IBM MQ for z/OS – Taking Advantage of the Hardware (and software!)

Lyn Elkins – elkinsc@us.ibm.com

Executive IT Specialist

IBM

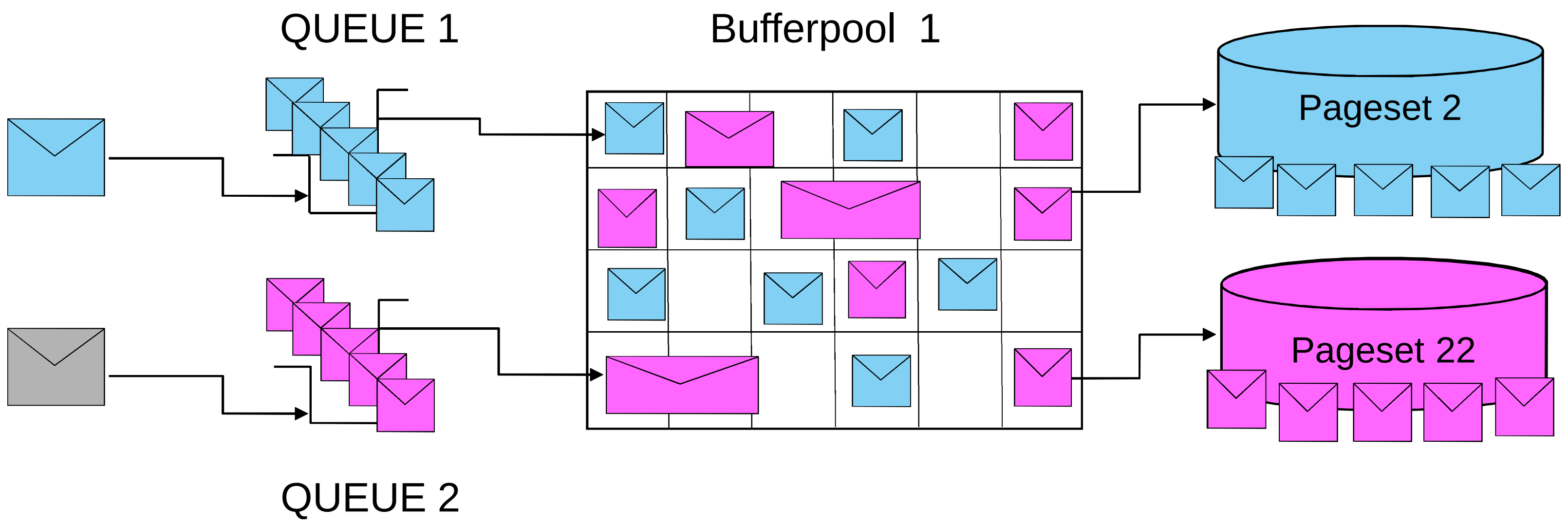
Agenda

- Above the Bar Bufferpools
- Message Suppression
- Logging Changes
- Z Hardware and Software Exploitation

Above the Bar Bufferpools

"This one enhancement will resolve all my MQ for z/OS performance problems forever."
Naive unnamed customer and IBMer

Bufferpools – how are the messages are mapped?



Message mapping to storage

N O T E S

Internally MQ keeps a space map, that contains pointers to every message and where it is physically located.

Remember, this does not change no matter where the buffer pools are located.

This example shows pre-V8 common use, where having multiple queues mapped to one bufferpool but to individual pagesets is an efficient use of resources.

When an MQPUT is issues, the message is initially put to the bufferpool

If there is no room on the BP, existing messages are moved from the bufferpool to the pageset until space is available

When an MQGET is issued, the space map tells the queue manager where the message resides

If it is on a pageset, the message is read back into the bufferpool . Again, if there is no room on the BP, existing messages are moved from the bufferpool to the pageset until space is available.

MQ for z/OS: 64-bit bufferpools

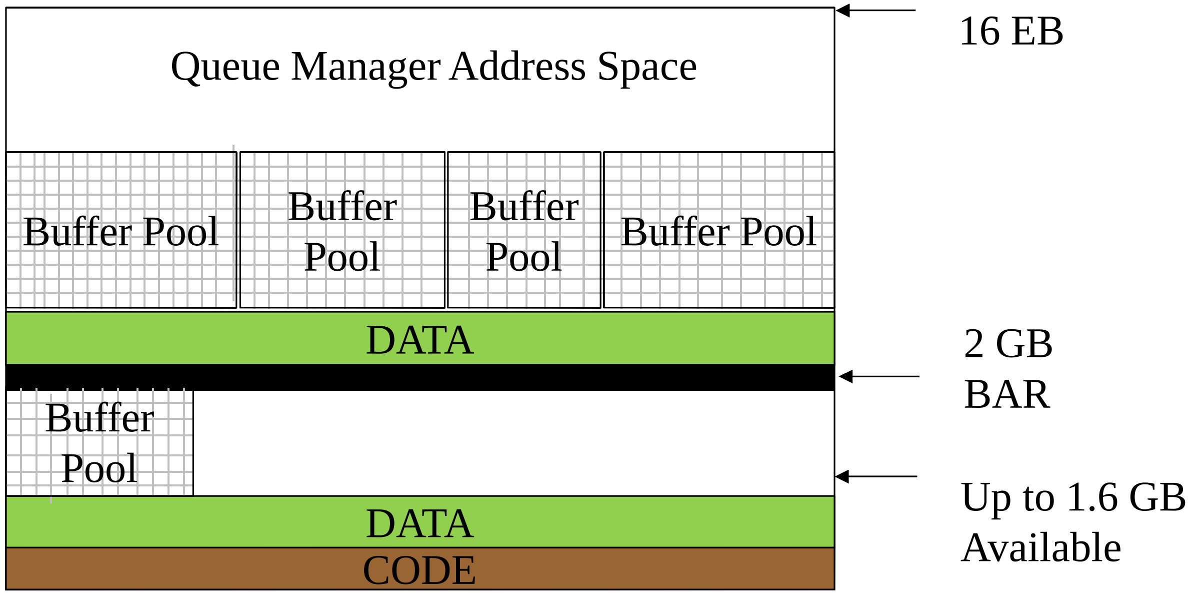
- **64-bit buffer pools in MQ for z/OS**
  - Allows large numbers of messages to be cached before writing to pagesets
  - Allows MQ to exploit the vast amount of storage on today's machines
- **Improves performance of putting/getting messages by minimizing disk I/O**
  - What's the BEST I/O?
- **Minimizes administrative overhead of managing buffer pools**
  - You can now have a one to one relationship between pagesets and bufferpools
- **Buffer pool LOCATION attribute says where it is located relative to the bar**
  - BELOW: The default. Buffer pool is located below the bar in 31 bit storage
  - ABOVE: Buffer pool is located above the bar in 64 bit storage
  - This can be altered dynamically
- **Storage can be pagefixed based on pageclass attribute**

Buffer pool problems before V8

- **There is not much space below the bar for buffer pools once queue manager code and data is taken into account. Maximum 1.6GB available for buffer pools depending on common area**
- **Putting/getting messages into the buffer pool works at 'memory' speed, putting/getting messages from the page set works at 'disk' speed**
- **For scenarios where several applications read and/or write large number of messages to the same buffer pool a lot of time is spent getting pages from page set into buffer pool and getting pages from buffer pool into page set**
  - This is detrimental for performance
- **A maximum of 16 buffer pools are supported while up to 100 page sets are supported**
- **This can result in a lot of time spent tuning buffer pool sizes and the relationship between queues, buffer pools and page sets**
- **These problems are resolved by allowing bufferpools to be defined in 64-bit space, above the bar**

MQ for z/OS: 64-bit bufferpools

- **Buffer pools above the bar can (theoretically) use up to 16 EB storage**
- **Increased maximum size of pool to 999,999,999 buffers**
  - Was 500,000
- **Allows up to 100 buffer pools**
  - Was 16
- **The storage available above the bar**
- **Set by MEMLIMIT or systems settings**

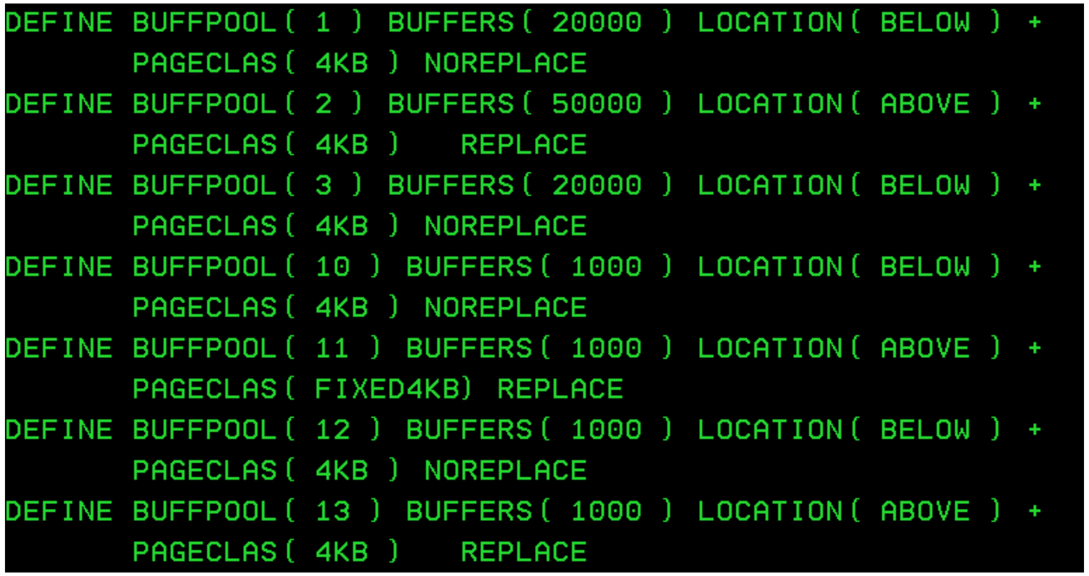


MQ for z/OS: 64-bit bufferpools

- **64-bit bufferpools can now theoretically use a huge amount of storage**
- **Having the same number of pools and pagesets makes it much simpler to monitor and control the usage of these resources**
- **SMF 115, subtype 215**
  - Up to 100 buffer pools are now supported. However there is not enough space for 100 buffer manager statistics records (QPST) in the existing SMF 115, subtype 2, record
  - If OPMODE(NEWFUNC,800) is specified in system parameters then an SMF 115, subtype 215, record will be cut for buffer manager statistics. The SMF 115, subtype 2, record will still contain the self-defining section for buffer manager but it will be all zeros
  - If OPMODE(COMPAT,800) is specified in system parameters then buffer manager statistics will continue to be in SMF 115, subtype 2.

