Guide MQ du 10/03/2015

WebSphere MQ Internals for Best Application Performance

Origine :

- Présentation IBM Impact 2013 : WebSphere MQ Internals Deep Dive for Best Application Performance - session 1997
- Présentation IBM InterConnect 2015 : IBM MQ Better Application Performance - session 2279

Speaker : Luc-Michel Demey, Demey Consulting

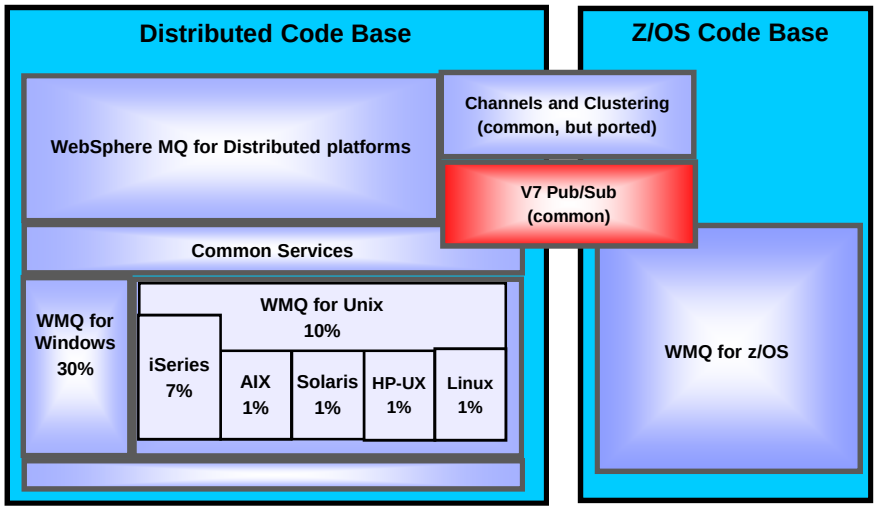
Agenda

- What is distributed WebSphere MQ?
- Structure of the Queue Manager
- Function Walkthroughs
- Channels
- Logging and Recovery
- Other ways to improve application performance

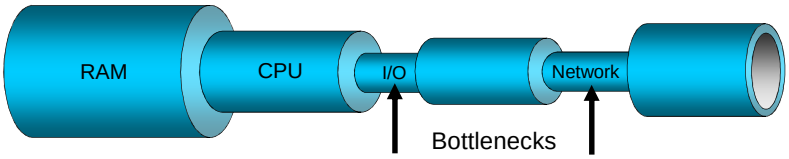
Agenda

- What is distributed WebSphere MQ?
- Structure of the Queue Manager
- Function Walkthroughs
- Channels
- Logging and Recovery
- Other ways to improve application performance

What is Distributed WMQ?



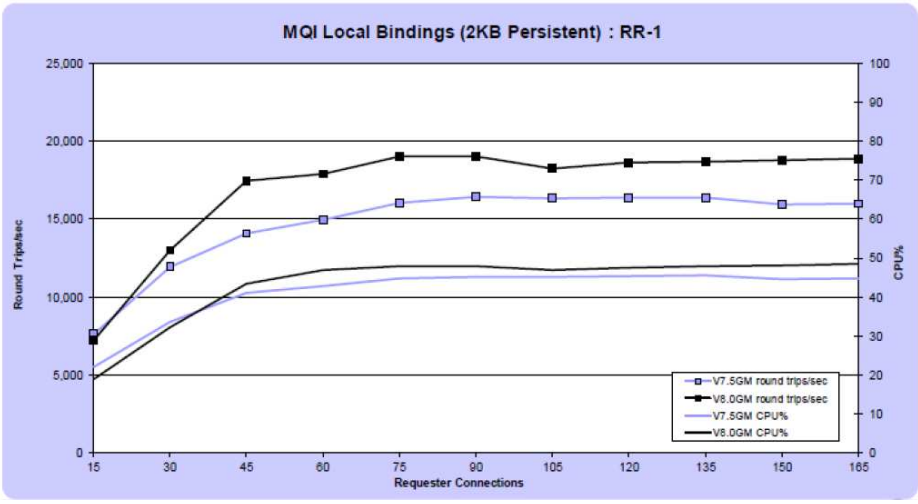
Performance Bottlenecks



- Business Systems are complex
  - Often no single bottleneck limiting performance
  - Performance can mean different things to different people
  - Throughput
    - Scalability
    - Low resource usage
- Not only limited by physical resources
  - Application design, such as parallelism can have major effect
- Performance Reports (from SupportPac site) show range of scenarios

