - Queue, buffer pool, and page set mappings

Buffer Pools: The Problems

NOTES

- There is not much space below the bar for buffer pools once queue manager code and data is taken into account. There is generally a maximum of 1.6 gigabytes available for buffer pools depending on common storage usage.
- Putting and getting messages to/from buffer pools works at 'memory' speed, where as putting and getting messages to/from page sets works at 'disk' speed.
- For scenarios where several applications read and/or write large numbers of messages to the same buffer pool, a lot of time is spent getting pages from the page set into the buffer pool and putting pages from the buffer pool into the page set. This is detrimental to performance.
- A maximum of 16 buffer pools are supported while up to 100 page sets are supported, meaning that if you have more than 16 page sets, you need to share the same buffer pool for some of the page sets.
- Because of these reasons, a lot of time and effort can be spent in tuning the buffer pool sizes, and the mapping of queue to page sets and page sets to buffer pools.

64 Bit Buffer Pools: The Solution

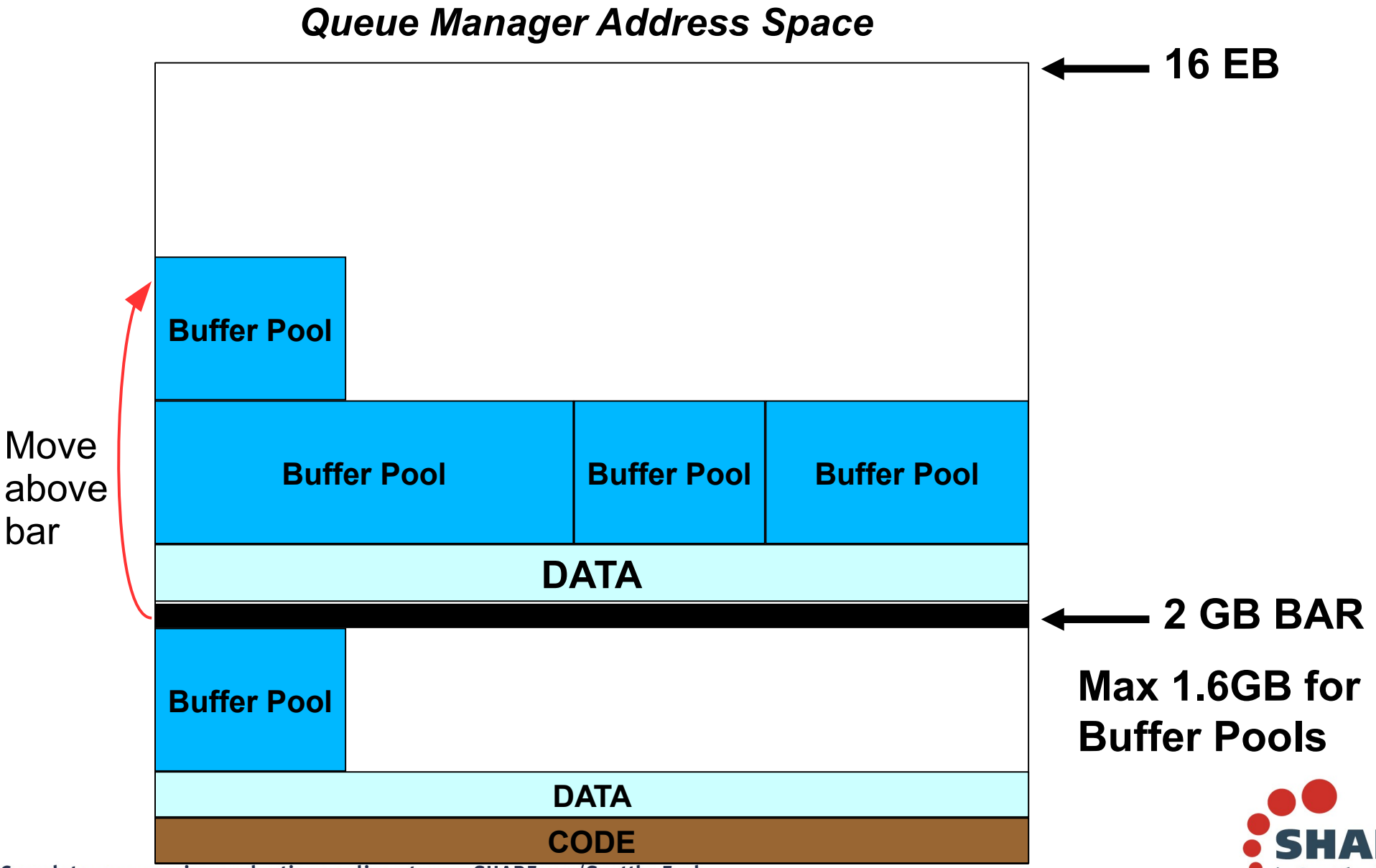
- Buffer pools above the bar.
  - Buffer pools can (theoretically) make use of up to 16 EB of storage
- More buffer pools
  - Up to 100 buffer pools
  - Can have 1-1 mapping between page set and buffer pool
- More buffers per pool
  - Above the bar
    - Up to 999,999,999 4K buffers per pool
  - Below the bar
    - Up to 500,000 4K buffers per pool

64 Bit Buffer Pools: The Solution

NOTES

- MQ version 8 introduces the capability of having buffer pools above the bar. This means that buffer pools can - theoretically, make use of up to 16 exabytes of storage.
- The maximum number of buffer pools has been increased from 16 to 100. This enables a 1 to 1 mapping of page sets to buffer pools.
- The maximum size of an individual buffer pool has also increased to 9 9s buffers, if the buffer pool is above the bar. The previous limit was 500,000 buffers, which remains in force if the buffer pool is located below the bar.

Buffer Pools: Using 64 bit storage



Buffer Pools: Using 64 bit storage

NOTES

- This slide shows a representation of the queue manager address space, similar to the earlier diagram, but this time showing 64 bit storage in use for buffer pools.
- Other storage usage has remained the same, with queue manager code, and some of the data remaining in 31 bit storage. However, being able to support 64 bit storage for buffer pools means that buffer pools may be moved out of 31 bit storage, relieving the constraint for other users of 31 bit storage.

  The diagram shows that buffer pools may continue to use 31 bit storage, providing a migration path to using 64 bit storage. The diagram also shows that because of the greater availability of storage above the bar, the sizes of the buffer pools may be increased, not being constrained by the 1 point 6 gigabytes overall storage availability.

64 Bit Buffer Pools: What has changed?

```
DEFINE    BUFFPOOL(<id>)
          BUFFERS(<integer>)
          PAGECLAS(4KB/FIXED4KB)
          LOCATION(BELOW/ABOVE)
```

- BUFFPOOL id
  - 0 to 99
- BUFFERS integer
  - Up to 500,000 if LOCATION(BELOW)
  - Up to 999,999,999 if LOCATION(ABOVE)
- PAGECLAS can be:
  - 4KB, if LOCATION(BELOW)
  - FIXED4KB, if LOCATION(ABOVE)
    - permanent backing by real storage for life of Queue Manager
    - no need to programmatically page fix/unfix when doing I/O
    - better performance
    - ensure you have enough real storage available !
- LOCATION
  - BELOW – buffer pool is below the bar (default)
  - ABOVE – buffer pool is above the bar

64 Bit Buffer Pools: What has changed?

NOTES

- A new attribute called LOCATION, or LOC for short has been added to the buffer pool definition. This enables the location, relative to the bar to be specified.

  A value of “BELOW” indicates that the buffer pool should be below the bar in 31 bit storage, this is the default and matches what was available in MQ version 7.1 and earlier releases.

  A value of “ABOVE” indicates that the buffer pool should be located above the bar in 64 bit storage.

  The LOCATION value can be altered dynamically, and the queue manager will then dynamically move the buffer pool to the new location.

- The buffer pool BUFFERS attribute now has an extended valid range, being able to accept a value up to 999,999,999 if LOCATION(ABOVE) has been set.

- The buffer pool ID attribute can now have a valid range of 0 to 99.

- A new attribute called PAGECLAS has been added to the buffer pool definition. This attribute enables 64 bit buffer pools to be configured to be permanently backed by real storage for maximum performance. The default value of 4K means that the buffer pool will be page-able. Using a value of FIXED 4KB means that MQ does not have to page fix, page unfix buffers when doing I/O. This can give significant performance benefits if the buffer pool is under stress, and therefore doing lots of reads from, writes to, the page set.

- Note though, storage will be page fixed for the life of the queue manager so ensure you have sufficient real storage otherwise other address spaces might be impacted.

64 Bit Buffer Pools: Migration

- To use this function OPMODE(NEWFUNC,800) must be set
  - Otherwise behaviour is same as in version 7
  - Though, LOCATION(BELOW) is valid regardless of OPMODE

64 Bit Buffer Pools: Migration

NOTES

- To exploit 64 bit storage for the buffer pools, the queue manager must be running in version 8 OPMODE NEWFUNC. However, LOCATION(BELOW) can be specified when running in OPMODE COMPAT.
- Some console messages have changed, regardless of OPMODE. For example to display the location of the buffer pool when using the DISPLAY USAGE PSID command.

64 Bit Buffer Pools: Configuration

- CSQINP1
  - DEFINE BUFFPOOL(22) LOCATION(ABOVE) BUFFERS(1024000) REPLACE
  - DEFINE BUFFPOOL(88) BUFFERS(12000) REPLACE
- CSQINP1 or dynamically
  - DEFINE PSID(22) BUFFPOOL(22)
  - DEFINE PSID(88) BUFFPOOL(88)
- CSQINP2 or dynamically
  - ALTER BUFFPOOL(88) LOC(ABOVE)

```
CSQP024I !MQ21 Request initiated for buffer pool 88
CSQ9022I !MQ21 CSQPALTB ' ALTER BUFFPOOL' NORMAL COMPLETION
CSQP023I !MQ21 Request completed for buffer pool 88,  now has 12000 buffers
CSQP054I !MQ21 Buffer pool 88 is now located above the bar
```

64 Bit Buffer Pools: Configuration

NOTES

- This slide shows the enhanced buff pool commands. The places that above the bar buffer pools are defined are the same as when they are below the bar.

  MQ allows a buffer location to be moved dynamically. This slide shows the console messages when this is done.

64 Bit Buffer Pools: Migration

- Some messages have changed regardless of the value of OPMODE
  - Space has been added to allow for a larger number for buffers
  - PAGE CLASS and LOCATION can be seen on DISPLAY USAGE

```
CSQI010I !MQ21 Page set usage …
<REMOVED>
End of page set report
CSQI065I !MQ21 Buffer pool attributes ...
      Buffer   Available   Stealable   Stealable   Page        Location
        pool     buffers     buffers  percentage   class

_          0        1024        1000          99   4KB         BELOW
_         22     1024000      234561          23   FIXED4KB    ABOVE
_         88       12000        1200          10   4KB         ABOVE
      End of buffer pool attributes
```

64 Bit Buffer Pools: Migration

NOTES

- Some console messages have changed, regardless of OPMODE. For example to display the location of the buffer pool when using the DISPLAY USAGE PSID command.