Definition Changes

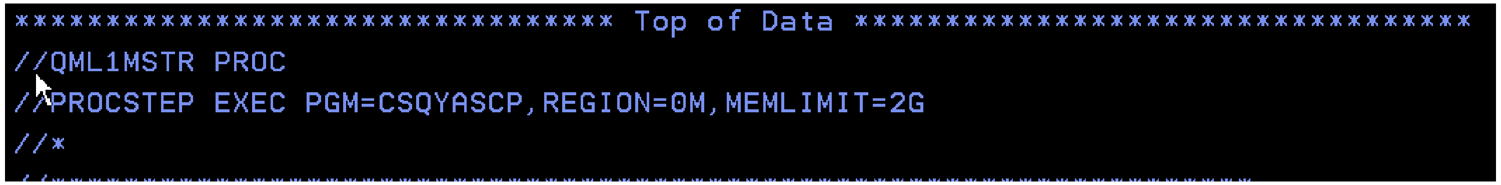
- **To implement expanded bufferpools**
  - New attributes added to the BUFFPOOL commands
    - LOCATION
      - BELOW – default, the pool is taken from the 2G address space
      - ABOVE – the pool is allocated above the 2G bar
    - PAGECLAS
      - 4KB – each 4K page is pageable by the operating system
        - This is the only option for pools defined in BELOW the bar storage
      - FIXED4KB – each 4K page is fixed in memory
    - REPLACE/NOREPLACE
      - Should this definition override the what is held in the log of the queue manager?



```
DEFINE BUFFPOOL( 1 ) BUFFERS( 20000 ) LOCATION( BELOW ) +
       PAGECLAS( 4KB ) NOREPLACE
DEFINE BUFFPOOL( 2 ) BUFFERS( 50000 ) LOCATION( ABOVE ) +
       PAGECLAS( 4KB )   REPLACE
DEFINE BUFFPOOL( 3 ) BUFFERS( 20000 ) LOCATION( BELOW ) +
       PAGECLAS( 4KB ) NOREPLACE
DEFINE BUFFPOOL( 10 ) BUFFERS( 1000 ) LOCATION( BELOW ) +
       PAGECLAS( 4KB ) NOREPLACE
DEFINE BUFFPOOL( 11 ) BUFFERS( 1000 ) LOCATION( ABOVE ) +
       PAGECLAS( FIXED4KB) REPLACE
DEFINE BUFFPOOL( 12 ) BUFFERS( 1000 ) LOCATION( BELOW ) +
       PAGECLAS( 4KB ) NOREPLACE
DEFINE BUFFPOOL( 13 ) BUFFERS( 1000 ) LOCATION( ABOVE ) +
       PAGECLAS( 4KB )   REPLACE
```

MEMLIMIT – setting on queue manager

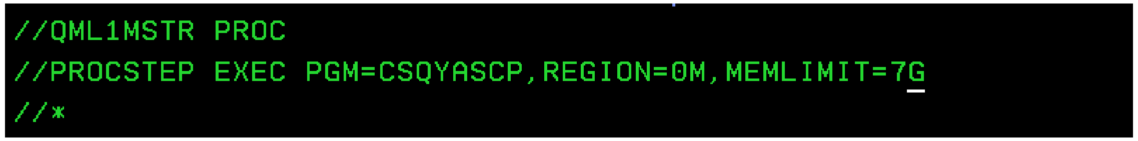
- **V7.1 MEMLIMIT**



```
********************************* Top of Data **********************************
//QML1MSTR PROC
//PROCSTEP EXEC PGM=CSQYASCP,REGION=0M,MEMLIMIT=2G
//*
```

  - Allocates 64-bit storage for MQ’s use
    - Indexes
    - Security cache
    - SMDS buffers

