#SHAREorg

Session 11453: They Came from Across the Pond: Performance Programming for CICS with WMQ

Lyn Elkins – elkinsc@us.ibm.com

IBM Advanced Technical Skills

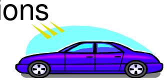
SHARE in Anaheim 2012

Agenda

- What is performance to me?
- Performance reports
- Performance Inhibitors
- Building Performance in applications
- Workload skewing
- Summary

SHARE in Anaheim 2012

What is performance to me?



- Performance can mean different things:
  - Meeting sub-second SLAs on critical transactions
    - In spite of workload fluctuations
  - 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

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"
  - 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

WMQ for z/OS Performance Reports

- Published after major releases
  - SupportPac MP1H available for WMQ V7.1
  - SupportPac MP1G available for WMQ V7.0.1
  - 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

WMQ 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 may contain more 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.

Building Performance into the applications

- Using the right queues
- Using the right verbs
- Using the right release
- Using the right hardware

Using the right queues

- Creating and using queues with the right characteristics is key to good application performance
  - Dynamic queues
  - Queue Indexes
  - Shared vs private queues

WMQ 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	335%
Put	106	115	-8%
Inquire	17	18	-5%

WMQ Application Performance - Queues

- Choose the right queue - continued:
- The Suspend count comparison

Verb	TDQ	Permanent	Difference
Open	727	0	Divide error
Close	0	0	0
Put	0	0	0

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.C9422A60F4306075                             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
```


Alignment Page

Choosing the right queues - Permanent Queues

This information was taken from the SMF116 class 3 records

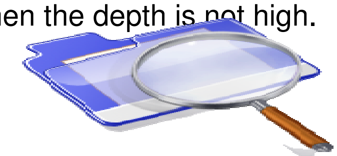
```
== Task token : 12-03-2012 21:24:23.42,  55FE03F0,  55FD0000