64 Bit Buffer Pools: Performance Summary

Single Requester per Queue:

Test	Transaction Rate (per second)	Transaction Cost (cpu microseconds)	LPAR %Busy	Channel Path %Busy
31-bit	232762	35.92	54%	56%
64-bit	235217	37.48	57%	57.4%
64-bit (enough buffers)	324213	38.12	83%	0.07%
64-bit (4GB per buffer pool)	341412	38.23	83%	0.08%

2 Requesters per Queue:

Test	Transaction Rate (per second)	Transaction Cost (cpu microseconds)	LPAR %Busy	Channel Path %Busy
31-bit	149140	42.3	42%	75.4%
64-bit	145623	44.84	43.5%	75.9%
64-bit (enough buffers)	384062	40.65	99.59%	0.08%
64-bit (4GB per buffer pool)	370546	52.15	99.69%	0.07%

- 16 Central Processor LPAR
- Each transaction puts and gets a random message from a pre loaded queue.
- Second test requires a doubling in buffer pool size

64 Bit Buffer Pools: Performance Summary

NOTES

The previous slide shows two tables comparing the performance of 31 bit buffer pools and 64 bit buffer pools.

The first table shows the results when running tests using a single requester on a queue. There is a small increase in transaction cost when using 64 bit buffer pools vs 31 bit buffer pools, with the CPU microseconds increasing from 35.92 to 37.48. However, when we increase the number of buffers in use in the 64 bit case, the channel path %busy drops to nearly 0, indicating that we are no longer needing to access the page set, and all requests are satisfied from the buffer pool. The transaction rate has also increased by about 40%.

The second table shows that when using two requesters against the queue, there is a high channel path %busy rate, of about 75%, for both the 31 bit and 64 bit buffer pool case. However, when extra buffers are added in the 64 bit case, this channel path busy drops to nearly 0 and the transaction rate more than doubles. The LPAR busy % also increases from about 43% to very close to 100%, showing that we are now driving the system as hard as possible.

Being able to provide more buffers by using 64 bit storage means that we can drive the system much more efficiently for a slight increase in per transaction cost.

8 Byte Log RBA

8 byte log RBA

NOTES

- Next, we'll take a look at what we have done in MQ V8 to increase the size of the log range.

6 byte log RBA: The Problem

- MQ for z/OS V7.1 (or earlier):
  - Implements 6 byte Log RBA (Relative Byte Address)
  - Gives an RBA range of **0 to x'FFFFFFFFFFFF' (= 255TB)**
  - Some customers reach this limit in 12 to 18 months
  - At 100MB/sec, end of log range would be reached in 1 month

6 byte log RBA: The Problem

NOTES

- With MQ version 7.1 or earlier the relative byte address, RBA is 6 bytes long. MQ uses RBA to track log records, when MQ handles persistent data, or units of work which involves writing to the log, the RBA increases depending on the size of the log record. The length of 6 bytes gives a range from 0 to x'FFFFFFFFFFFF' which is approximately 255 terabytes.
- Although this sounds like a large amount of data, with speeds of the modern z systems, the throughput that customers are pumping through MQ means that some customers have been reaching this limit in between 12 and 18 months. If MQ is writing to the log constantly at 100 MB per second, which is achievable on the modern hardware, it would only take a month to reach the end of the log. At which point, the queue manager terminates and requires a cold start. This could also result in loss of persistent data.

Warning Messages and abend

- V7.1 Queue Managers do issue warning messages as log RBA gets high:

CSQI045I when log RBA is x'700000000000', x'7100..', x'7200..' and x'7300..'
CSQI046E when log RBA is x'740000000000', x'7500..', x'7600..' and x'7700..'
CSQI047E when log RBA is x'780000000000', x'7900..', x'nn00..' and x'FF00..'

- **APAR PM48299 (**WebSphere MQ V7.0.1 and above) added messages:

CSQJ032E when log RBA is higher than x'F80000000000'
CSQJ031D to confirm restart even though log RBA has passed x'FF8000000000'

- To prevent loss of data, Queue Managers with **APAR PM48299** applied:

Terminate with abend 00D10257 when log RBA reaches x'FFF800000000'

Warning Messages and abend

NOTES

- As the end of the log approaches, MQ issues warning messages, initially informational (I), then eventual action (E) indicating that a log reset is required.
- An APAR was taken to modify the thresholds as some customers felt they were premature. In the same APAR, we took the opportunity to be more robust (ABEND and WTOR to restart) in enforcing that it would be hard to run a queue manager through the end of log and RBA wrap!

6 byte log RBA: The Problem

- If 'end' of the Log RBA range is reached:
  - You get an unplanned outage
    - Queue Manager terminates
    - Requires a “cold” start – a disruptive outage !
    - Potential for loss of persistent data
- To avoid an unplanned outage, at regular planned intervals:
  - Quiesce the Queue Manager
  - Run CSQUTIL RESETPAGE
    - RESETs the LOG RBA in header of each page
  - Restart the Queue Manager
  - Some customers are happy to do this, but others are not !

6 byte log RBA: RESETPAGE

NOTES

- MQ does provide a procedure to avoid a disruptive outage. MQ has a RESET PAGE function on CSQUTIL to reset the log RBA, this should be planned at regular intervals but does require the queue manager to be stopped while the utility is run.
- To RESET the log RBA (procedure is documented in the InfoCenter):
  - Ensure there are no unresolved units of work
  - Shut down the queue manager cleanly
  - Define new logs and BSDS
  - Run the **CSQUTIL RESETPAGE** utility against all the queue manager's page sets
- The CSQUTIL step can take time to complete, resulting in a long outage
  - Needs to reset the log RBA in the header of each page in each page set
- Once the log RBA in the page sets has been reset:
  - Restart with the new logs and BSDS

Note: Don't restart the Queue Manager with the old logs and BSDS.

8 byte log RBA: The Solution

- Implement an 8 byte (64-bit) log RBA
  - Gives an RBA range of **0 to x'FFFFFFFFFFFFFFFF'**
    - Upper limit on logical log is **now 64K times bigger**
      - At a 100MB/sec, this **would take >5000 years** to fill
  - Format of BSDS and log records has changed to accommodate 8 byte RBAs
  - URIDs are now 8 bytes long
  - Utilities or applications that read the BSDS and Logs have been updated
  - Console messages that contain the log RBA or URID have been updated
  - Queue Manager uses 6 byte log RBAs until 8 byte log RBAs are enabled

8 byte log RBA: The Solution

NOTES

In MQ version 8 we have extended the log RBA from 6 to 8 bytes which doesn't sound a lot but the log is approximately 65,000 times larger. Thinking about the logging rate example it would take >5,000 years to fill at 100 megabytes per second.

This feature has changed the BSDS and log records to handle the new 8 byte RBAs and URIDs. Utilities or applications that read the BSDS, the logs, or the console messages, that contain the log RBA or URID have been updated (**Note:** Vendor utilities may be impacted by these changes. You will need to check with your vendors).

When the queue manager is running at MQ version 8, it has been designed to use 6 byte log RBA until 8 byte log RBA is enabled. This allows both backwards migration and co-existance with queue managers at a prior release.

Console messages at version 8 show the full 8 bytes regardless to whether 8 byte log RBA is enabled, the 6 byte log RBA values have four zeroes pre-pended.

A BSDS conversion utility, CSQJUCNV (see later slides), has been provided for migration to 8 bytes. This is a one way migration process and its the same model that DB2 uses. Once a queue manager has been enabled to use 8 byte RBAs, there is no option to backwards migrate it.

Enabling 8 byte log RBAs

- Procedure to enable 8 byte log RBAs:
  - Stop the QMgr cleanly
  - Enable OPMODE(NEWFUNC,800)
    - In a QSG, new function mode is entered once all QMgrs have been started at NEWFUNC

      Important that all QMGRs in a QSG have been restarted with OPMODE(NEWFUNC,800) !
  - Define new BSDSs in V1 format (these will be used to create the V2 format BSDSs)
    - V2 format BSDS contains more data than V1 format BSDS
    - Recommended space allocation is now RECORDS(850 60)
    - CSQ4BSDS sample job has been updated with this value
  - Run BSDS conversion utility (CSQJUCNV) to convert the V1 BSDS to V2
    - Creates a copy of a V1 format BSDS in V2 format
    - Checks all QSG QMgrs are running OPMODE(NEWFUNC,800)
  - Rename BSDSs so that V2 BSDSs are used during next restart of QMgr
  - Restart the QMgr

Enabling 8 byte log RBAs

NOTES

So how do you use this log RBA change to practically eliminate the need to reset the RBA.

First the queue manager needs to be stopped cleanly.

A queue manager has to be running MQ version 8 and in OPMODE NEWFUNC, **this indicates that you promise not to fall back to a prior version**. If the queue manager is part of a QSG, the entire group must be at version 8 and in OPMODE NEWFUNC.

Then the new CSQJUCNV utility needs to be run. It performs QSG checks to ensure the levels are correct. CSQJUCNV will then convert the BSDS data from old primary/ secondary BSDS pair to new pair (the old data is not changed in case the queue manager should fail to start with the new BSDSs for some reason and you need to start it with the old BSDSs again).

Finally the queue manager needs to be restarted with the new 8 byte BSDSs. The queue manager will ONLY start if in NEWFUNC mode. A new console message, CSQJ034I, is issued to indicate that the queue manager is running in 8 byte RBA mode (see later slide). All subsequent log data will be written in the new format.