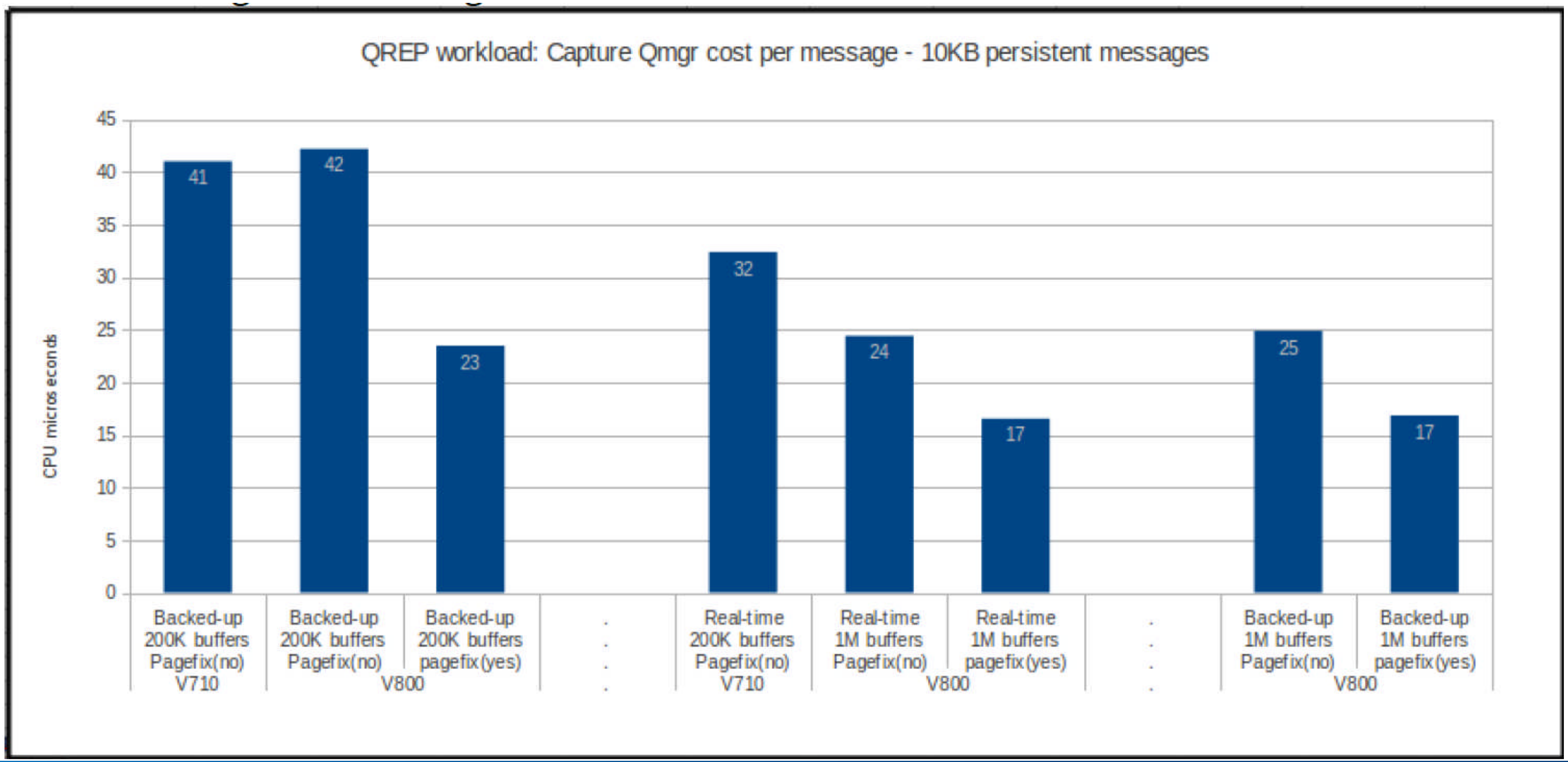
- **V8.0 MEMLIMIT**



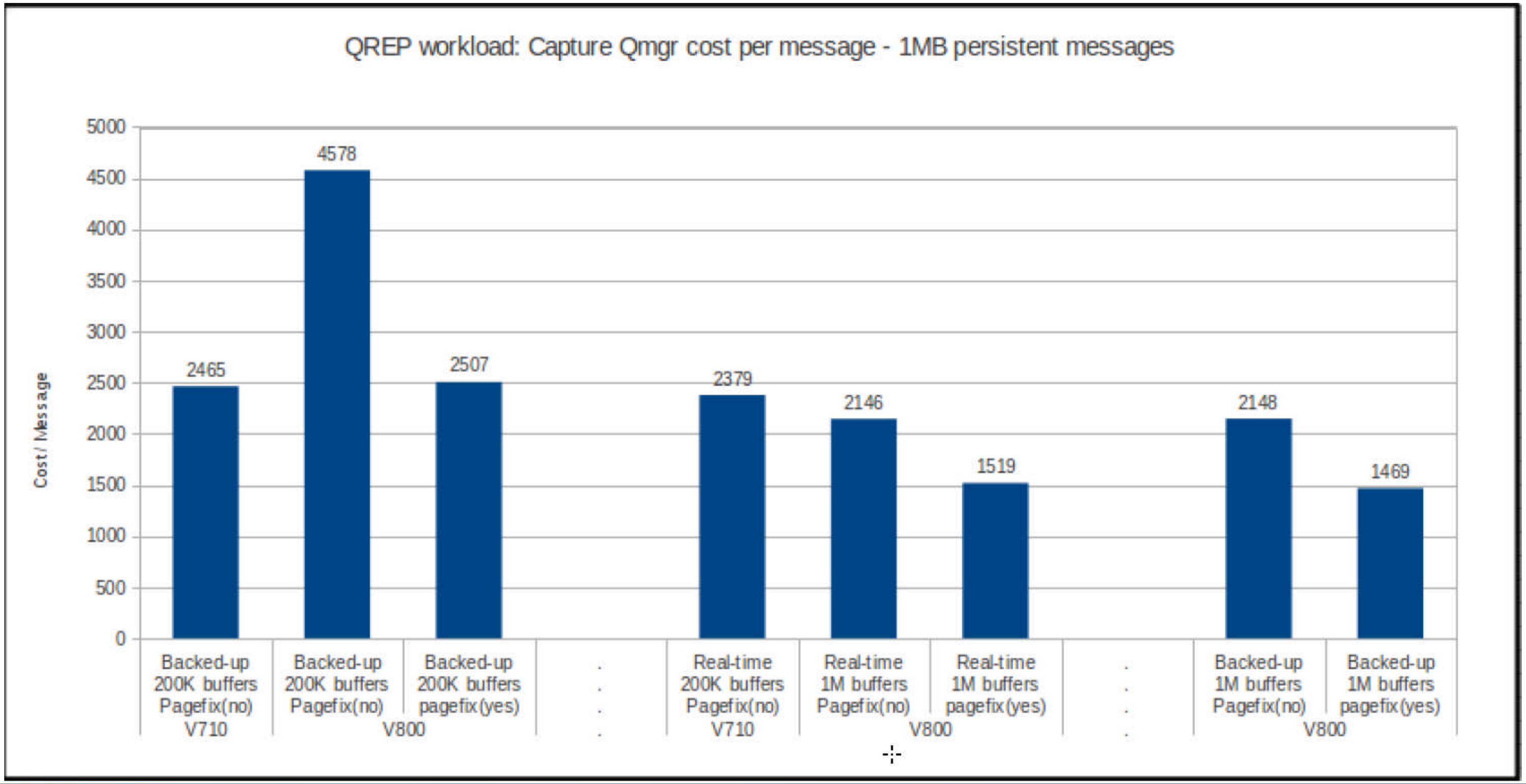
```
//QML1MSTR PROC
//PROCSTEP EXEC PGM=CSQYASCP,REGION=0M,MEMLIMIT=7G
//*
```

  - Same storage areas and the above the bar bufferpools (sized for example in MQ V8 Redbook)

QREP and similar workload performance



QREP and similar workload performance



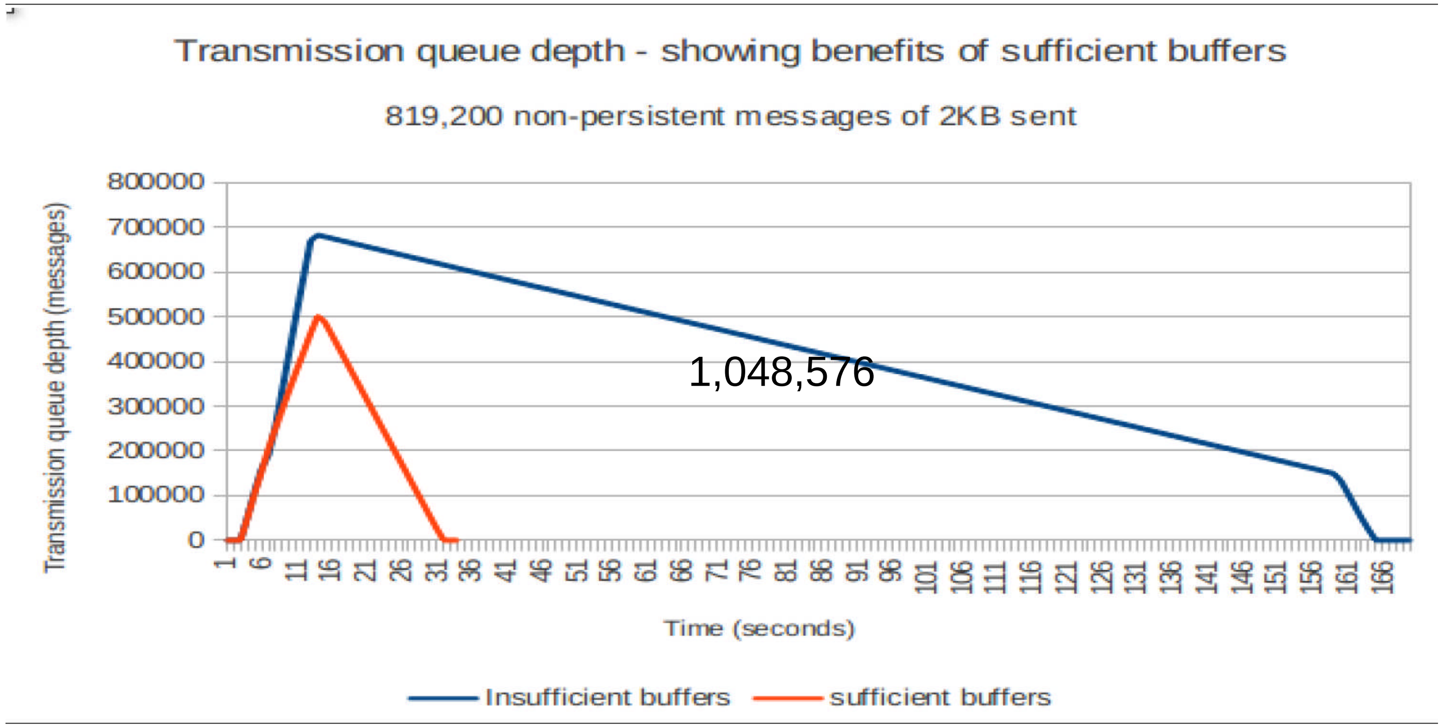
QREP and similar workload performance – notes

- **The throughput rate for version 8.0 has not been significantly affected by page-fixing the buffers, however the tests are configured to have sufficient CPU at all times.**
- **Only the capture queue manager costs varied by any significant value.**
- **Moving from version 7.1 to version 8.0 with PAGECLAS(4KB) saw an increase in capture queue manager cost, particularly for 1MB messages.**
  - Specifying PAGECLAS(FIXED4KB) brought costs back in-line with version 7.1 for large messages (reducing version 8.0 costs by ~36%) and significantly reduced the 10KB message costs (~48%).
  - Comparing the first and second sets of tests show that increasing the size of the buffer pool brings the costs below those observed when using 31-bit buffer pools.
- **•When larger buffer pools were used (version 8.0 only), PAGECLAS(FIXED4KB) reduced the cost (~30%) for both small and large message workloads.**

The benefit of sufficient buffers

- **While QREP was one of the focus areas when testing above the bar bufferpools, any application that overfills a bufferpool will benefit.**
- **Another example illustrated in the upcoming MP1J, MQ for z/OS performance, is the effect of having sufficient buffers on a normal transmission queue.**
- **In this example a transmission queue has filled due to an unavailable receiver queue. Please note the following:**
  - The below the bar bufferpool had 200,000 pages – which is quite large for a buffer pools
  - The above the bar bufferpools had a **1,048,576 page (**4GB) buffer pool for the transmission queue.
  - Each test put the same number of messages, batching them at 200 messages per commit.
  - The large bufferpool peaked at 500,000 messages and did not have to write any to the pageset during processing.

Comparing time to drain a transmission queue



Comparing time - Notes

- **With the 1,048,576 page bufferpool the 819,200 2Kb messages were transmitted in about 32 seconds**
- **The same volume took almost 165 seconds when there were only 200,000 pages available**
  - **Writing buffer pages to disk, due to the bufferpool filling is the difference.**

on the BP, existing messages are moved from the bufferpool to the pageset until space is available

Above the Bar Bufferpools – Not always the ultimate solution!