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  10-10-2019 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        10        10
Maximum depth encountered          3
```


Choosing the Right queues - Temporary Dynamic Queues - Notes

- The data shown is taken from one of the sample SMF print programs available with 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.

WMQ Application Performance - Queues

- Queue index specification is unique to WMQ 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.

WMQ 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, one of the MQ SMF print programs provided in 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 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%

Alignment page

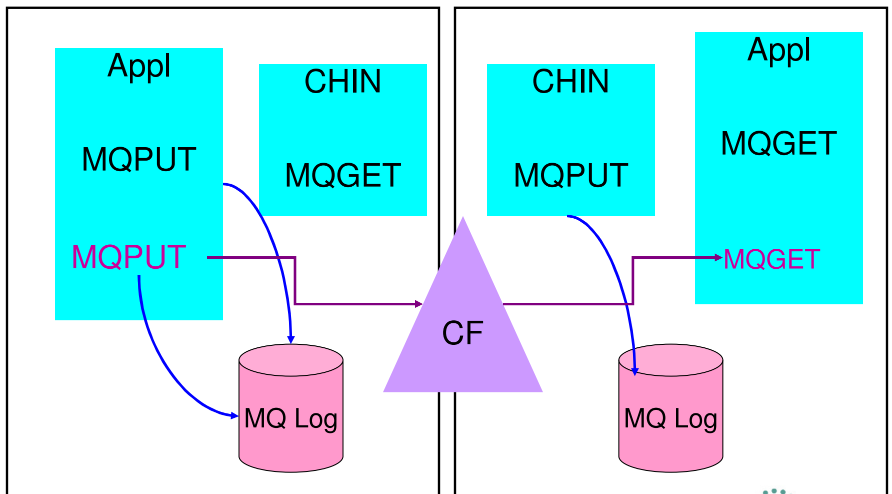
Shared vs Private queues

- Availability can trump pure performance
  - CPU usage is always higher with shared queues
    - But messages (and the queues) are much more available
    - Typically better throughput on QMGR to QMGR communication in the same ‘plex
    - Can eliminate multiple logging for persistent messages

Shared vs Private queues - Notes

- Availability can trump pure performance
  - CPU usage is always higher with shared queues
      - *On a message by message basis, the CPU usage is higher for shared queues.*
    - But messages (and the queues) are much more available
      - *Queues and Messages are on the coupling facility*
    - Typically better throughput on QMGR to QMGR communication in the same ‘plex
      May eliminate multiple logging
      - *Persistent messages that are ‘stored and forwarded’ between z/OS queue managers can benefit quite a bit*
-

Multiple Logging



27

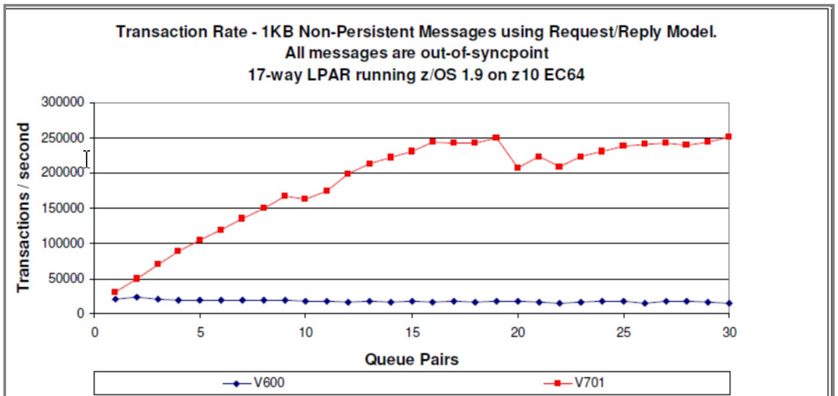
Multiple logging notes

- When an application is using private queues to communicate between two z/OS queue managers, double logging is done for persistent messages.
  - A message is logged at the initial MQPUT
  - When passed across the channel, the MCA for the receiving queue manager PUTs the message and the entire message is logged again
- If a shared queue is used, the MQPUT is logged at the originating queue manager
- Other log records are written as part of normal processing, but the largest log message (where the entire message is written to the log) has been performed twice.

28

Using the right Release - Small Message Storage improvements

