MQ for z/OS – Performance, Performance

Lyn Elkins, IBM WSC
elkinsc@us.ibm.com

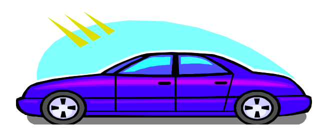
Mitch Johnson, IBM WSC
mitchj@us.ibm.com

Agenda

- What is performance to me?
- Performance reports
- Performance Inhibitors
- Performance Tuning in the queue manager
- Performance Tuning in applications
- Summary

What is performance to me?

- **Performance can mean different things:**
  - Meeting sub-second SLAs on critical transactions
    - In spite of workload fluctuations
  - Meeting batch windows
  - Meeting previously set expectations
  - Performance is not availability
    - Though if resources are not available it can show up as a performance problem



- **Performance is a matter of perception**

- **It can also mean different things to different applications in the same organization!**

What is performance to me? Notes

- **Performance can mean different things:**
  - Workload fluctuations can be predictable
    - Daily - Market open, ‘lunch time’ spikes
    - Weekly – Monday morning blues, Friday payday
    - Monthly – Pension day payouts
    - Annual – ‘Black Friday’, enrollment periods
  - Some workload fluctuations are not as predictable
    - “The market went nuts today”
  - Batch window are still critical to many businesses
    - SOA evolution has shortened or eliminated some, but the work still has to get done
  - Meeting previously set expectations
    - Yesterday my request was back in less than a second, today it is two seconds. MQ must be broken
  - Just as availability is not performance, performance is not availability
    - A sharp slowdown caused by performance problems may be perceived as an outage, just as a real outage may be reported as a performance problem

MQ for z/OS Performance Reports

- **Published after major releases**
  - SupportPac MP1G for MQ V7.0.1
  - SupportPac MP1H for MQ V7.1
  - SupportPac MP1J for MQ V8.0
  - SupportPac MP1K for MQ V9.0
  - Emphasis is on new functionality and major areas of change
  - Typically existing features and functions are not retested
- **Hardware & Software versions**
  - 'Best available' at the time of testing
- **Benchmark environment not a production environment**

MQ for z/OS Performance Reports - Notes

- Generally, if you need information about performance characteristics for a feature introduced in an earlier release, you will need to look at the report for that release. SupportPac MP16 contains some consolidated information about a particular feature, but it may be from older hardware and software.

- The benchmark environment is relatively pristine, it's not running a production workload. Numbers from benchmark reports should only be used as guidelines, not as absolutes.
  - YOUR MILEAGE WILL VARY!

Performance Inhibitors

- Unnatural expectations
  - Performance reports
  - Other peoples ideas
- Lack of resources
- Volume growth over time
  - DASD response times
  - Channel waits on shared queues
  - CPU
  - Storage
- Unexpected volume
- Applications

Performance Inhibitors - Notes

Unnatural expectations

- Performance reports
  - 'Why does IBM report X when we can only get Y'?
- Other peoples ideas
  - 'In my environment, I get 2500 transactions per second'
    - Sometimes it is a difference in measurement criteria

Lack of resources

- The overall system may have constraints that MQ has no control over

Volume growth over time

- This is what I think of as the 'creeping syndrome', a process that is executed once an hour initially, every minute after a full rollout, and going to millions of executions per second when exposed as a service can impact performance in surprising ways. If planned, the impact can be mitigated in various ways.

Unexpected volume

- Stock market meltdowns, recovering from network outages, complete catalog updates, initial database loads
- Unexpected volume growth – anticipating demand for this process is underestimated by a substantial factor
- If not prepared, these events can cause critical performance problems

Applications

- Always an opportunity.

What kind of performance problem do I have?

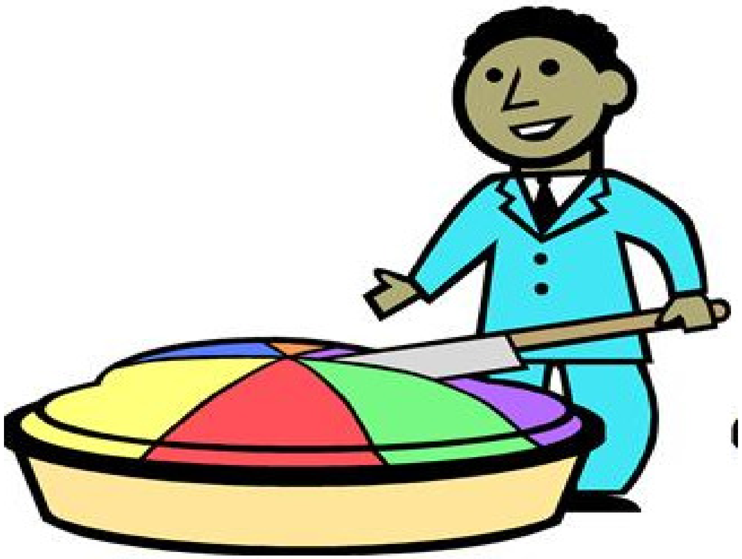
- **Missing SLAs?**
  - Is my workload smooth or spiky?
- **Occasional slowdowns?**
  - Cyclical or random?
- **Over-consumption of resources?**

What kind of performance problem do I have? Notes

- Smooth workload, that is almost a steady state of messages being processed, is much easier to tune to and to measure than a spikier workload. Periodic, predictable spikes, can also be managed.
- Unpredictable volume spikes are the bane of any system, but can often be absorbed with minimal impact if there is reserve capacity in the system. This is not an MQ statement, but a whole environment statement.
- Performance problems are often reported as MQ problems because it is easy to see queue depth growing, therefore it must be MQs fault.
- As a systems administrator there are some things that can be done to improve performance, or more importantly to avoid performance problems.

Performance issues with MQ

- MQ uses system provided resources, we cannot cover everything so here are the high points
  - Storage – bufferpools and pagesets
  - Logging



Performance issues with MQ - Notes

- MQ can only perform as well as the underlying resources that are used. There are a number of things that can be done with the queue managers to help control performance issues, some will be highlighted in this presentation, along with what has been exposed via the MQ SMF data that aid in finding tuning opportunities.
- Changes to MQ for z/OS itself that have been made in recent releases are highlighted. Often performance gains can be made by upgrading to the most current release.

- **Virtual Storage – Largest user is the bufferpools**
  - Bufferpools
    - For private queues, messages are put into bufferpools
    - Bufferpool allocation and tuning are critical for private queue performance
    - MQ Statistics records track use over time
      - Real time messages are indicators that the pool is currently constrained
    - Bufferpool thrashing
      - Caused by messages being put at the same time they must be read into a constrained bufferpool

Queue Manager Performance – Bufferpools - Notes

- **Often the biggest 'bang for the buck' in tuning MQ on z/OS can be from evaluation of the bufferpool use**
  - This is even true for MQ V8 users, where large above the bar bufferpools are supported
- **Even though many are aware of the benefits of bufferpool tuning we still see problems in this area**
- **The next slides illustrate how to find 'hotspots' and potential problem areas.**
- **Know your environment:**
  - As a warning to casual observers of performance data; if you don't know what looks normal, it can be difficult to figure out what may be a problem.
  - As a warning to serious observers of performance data; what looks normal, may not be right.

Bufferpool Use – General Recommendations

- **MQ Bufferpool and Pageset reservations:**
  - Reserve Buffer pool 0 and Page set 0 for MQ
  - Buffer pool 1 for 'SYSTEM' queues that do not get deep
  - Buffer pool 2 for 'SYSTEM' queues that may get deep
  - Reserve a separate bufferpool and pageset for the SYSTEM.CLUSTER.TRANSMIT.QUEUE

Bufferpool Use – General Recommendations - Notes

- **Bufferpool and Pageset 0 should always be reserved to MQ to use**
- **As of V6, MQ also began using Pageset 1. If you have queues using PSID 1, you may want to gradually move them to other resource pools.**
  - Please note this is NOT formally documented in the Knowledge center.
- **If you have a large and active cluster, especially if you are using pub/sub in a cluster; please use a separate pageset and bufferpool for the SYSTEM.CLUSTER.TRANSMIT.QUEUE. This will prevent the chattiness of the cluster, and possible cluster member outages from impacting other workload.**

Bufferpool Use – General Recommendations

- **Application Bufferpool use:**
  - Buffer pools 3-99 for application data
  - One or more Buffer pools for short lived messages
    - Buffer pool should not fill up
    - Short lived may be seconds or minutes
    - Make buffer pool as large as necessary
    - Keep Buffer pool < 85% full
      - DWT = 0
    - Allow for unexpected spikes
    - High volume request and reply queues should use different bufferpools where possible
      - And if not, separate pagesets are recommended
  - Buffer pool for long lived messages
    - Expect messages to be moved to the page set
- **Keep batch processing separate from transactional**

Bufferpool Use – General Recommendations - Notes

- Bufferpools for short-lived messages should have enough pages without ever getting over 85% full.
- Once the buffer pool gets over 85% full then the queue manger starts moving pages out to the page set to free up space in the buffer pool (the deferred write task or DWT in the statistics printing). Applications getting these messages on these pages may have to do disk I/O to retrieve them – which will slow down the applications.
- You should also ensure you have enough capacity to handle peak work loads, for example at busy times, and if there was an outage, so the work is now flooding in to the queue manager.
- If you have long lived messages, they will typically be flushed to disk, either to free up pages or if they have been in the bufferpool for 3 checkpoints. A relatively small bufferpool has an advantage, in that if the DWT is started frequently it is only writing a few pages at a time. If the bufferpool is very large, then the task may write a large number of pages when space is needed. This may have a small, but possibly noticeable, blip on the CPU resources used by the queue manager.