- **Having more bufferpools and pages may NOT be the ultimate performance solution**
  - If your performance problems are log I/O related this will not help you
  - If your problems are due to CPU shortages, it might not help you
  - And, finally allocating more, and larger, bufferpools may just expose other issues
- **Furthermore, getting the most out of above the bar bufferpools may require the purchase of additional memory**

Message Suppression - Reducing CPU

- **Or why using clients can hurt me**

- **This includes separate messages for “real” and client channels**
  - CSQX511I and CSQX512I for client connection to a SVRCONN
  - CSQX500I and CSQX501I report other channel start/stop events

- **Message suppression**
  - Messages that are suppressed are not written to the JES log, so can reduce CPI
  - Implemented via a new ZPARM, EXCLMSG
  - Can be altered via a SET SYSTEM command
  - Up to 16 messages can be suppressed
  - Significant performance improvements have been observed noted when the SVRCONN messages are suppressed
    - Up to a 20% CPU reduction seen for poorly behaved clients

Message Suppression

- **We start with clients connecting to z/OS queue managers**
  - With MQ V8, attaching clients to z/OS queue managers became free (no CAF license).
    - However, the CPU costs (and therefore the MLC) can increase dramatically for the CHIN if clients are not well behaved.
  - A poorly behaved client is defined as one following this general pattern:
    - Connect
    - Open
    - Put or get a single message
    - Close
    - Disconnect
    - Repeat until CPU is exhausted
  - Client applications that issue more MQ API requests see less CPU recovery from the message suppression

- **Care should be taken when selecting messages to suppress.**

Reducing Outages - 8 byte log RBA

- **WebSphere MQ for z/OS V7.1 (or prior):**
  - Implements a 6 byte Log RBA (Relative Byte Address)
  - This give an RBA range of 0 to x'FFFFFFFFFFFF' (= 255TB)
  - Some customers reach this limit in
  - At 100MB/sec, log would be full in 1 month

- **If we reach the end of the Log RBA range (i.e. “wrap”):**

~~12 to 18 months~~ 3 to 6 months

  - Queue Manager terminates and requires a cold start – a disruptive outage !
  - Potential for loss of persistent data

- **To avoid an outage:**
  - Run CSQUTIL RESETPAGE, at regular planned intervals, to RESET the LOG RBA

8 byte log RBA: The Problem

- **For versions of WMQ 7.1 or earlier, a 6 byte log RBA (Relative Byte Address) was used. This gave a maximum log range of 0 to x'FFFFFFFFFFFF' which is 255TB. Some customer were reaching this limit within about 12 to 18 months. When the end of the log is reached, a cold start of the queue manager is required, causing a disruptive outage and potential loss of persistent data. To avoid the disruptive outage, it is necessary to have a planned outage to run the CSQUTIL RESETPAGE function to RESET the LOG RBA.**
- **With APAR PM48299, WMQ V7.0.1 (and above) Queue Managers will issue more message to warn of the log being exhausted.**
- **CSQI045I/CSQI046E/CSQI047E to warn if RBA is high (>= x'700000000000')**
  - CSQI045I, when RBA is x'700000000000', x'7100..', x'7200..' and x'7300..'
  - CSQI046E when RBA is x'740000000000', x'7500..', x'7600..' and x'7700..'
  - CSQI047E when RBA is x'780000000000', x'7900..', x'nn00..' and x'FF00..'
- **CSQJ031D on restart to confirm Queue Manager start-up even though log RBA has passed x'FF8000000000'**
- **Terminates, to prevent loss of data, if log RBA reaches x'FFF800000000'. The LOG RBA will need to be RESET to allow the Queue Manager to continue to operate**

8 byte log RBA: The Solution

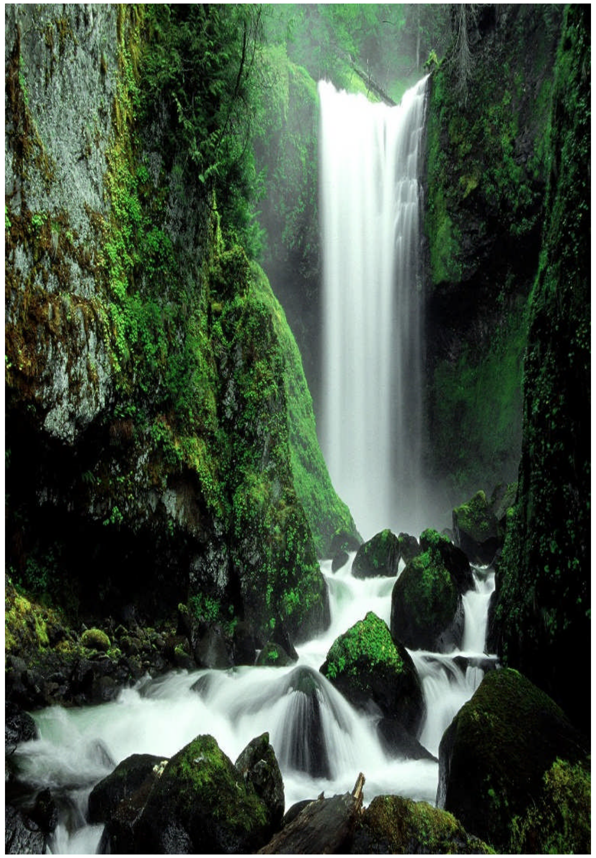
- **Upper limit on logical log will be 64K times bigger**
  - At 100MB/sec this will now take about 5578 years to fill!
- **BSDS and log records changed to handle 8 byte RBAs and URIDs**
  - Utilities or applications that read the following are impacted:
    - The BSDS
    - The Logs
    - Console messages that contain the log RBA or URID
- **Queue Manager will use 6 byte RBA until 8 byte RBA is enabled**
  - BSDS conversion utility (same model as used by DB2) to migrate to 8 byte RBA

8 byte log RBA: The Solution

- Using an 8 byte RBA will increase the maximum log RBA by 65536 times, compared to using a 6 byte RBA. This means that it would now take about 5578 years to use the full range is writing constantly at 100MB/sec.
- The BSDS and log records have been changed to handle 8 byte RBAs and URIDs, so any application or utility that reads these will need to be updated to handle them. Console messages and MQ utilities have also been updated to output 8 byte RBAs, so if these are processed by other applications they may need to be modified to handle the extra data.
- The queue manager will continue to use 6 byte RBAs until 8 byte RBAs are enabled by using the BSDS conversion utility to migrate. This is the same model as used by DB2.

8 byte log RBA: Migration

- **Get MQ V8 running in NEWFUNC mode**
  - You CANNOT fall back to V7
- **In QSG, ALL queue managers MUST BE in V8 NEWFUNC**
- **STOP queue manager**
- **Run the new CSQJUCNV utility**
  - Utility will check that entire QSG is at correct level
  - Copies data from old primary / secondary BSDS pair to new pair
  - Old data is NOT changed for fallback
- **Restart queue manager with new 8 byte format BSDS**
  - Queue manager will ONLY start if in NEWFUNC mode
  - New message CSQJ034I issued indicating 8 byte RBA mode
  - All subsequent log data in new format



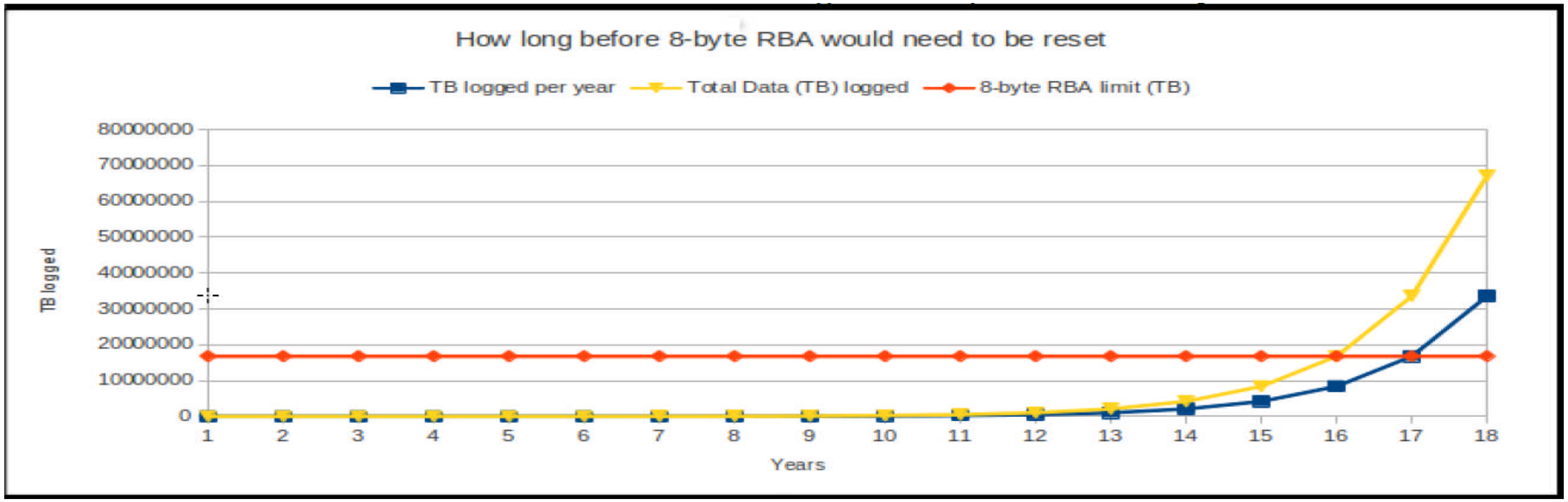
8 byte log RBA: Migration

- To be able to migrate to using 8 byte RBA, you first need to have your queue manager running in V8 NEWFUNC mode. If in a QSG then all of the other queue managers in the QSG also need to be in NEWFUNC mode. This is because until that point you would be able to migrate the queue manager back to a previous release, and only V8 understands how to handle 8 byte RBA records.

- To convert a queue manager to 8 byte RBAs you need to run the CSQJUCNV utility. The utility will check that the entire QSG is at the correct level, if successful then it will copy the data from the primary and secondary BSDS pair into a new set enabled for 8 byte RBAs. The original BSDS pair are un-modified in-case fallback is necessary. Once the utility has been run, the queue manager can be restarted with the new 8 byte format BSDS. The queue manager will ONLY start if in NEWFUNC mode. A new message, CSQJ034I, will indicate that the queue manager is running in 8 byte RBA mode.