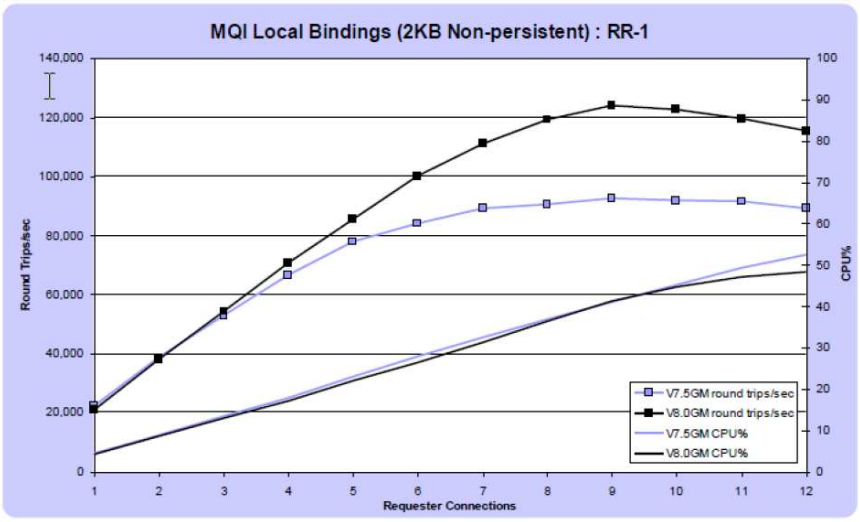
Améliorations MQ V8

- Messages persistants : de 16 K à 19 K messages/s sur Aix



Améliorations MQ V8

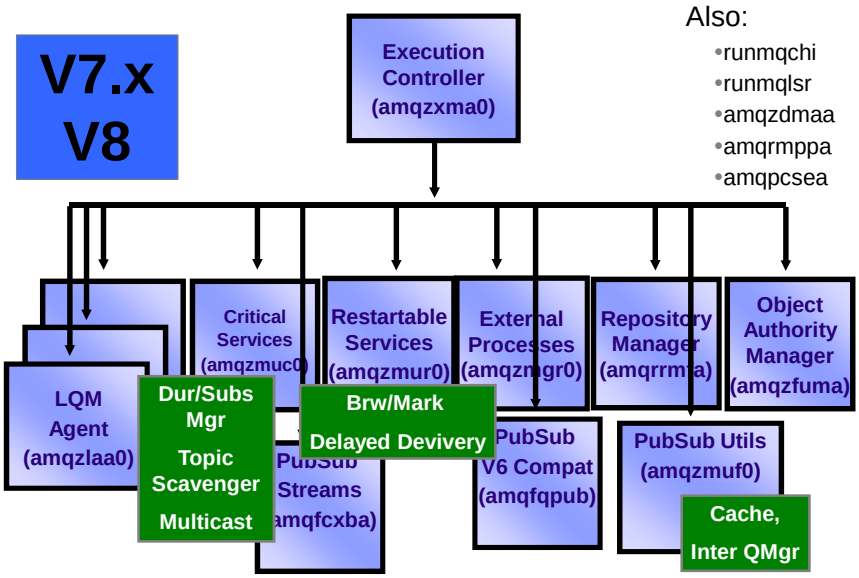
- Messages non persistants : de 90 K à 121 K messages/s sur Aix