What does bufferpool stress look like?

- **Free pages at 20% or less – for short term messages**

QMGR	BP	NumBuff	%now	%low	dwt	dmc	stl	stla	sos
QML4	2	70000	53	19	0	0	46571	0	0
QML4	3	70000	98	20	0	0	46028	0	0
QML4	3	70000	75	20	0	0	0	0	0

- **Free pages at 5% or less – for all messages**

QMGR	BP	NumBuff	%now	%low	dwt	dmc	stl	stla	sos
QML2	3	70000	18	0	109	198906	922354	1	50
QML2	3	70000	19	0	68	143872	387873	1	13

What does bufferpool stress look like? - Notes

- **What is shown is from an older Buffer Manager report, imported into a spreadsheet**

- **Things to watch out for include:**
  - A concentration of highly active queues in one bufferpool
    - Avoid the ‘define like’ syndrome
  - A mixture of long lived and short lived messages in the same bufferpool
  - High volume request and reply queue on the same pageset and bufferpool
  - Change in usage patterns over time

- **The best I/O is no I/O**

- **In the samples shown**
  - The free pages at 15% is the point where the async write task normally kicks off
    - Not a problem for long lived messages, but may be an indicator of problems if the messages are short lived
  - The free page at 5% or less is a real sign of problems, no matter the type of messages
    - The synchronous write process will start, delaying work and consuming additional CPU

What does bufferpool thrashing look like?

- **Bufferpool churn example Note the 'low' value of '0' and the SOS value of 413**

```
> 01 Buffs      15000  Low        0  Now      1844  Getp    351632  Getn    198775
  01 Rio       102140  STW   472341  TPW    260049  WIO     129209  IMW      85105
  01 DWT          137  DMC    81686  STL    276198  STLA         4  SOS        413
```

  - The bufferpool went to short on storage 413 times in a 5 minute interval
  - There were 102,140 reads from the pagesets
  - There were 129,209 writes to the pagesets, and 102,140 reads from the pagesets
  - The async write process threshold was hit 137 times
  - The synchronous write process threshold was hit 81,686 times!

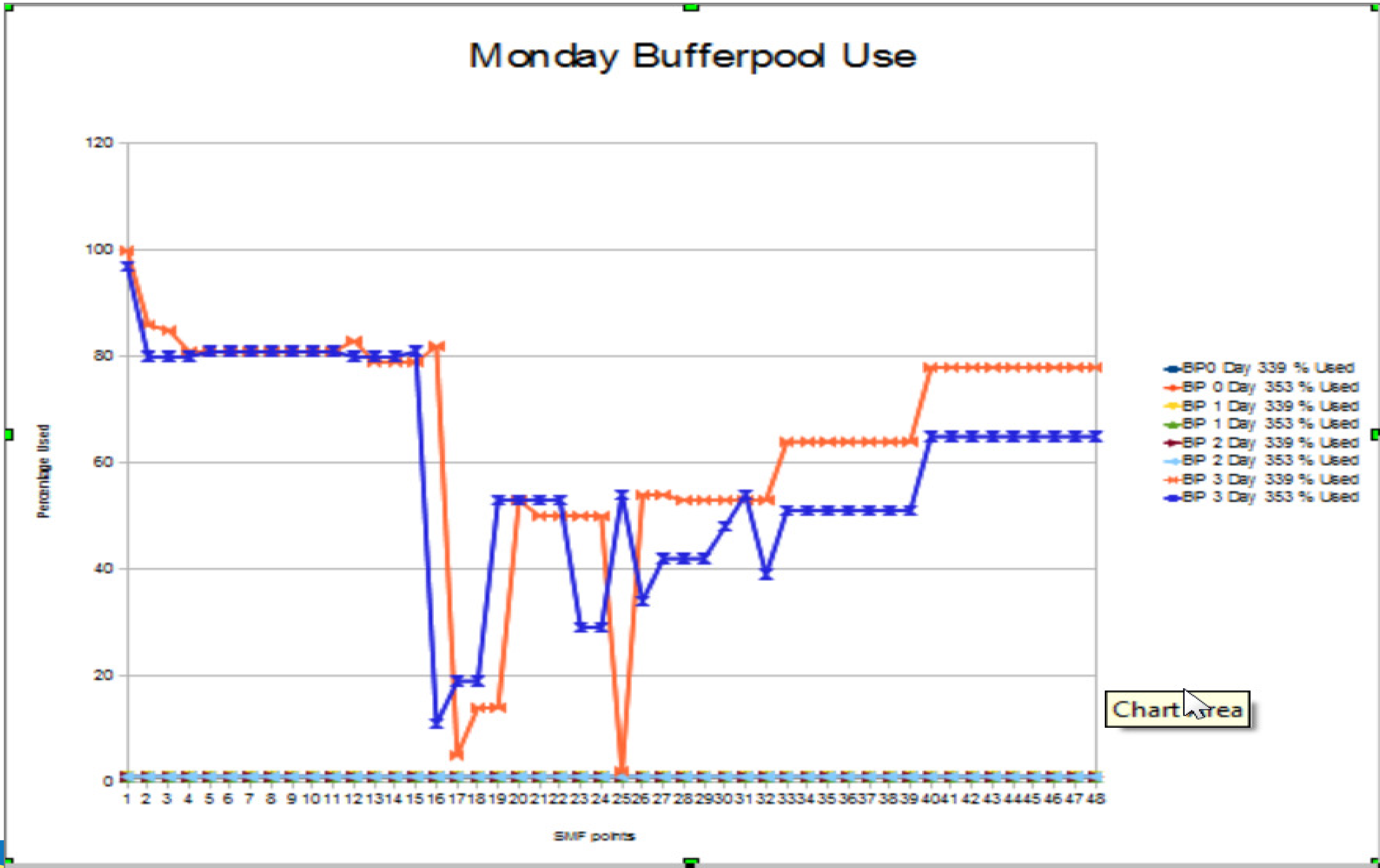
- **JES log also had repetitions of the following messages**

```
CSQP020E QML1 CSQP1RSW Buffer pool 1 is too small
CSQP020E QML1 CSQP3GET Buffer pool 1 is too small
```

What does bufferpool thrashing look like? Notes

- This information is taken from the old MQ1150 SMF print program
- In this truly fine 'bad example', the volume and overuse of a single bufferpool and pageset
- The SMF interval was really short for this evaluation, 2 minutes
- Messages were being read (RIO) at the same time they were having to be flushed to disk
- This created a huge amount of overhead in the queue manager, and caused a number of slowdowns as real I/O had to take place.
- In this case breaking up the queues in this pageset into multiple pagesets greatly reduced the contention. Contention reduction made a substantial improvement in performance, and lower CPU utilization for the queue manager as a whole.

Bufferpool Trends and Analysis



SMF115 – Bufferpool Trends and Analysis - Notes

- **In the chart shown two high volume days were compared to see if there was a pattern to the BP use.**
  - The ‘Y’ axis shows the percent of the bufferpool that is used
  - The ‘X’ axis shows the SMF point during the 24-hour periods (SMF is set to 30 minutes)
  - BP 0, 1 an 2 showed almost no utilization.
  - BP 3 was in very heavy use, some of the time.
  - BP 3 is under some stress.