Logging Changes

- **Log RBA constraint relief**
  - Already improved messages to warn of approaching RBA
  - Now widening RBA field from 6 to 8 bytes
  - At 100MB/sec this will now take about 5578 years to fill



Warning message thresholds in V8

- **The message numbers 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**

Log capacity improvement

- **Log access changed to BSAM, previously used BDAM**
  - Archive logs can now hold 4G of data when written to DASD, active logs can be defined and will use the entire 4G
  - Larger archive logs may increase CPU cost in the QMGR
    - Recommendations:
      - Test in your environment!
      - The reduced number of log switches may offset the additional cost

Increased Active Log Size - notes

- **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 a max of 4GB in size

More Notes

Note recommendations from KnowledgeCenter:

> The maximum log size is 4 GB. You need to take care when defining a log of 4 GB, as the system might round up the number of records specified to a value that results in a log being greater than 4 GB.
>
> When the queue manager reads such a log, the log is seen as being much smaller than it actually is, giving undesirable results; for example, very frequent log switches.
>
> When defining a large log, you should check the HI-A-RBA value of the log using IDCAMS LISTCAT, to ensure that the log is strictly less than 4 GB (4 294 967 296 bytes).

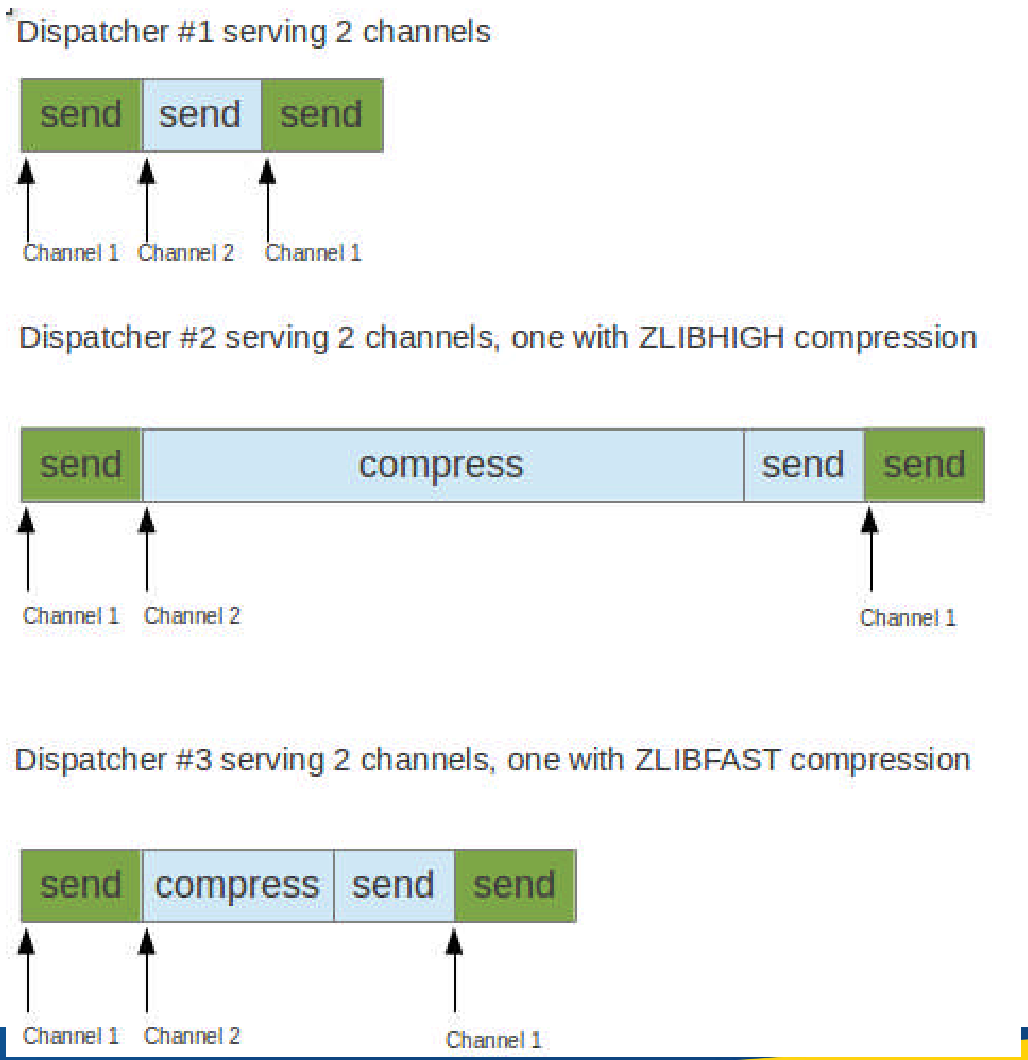
MQ for z/OS: Taking Advantage of zEDC

- **Exploitation of zEDC compression accelerator**
  - This is a combination of hardware and software, and is a separately chargeable feature
  - SMF
    - We are awaiting performance numbers on this, but initial results looks promising!
    - Lower costs for the prolific MQ SMF 116 class 3 and 4 data.
  - Channel compression (zlibfast option) can be useful when using SSL
    - In benchmark tests zEDC hardware compression can reduce the message costs by 80% vs software compression
    - Note that this can impact the dispatcher tasks, please see MP1J for additional information

Channel Compression using zEDC hardware

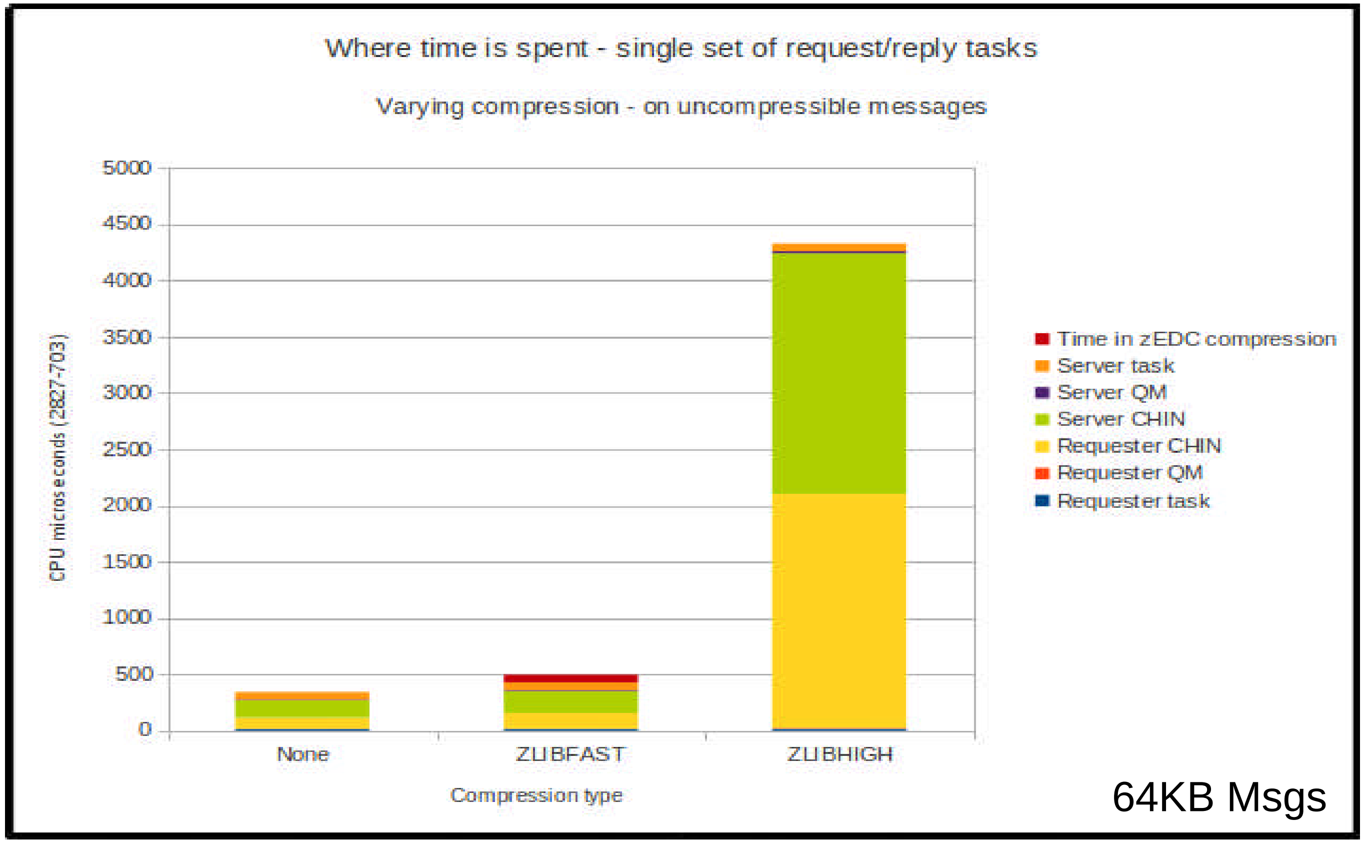
- **Channel Compression**
  - Typically used on **high latency/low-bandwidth** networks
  - Reduces amount of data being flowed
  - Reduces CPU cost per message
    - Can yield higher reduction in CPU costs for SSL channels (dispatcher compresses before encrypting)
  - Channel attributes:
    - **COMPHDRS(ZLIBFAST)**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