- In version 7.0.1 changes were made to the way small messages are stored internally by the queue manager.



29

Complete your sessions evaluation online at SHARE.org/AnaheimEval

Small Message Storage – WMQ V7.0.1 - Notes

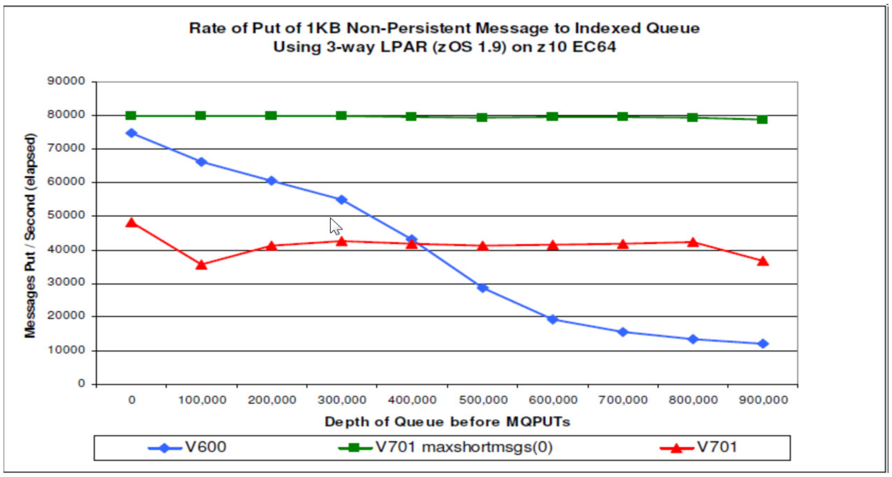
- From SupportPac MP1G:
  - Prior to version 7.0.1, small messages were stored such that multiple messages could co-exist on the same page. This meant that the scavenger could not run once a message was deleted as there were potentially other messages still on the page.
  - Instead, the small message 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**)

30

Complete your sessions evaluation online at SHARE.org/AnaheimEval

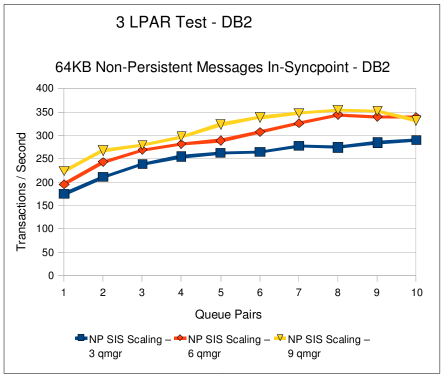
Queue Manager Performance Improvements – Indexing – WMQ V7.0.1

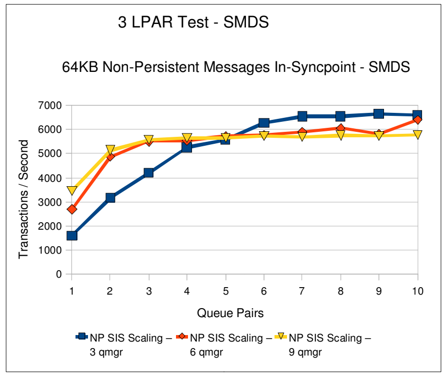


Queue Manager Improvements – Indexing Improvements – WMQ V7.0.1

- 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.

Choosing the right release - Shared Message Data Sets with WMQ V7.1





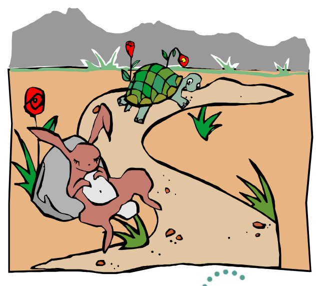
- Early Test Results on z196
- Tests show comparable CPU savings making SMDS a more usable feature for managing your CF storage
- SMDS per CF structure provides better scaling than DB2 BLOB storage

Queue Manager Performance – SMDS

- In WMQ v7.1 Shared message data sets were introduced
- Messages larger than 63K can be offloaded to VSAM linear datasets, rather than DB2 Blobs.
- The thoughput rates increased dramatically, and CPU usage improved.
- In addition, users can now choose the offload size.

WMQ Application Performance

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



Alignment Page

WMQ 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**

WMQ Application Performance – Message Style

- The CPU cost comparison

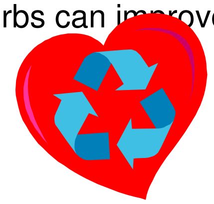
Verb	Persistent	NonP	Difference
Open	125	38	238%
Close	111	26	327%
Put	104	113	-8%
Inquire	17	18	-5%

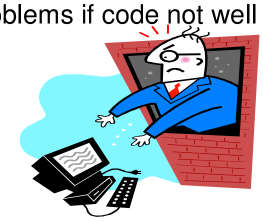
- The Elapsed Time comparison

Verb	Persistent	NonP	Difference
Open	850	39	2079%
Close	113	26	337%
Put	106	115	-8%
Inquire	17	18	-5%

WMQ 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





39

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

40

Effect of MQPUT1

- Each MQPUT1:
  - 117 ųs CPU, for a total 351,000 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
```