- **Having multiple days worth of data is vital, had there just been one heavy day the bufferpool use shown may have been an anomaly. Data from longer periods of time, when compared like this can be very useful in tracking usage, and predicting when there may be issues.**

- **In this case there was a clear pattern of overuse of bufferpool 3, in further evaluation the SMF116 data showed that all the queues that were being used for this queue manager were defined on the same pageset/bufferpool. By moving some of the queues to another resource pool, the stress was reduced, work flowed faster and the CPU usage was reduced.**

Queue Manager Performance Improvements - MQ V7.0.1 and beyond

- **MQ V7.0.1 included several queue manager performance changes, including:**
  - **Small Message storage change**
  - **Above the bar Queue Indexing**
    - Allows deeper indexed queues
    - Constant cost for PUTs to deep queues
    - Somewhat higher cost for MQGETs from deep indexed queues
    - Can increase queue manager restart or first access time
  - **Above the bar Security Cache**
    - Allows larger security cache structure
    - Some users have increased security scan interval
      - Scans being done less frequently, less overhead during peak times

Queue Manager Performance Improvements - MQ V7.0.1 and beyond

- **MQ V7.1 introduced Shared Message Data Sets**
  - Huge performance increases over using DB2 for message body offloading

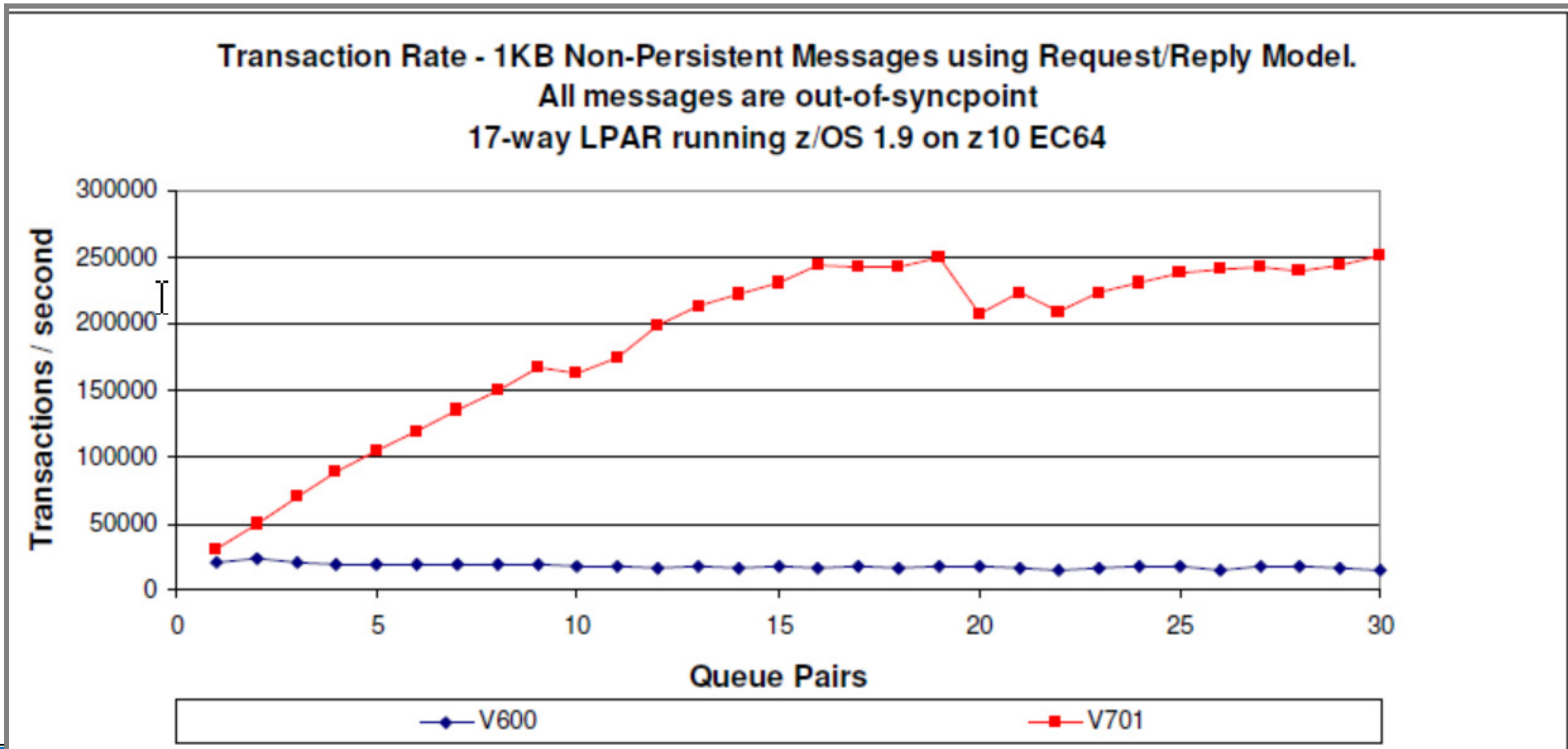
- **MQ V8 included:**
  - Above the bar bufferpools
    - I/O avoidance for high volume queues
    - 4GB Log files

Notes on improvements to MQ

- **This is not an all inclusive list of performance improvements made to MQ, just some of the heavy hitters (and often requested features).**
- **The features mentioned have been shown to improve performance and in some cases capacity of z/OS queue managers without having to make application changes to take advantage of them.**
  - There may be admin changes required to take advantage

Small Message Storage – MQ V7.0.1

- In version 7.0.1 changes were made to the way small messages are stored internally by the queue manager. Throughput was substantially increased, and CPU costs decreased for small messages.



Small Message Storage – MQ V7.0.1 - Notes

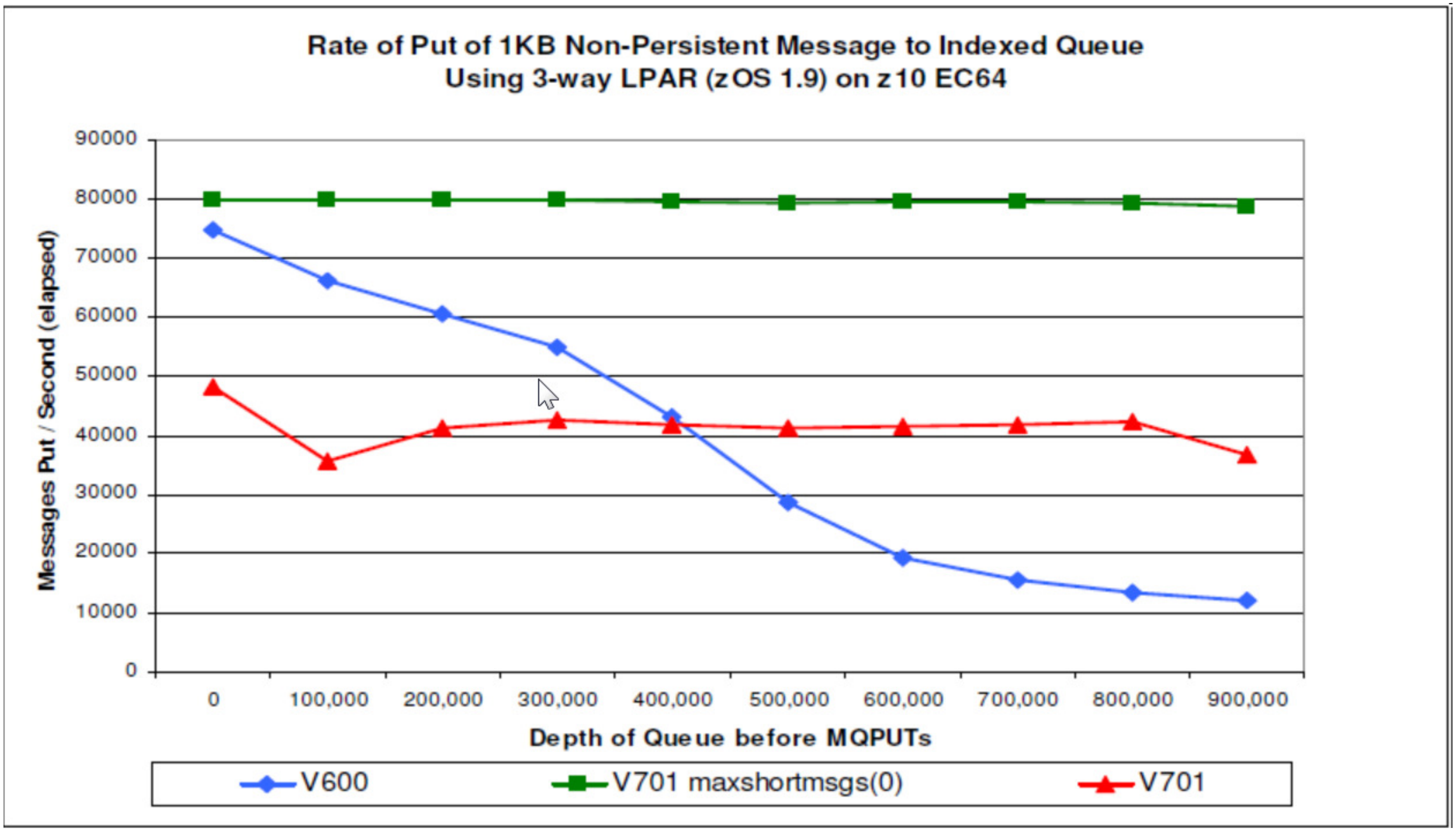
From SupportPac MP1G:

- Prior to version 7.0.1, small messages were stored such that multiple messages could co-exist on the same page. This meant that the scavenger could not run once a message was deleted as there were potentially other messages still on the page.
- Instead, the small message scavenger would run periodically – up to every 5 seconds. This means that a workload using small messages could see a build up of dead pages that were waiting to be scavenged. With ever faster processors, the time taken to fill the bufferpool with dead pages becomes significantly reduced. In turn, this meant that the messages would overflow onto the pageset and potentially the queue manager could be spending time performing I/O – causing slower MQGETs and MQPUTs.
- To allow the scavenger to work more efficiently with small messages, each message is now stored in a separate page. In addition a separate index page is used to hold data for approximately 72 messages. Once the message is deleted, the page holding the message data can be re-used immediately but the index page can only be scavenged once all messages referenced are deleted.