Channel Compression - Impact on Dispatcher Task



Dispatcher #1 serving 2 channels
send
send
send
Channel 1
Channel 2
Channel 1
Dispatcher #2 serving 2 channels, one with ZLIBHIGH compression
send
compress
send
send
Channel 1
Channel 2
Channel 1
Dispatcher #3 serving 2 channels, one with ZLIBFAST compression
send
compress
send
send
Channel 1
Channel 2
Channel 1

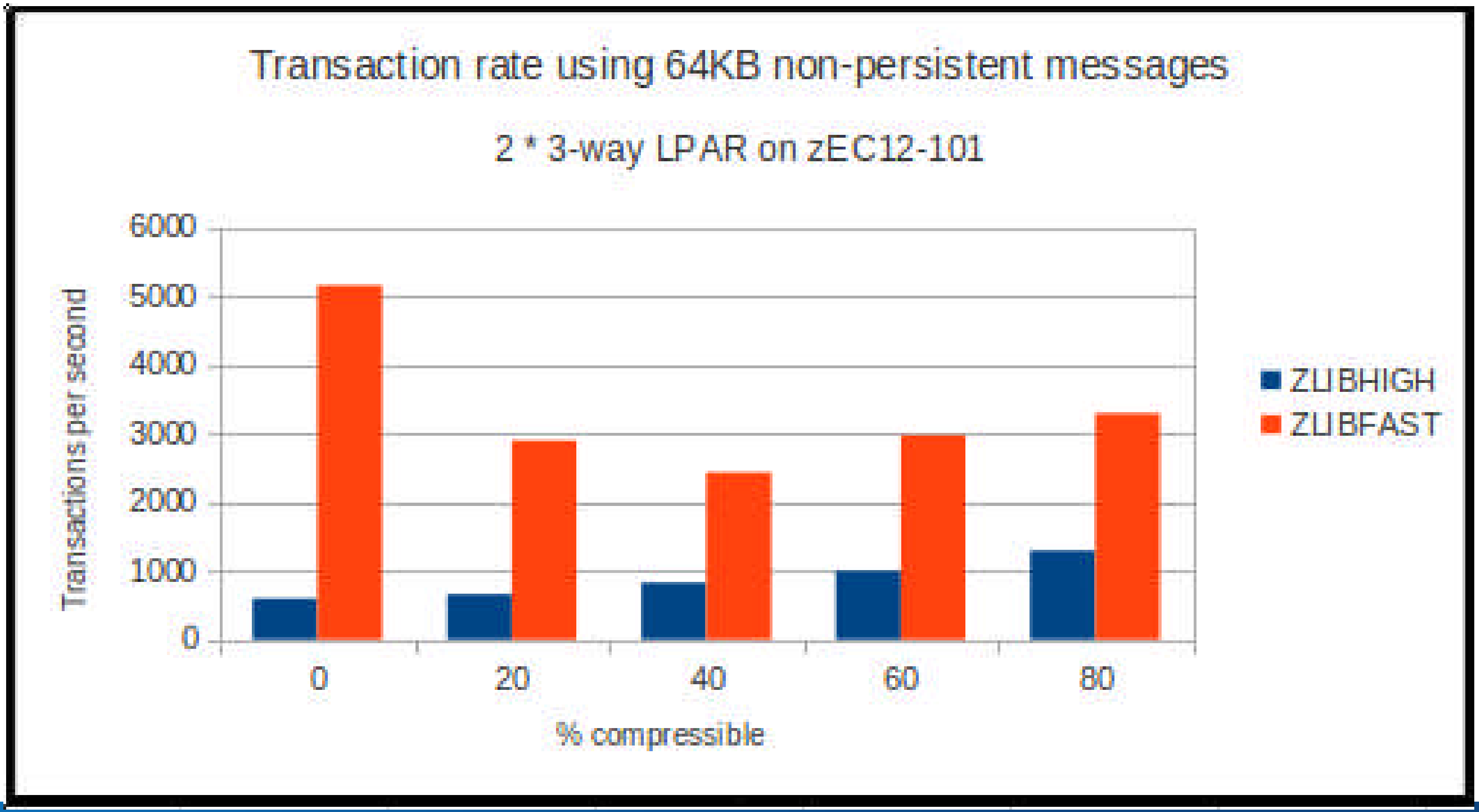
Channel Compression – Uncompressible messages



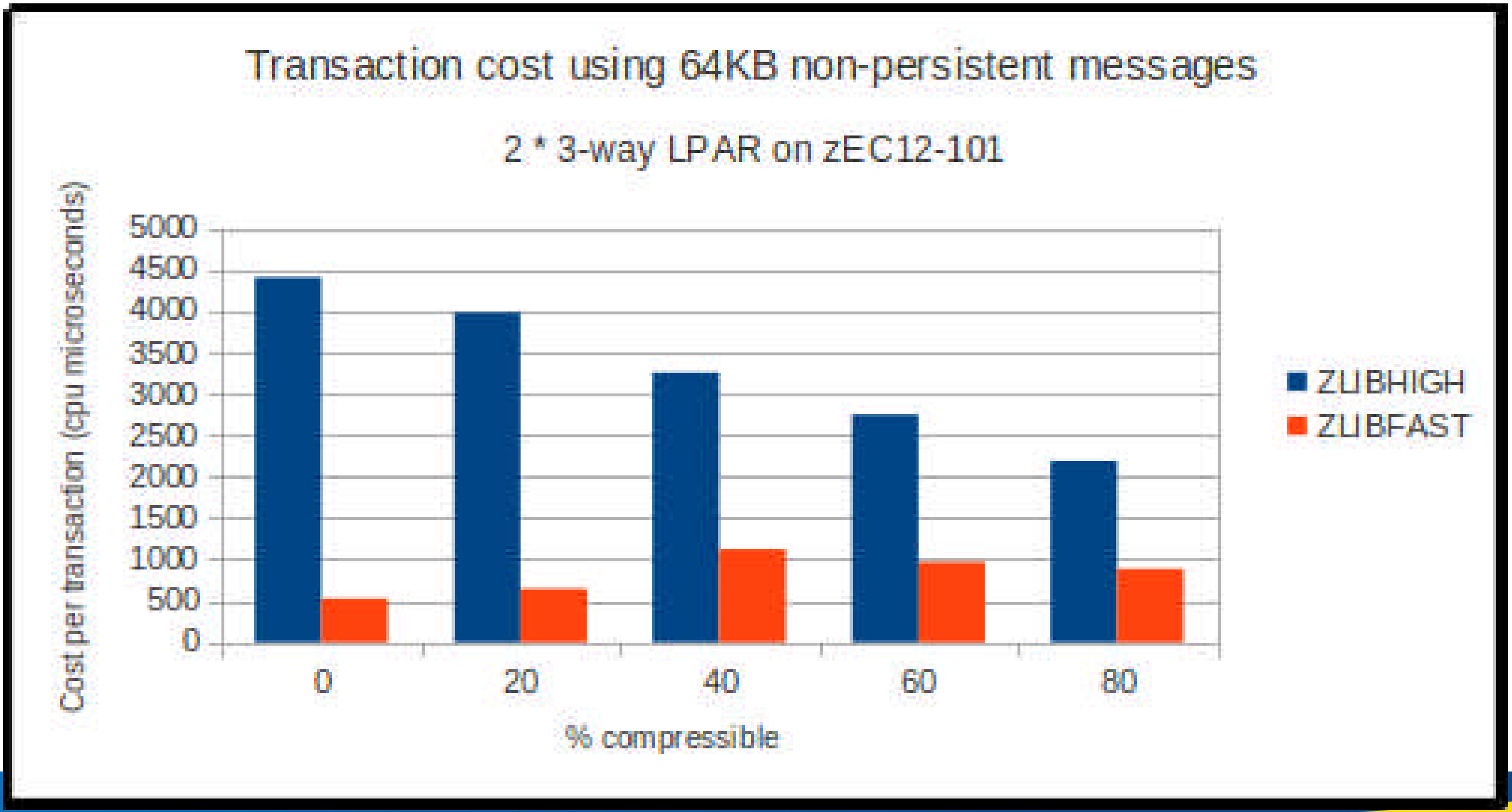
64KB Msgs

ZLIBFAST Costs Roughly 50% more Than NONE !!

Channel Compression – Compressible messages



Channel Compression – Compressible messages



MQ for z/OS: Taking Advantage of SCM

- **Support for Storage Class Memory (SCM) (Flash)**
- **Improves resiliency of Coupling Facility with cost-effective standby capacity to handle overflow of shared queues**
  - Messages <63K fully held in Flash
  - Messages >63K have pointers in Flash, body in SMDS just as for traditional CF structures
  - More information to come on this!