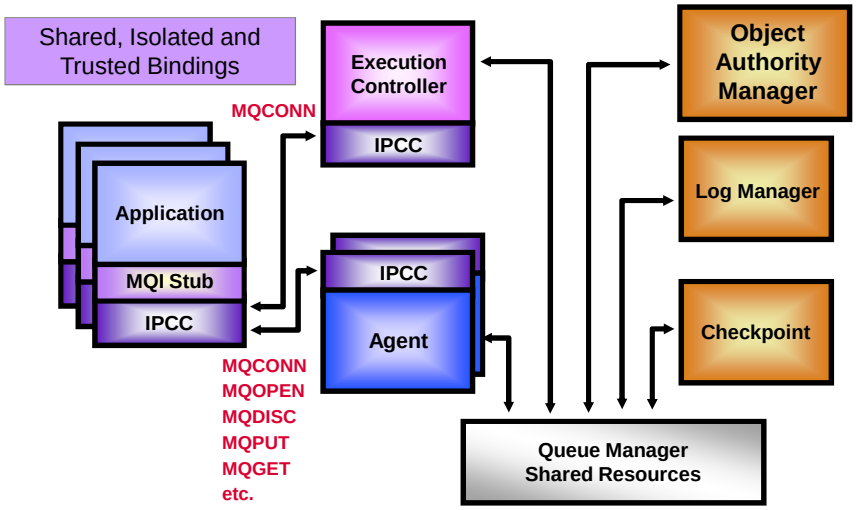
Agenda

- What is distributed WebSphere MQ?
- Structure of the Queue Manager
- Function Walkthroughs
- Channels
- Logging and Recovery
- Other ways to improve application performance

Queue Manager Process Tree



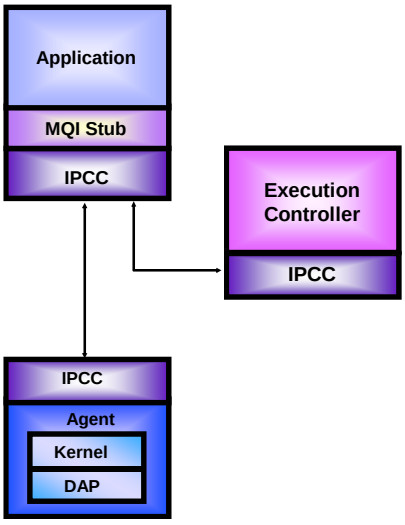
Queue Manager: Process Model



Agenda

- What is distributed WebSphere MQ?
- Structure of the Queue Manager
- Function Walkthroughs
- Channels
- Logging and Recovery
- Other ways to improve application performance

MQCONN



Application (MQI Stub)
- Verify parameters and handles
- Construct a Connect message
- Call API crossing exit
- Send a message to the EC

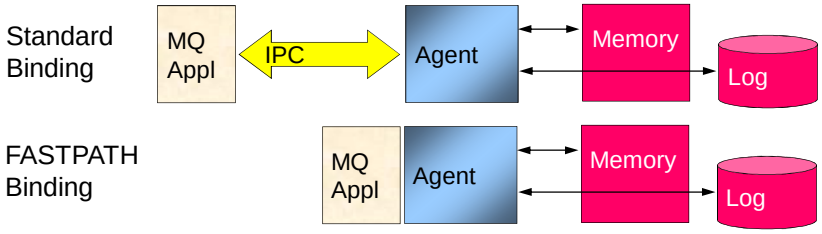
Execution Controller
- Choose an agent or start a new one
- Construct a reply IPCC message
- Return reply to application

Application (MQI Stub)
- Receive the reply
- Construct an IPCC message
- Send the message to the agent

Agent
- Check App's permission to connect
- Allocate and assign agent resources
- Send IPCC reply back to application