The short message feature can be turned off with a tuning parameter:

REC QMGR (TUNE MAXSHORTMSGS 0)

Queue Manager Performance Improvements – Indexing – MQ V7.0.1



Queue Manager Improvements – Indexing Improvements – MQ V7.0.1

- **With V7.0.1 the cost to put a message to an indexed queue is almost constant, an indexed queue can be much deeper. In performance testing there have been indexed queues with as many as 100M messages, previously the limit had been 7.2M.**

- **Costs of using 64-bit storage**
  - There is a slightly higher cost for putting messages to an indexed queue, until the queue depth grows. In performance testing putting message to an indexed queue was slightly more expensive until the depth reached 200,000 messages. Depths greater than that showed a dramatic increase in costs for 'below the bar' indexing.
  - There is a slightly higher cost for getting messages, and it grows as the queue depth grows.
  - These costs are due to the additional overhead of 64-bit addressing.

- **Deep indexed queues may also contribute to slower recovery time in the event of a failure.**

- **The use of indexed vs. non indexed queues is in the application section of this presentation.**

Queue Manager Performance - Logging

MQ logging

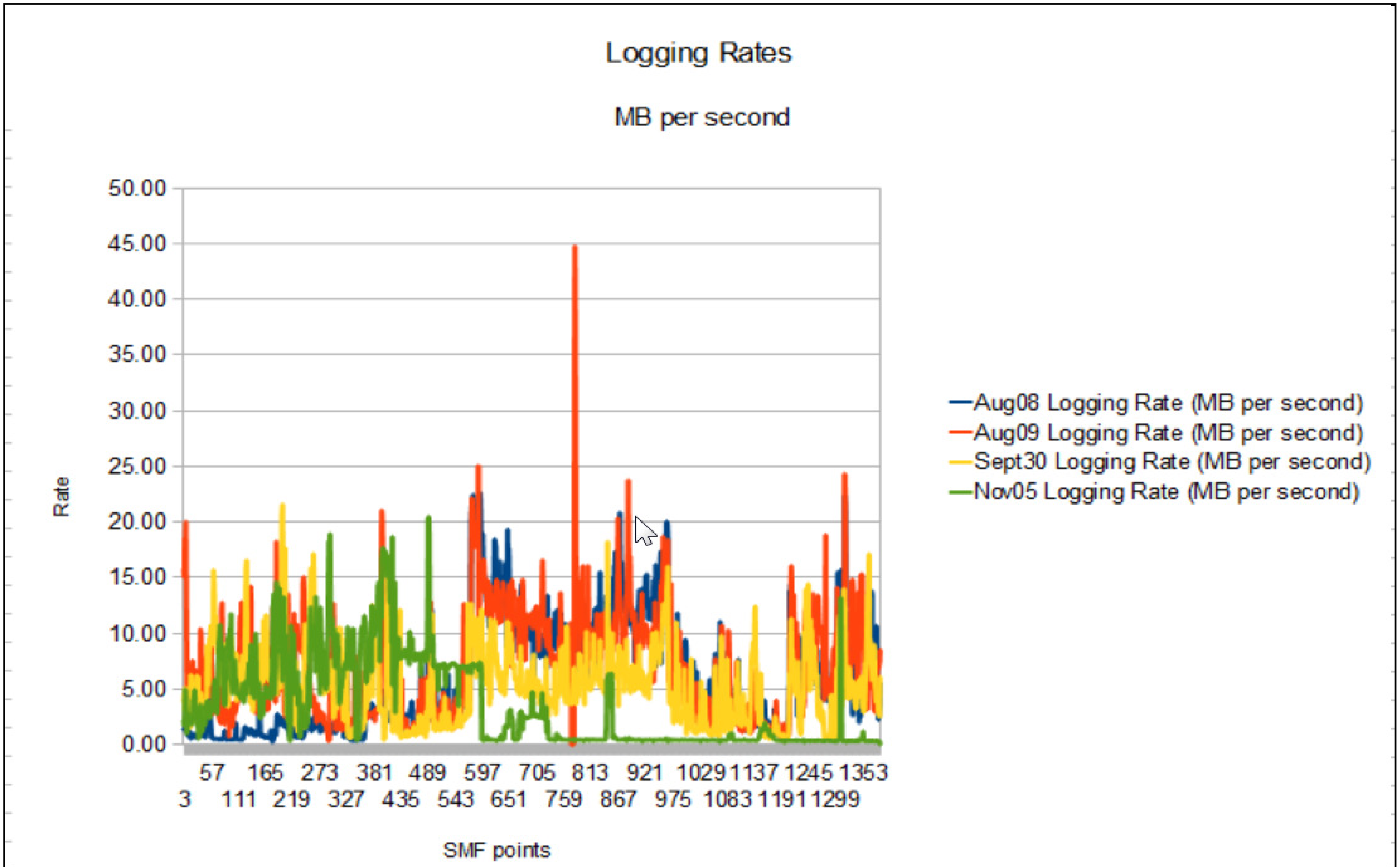
- **All Persistent messages are logged to disk**
  - MQ log files are limited to 4G
    - This limit is made obvious in V7.1
      - Message - CSQJ499I: Log data set is larger than 4GB
  - Logs are switched when full
  - Checkpoints are issued at log switches and at LOGLOAD points
  - Proper positioning of the log files is a critical factor
  - When examining real I/O
    - Look not only for averages, but for outliers

Performance issues with MQ - Notes

- **Logs can be defined at larger than 4G, but MQ only uses 4g of them. A log switch will occur when it reaches that limit.**
- **Checkpoints counted in the SMF115 data are only those that are LOGLOAD driven, checkpoints at log switches are not included in that count.**
  - This changed, and is undocumented (to date) in V9. The checkpoints driven by log switches are now included in the count.
- **I/O rates can vary dramatically, there have been a number of performance issues related to work backing up behind a single or a very few slow responses from the I/O subsystem.**
- **Many documents would have people believe that with more current hardware and software the positioning on devices is less critical. In the past few years a number of critical performance issues have been traced back to things like active and archive logs on the same physical device causing periodic slowdowns.**
- **Average I/O rates, like any averages, can be misleading. While my average may be quite good, included in that average may be the one or two long response times that is causing serious back-up.**

.

Logging Rate Trends and Analysis



Performance issues with MQ - Notes

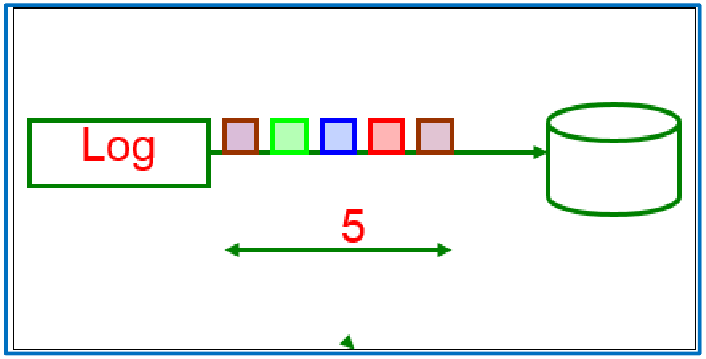
- **The chart is taken from real data. The spike, which had to be investigated was not a volume spike, but actually from a change in the SMF interval.**

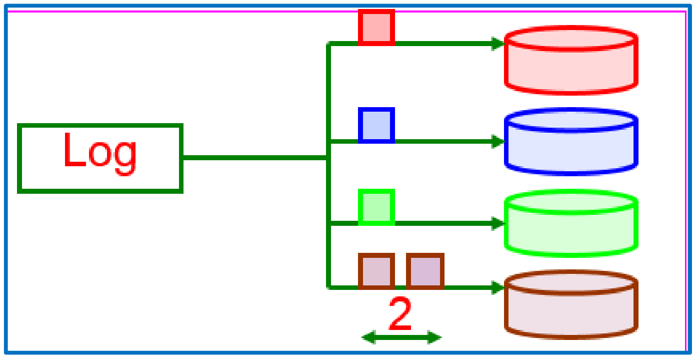
- **In this case the queue manager was being almost constantly throttled by logging I/O limit (old hardware, max. I/O rate was 25 MB/second), in this environment.**
  - They moved to newer hardware and were able to achieve >50 MB/second

Queue Manager Performance - Logging VSAM striping

- **What is VSAM Striping**
  - Simply it is using more than one physical location for the data
  - With VSAM striping writing 4 consecutive pages, each page will be mapped to one of 4 volumes, frequently reducing contention. Data transfer time is typically less.

- **Significant performance benefits for large messages**
  - But archive log cannot exploit this, so may be a bottleneck
  - Consider using VSAM striping if you have a peak rather than sustained high volume





VSAM Striping - Notes

- VSAM striping is part of DFP (VSAM). When you define your data sets you can define the data set as being striped.
- Without striping if you wanted to write 5 4K pages to a track, then one requests would be issued, and 6 pages of data sent to the device, for example page 1,2,3,4,5,6 on track 7. The time for the I/O requests is essentially the time to send 6 pages down the channel – perhaps 6 ms.
- With VSAM striping, the pages are spread across different volumes, so pages 1&5 would be written to volume 1, page 2 &6 to volume 2,page 3 to volume 3, and page 4 to volume 4. So we now have 4 I/Os in parallel, with at most 2 pages per I/O. This may take 2ms – which is less than the 6ms above.
- You can use VSAM striping for active logs, but not for archive logs. So if you have a sustained high log throughput you may run out of active logs, and have to wait for logs to be archived.