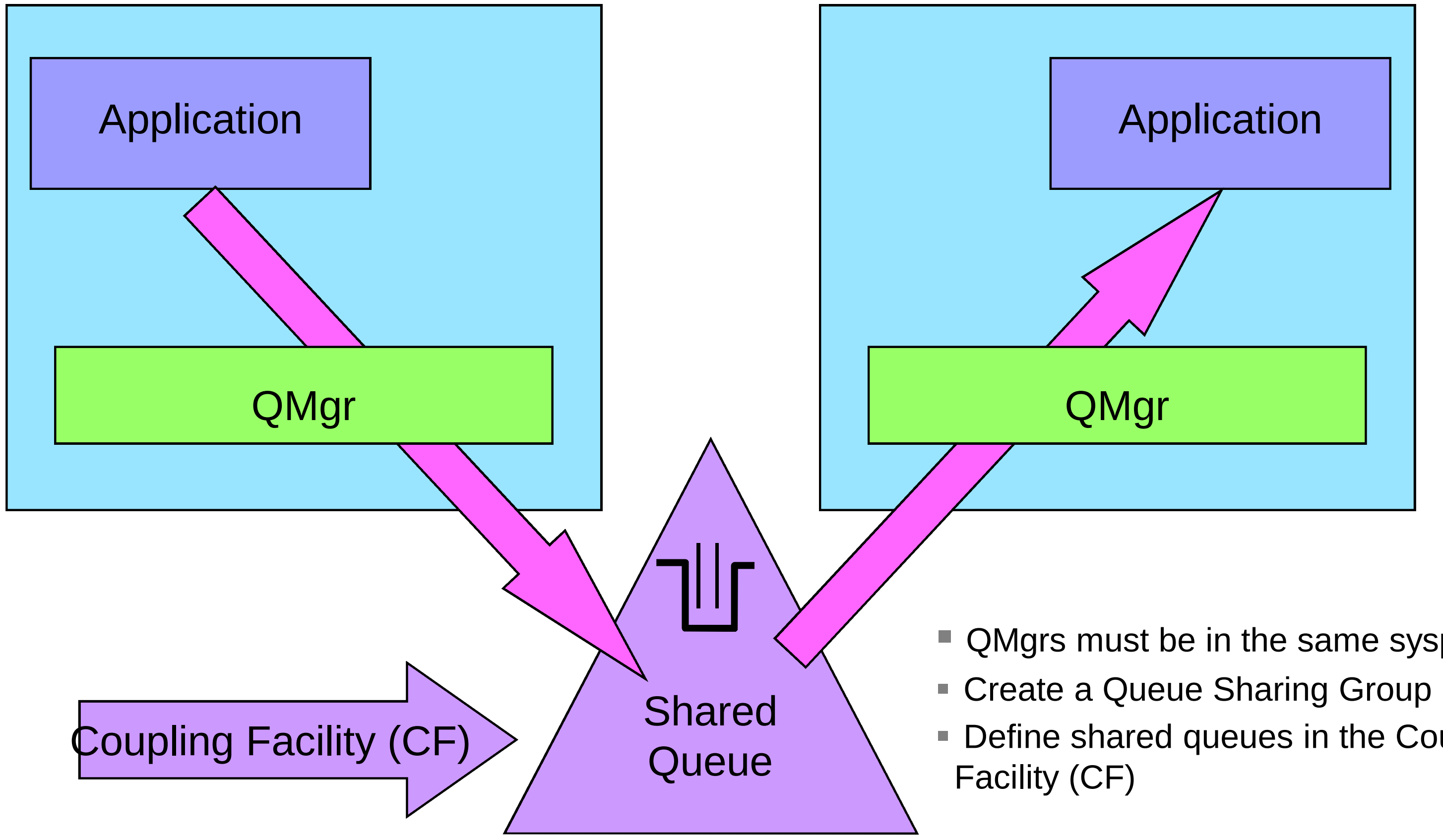
Storage Class Memory (SCM) (Flash)

- 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

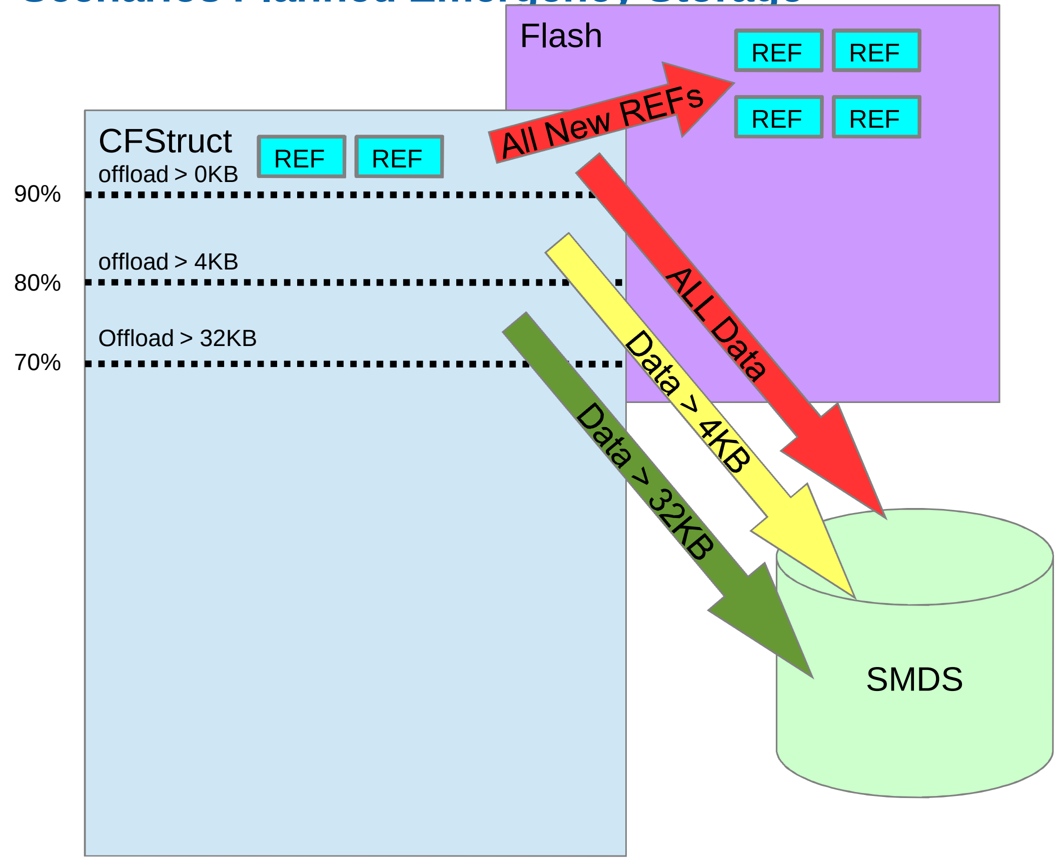


- QMgrs must be in the same sysplex
- Create a Queue Sharing Group (QSG)
- Define shared queues in the Coupling Facility (CF)

Shared Queues

- **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



Note: Assume all msgs < 63KB

CFSTRUCT OFFLOAD rules cause progressively smaller messages to be written to SMDS as the structure starts to fill. Leaving only the control information in the CF structure.

Once 90% threshold is reached, the queue manager stores the minimum data per message (control information) to squeeze as many messages as possible onto the shared queue.

Once at 90% threshold, CF Flash pre-staging algorithm also starts to move message control information 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.

CF Flash: Scenarios Planned Emergency Storage

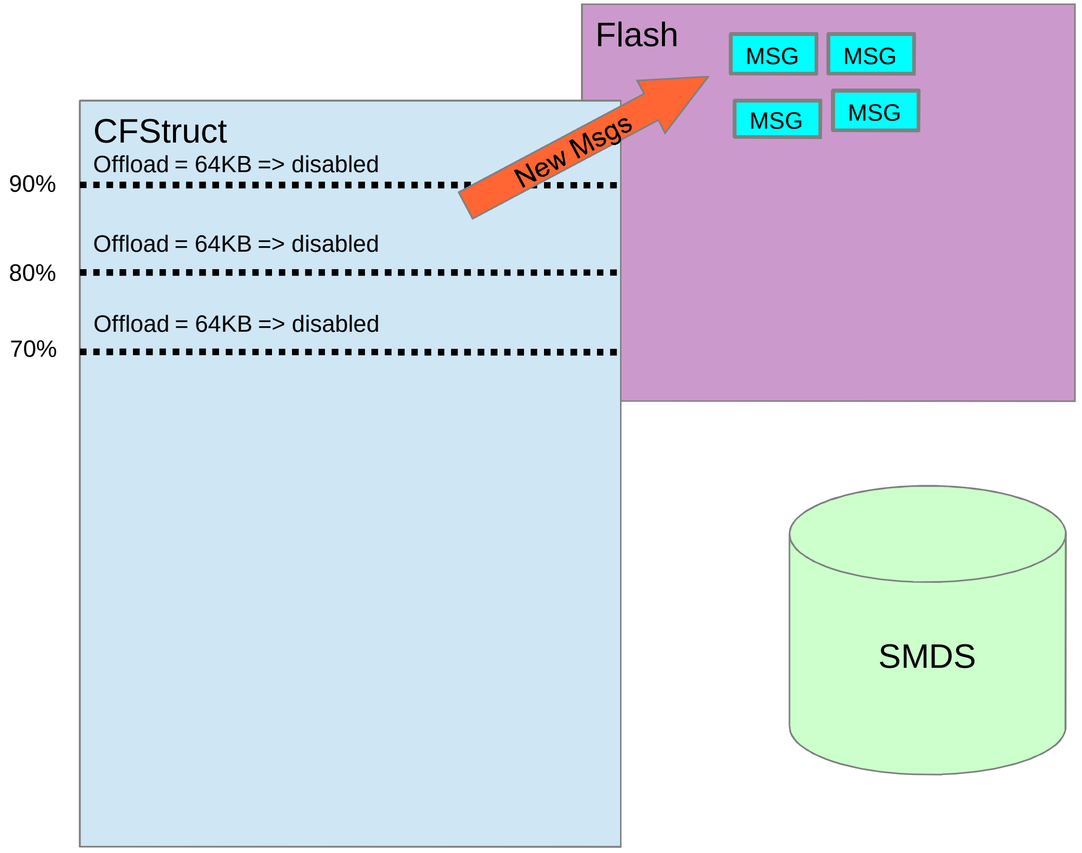
- This first slide shows a planned emergency storage scenario.