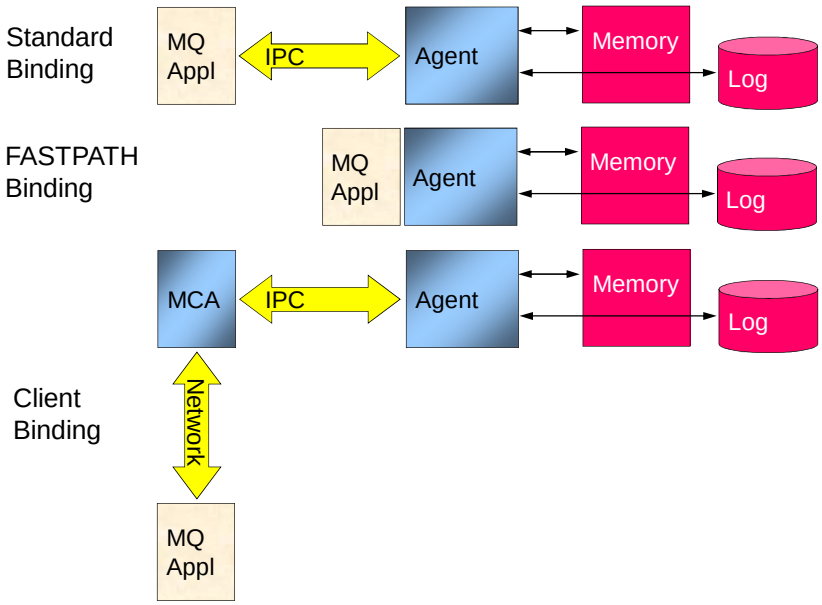
Application (MQI Stub)
- Receive reply and call API crossing exit
- Return HCONN

Performance Implications: Connection Binding

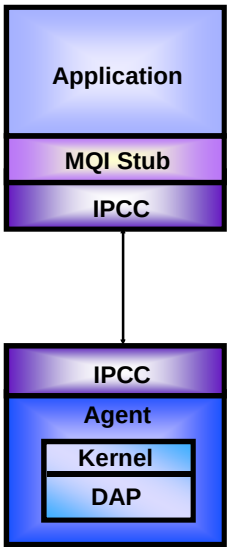


- Fastpath binding removes inter-process communications
  - Implies that the application is 'trusted'
  - MQCONNX option MQCNO_FASTPATH_BINDING
  - Application failure can corrupt queue manager
- Primary benefit is for non-persistent message processing
  - Use for MCAs, Broker
    - ➢ - 30% CPU saving

Performance Implications: Connection Binding



MQOPEN of a queue



Application (MQI Stub)
- Verify hConn
- Call API crossing exit
- Verify parameter addressability
- Place parameters in an Open message
- Send an IPCC message to the agent

Application (MQI Stub)
- Receive reply
- Call API crossing exit
- Return HOBJ

Agent

Application interface
- Verify open parameters

Kernel
- Verify operation validity
- Resolve target – including cluster lookup
- Check permissions on the queue

DAP
- Load the queue ready for gets and puts if required
  - This is the part that can use system resources

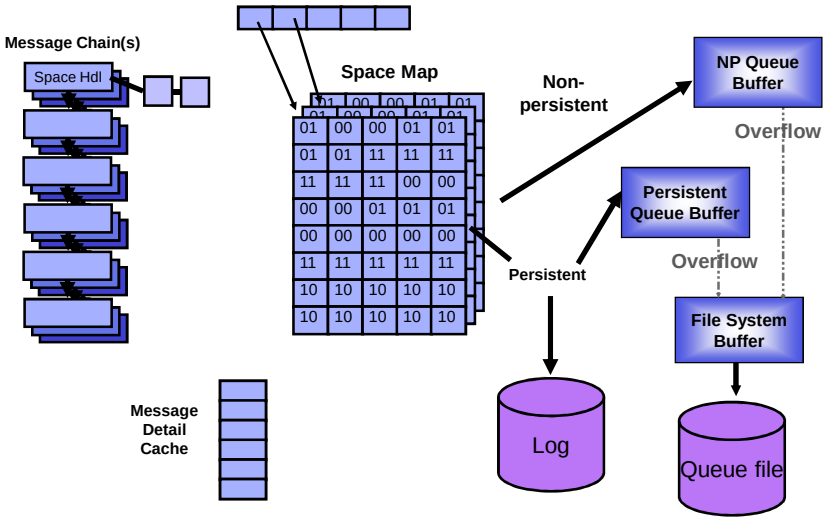
Kernel
- Generate handle to object for application
- Generate responses and event messages