Note: In order to enable 8 byte log RBA on a brand new V8 Qmgr:

1) The BSDS must first be defined and formatted in V1 by CSQJU003
2) Then the BSDS must be converted to V2 by CSQJUCNV

BSDS conversion utility (CSQJUCNV)

- Parameters
  - **NOQSG** (specify for a stand alone queue manager)
    - No OPMODE checks performed
  - **INQSG,*qsgname*,*dsgname*,*db2ssid*** (specify for a queue manager in a QSG)
    - Utility checks that all QMgrs in the QSG have been started at OPMODE(NEWFUNC,800) before allowing conversion to proceed
- Example JCL:

```
//CSQ4BCNV JOB
//CONVERT  EXEC PGM=CSQJUCNV,REGION=32M,PARM=('INQSG,SQ13,DB2,DB4A')
//STEPLIB  DD DSN=ANTZ.MQ.V000.CUR.SCSQAUTH,DISP=SHR
//         DD DSN=ANTZ.MQ.V000.CUR.SCSQANLE,DISP=SHR
//         DD DSN=SYS2.DB2.V10.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=VICY.MQ1O.BSDS01,DISP=SHR
//SYSUT2   DD DSN=VICY.MQ1O.BSDS02,DISP=SHR
//SYSUT3   DD DSN=VICY.MQ1O.NEW.BSDS01,DISP=OLD
//SYSUT4   DD DSN=VICY.MQ1O.NEW.BSDS02,DISP=OLD
```

Externals – BSDS conversion utility (CSQJUCNV)

NOTES

- DD statements
  - SYSUT1 (input BSDS copy 1) and SYSUT3 (output BSDS copy 1) always required
  - SYSUT2 (input BSDS copy 2) is optional (used for consistency check with SYSUT1)
  - SYSUT4 (output BSDS copy 2) is required if dual BSDSs are used

Externals – BSDS conversion utility (CSQJUCNV)

- Typical output

```
CSQJ445I CSQJUCNV BSDS CONVERSION UTILITY - 2014-06-04 15:02:48
CSQU526I CSQJUCNV Connected to DB2 DB4A
CSQU528I CSQJUCNV Disconnected from DB2 DB4A
CSQJ200I CSQJUCNV UTILITY PROCESSING COMPLETED SUCCESSFULLY
```

New message CSQJ034I on QMgr startup

- Issued during QMgr startup
- Indicates whether QMgr is running in 6 or 8 byte RBA mode
  - **0000FFFFFFFFFFFF** – 6 byte RBA mode

```
11.25.05 STC05120  CSQJ127I !MQ4E SYSTEM TIME STAMP FOR BSDS=2014-04-02 11:19:18.70
11.25.05 STC05120  CSQJ001I !MQ4E CURRENT COPY 1 ACTIVE LOG DATA SET IS  280
   280             DSNAME=VICY.MQ4E.LOGCOPY1.DS04, STARTRBA=00000000038F4000
   280             ENDRBA=0000000003B0FFFF
11.25.05 STC05120  CSQJ099I !MQ4E LOG RECORDING TO COMMENCE WITH  281
   281             STARTRBA=00000000039AF000
11.25.05 STC05120  CSQJ034I !MQ4E CSQJW007 END OF LOG RBA RANGE IS 0000FFFFFFFFFFFF
```

  - **FFFFFFFFFFFFFFFF** – 8 byte RBA mode

```
22.57.53 STC13100  CSQJ001I !MQ08 CURRENT COPY 2 ACTIVE LOG DATA SET IS  810
   810             DSNAME=VICY.MQ08.LOGCOPY2.DS01, STARTRBA=0000000002760000
   810             ENDRBA=0000000003B0FFFF
22.57.53 STC13100  CSQJ099I !MQ08 LOG RECORDING TO COMMENCE WITH  811
   811             STARTRBA=0000000002AA8000
22.57.53 STC13100  CSQJ034I !MQ08 CSQJW007 END OF LOG RBA RANGE IS FFFFFFFFFFFFFFFF
```

Warning message thresholds in V8

- The messages issued remain the same:

CSQI045I, CSQI046E, CSQI047E, CSQJ031D and CSQJ032E

- The thresholds at which the qmgr starts to issue warning messages are:
  - With 8 byte RBAs disabled:

```
HIGH RBA ADVICE    '0000F00000000000'x
HIGH RBA WARNING   '0000F80000000000'x
HIGH RBA CRITICAL  '0000FF8000000000'x
HIGH RBA ABEND     '0000FFF800000000'x
```

- The thresholds for issuing the CSQI messages have been increased to match those for the CSQJ messages:
  - With 8 byte RBAs enabled:

```
HIGH RBA8 ADVICE    'FFFF800000000000'x
HIGH RBA8 WARNING   'FFFFC00000000000'x
HIGH RBA8 CRITICAL  'FFFFFC0000000000'x
HIGH RBA8 ABEND     'FFFFFFC000000000'x
```

- Once the threshold is exceeded, the frequency at which messages are issued remain the same

Updates to existing messages

- URIDs and RBAs in command outputs and console messages are 8 bytes:
  - Output of the DISPLAY CONN command looks like:

```
CSQM201I !MQ1P CSQMDRTC  DIS CONN DETAILS
CONN(CC15CF64B98D0001) EXTCONN(C3E2D8C3D4D8F1D7404040404040404040)
TYPE(CONN)
QMURID(0000000000078599)
END CONN DETAILS
```

  - Console Message **CSQE130I** (for CF structure recovery) looks like:

```
CSQE130I !MQ1P CSQERCF2 Recovery of structure
APPLICATION1 started, using MQ1P log range from RBA=000000000007B663 to
RBA=000000000007B6AB
```

Print log map (BSDS) utility (CSQJU004)

- Changed to always print 8 byte log RBA values
- Now displays BSDS version

```
LOG MAP OF BSDS DATA SET COPY 1, DSN=VICY.MQ1P.BSDS01
  BSDS VERSION        - 1
  SYSTEM TIMESTAMP    - 2014-05-23  17:47:14.85
  UTILITY TIMESTAMP   - 2014-05-23  13:40:09.56
  HIGHEST RBA WRITTEN    0000000000090120  2014-05-23  17:47:14.5
  HIGHEST RBA OFFLOADED  0000000000000000
```

Change log inventory utility (CSQJU003)

- Now accepts RBA values up to 16 characters long in parameters...

```
//CSQJU003 EXEC PGM=CSQJU003,REGION=0M
//SYSUT1   DD   DISP=SHR,DSN=VICY.MQ1O.BSDS01
//SYSUT2   DD   DISP=SHR,DSN=VICY.MQ1O.BSDS02
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 CHECKPT STARTRBA=00ABCD0000040000,ENDRBA=00ABCD0000045000,
         TIME=20140201650000
```

- But, a value greater than 0000FFFFFFFFFFFF results in error for V1 BSDS:

```
CSQJ443I CSQJU003 CHANGE LOG INVENTORY UTILITY - 2014-06-03 17:20:16
CHECKPT STARTRBA=00ABCD0000040000,ENDRBA=00ABCD0000045000,
        TIME=20140201650000
CSQJ456E STARTRBA PARAMETER ARGUMENT EXCEEDS MAXIMUM VALUE FOR BSDS VERSION 1
CSQJ456E ENDRBA PARAMETER ARGUMENT EXCEEDS MAXIMUM VALUE FOR BSDS VERSION 1
CSQJ221I PREVIOUS ERROR CAUSED CHECKPT OPERATION TO BE BYPASSED
CSQJ201I CSQJU003 UTILITY PROCESSING WAS UNSUCCESSFUL
```

Changes to log print utility (CSQ1LOGP)

- CSQ1LOGP has been updated to handle 8 byte RBAs and URIDs
  - RBA of log records, and URIDs in log records, displayed in 8 byte format
- LRSN value now formatted as timestamp in log record header information

8 byte Log RBA

```
000000000276469D  URID(0000000002764000)  RM(DATA)  LRID(00000000.00003001)  TYPE( COMPENSATING LOG RECORD UNDO REDO )
      SUBTYPE( DECREMENT BY )
      LRSN(CCE1B6A55ECE)                                                                    16:07:43.982816 20140320
     ****  00400034 0601000F C9880000 00000276 40000000 00000276 46698024 CCE1B6A5  *        Ih                      v
     ****  5ECE0002 00000000 027644C4                                              *;         D
     0000  00000000 00003001 00040416 00000001 00000001                            *


000000000276470D  URID(0000000002764000)  RM(DATA)  TYPE( COMPENSATING LOG RECORD UNDO REDO )
      SUBTYPE( NULL )
      LRSN(CCE1B6A55ECF)                                                                    16:07:43.982832 20140320
     ****  00640040 06010001 C9880000 00000276 40000000 00000276 469D8024 CCE1B6A5  *        Ih                      v
     ****  5ECF0002 00000000 02764468                                              *;
     0000  00000000 00003001 E2E3E2E3 C5D44BC3 D3E4E2E3 C5D94BC3 D6D4D4C1 D5C44BD8  *        SYSTEM.CLUSTER.COMMAND.Q
     0020  E4C5E4C5 40404040 40404040 40404040 40404040 40404040                    *UEUE
```

8 byte URID

Formatted LRSN

Changes to log print utility (CSQ1LOGP)

- New message **CSQ1219I** issued at the start of the output and whenever the format of the log record changes to indicate:
  - Whether the log records are in 6 or 8 byte RBA format
  - Whether the qmgr is in a QSG

```
CSQ1219I LOG RECORDS CONTAIN 6 BYTE RBA - QSG(NO)
CSQ1219I LOG RECORDS CONTAIN 6 BYTE RBA – QSG(YES)

CSQ1219I LOG RECORDS CONTAIN 8 BYTE RBA - QSG(NO)
CSQ1219I LOG RECORDS CONTAIN 8 BYTE RBA - QSG(YES)
```

Changes to log print utility – EXTRACT(YES)

- Irrespective of whether the log record being read contain 6 or 8 byte URID and RBA values, the Extract function has been changed to produce records (in output datasets) with 8 byte URIDs and RBAs.
- The following example shows records written to the CSQCMT dataset (the 8 byte URIDs are marked):

```
                                    | URID | (now 8 bytes)
01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
2014.079 15:07:43.582      0ö÷¶uÙ.ô¯.....Î.m003.RCRSC 02SYSOPR  ö÷¶uÙ.ô¯        ........MQ08    BUR........
2014.079 15:07:43.582      0ö÷¶uÙ.ô¯.....Î.m003.RCRSC 02SYSOPR  ö÷¶uÙ.ô¯        ........MQ08    BUR........
2014.079 15:07:43.702      0ö÷¶v.ê.].....Î.*201.SCAVNG01SYSOPR  ö÷¶v.ê.]        ........MQ08    BUR........
2014.079 15:07:43.702      0ö÷¶v.ê.].....Î.*201.SCAVNG01SYSOPR  ö÷¶v.ê.]        ........MQ08    BUR........
...
...
```

- The records are mapped by CSQ4LOGD which has been updated to allow for 8 byte URIDs and RBAs
- Commit output dataset can be used to replay the logs

Replaying the logs

- **CSQ4LOGD** (format of records produced by **CSQ1LOGP EXTRACT(YES)**) now declares 8 byte URID and RBAs
- Sample **CSQ4LOGS** has been updated to replay messages based on 8 byte URID field e.g.