- 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


Flash
MSG
MSG
MSG
MSG
CFStruct
Offload = 64KB => disabled
New Msgs
90%
Offload = 64KB => disabled
80%
Offload = 64KB => disabled
70%
SMDS

Note: Assume all msgs < 63KB

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.

CF Flash: Scenarios Maximum Speed

- 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

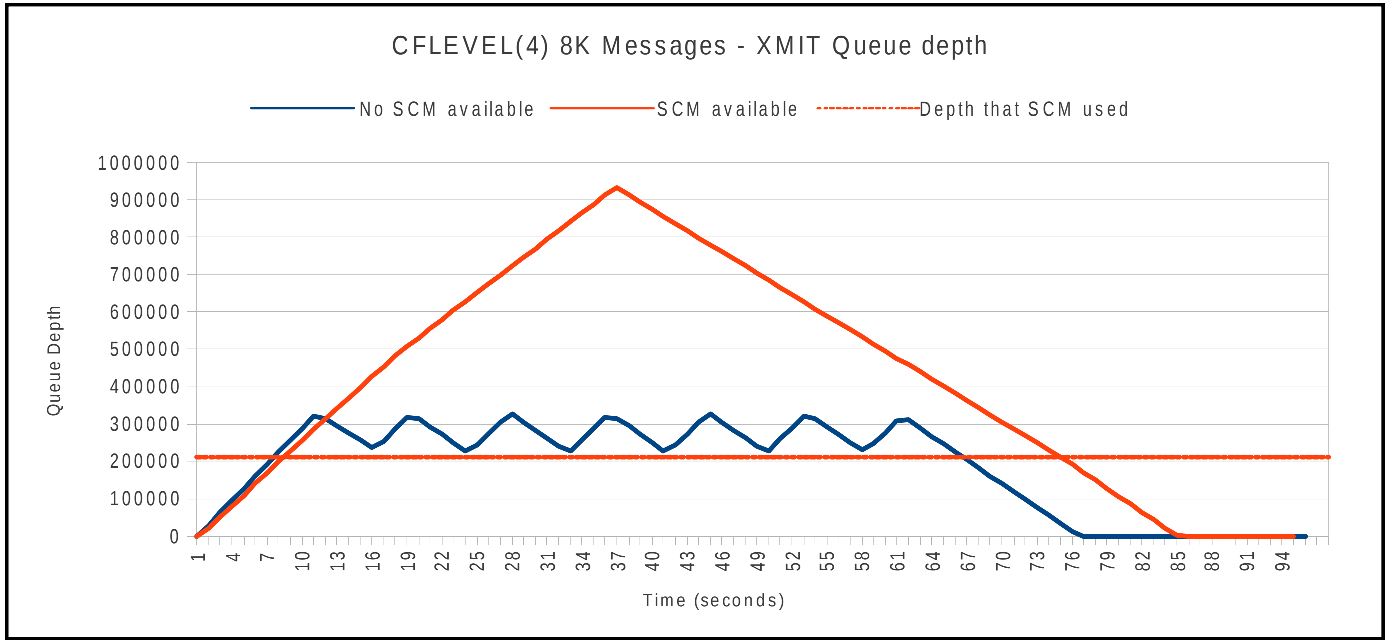
CF Structure size = 4GB

CF Flash: Storage

- 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

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.

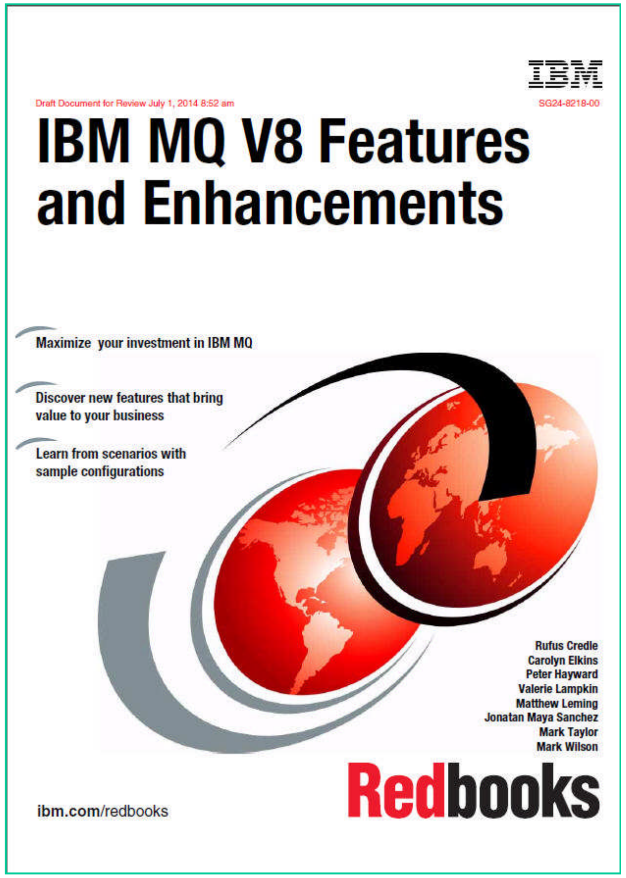
The test case has a message generator task putting 8kB messages on to a transmission queue. A channel is getting these in FIFO order. The message generator pauses for a few seconds when it hits storage media full leading to the saw-tooth shape of the blue line where no Flash memory is available.

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.

For more info … Already available



IBM

Draft Document for Review July 1, 2014 8:52 am

SG24-8218-00

IBM MQ V8 Features and Enhancements

Maximize your investment in IBM MQ

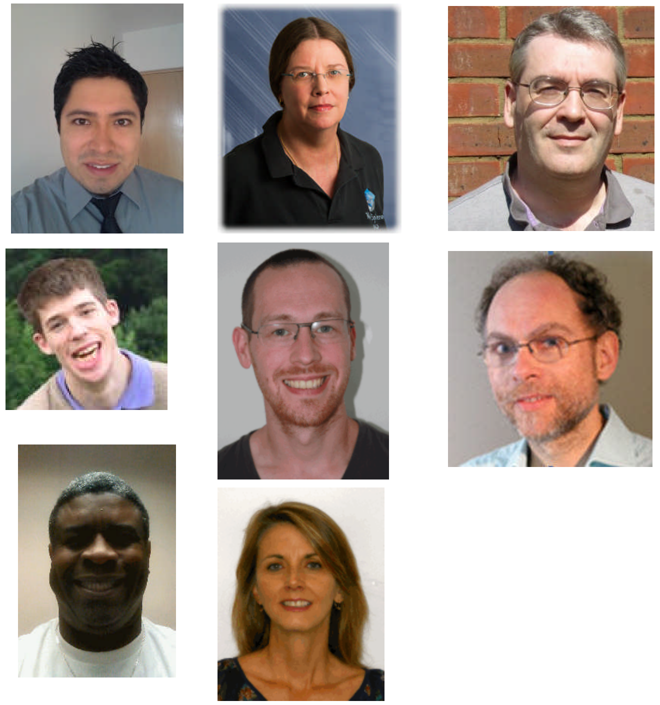
Discover new features that bring value to your business

Learn from scenarios with sample configurations

Rufus Credle
Carolyn Elkins
Peter Hayward
Valerie Lampkin
Matthew Leming
Jonatan Maya Sanchez
Mark Taylor
Mark Wilson

Redbooks

ibm.com/redbooks



https://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248218.html

SHAMELESS PROMOTION

- **MQ V8 for z/OS Wildfire Workshop**
  - Extensive hands on lab exercises
- **MQ V8 for z/OS Smart Seminars**
  - Can be brought onsite
  - No labs
- **FREEEEEEEEEEEEEEEEEE!**

Wildfire

Technical Hands-On Workshops

IBM zGrowth Team

IBM MQ v8 for z/OS Wildfire Workshop (IMQ08)

The Dallas Systems Center MQ technical team is offering a valuable "no-charge" hands-on workshop focusing on IBM MQ V8

Enrollment Procedure:
Contact your IBM Representative to find out when an event is scheduled close to YOU. .

(Additional Wildfire courses are available ….please contact your IBM Representative.)

Workshop Description

This workshop gives a guided hands on experience with IBM's premier connectivity product for z/OS, IBM MQ V8. It begins with implementation and administration of the products on z/OS that will help you plan your implementation or migration. The emphasis is on the new z/OS capabilities – bufferpools above the bar, channel initiator statistics SMF data, and channel accounting SMF data. You also get hands on experience with how applications can use features of MQ, for example how a CICS/COBOL application can use cooperative browsing..

Audience

This workshop is designed for administrators and developers new to the MQ family and experienced MQ personnel who need an introduction to MQ V8.

Duration: 2 Days

Skills taught
• Understand how IBM MQ for z/OS V8 is implemented
•Understand how bufferpools above the bar work, and where they can help with performance and capacity problems
• Understand how to use and administer the new channel initiator statistics and channel accounting records
•Understand the administration improvements
• Understand the new authentication features of MQ V8

Workshop outline
• Introduction to IBM MQ or a level set for those familiar with the product
• What's new in MQ V8.0 – across the board
• MQ V8.0 for z/OS Implementation and Customization
• MQ V8 for z/OS – Buffers above the bar
• MQ V8.0 for z/OS – New SMF data and reports
• MQ V8 – new authentication features
• Advanced Message Security for MQ V8

Questions???