VSAM Striping - Notes

- **One customer, who resisted the advice to stripe the logs for years (and the advice came from many sources), saw a 35% increase in throughput when they moved to striped logs.**
  - They had tried using striped logs some years prior and found there was no benefit at the time
  - What they had not considered is that in the intervening years, their message sizes and volume had grown substantially.
  -

Queue Manager Performance – Shared Queues

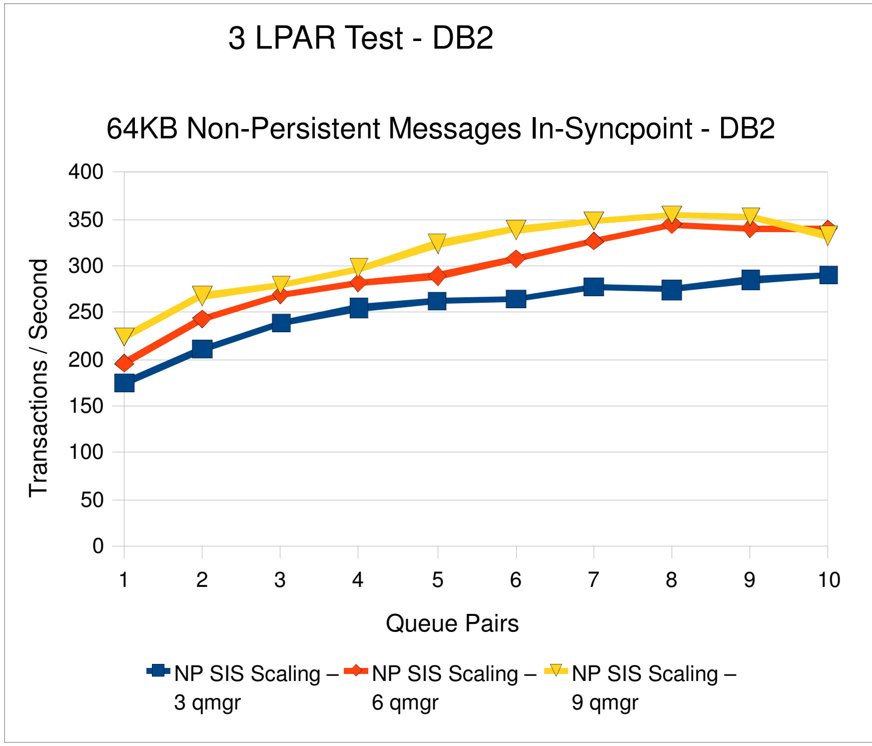
- **Coupling Facility**
  - CF storage constraints and large message performance problems can be mitigated by Shared Message Data Sets
    - New with V7.1
    - Reduces (BUT DOES NOT ELIMINATE!) the dependence on DB2 - large (over 63K) messages can be stored on SMDS
  - Links to the CF can become saturated, causing delays and performance problems

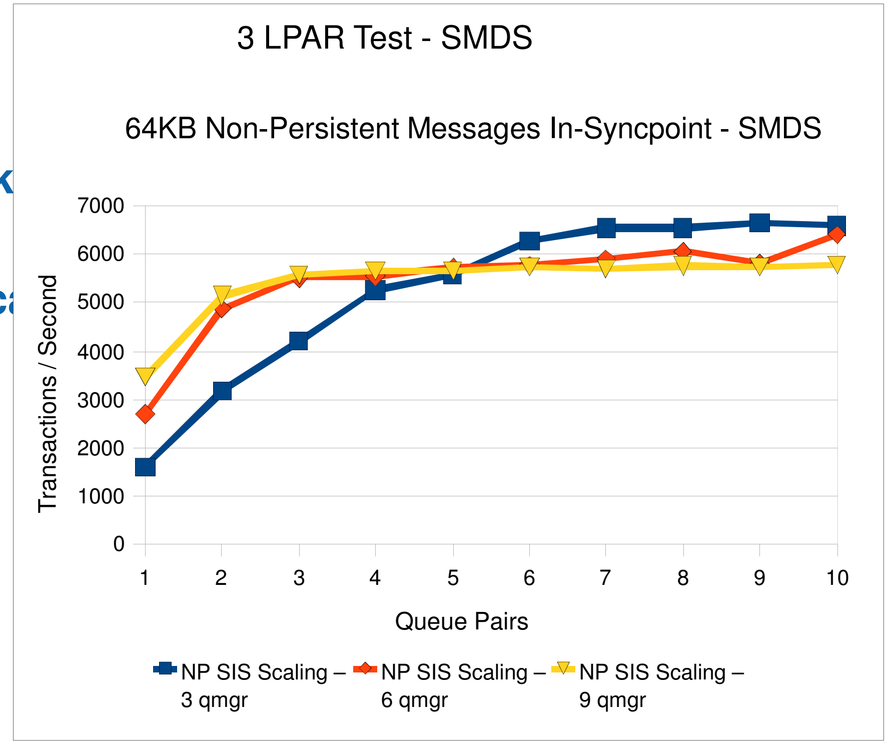
Performance issues with MQ - Notes

- The Coupling Facility can be a performance bottleneck when links to the CF from the LPAR where MQ resides become saturated. The CF activity reports, based on the RMF data show issues like this. The systems programmers or capacity group should be monitoring the CF activity reports.

- We do recommend that you become familiar with the CF activity reports.

Queue Manager Performance - SMDS



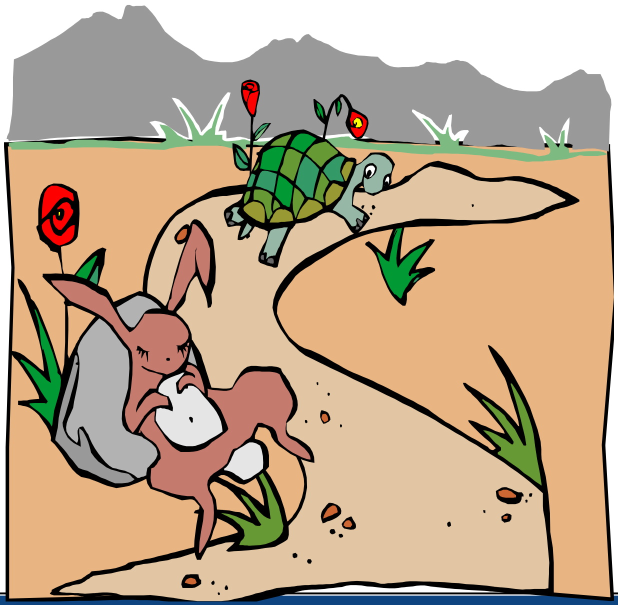


Queue Manager Performance – SMDS

- Using VSAM datasets for large message storage is a remarkable improvement over DB2

MQ Application Performance

- Choose the right queues
- Choose the right messaging styles
- Choose the right verbs



Alignment Page

Choosing the Right queues - Temporary Dynamic Queues – SMF data

This information was taken from the SMF116 class 3 records

```
Open name TEAMXX.MODEL                                    Object type:Local Queue
Base name AMQ.C9422A60F4386075                            Base type  :Queue
Queue indexed by NONE
First opened 12-03-2012 21:24:16.34
Last closed  23-09-2019 17:52:14.24
Page set ID           0,  Buffer pool        0
Current opens         0,  Total requests       10
Generated messages :        0
Persistent messages: GETs        0,  PUTs          0,  PUT1s         0
Put to waiting getter: PUT        0,  PUT1         0
PUTs: Valid        3,  Max size        9,  Min size        9,  Total bytes     27
-MQ call-          N         ET         CT        Susp       LOGW       PSET Epages   skip expire
 Open   :          1        850        125         727
 Close  :          1        113        111           0
 Put    :          3        106        104           0          0
 Inquire:          5         17         17
Maximum depth encountered          3
```

Choosing the right queues - Permanent Queues

This information was taken from the SMF116 class 3 records

```
== Task token : 12-03-2012 21:24:23.42,  55FE03F0,  55FD0000

 Open name TEAMXX.NOT.TEMP                              Object type:Local Queue
 Base name TEAMXX.NOT.TEMP                              Base type  :Queue
 Queue indexed by NONE
 First opened 12-03-2012 21:25:09.23
 Last closed  18-10-2019 00:31:46.22
 Page set ID           0,  Buffer pool          0
 Current opens         0,  Total requests         10
 Generated messages :         0
 Persistent messages: GETs         0,  PUTs          0,  PUT1s         0
 Put to waiting getter: PUT         0,  PUT1          0
 PUTs: Valid        3,  Max size         9,  Min size         9,  Total bytes     27
 -MQ call-         N        ET        CT       Susp      LOGW      PSET Epages   skip expire
  Open   :         1        39        38          0
  Close  :         1        26        26          0
  Put    :         3       115       113          0          0
  Inquire:         5        18        18
 Maximum depth encountered         3
```

MQ Application Performance - Queues

Choose the right queue:

- On z/OS Temporary Dynamic queues should be avoided
  - Higher CPU costs
  - Elapsed time can be significantly longer

The CPU cost comparison

Verb	TDQ	Permanent	Difference
Open	125	38	238%
Close	111	26	327%
Put	104	113	-8%
Inquire	17	18	-5%

The Elapsed Time comparison

Verb	TDQ	Permanent	Difference
Open	850	39	2079%
Close	113	26	3347%
Put	106	115	-8%
Inquire	17	18	-5%

- Choose the right queue - continued:

- The Suspend count comparison

Verb	TDQ	Permanent	Difference
Open	727	0	WOW!!!
Close	0	0	0
Put	0	0	0

Choosing the Right queues - Temporary Dynamic Queues - Notes

- **The data shown is taken from one of the SMF print programs from an older version of MP1B (MQ116S)**

- **The information presented here is a mixture of counts and averages**
  - The MQ calls made is a count of the calls made as part of this unit of work, or during the interval for long running tasks.
  - The ‘ET’ (elapsed time) and ‘CT’ (CPU time) are averages for the unit of work.
  - The remaining fields are counts.