Snippet from the **CSQ4LOGS** job output that shows 5 msgs being replayed to queue TEST using log records that contain 8 byte URIDs. The Summary shows that 400 msgs in total were replayed/MQPUT1 (I had 400 msgs on queue TEST):

```
....
==> New Unit of Recovery started <==
Unit of recovery ID ....' (hex) '0000000002AF1508' : (disp) '........'
Correlator ID...........' (hex) '1B414840E7E75C5C1B414B00' : (disp) '... XX**....'
Authorisation ID........' (hex) 'D4C1E8E4D9404040' : (disp) 'MAYUR   '
Thread start TOD value..' (hex) 'CCEB892D8156ABB1' : (disp) '..i.a...'
Resource identifier.....' (hex) '4040404040404040' : (disp) '        '
Connection type.........' (hex) 'C3C8C9D540404040' : (disp) 'CHIN    '
Connection identifier...' (hex) 'D4D8F0F8C3C8C9D5' : (disp) 'MQ08CHIN'
MQPUT1 - Length=100,Queue=TEST                                        CC=0, RC=0
MQPUT1 - Length=100,Queue=TEST                                        CC=0, RC=0
MQPUT1 - Length=100,Queue=TEST                                        CC=0, RC=0
MQPUT1 - Length=100,Queue=TEST                                        CC=0, RC=0
==> Summary of UR activity       <==
==> Message puts : 00000005, ==> Message gets : 00000001, ==> Message expired : 0000...
alters : 00000000.
==> End summary of UR activity   <==
....

=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=
==> Total records read from input file.: 8153
==> Records read of unknown type.......: 0
==> Total Units of recovery processed..: 2666
==> Total MQPUT1(replay) requests......: 400
==> Successful MQPUT1(replay) requests.: 400
=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=
```

8 byte URID displayed

Backwards Migration

- Backwards migration is NOT possible once NEWFUNC is enabled
- Cannot start a QMgr previously run in 8 byte RBA mode, in 6 byte RBA mode

00D92023 – 8 byte RBA log record read during restart in 6 byte log RBA mode

Channel Initiator (CHINIT) SMF Data

Channel Initiator (CHINIT) SMF data

NOTES

In V8 we have added the recording of Channel Initiator Statistics and Channel Accounting Data to SMF.

Chinit SMF: The Problem

- Prior to MQ V8 no SMF data for:
  - CHINIT address space
  - Channel activity
- Many customers have had to create their own ‘monitoring’ jobs
  - With periodic **DISPLAY CHSTATUS** commands
- Difficult to:
  - Manage historical data
  - Investigate performance issues
  - Perform capacity planning

CHINIT SMF: The Solution

- New SMF data for CHINIT address space:
  - **Channel Initiator Statistics** (SMF 115, SubType 231)
    - High level view of activity in CHINIT
      - Number of channels and TCB usage
      - Dispatchers, Adapters, DNS, SSL
        - Do I have spare capacity ?
        - Do I need more or less dispatchers/adapters ?
  - **Channel Accounting Data** (SMF 116, SubType 10)
    - Detailed view of individual channels
      - What work have channels been doing ?
      - Which channels are being heavily utilised ?
    - Controlled by STATCHL attribute on QMgr and Channel definition

Chinit SMF: The Solution

- Useful for
  - Monitoring
  - Capacity planning
  - Tuning
- Separate controls from queue manager SMF allows 'opt in'
- Supportpac MP1B updated to:
  - Format new data

Please attend

Session 17044 – But Wait, There's More MQ SMF Data Now?!?! - Monitoring your Channels Using V8's New Chinit SMF Data [z/OS]

On

Weds Mar 4th at 4:30pm in Seneca

for more details

Storage Class Memory (SCM) (Flash)

Storage Class Memory (SCM) (Flash)

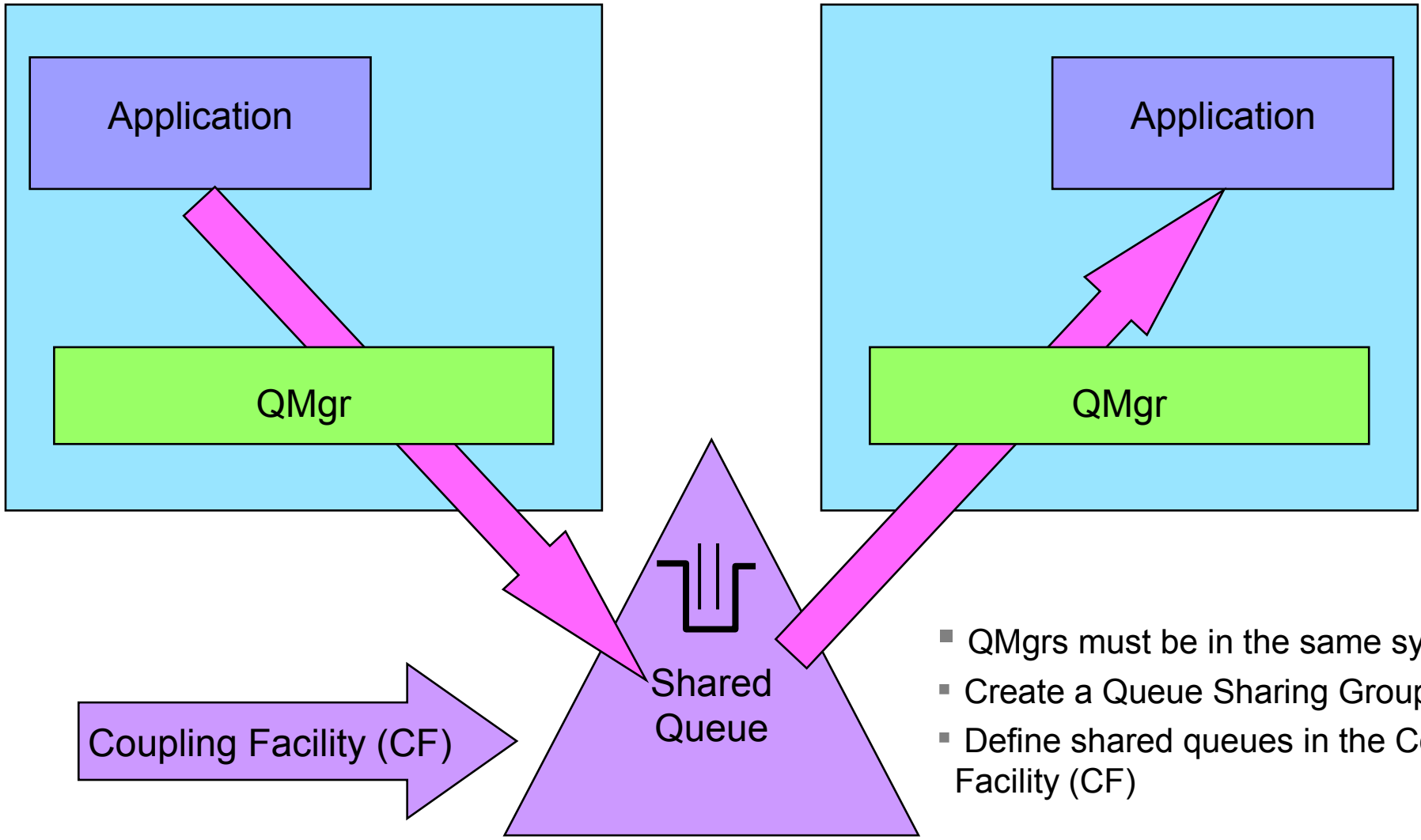
NOTES

This is a change that the Coupling Facility (CF) development team have implemented to help a Queue Sharing Group (QSG) store messages. The cost of storing messages in the CF is reduced but some augmented storage (used from the CF) is required to achieve this gain.

This is not a true version 8 feature because all the configuration is done at the CF level, with the right hardware (zEC12 or zBC12 with Flash Express cards (Solid State Drives) installed) and software. This means that this CF flash memory can be used with MQ version 8 as well as prior versions. Each Flash Express card has a capacity of 1.6 TBs and up to 4 cards can be installed giving a total of 6.4 Tbs.

With the z/OS V1R13 RSM Enablement Offering web deliverable (FMID JBB778H) for z/OS V1R13, z/OS exploits Flash through a new tier of memory called Storage Class Memory (SCM) for paging and SVC dump processing. This function is expected to provide faster paging and dump processing because flash storage is faster compared to hard disk storage. In addition to support for the existing large (1 MB) pages and frames, zEC12 supports pageable large pages when SCM is configured and allocated to z/OS.

Shared Queues



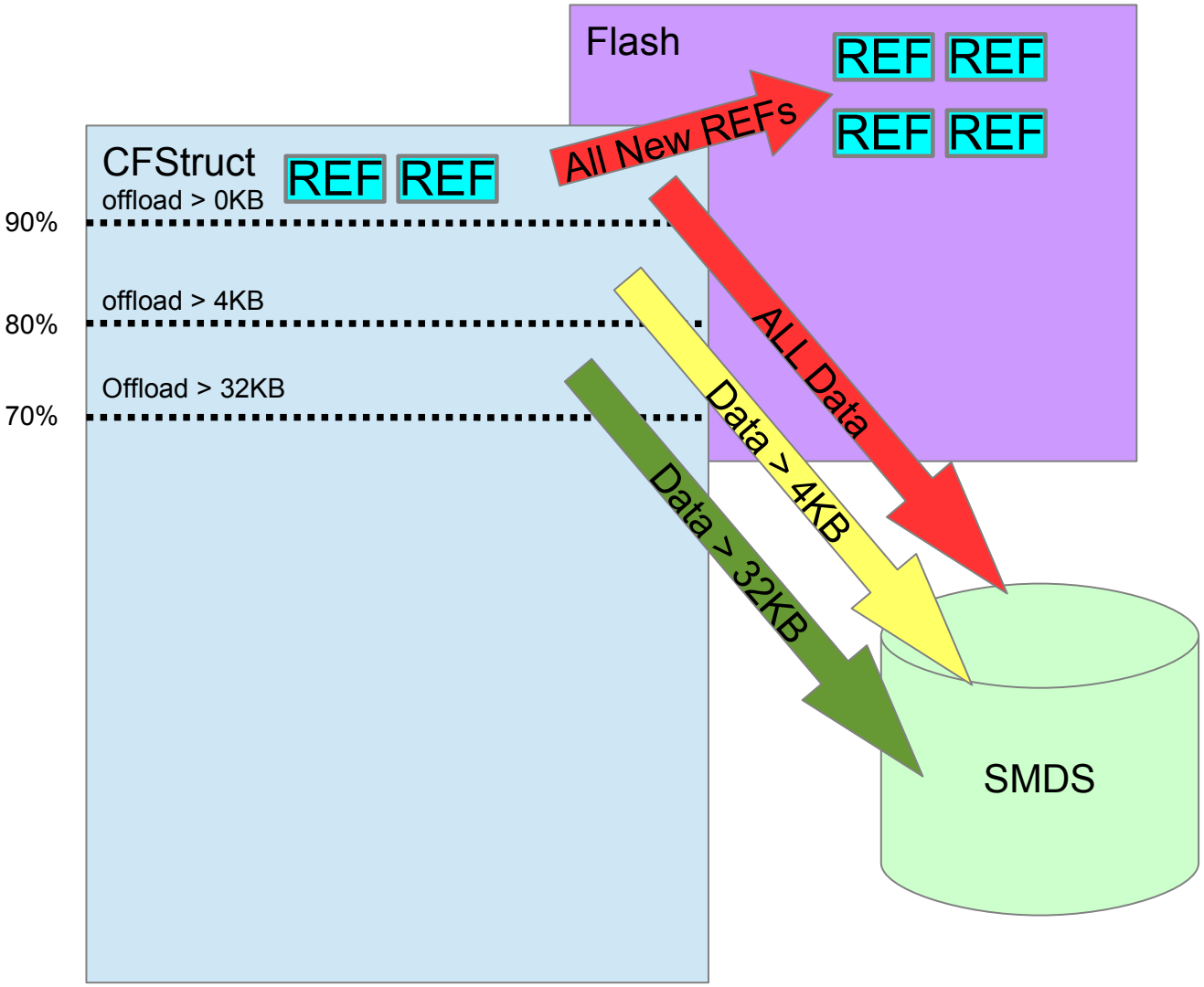
- QMgrs must be in the same sysplex
- Create a Queue Sharing Group (QSG)
- Define shared queues in the Coupling Facility (CF)

Shared Queues

NOTES

- A **Parallel Sysplex** is a Sysplex that uses one or more **coupling facilities (CFs)**, which provide high-speed caching, list processing, and lock processing for any applications on the Sysplex.
- MQ exploits the CF to implement Shared Queues.
- Queue Managers configured in a Queue Sharing Group (QSG) connect to the CF to process messages on shared queues. This provides for high availability.