- Each MQPUT:
  - 72 ųs CPU, for a total of 216,000 ųs

```
PUTs: Valid      3000,  Max size        80,  Min size       80,  Total bytes 240000
-MQ call-           N         ET        CT       Susp       LOGW       PSET Epages   skip expire
 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 MQ116S report is the average time, not the total time.

Effect of MQPUT1

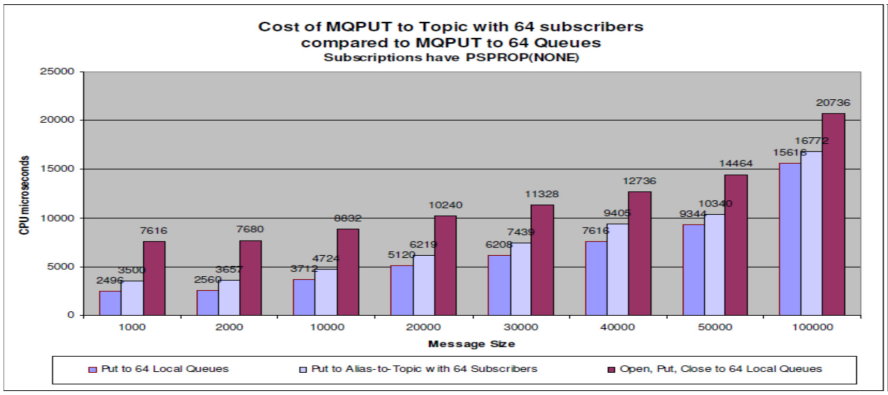
- For one PUT it is less expensive to use an MQPUT1
  - MQPUT1 - 117 total ųs
  - MQPUT - 171 total ųs
- For two PUTs it is less expensive to use an MQOPEN, MQPUT and MQCLOSE
  - MQPUT1 - 234 total ųs
  - MQPUT - 213 total ųs
- Draw your own conclusions

MQPUT vs MQPUT1 comparison - Notes

- In one particularly good example of this, ATS was 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.

Choice of WMQ Verbs – Pub/Sub

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



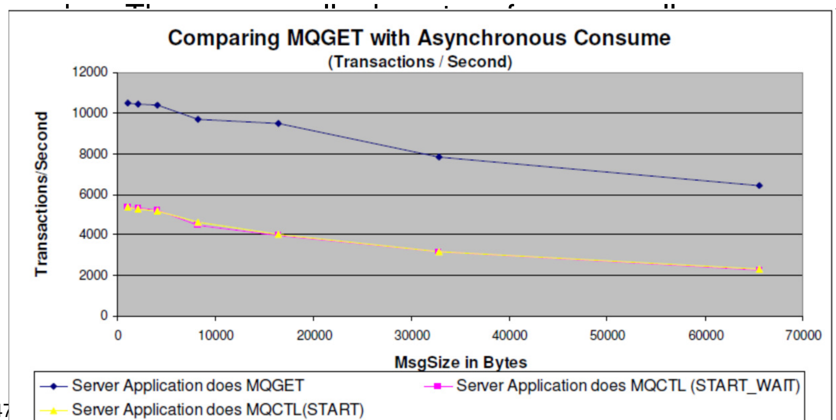
Choice of WMQ Verbs – Pub/Sub - Notes

- 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 WMQ Verbs – MQGET vs Async Consume

- WMQ V7 introduced the Asynchronous Consumer, the MQCB (register) and MQCTL (start and stop the action)



Choice of WMQ 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.

Workload Skewing

- Ideally messages are delivered evenly across a Q'plex
- That does no always happen
  - Workload distribution is skewed to one or more 'favored' queue managers or CICS regions in some shared queue environments.
  - This can be fairly dramatic, and can impact the software costs.
- There are usual suspects……..

Workload Skewing – Put to Waiting Getter

- P2WG:
  - Implemented in WMQ V6, a performance enhancement
  - Out of syncpoint, nonpersistent messages are put directly to a waiting application buffer
    - No notification, no I/O, nothing
  - Reduces CPU by 15-20%
- But
  - If messages arrive at a queue manager that has applications with waiting buffers, they will be processed locally
  - No distribution to other consumers until saturation

Put to waiting getters

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.

Workload Skewing – Local Notification

- Messages destined for shared queues can benefit from 'local notification'
  - The queue manager where messages are received often gets notification of availability before other queue managers in the QSG
  - Takes advantage of a shorter path length

Workload Skewing – Local Notification

- There is a way to turn this off via a service parm

Workload Skewing – Other contributors

- CF 'Favoritism'
  - Skewing may be the result of faster links or proximity to the CF hosting the structures
  - Observed a 40% skew at one customer
- Asymmetrical Sysplex
  - One or more LPARs are on a more powerful engine
- WLM influence
- More transaction instances

Workload Skewing – Other contributors

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
  - 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 WebSphere MQ for z/OS
- Coming soon (we sincerely hope at any rate!)
  - Updates to MP16 and MP1B

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?rs=171&uid=swg24005907&loc=en_US&cs=utf-8&lang=en
  - 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

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00					Free MQ! - MQ Clients and what you can do with them
09:30	Clustering – the easier way to connect your Queue Managers	MQ on z/OS – vivisection	The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation		**They Came from Across the Pond: Performance Programming for CICS with WMQ**
11:00		Diagnosing problems for Message Broker	Lock it down - WebSphere MQ Security	Using IBM WebSphere Application Server and IBM WebSphere MQ Together	Spreading the message – MQ pubsub
12:15	Highly Available Messaging - Rock solid MQ	Putting the web into WebSphere MQ: A look at Web 2.0 technologies	The Doctor is In and Lots of Help with the MQ family - Hands-on Lab		
01:30	WebSphere MQ 101: Introduction to the world's leading messaging provider	What's new in the WebSphere MQ Product Family	Extending IBM WebSphere MQ and WebSphere Message Broker to the Cloud	MQ Performance and Tuning on distributed including internals	
03:00	First steps with WebSphere Message Broker: Application integration for the messy	What's new in Message Broker V8.0	Under the hood of Message Broker on z/OS - WLM, SMF and more	The Do's and Don'ts of z/OS Queue Manager Performance	
04:30	The MQ API for Dummies - the Basics	What the **** is going on in my Queue Manager!?	Diagnosing problems for MQ	Shared Q using Shared Message Data Sets	
06:00			For your eyes only - WebSphere MQ Advanced Message Security	MQ Q-Box - Open Microphone to ask the experts questions	