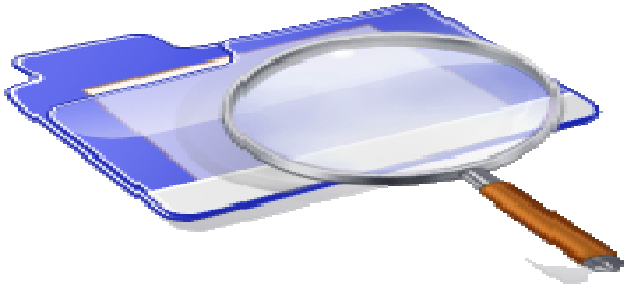
- **Not only is the CPU noticeably higher, note the suspend count. If the application has very strict SLAs avoiding the opportunity for suspensions can be critical in a heavily loaded system.**

- **TDQs are often used as reply queues for online monitors, which seems like a such an oxymoron. Most monitors have optional permanent queues.**

MQ Application Performance - Queues

- **Queue index specification is unique to MQ on z/OS**
  - Messages that are retrieved using an index-able field benefit from being indexed even when the depth is not high.
    - Message ID
    - Correlation ID
    - Token
    - Group ID

- **The use of a proper index can substantially improve performance an CPU consumption.**



MQ Application Performance - Queues

- The information that follows illustrates the need for proper queue definition based on application use. It also shows where the MQ Admins and application programmer can find out what is going on within an application.
- Many of the slides are from SMF 116 data that has been printed using MQ116S, from an older version of SupportPac MP1B.

Non-Indexed Queue retrieval

```
Open name TEAMXX.NON.INDEXED                    Object type:Local Queue
Base name TEAMXX.NON.INDEXED                    Base type  :Queue
Queue indexed by NONE
First opened 12-03-2012 15:12:58.55
Last closed  **-**-**** **:**:**.**
Page set ID           4,  Buffer pool        3
Current opens         1,  Total requests      61
Generated messages :        0
Persistent messages: GETs        0,  PUTs        0,  PUT1s        0
Put to waiting getter: PUT        0,  PUT1        0
GETs: Valid       28,  Max size        80,  Min size        80,  Total bytes     2240
GETs: Dest-S       28,  Dest-G        0,  Brow-S        0,  Brow-G        0, Su cessful destructive        28
Time on queue : Max 4503.730054,  Min  257.434901, Avg 3958.326341
-MQ call-        N        ET        CT       Susp       LOGW       PSET Epages   skip expire
 Get    :       28       384       369         0          0          0       0   3505     0
 Inquire:       28        22        21
Maximum depth encountered        258
```

Non-Indexed Queue retrieval - Notes

- **How can you tell if a queue is being read for a specific message?**
  - In V8 and above there is a JES log message that shows when a queue should be indexed, CSQI004I .
  - Some admins have already suppressed this message because of the frequency. The SMF116 class 3 data will also indicate this is an area of opportunity.

- **In the SMF 116 class 3 data record, the fields of interest are:**
  - The Queue Indexing
  - The Type of GET request being made. Those with a '-S' are for specific messages (Get by correlid, get by message id, etc.). Those with a –G are generic, get the next message on the queue.
  - The average CPU expenditure for the successful gets – the 'CT' column highlighted
  - The number of pages skipped while finding matching messages

Indexed Queue Retrieval

```
Open name TEAMXX.INDEXED                                Object type:Local Queue
Base name TEAMXX.INDEXED                                Base type  :Queue
Queue indexed by CORREL_ID
First opened 12-03-2012 15:16:01.44
Last closed  12-03-2012 15:16:50.35
Page set ID             4,  Buffer pool         3
Current opens           0,  Total requests        59
Generated messages :          0
Persistent messages: GETs         0,  PUTs          0,  PUT1s          0
Put to waiting getter: PUT         0,  PUT1          0
GETs: Valid         27,  Max size        80,  Min size         80,  Total bytes     2160
GETs: Dest-S        27,  Dest-G          0,  Brow-S          0,  Brow-G          0, Su cessful destructive         27
Time on queue : Max 4780.946117,  Min  422.046309, Avg 4288.437716
-MQ call-            N          ET          CT         Susp        LOGW        PSET Epages  skip expire
 Get    :           27         105          99           0           0           0      0      0      0
 Inquire:           26          21          20
Maximum depth encountered        258
```

Indexed Queue retrieval - Notes

- Note the differences between the non-indexed and indexed retrieval. In particular, no pages had to be skipped during the MQGET process. That saves both CPU and elapsed time.
- In practice, differences were seen with queue depths as low as 5-10 messages.

Indexed vs Non - comparison

- **Comparing the CPU time, both queues with the same max message depth:**
  - Indexed - 27 messages at an average of 99 CPU microseconds
    - 2673 µs for 27 messages retrieved
  - Non-indexed 28 messages at an average 369 CPU microseconds
    - 9963 µs for 27 messages retrieved
  - Difference 272%

- **Comparing the elapsed time**
  - Indexed - 27 messages at an average 105 microseconds
    - 2835 µs elapsed time for the messages
  - Non-Indexed 28 messages at an average 384 microseconds
    - 10368 µs elapsed time for 27 messages
  - Difference 252%

MQ Application Performance - Queues

- **Choose the right messaging style:**
  - Persistent messages are more costly than non-persistent
  - Use nonpersistent messaging –
    - When the message is a simple query
    - Easy to discover and recover
  - Use Persistent messaging
    - When the message drives an update transaction that must be coordinated
      - When designing/writing/testing the application recovery code is too challenging
    - Difficult to recreate the request
    - When required to by a business
    - A ‘C’ level executive is watching
    - **It’s OUR paychecks**

MQ Application Performance – Message Style

- **The CPU cost comparison**

Verb	Persistent	NonP	Difference
Open	125	38	238%
Close	111	26	327%
Put	104	113	-8%
Inquire	17	18	-5%

- **The Elapsed Time comparison**

Verb	Persistent	NonP	Difference
Open	850	39	2079%
Close	113	26	3347%
Put	106	115	-8%
Inquire	17	18	-5%

MQ Application Performance – Choose the right verbs

- **Like any other subsystem, the choice of verbs can improve performance and scalability.**
  - **Recycling code is a positive**
    - Reduces development time and effort
    - Often enforces best practices
    - Can reduce testing time
  - Recycling code is a negative
    - Can introduce performance problems if code not well understood
  - Increased use of a transaction can expose underlying issues

Choose the Right Verbs

- **Misuse of MQPUT1**
  - MQPUT1 combines an MQOPEN, MQPUT and MQCLOSE into one verb
  - Typically used for the reply messages on request/reply processing
  - More efficient if just putting one message
  - Substantial performance impact if putting multiple messages to the same queue

Effect of MQPUT1

- **Each MQPUT1:**
  - 117 ųs CPU, for a grand total **351,000** ųs
  - 121 ųs Elapsed time, for a grand total of 363,000 ųs
- **Each MQPUT:**
  - 72 ųs CPU, for a grand total of **216,000** ųs
  - 74 ųs Elapsed time, for a grand total of 222,000 ųs

```
PUTs: Valid       3000,  Max size         80,  Min size        80,  Total bytes 240000
-MQ call-            N          ET        CT         Susp       LOGW        PSET Epages    skip expire
 Put1    :       3000         121       117            0          0
```

```
PUTs: Valid       3000,  Max size         80,  Min size        80,  Total bytes 240000
-MQ call-            N          ET        CT         Susp       LOGW        PSET Epages    skip expire
 Open    :          1          84        81            0
 Close   :          1          18        18            0
 Put     :       3000          74        72            0          0
Maximum depth encountered        6000
```

Effect of MQPUT1 - Notes

- Remember that both elapsed time and CPU time reported in this section of the old MQ116S report is the average time, not the total time.
- The TASK output of the new MQSMF program also reports average CPU time.