CF Flash: Scenarios Planned Emergency Storage



CFSTRUCT OFFLOAD rules cause progressively smaller messages to be written to SMDS as the structure starts to fill.

Once 90% threshold is reached, the queue manager stores the minimum data per message (reference message) to squeeze as many message references as possible into the remaining CF storage.

Once at 90% threshold, CF Flash pre-staging algorithm also starts to move reference messages for new messages arriving into the CF structure into SCM (assume msgs are of the same priority). Older messages, which are likely to be got first are kept in the faster CF storage.

Note: Assume all msgs < 63KB

CF Flash: Scenarios Planned Emergency Storage

NOTES

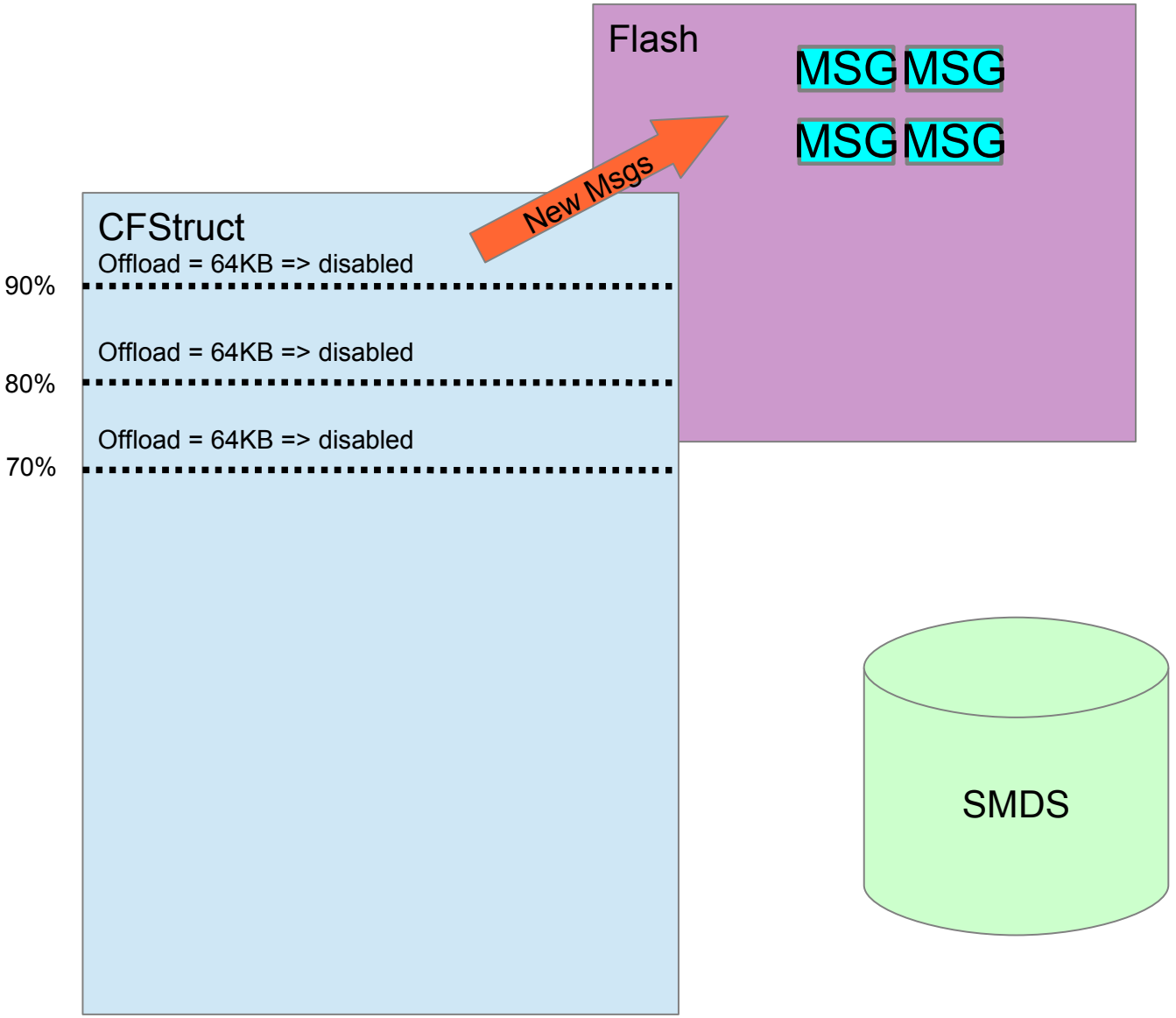
This first slide shows a planned emergency storage scenario.

The CF is being used mainly for messages but, Shared Message Data Sets (SMDS) are configured to hold messages once some the CF structure offload (CFSTRUCT OFFLOAD) rules come into play. The rules cause progressively smaller messages to be written to SMDS as the structure starts to fill up.

Once the 90% threshold is reached, the queue manager stores the minimum data per message in the CF to squeeze as many message references as possible into the remaining storage in the CF structure.

The CF development team have used queuing theory to develop the Flash algorithm. Messages that have been put most recently and have the same (i.e. all messages on the queue have the same) or lowest priority are the most unlikely to be got next, so the CF also starts moving theses new reference messages out to flash storage, keeping the faster CF storage for messages most likely to be gotten next.

CF Flash: Scenarios Maximum Speed



We want to keep high performance messages in the CF for most rapid access.

CFSTRUCT OFFLOAD are configured with special value '64k' to turn them off.

Once 90% threshold is reached, the CF Flash algorithm starts moving new messages to flash storage, keeping the faster 'real' storage for messages most likely to be gotten next.

As messages are got and deleted, the CF flash algorithm attempts to pre-stage the next messages from flash into the CFSTRUCT so they are rapidly available for MQGET.

In this scenario the flash storage acts like an extension to 'real' CFSTRUCT storage. However it will be consumed more rapidly since all message data is stored in it. Though, you could define a threshold to offload >16KB messages to SMDS if the CF structure is say 40% full. This would mean that only messages <=16KB ever get moved to flash storage.

Note: Assume all msgs < 63KB

CF Flash: Scenarios Maximum Speed

NOTES

In this scenario, we want to keep high performance messages in the CF for most rapid access.

The CFSTRUCT OFFLOAD rules are configured with special value 64K to disable off loading.

Once the 90% threshold is reached, the CF Flash algorithm starts moving new messages out to flash storage, keeping the faster - real storage for messages most likely to be gotten next.

As messages are got and deleted, the CF flash algorithm attempts to pre-stage the next messages from flash into the CF structure so they are rapidly available for MQGET.

In this scenario the flash storage acts like an extension to, real CF structure storage. However, it will be consumed more rapidly since all message data is stored in it. However, you may choose to use one rule to alter the large data threshold to indicate that all messages >16KB should be off-loaded to SMDS if the CF structure is 40% full say. This would mean that only messages <=16KB get moved to SCM.

CF Flash: Storage

Scenario	Offload Rule	Msg Size	Total Msgs	# in 'real'	SMDS space	# in 200 GB flash	Augmented (limit 30GB)
No SMDS No Flash		1kB	3M	3M			
		4kB	900,000	900,000			
		16kB	250,000	250,000			
SMDS No Flash	MQ 90%	1kB	3.2M	3.2M	800MB		
	MQ 80%	4kB	1.8M	1.8M	5GB		
	MQ 80%	16kB	1.3M	1.3M	20GB		
„Emergency Scenario	MQ 90%	1kB	190M	2M	270GB	188M	30GB
	MQ 80%	4kB	190M	600,000	850GB	189M	30GB
	MQ 80%	16kB	190M	150,000	3TB	189M	30GB
“Speed” Scenario	CF 90%	1kB	150M	2M		148M	26GB
	CF 90%	4kB	48M	600,000		47M	8GB
	CF 90%	16kB	12M	150,000		11M	2GB

CF Structure size = 4GB

CF Flash: Storage

NOTES

In the table, a CFSTRUCT with SIZE 4 gigabytes is defined. There is a maximum of 200 GB of flash memory and an additional 30 GB of augmented storage available.

The table addresses 4 scenarios, and shows the effect of message size in each, the amount in real, estimates how many MQ messages in CF real storage.

First scenario has no flash nor SMDS. Entire structure is available to store messages. A total of 250,000 16 kilobyte messages can be stored.

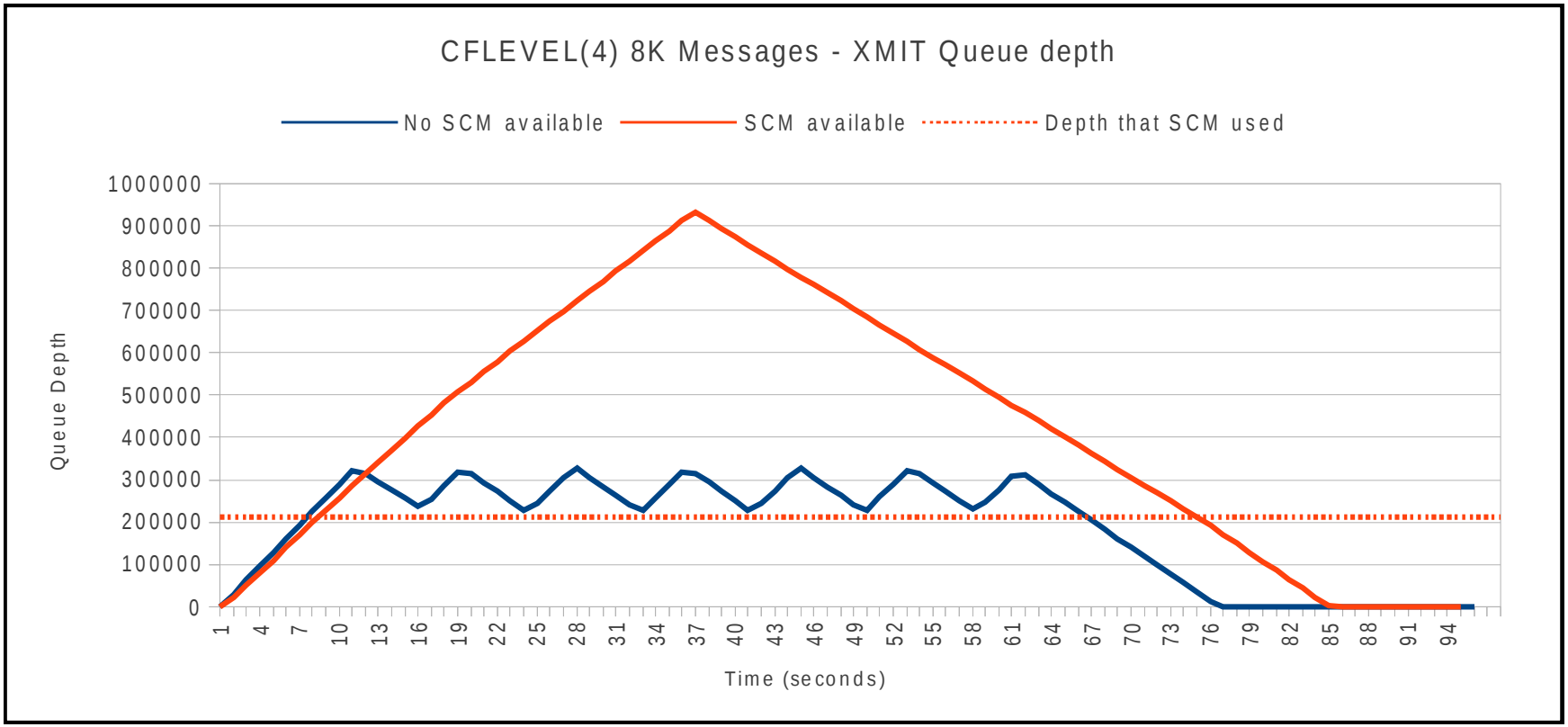
Second scenario introduces SMDS off-load with default rules. The 1k messages don't get to SMDS till 90% full, but the 4k and 16k cases both start off-loading at 80%. The CF space released by off-loading data can hold more message pointers, so the 1k case doesn't increase number of messages greatly, but 4k number doubles, and 16k gets 5 times as many to 1.3 million.

The third scenario is our “Emergency Flash”. Because flash is configured, there is less 'real' storage for storing data, I've assumed 1 gigabytes less for 200 gigabytes flash. Here flash only holds the message references. This means the message size in flash is small. Our experiments show about 175 bytes of Augmented storage per message in flash. We have chosen to limit Augmented storage to 30 gigabytes, so message numbers are limited by augmented storage. The SMDS holds the actual data. In this scenario 190 messages can be stored.

The last scenario is entitled “Max Speed”. All message sizes are below SMDS threshold, so SMDS not used. Limit is now how many messages will fit in 200 gigabytes flash. We show approximate size of augmented storage needed to support these volumes of messages in flash. This gives us space for 12 million messages.

The numbers are estimates only, based on our limited test scenarios, and CFSIZER data.

CFLEVEL(4) using 8KB Messages



- Saw-tooth effect
  - Occurs when putting task goes into retry mode due to MQRC_STORAGE_MEDIUM_FULL
  - Results in 5 sec pauses
- Non-SCM workload still completes in 90% of the time of SCM workload
- CPU cost of non-SCM v's SCM workload in MVS differs by less than 2% (hence, insignificant)
- Get rate once the putting task has completed:
  - Non-SCM: 21100 x 8K messages / second ~ 164MB/sec
  - SCM: 19000 x 8K messages / second ~ 148MB/sec

CFLEVEL(4) using 8KB Messages

NOTES

Chart demonstrates that put and get rates do not significantly alter as CF 'real' storage overflows and data is offloaded to, or read back from, CF flash. This is demonstrated by comparing the slopes of the red and blue lines and noticing no significant kink in red line as we overflow CF 'real' 90% threshold.

NOTE: SCM = Storage Class Memory, alternative terminology for flash storage.

The scenario being tested uses CFLEVEL(4) so SMDS config was not required. However, it corresponds identically with our “Max Speed” scenario above.

The test case has a message generator task putting 8kB messages on to a transmission queue. A channel is getting these in FIFO order. The message generator pauses for a few seconds when it hits storage media full leading to the saw-tooth shape of the blue line where no Flash memory is available.

The red dotted line indicates the threshold at which messages in the red line test, started to be written to flash storage. Notice that it is significantly lower than 90% of the 'media full' blue peaks, because some of the structure storage has gone to control flash, and the threshold is 90% of the remainder.

Final point is that cpu costs are not significantly different whether flash is being used or not.

From performance perspective, using sequential queues, flash memory behaves like CF real.

Other Enhancements

Other Enhancements

NOTES

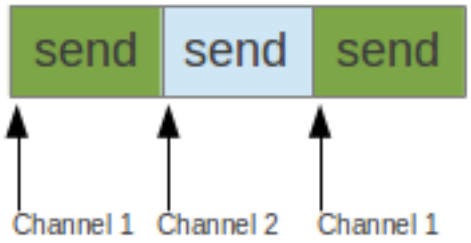
The final section of this presentation will cover other enhancements to MQ in V8.

Channel Compression using zEDC hardware

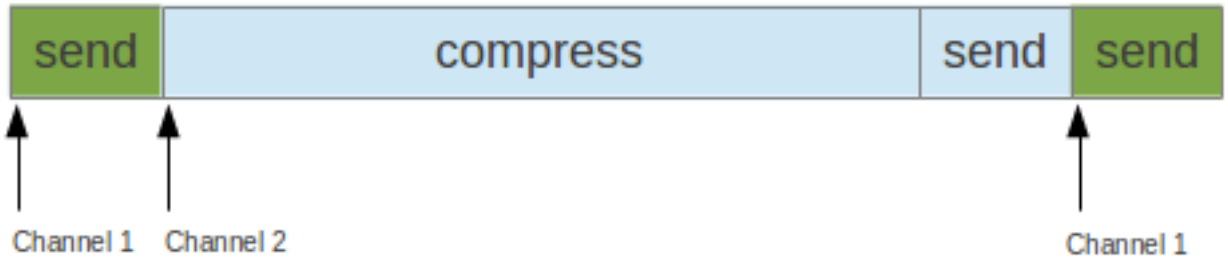
- **ZEC12 - zEnterprise Data Compression (zEDC) card**
  - Hardware compression
- **Channel Compression**
  - Typically used on **high latency/low-bandwidth** networks
  - Reduces amount of data being flowed
  - Reduces CPU cost per message
    - Can yield higher reduction in CPU costs for SSL channels (dispatcher compresses before encrypting)
  - Channel attributes:
    - **COMPHDRS(ZLIBFAST)**
    - **COMPMSG(ZLIBFAST)**
  - Performed by **Dispatcher tasks** in the Channel Initiator address space

Channel Compression - Impact on Dispatcher Task

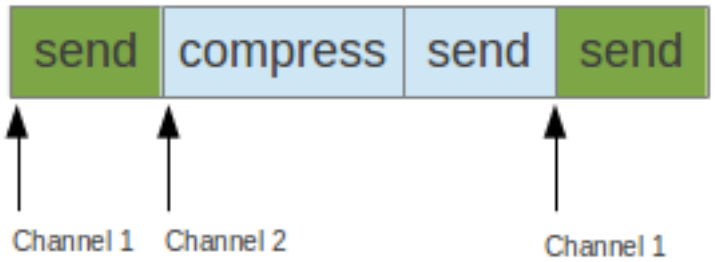
Dispatcher #1 serving 2 channels



Dispatcher #2 serving 2 channels, one with ZLIBHIGH compression



Dispatcher #3 serving 2 channels, one with ZLIBFAST compression



Channel Compression - ZLIBHIGH V's ZLIBFAST

- **Dispatcher tasks:**
  - 7 has 1 channel using ZLIBHIGH for compression
  - 8 has 1 channel using ZLIBHIGH for decompression

```
Task,Type,Requests,Busy %,     CPU used, CPU %,"avg CPU","avg ET"
   ,   ,       ,     ,       Seconds,      , uSeconds,uSeconds
  7,DISP,  146303,  98.5,   58.729109, 97.9,       401,     404
  8,DISP,  147030,  25.7,   15.514522, 25.9,       106,     105
```

98.5% Busy !!

- **Dispatcher tasks:**
  - 7 has 1 channel using ZLIBFAST for compression
  - 8 has 1 channel using ZLIBFAST for decompression

```
Task,Type,Requests,Busy %,     CPU used, CPU %,"avg CPU","avg ET"
   ,   ,       ,     ,       Seconds,      , uSeconds,uSeconds
  7,DISP, 1443847,  71.6,   19.130398, 31.9,        13,      30
  8,DISP, 1431899,  20.4,   12.655084, 21.1,         9,       9
```