Effect of MQPUT1

- **For one PUT it is less expensive to u an MQPUT1**
  - MQPUT1 - 117 total ųs
  - MQPUT - 171 total ųs
- **For two PUTs it is less expensive to an MQOPEN, MQPUT and MQCLOSI**
  - MQPUT1 - 234 total ųs
  - MQPUT - 213 total ųs
- **Draw your own conclusions**

MQPUT vs MQPUT1 comparison - Notes

- In one particularly good example of this, the WSC MQ people were reviewing CPU use for a very high volume queue manager. A single CICS transaction was issuing 7,000+ MQPUT1s to the same queue for each execution. The transaction, once executed a few hundred times a day had become a service. It was now being executed thousands of times a minute.
- Like the Inquisition, no one expected the dramatic jump in CPU.

Performance is a topic both deep and wide

- **We cannot cover everything in an hour**
  - Other things to look for include (but not limited to!):
    - Using zEDC for channel compression – See MP1J
    - Coupling Facility activity report
    - Using Storage Class Memory (SCM) to offload shared queue messages
    - Batch size, batch limit, and batch interval tuning on channels
    - Dispatcher and adapter task tuning in the CHIN
    - Throughput and CPU improvements for transmission used when using large fixed bufferpools
    - Client application tuning
      - Suppressing the CSQX511I and CSQX512I messages

More information

- **Performance is a huge topic, we have only scratched the surface. There is a lot more investigation that can be done, and more information being published regularly.**

- **There are a number of SupportPacs available:**
  - MP16 - Capacity Planning and Tuning for WebSphere MQ for z/OS
  - MP1K - Performance Report – IBM MQ for z/OS V9.0
  - MP1J – Performance Report – IBM MQ for z/OS V8.0
  - MP1H - Performance Report - WebSphere MQ for z/OS V7.1
  - MP1G - Performance Report - WebSphere MQ for z/OS V7.0.1
  - MP1F – Performance Report - WebSphere MQ for z/OS V7.0.0
  - MP1B - Interpreting accounting and statistics data IBM MQ for z/OS

More information

- **There are a number of SupportPacs available:**
  - MP16 - Capacity Planning and Tuning for WebSphere MQ for z/OS
    - http://www-01.ibm.com/support/docview.wss?uid=swg24007421
  - MP1K - IBM MQ Version 9.0 for z/OS
    - http://www-01.ibm.com/support/docview.wss?uid=swg24042470
  - MP1J - IBM MQ Version 8.0 for z/OS
    - http://www-01.ibm.com/support/docview.wss?uid=swg24038347
  - MP1H - Performance Report - WebSphere MQ for z/OS V7.1
    - http://www-01.ibm.com/support/docview.wss?uid=swg24031663
  - MP1G - Performance Report - WebSphere MQ for z/OS V7.0.1
    - http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24024589&loc=en_US&cs=utf-8&lang=en
  - MP1F – Performance Report - Performance Report - WebSphere MQ for z/OS V7.0.0
    - http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24020142&loc=en_US&cs=utf-8&lang=en
  - MP1B - Interpreting accounting and statistics data WebSphere MQ for z/OS
    - http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24007421&loc=en_US&cs=utf-8&lang=en

Final words

- **Many hands make light work**
  - Over the years we have worked with some wonderful people sharing their time and expertise
    - Tony Sharkey – MQ for z/OS Performance
    - Colin Paice – MQ Scenarios team
    - Pete Siddall – MQ for z/OS development
    - Paul Dennis – MQ for z/OS development

Other features

- These additional slides include performance features not covered due to time constraints

Log Compression

- **Log compression should be called 'log compaction'**
  - Run Length encoding used for message compression when being written to the log
  - The benefit is completely dependent on the data
    - If the messages are concise, there may only be cost
    - If the messages are mixed, there may be some benefit
    - If the messages are bloated, there may be noticeable benefit

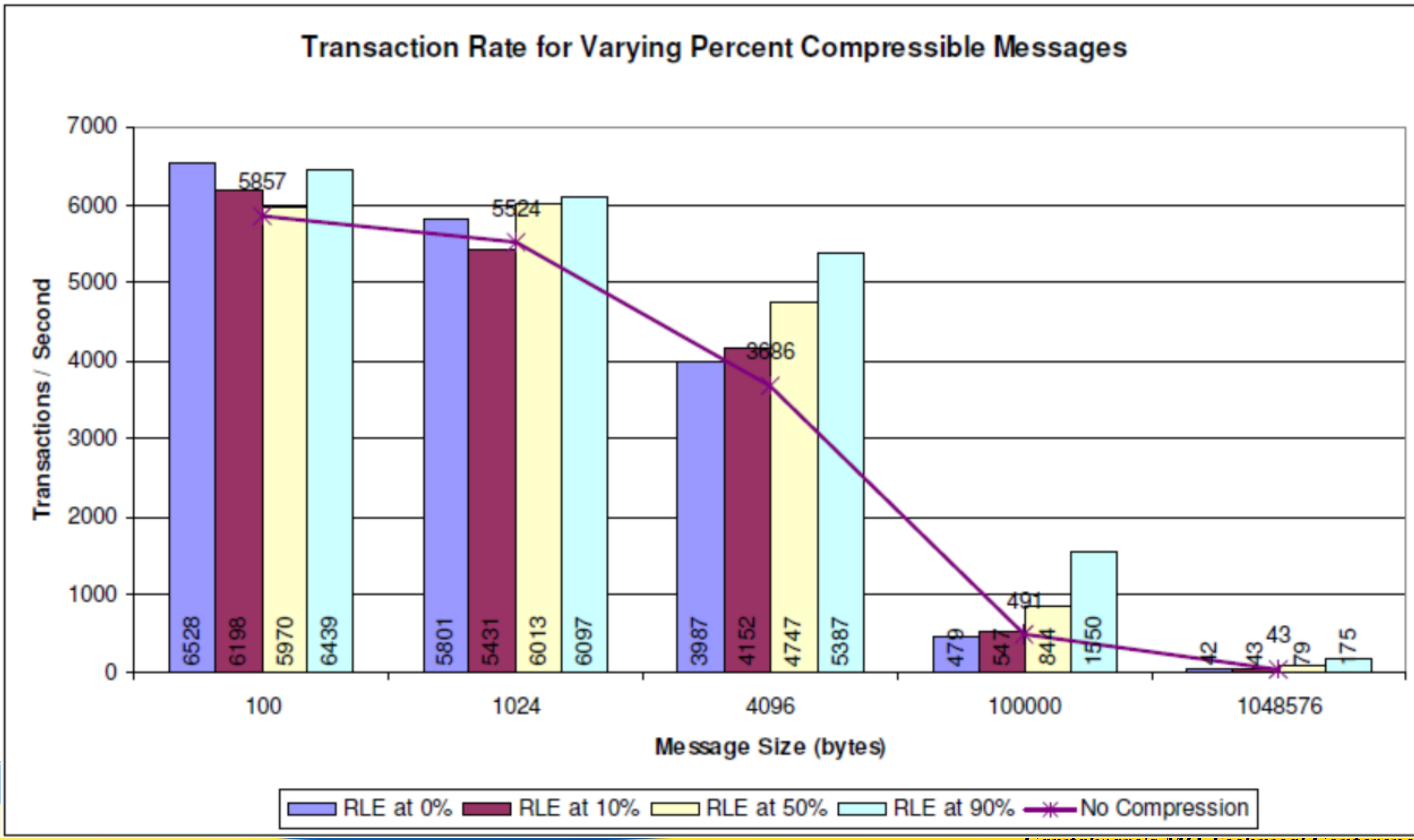
Queue manager Performance Improvements - Log Compression - Notes

- Log compression always incurs some CPU costs. On the next slide it shows that there is some benefit to throughput even when the data is not especially compressible.

- The SMF 115 records were enhanced for V7.01 to include information on how successful compression has been. There is no predictor, other than knowing your data. One customer analyzed their existing MQ logs and found that less than 10% of their message volume was a good fit, and that the CPU costs for checking all messages were more than they wanted to absorb for the limited benefit.