17 microsecs spent waiting for hardware compression

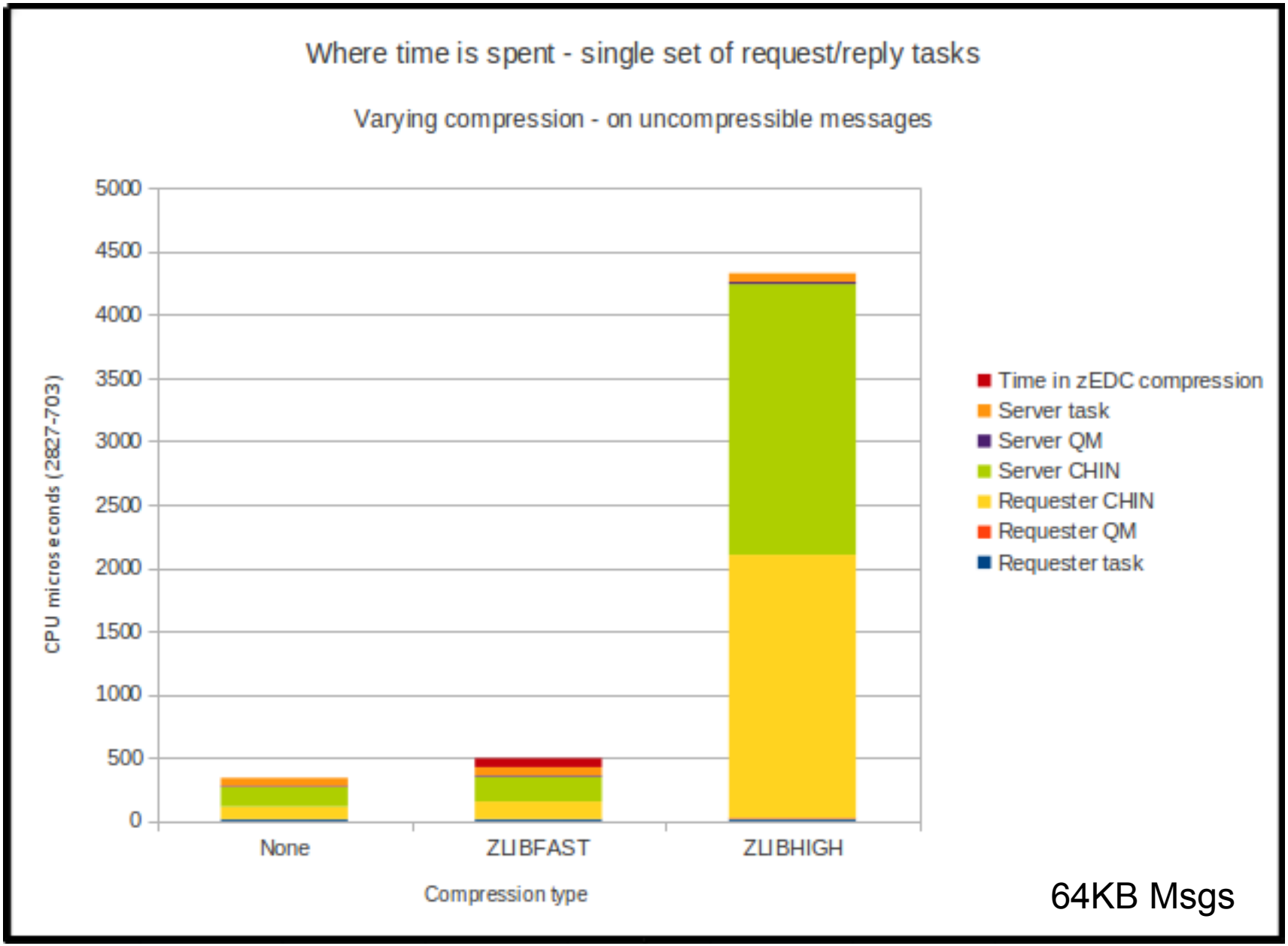
But, overall cost much cheaper than ZLIBHIGH !!

Channel Compression - ZLIBHIGH V's ZLIBFAST

NOTES

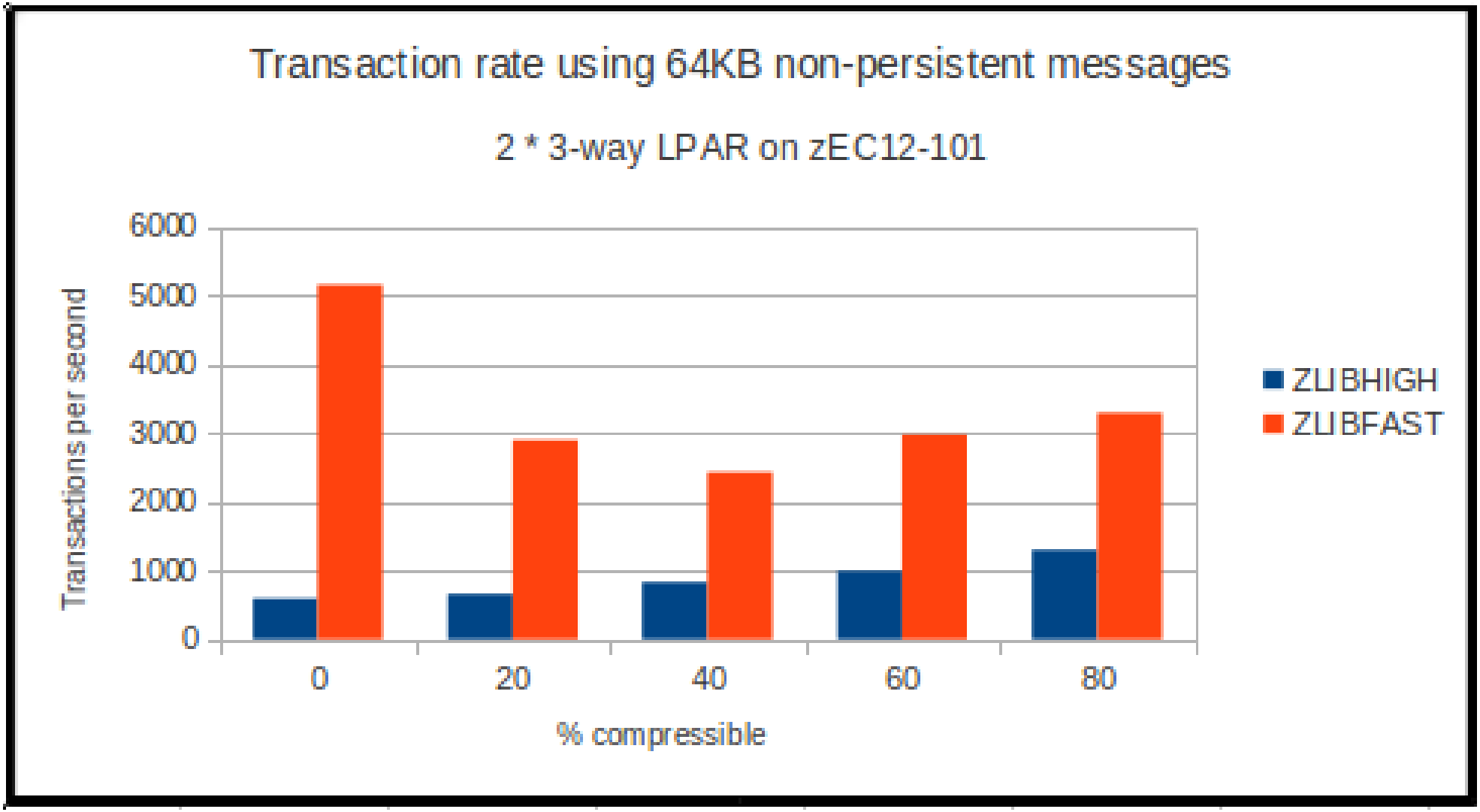
- Data is produced using Channel Initiator Statistics and Account SMF data
  - Dispatcher Report

Channel Compression – Incompressible messages

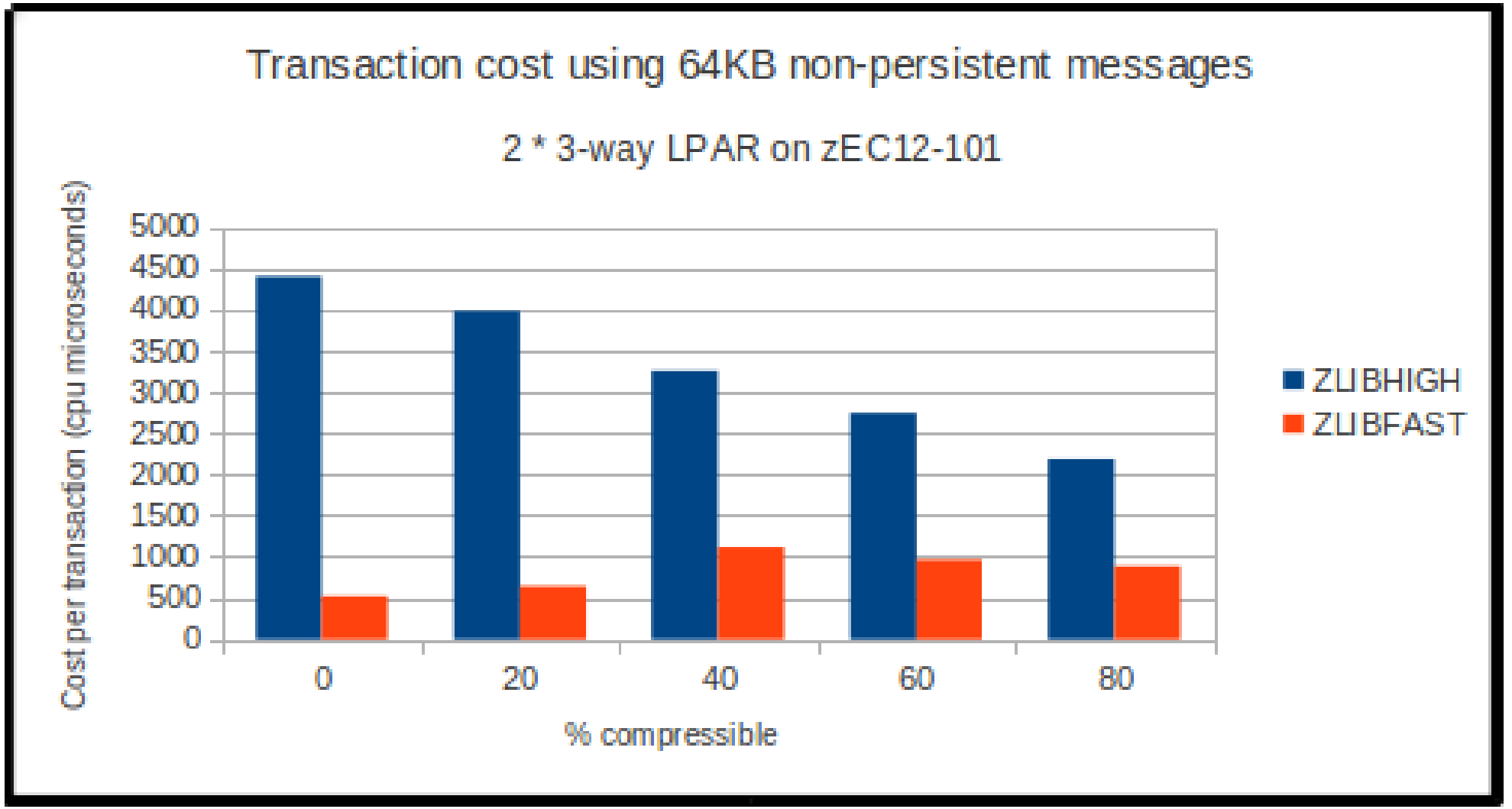


ZLIBFAST Costs Roughly 50% more Than NONE !!

Channel Compression – Compressible messages



Channel Compression – Compressible messages



Increased Active Log Size

- **Pre MQ V8:**
  - Archive to DASD used BDAM
  - Datasets accessed via BDAM are limited to a max of approx. 3.5GB in size (I.e. 65535 tracks)
  - Since MQ does not split an active log into multiple archive logs
    - Active logs are limited to a max of approx. 3.5GB
- **In MQ V8:**
  - Archive to DASD uses BSAM
  - Datasets accessed via BSAM are limited to 4GB in size
  - Hence Active Logs can now be up to 4GB in size
  - But note, larger archive logs result in additional CPU cost in the QMGR
  - So, only use if QMGR is short of active log space

Note: Active logs archived to tape are still limited to amax of 4GB in size

64 bit application support

- 64 bit application support for C language
  - LP64 compile option
  - supported by cmqc.h
- Restricted environments
  - Batch, TSO, USS
  - CICS® and IMS® do not support 64 bit apps
  - WebSphere Application Server is already 64 bit
- Must use sidedeck & DLL, not stubs:
  - csqbmq2x (uncoordinated batch & USS)
  - csqbrr2x (RRS coordinated, srrcmit())
  - csqbri2x (RRS coordinated, MQCMIT)

64 bit application support

NOTES

MQ now supports 64 bit applications written in the C language. There is no support for COBOL, PL1 or assembler. The LP64 option has to be used. The MQ C header file cmqc.h supports both 31 and 64 bit applications.

The support is restricted to batch, T S O and USS. There is no support in CICS and IMS 64 for bit applications. WebSphere Application Server already connects with 64 bit connections.

In order to use the 64 support, include the definition side decks (and not the stubs) in the set of modules to bind. Essentially, programs need to be linked with the definition side decks because the binder uses the definition side decks to resolve references to functions and variables defined in the DLL.

Client Attachment Feature (CAF)

- Now shipped as part of the base MQ for z/OS product
- No longer chargeable on earlier releases of MQ
  - APAR available to enable functionality without installing CAF
- This means that client capability is available by default
  - Use CHLAUTH rules if you don't want Clients to connect to your QMgr

Client Attachment Feature

NOTES

The Client Attachment Feature, which was required on earlier versions of MQ to connect client applications into a z/OS Queue Manager, no longer exists in MQ version 8. Instead, the client capability exists by default on a version 8 Queue Manager.

If you did not previously allow Client applications to connect into a z/OS queue manager, you will need to consider using Channel Authentication (CHLAUTH) rules to prevent client connections and hence protect the Queue Manager.

Other z/OS Items

- Message suppression
  - CSQ6SYSP / SET SYSTEM property **EXCLMSG**
  - Formalizes service parm to suppress Client channel start/stop messages
  - Extended to be generalized
    - Applicable for most MSTR and CHIN messages
- DNS reverse (ip address → host name) look up
  - Queue Manager attribute **REVDNS(DISABLED/ENABLED)**
  - If DISABLED, prevents channel hangs if DNS infrastructure impacted
    - But CHLAUTH rules that use hostnames are not matched

Other z/OS Items

NOTES

Message suppression - Exclude Message (EXCLMSG) is a CSQ6SYSP (hence CSQZPARM) property which can also be specified using the SET SYSTEM command. It can be used to specify a list of message identifiers to be excluded from being written to any log. Messages in this list are not sent to the z/OS console and hardcopy log. As a result, using EXCLMSG is more efficient (from a CPU perspective) than using the message processing facility list mechanism of z/OS and should be used where possible. The default value is an empty list.

Message identifiers are supplied without the CSQ prefix, and without the action code suffix (I, D, E, or A). For example, to exclude message CSQX500I, add X500 to this list. This list can contain a maximum of 16 message identifiers.

Reverse look up of IP address - For some customers the time taken to reverse look up an IP address to a host name can cause issues. MQ has added a queue manager attribute called REVDNS which controls whether reverse lookup of the hostname from a Domain Name Server (DNS) is done for the IP address from which a channel has connected. This attribute has an effect only on channels using a transport type (TRPTYPE) of TCP and has values of:

DISABLED - DNS host names are not reverse looked-up for the IP addresses of inbound channels. With this setting any CHLAUTH rules using host names are not matched. This is the initial default value for the queue manager.

ENABLED - DNS host names are reverse looked-up for the IP addresses of inbound channels when this information is required. This setting is required for matching against CHLAUTH rules that contain host names, and to include the host name in error messages. The IP address is still included in messages that provide a connection identifier.

In earlier releases a Service parameter was provided to some customers so that channel hangs could be avoided in situations where a DNS infrastructure became non-responsive.

zEDC compression hardware exploitation - When COMPMSG(ZLIBFAST) is used on channels to compress messages, it exploits the zEDC card (provided zEC12 GA2 is in use) for compression. This can yield higher throughputs and reduced CPU cost for SSL channels.

MQ platform and product updates

- Split Cluster Transmit Queue availability in MQ for z/OS
  - Ability to separate workloads
  - Reduce load on SYSTEM.CLUSTER.TRANSMIT.QUEUE

Please attend Session 17039
Clustering Queue Manager – Making Life Easier by Automating Administration and Scaling for Performance
[z/OS and Distributed]
by Neil Johnston
on Thursday March 5th at 3:15pm in Seneca

- Advanced Message Security (AMS)
  - Now integrated into the base MQ for z/OS product
  - Offers improved performance and usability

- MFT has been updated to reduce reliance on USS

MQ platform and product updates

NOTES

As MQ for z/OS didn't release a version 7.5, MQ V8 supports:

- Split cluster transmit queues

This allow a multiple transmission queues to be defined in a cluster so different work can be transported over different channels. This for example, enables different quality of service to be offered depending on the queues used.

- Advanced Message Security (AMS) has been integrated into the product to improve performance and usability. There is a separately installable product which enables the AMS feature.

- MFT has been improved to have less reliance on USS.

Breaking news

- **JMS support in CICS TS**
  - Can use the IBM MQ classes for JMS in certain versions of the CICS® Open Services Gateway initiative (OSGi) Java™ Virtual Machine (JVM) server
  - JMS 1.1 if using MQ for z/OS V7.1
  - JMS 2.0 (requires Java 7) if using MQ for z/OS V8.0
  - CICS TS APAR:
    - V5.2: PI32151
  - MQ APARs for:
    - V7.1: JMS PI29770 (supersedes 7.1.0.6) or later CSD
    - V8: JMS 8.0.0.2 PI33038 or later CSD **+** MQ base PI28482
  - For more details, see: http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q121740_.htm

Breaking News

NOTES

MQ classes for java have long been available to CICS apps

This recent CICS announcement brings MQ JMS support to java apps running in CICS JVM server. The JMS spec supported is dependent on version of MQ used. JMS 1.1 for MQ V7.1 and JMS 2.0 (requires java 7) on MQ V8.0

In MQ for z/OS we ship APARs for the JMS feature (USS Components FMID) containing identical levels of JMS to the MQ distributed CSDs. The CICS JMS updates just missed the APAR for CSD (or fixpac) 7106, so the APAR is equivalent to 7106+CICS function. It will be rolled into subsequent CSDs. On V8, as well as the APAR equivalent to 8002 CSD (or later) for JMS, there are some additional changes for the MQ base FMID relating to enabling JMS 2.0 features for the CICS implementation.

Breaking News

NOTES

Enhanced Java SE support

Enhanced support is provided for Java SE developers who need access to WebSphere MQ. Java SE programs that run in a CICS OSGi JVM server can now use the WebSphere MQ classes for the Java Message Service (JMS), as an alternative to the proprietary WebSphere MQ classes for Java.

Developers familiar with the JMS API can easily access WebSphere MQ resources. The CICS MQ attachment facility is enhanced to support the necessary new commands.

Support is limited to Java programs that run in an OSGi JVM server. There is no support in a Liberty profile JVM server.

Support from WebSphere MQ that use the WebSphere MQ classes for JMS is provided in WebSphere MQ for z/OS V7.1 and V8:

- V7.1 requires MQ APAR PI29770 (built on fix pack 7.1.0.6) or any later fix pack level.
- V8.0 requires base APAR PI28482 and fix pack 8.0.0.2 or any later fix pack level.
- CICS TS V5.2 is also supported and requires APAR PI32151.

Breaking news

- **Placeholder for z13 Performance**
  - Apologies but this data is not yet available
  - Look out for an update to one of the MQ Performance reports

New Redbook covers MQ V8

New Redbook covers MQ V8

NOTES

This redbook covers the new features in version 8. The book is still being edited but is due to be available soon.

Questions ?

This was session #17035. The rest of the week .....

	Monday	Tuesday	Wednesday	Thursday	Friday
08:30			17060: Understanding MQ Deployment Choices and Use Cases	17051: Application Programming with MQ Verbs [z/OS & Distributed]	16544: Why Shouldn't I Be Able To Open This Queue? MQ and CICS Security Topics Room: Willow B
10:00	17036: Introduction to MQ - Can MQ Really Make My Life Easier? [z/OS & Distributed]		17052: MQ Beyond the Basics - Advanced API and Internals Overview [z/OS & Distributed] 17035: MQ for z/OS, Using and Abusing New Hardware and the New V8 Features [z/OS] Room: Willow B	17054: Nobody Uses Files Any More do They? New Technologies for Old Technology, File Processing in MQ MFT and IIB [z/OS & Distributed]	17057: Not Just Migrating, but Picking up New Enhancements as You Go - We've Given You the Shotgun, You Know Where Your Feet Are [z/OS & Distributed]
11:15	17041: First Steps with IBM Integration Bus: Application Integration in the New World [z/OS & Distributed]		16732: MQ V8 Hands- on Labs! MQ V8 with CICS and COBOL! MQ SMF Labs! Room: Redwood	17046: Paging Dr. MQ - Health Check Your Queue Managers to Ensure They Won't Be Calling in Sick! [z/OS]	17053: MQ & DB2 – MQ Verbs in DB2 & InfoSphere Data Replication (Q Replication) Performance [z/OS]
01:45	17037: All About the New MQ V8 [z/OS & Distributed]	17034: MQ Security: New V8 Features Deep Dive [z/OS & Distributed]	17040: Using IBM WebSphere Application Server and IBM MQ Together [z/OS & Distributed]	17062: End to End Security of My Queue Manager on z/OS [z/OS]	All sessions in Seneca unless otherwise noted.
03:15	17042: What's New in IBM Integration Bus [z/OS & Distributed]	17065: Under the hood of IBM Integration Bus on z/OS - WLM, SMF, AT-TLS, and more [z/OS]	17043: The Do's and Don'ts of IBM Integration Bus Performance [z/OS & Distributed]	17039: Clustering Queue Managers - Making Life Easier by Automating Administration and Scaling for Performance [z/OS & Distributed]	
04:30	17059: IBM MQ: Are z/OS & Distributed Platforms like Oil & Water? [z/OS & Distributed]	17055: What's the Cloud Going to Do to My MQ Network?	17044: But Wait, There's More MQ SMF Data Now?!?! - Monitoring your Channels Using V8's New Chinit SMF Data [z/OS]	17068: Monitoring and Auditing MQ [z/OS & Distributed]	

Remember to submit your evaluation please

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

Copyright and Trademarks