Log Compression

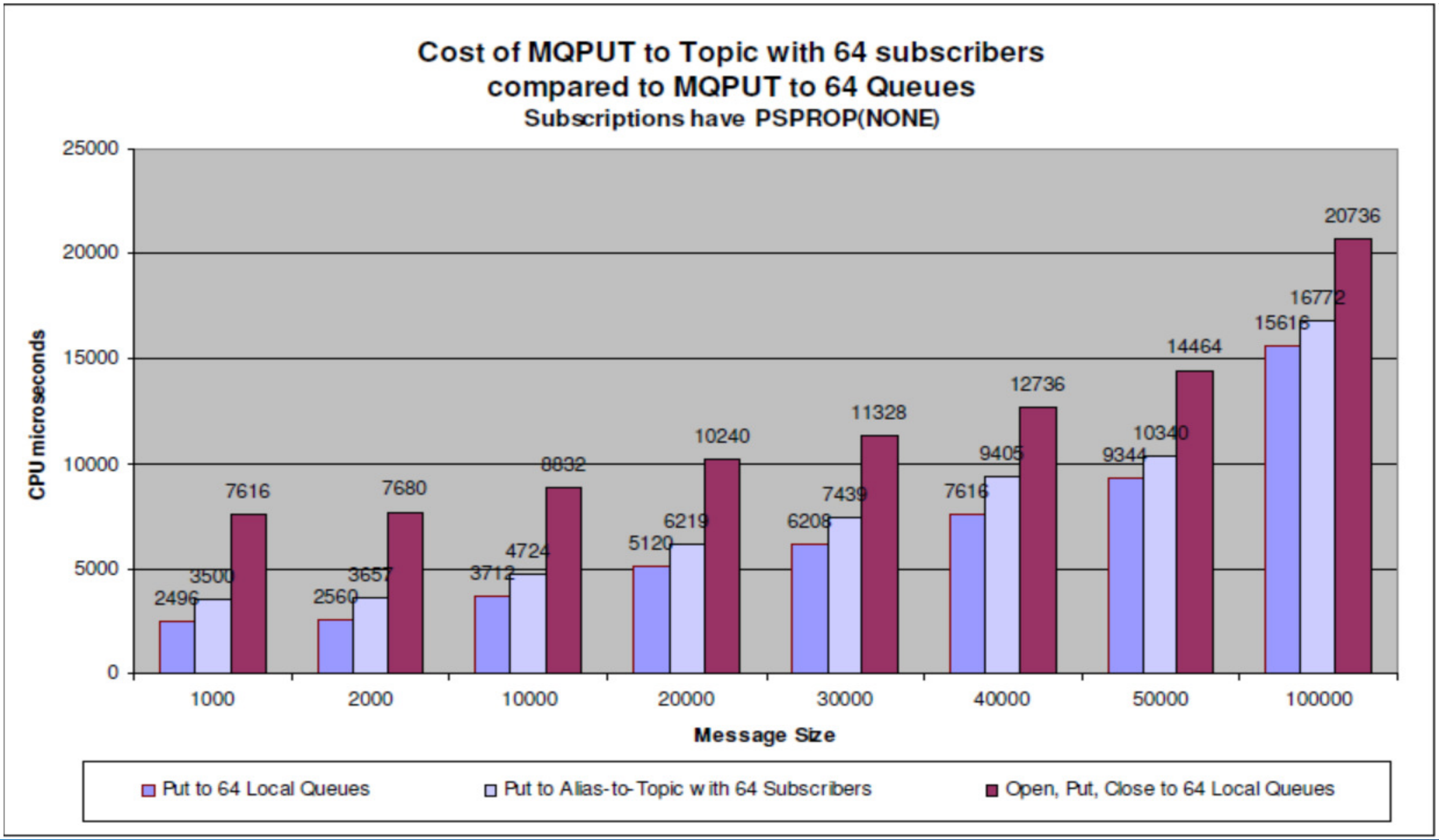
- **Log compression impact on message throughput rates**



Alignment Page

Choice of MQ Verbs – Pub/Sub

- Pub/sub is more expensive than point-to-point.
  - However putting to multiple queues can quickly add up too.



Choice of MQ Verbs – Pub/Sub - Notes

- Pub/sub, especially when implemented with a limited number of subscribers is more expensive than point-to-point messaging. However, when trying to 'emulate' a real pub/sub scenario – that is where a point to point application is putting messages to different targets, the CPU costs of true pub sub are lower.

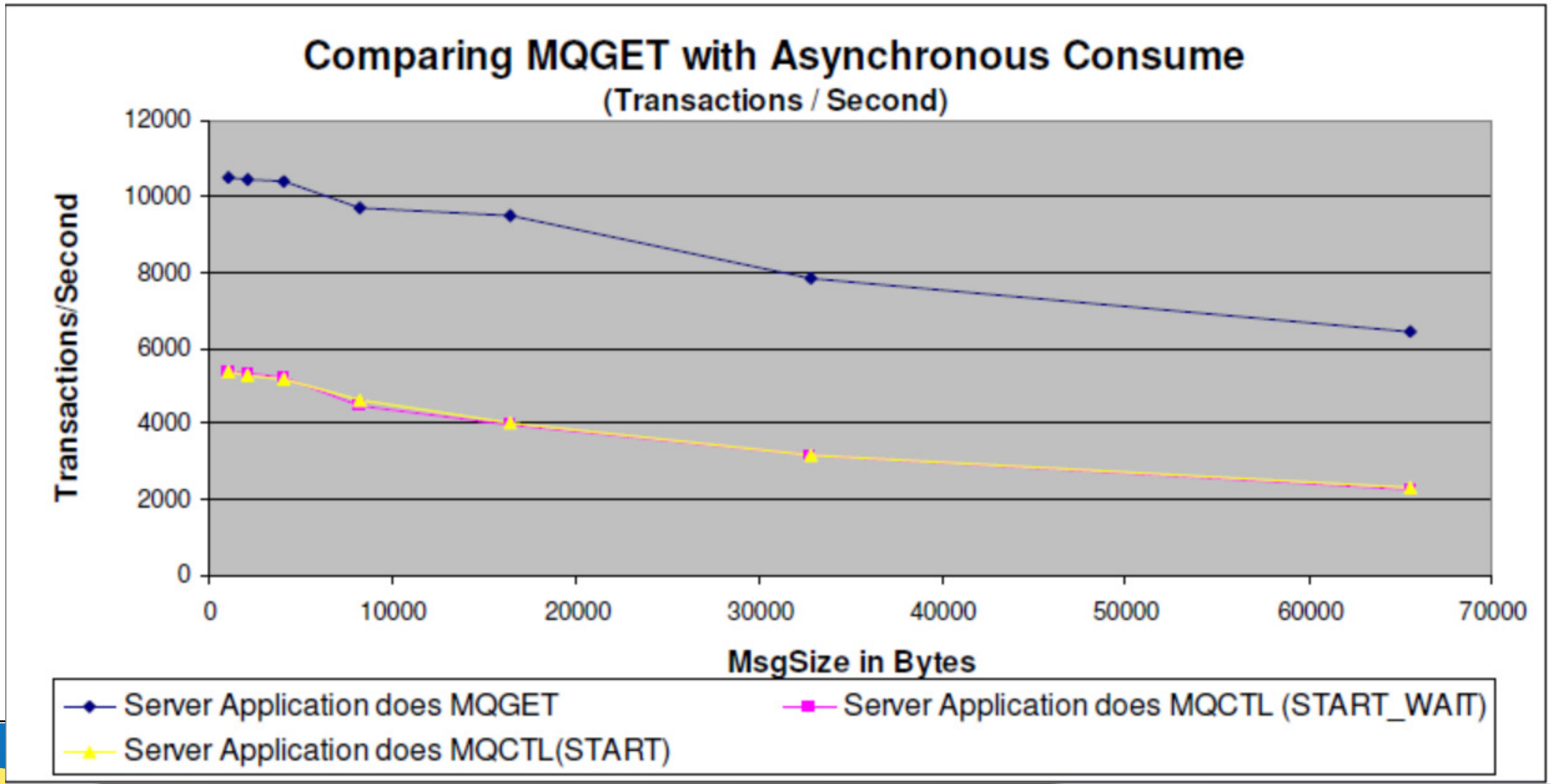
- The chart shown is from the MP1F SupportPac

- The publishing side, is only half the story. For more complete information, see the SupportPac.

- http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24020142&loc=en_US&cs=utf-8&lang=en

Choice of MQ Verbs – MQGET vs Async Consume

- **MQ V7 introduced the Asynchronous Consumer, the MQCB (register) and MQCTL (start and stop the action) verbs. These generally do not perform as well as more traditional processing, but do have practical application.**



Choice of MQ Verbs – MQGET vs Async Consume - Notes

- One particular use of the Async consume is when there are multiple queues that need to be monitored. A single application program cannot issue a MQGET with a wait for multiple queues simultaneously. Async consume allows you to do just that.

Put to waiting getters – old improvement, but not always visable

For out-of-syncpoint nonpersistent messages

- **If there is a get wait for the message – then putting application moves it directly to the get buffer, and posts the ECB. The message does not touch the queue, true for both private and shared queues**

	Put CPU	Get CPU	Total CPU
Put only (load)	147		147
Get only (drain)		165	165
Total			312
Put and get	124	132	256

Other Queue Manager Performance improvements- Put to waiting getters Notes

- **For out-of-sync point non persistent messages a putting application can put a message directly into the buffer for an application which issued a get with wait request. The ECB is then posted, and the getting application can continue with the data.**
- **The message has to match, that is the msgid and correlid, and there has to be space in the users buffer for the message.**
- **For other cases the messages it put onto the queue, and the ECB for the getting application is posted. The getting application will re-issue the get request and get the message.**
- **The statistics for a get are collected as the request goes into and out of the queue manager. When the optimised put/get occurs there is no call to the queue manager, so the accounting info cannot be collected for the getter.**
  - Field PUT1PWG and PUTPWG in Queue Accounting for the putter is incremented.
- **Note: Put to waiting getters can alter workload distribution in a shared queue environment.**
- **'Diagnostic' APAR PK55496**
  - Turns off Put to Waiting Getter via service parm – contact L2.