IPCC
- Send reply back to application

Performance Implications: Heavyweight MQI Calls

- MQCONN is a “heavy” operation
  - Don’t let your application do lots of them
  - Wrappers and OO interfaces can sometimes hide what’s really happening
  - Lots of MQCONNs can drop throughput from 1000s Msgs/Sec to 10s Msgs/Sec
- MQOPEN is also ‘heavy’ compared to MQPUT/MQGET
  - Depends on the type of queue and whether first use
    - Loading pre-existing queue; creating dynamic queue
  - It’s where we do the security check
    - Try to cache queue handles if more than one message
  - If you’re only putting one message consider using MQPUT1
    - Particularly client bindings
- Try to avoid exclusive access to the Queue
  - Makes it harder to scale the solution
    - For example adding more instances of application
  - Implies that reliance on message order is not required
    - Partition the data to allow parallel processing?

In the Depths of a Queue



Tuning Queue Buffers

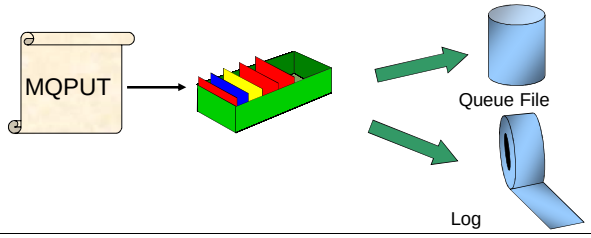
- Increasing buffers can improve performance
  - More information can be kept in memory, without flushing to disk
  - Costs more memory per modified queue
- But no documented external mechanism to do it
  - Performance supportpacs indicate how to do it
  - DefaultQBufferSize / DefaultPQBufferSize
  - SupportPac MS0P (Cat2 – ie “as-is”) includes “QTune” program

```
c:\> java -jar qtune.jar -d c:\mqm\qmgrs\QMA\queues\SYSTEM!DEFAULT!LOCAL!QUEUE

File c:\mqm\qmgrs\QMA\queues\SYSTEM!DEFAULT!LOCAL!QUEUE\q
Stored npBuff     = 64 kB
Stored pBuff      = QMgr default
Stored maxQSize   = 2,097,151 MB
```

Performance Implications: Persistence

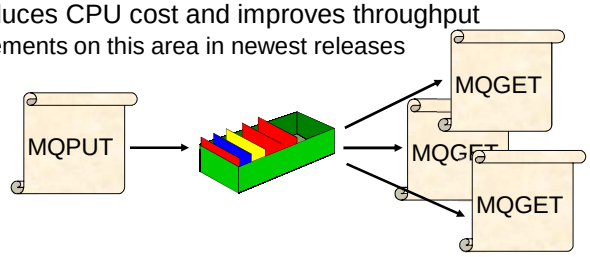
- Log bandwidth is going to restrict throughput
  - Put the log files on the fastest disks you have, separate from queue file
  - Persistent messages are the main things requiring recovery after an outage
  - Can significantly affect restart times
- Why use persistence?
  - False assumption that persistence is for "important" data and nonpersistent for when you don't care
  - The real reason for persistent messages is to reduce application complexity
  - With persistent, apps do not need logic to detect and deal with lost messages
  - If your app (or operational procedures) can detect and deal with lost messages, then you do not need to use persistent messages



Guide MQ du 10/03/2015 WebSphere MQ Internals for Best Application Performance 27

Put to a waiting getter

- MQPUT most efficient if there is getting application waiting
  - Having multiple applications processing queue increases percentage
  - May not appear 'balanced' – May keep one 'hot'
- Only for out of syncpoint messages
  - Both persistent and non-persistent
  - If Persistent outside of syncpoint, think carefully about why using persistence
    - Got message could be lost if crash before returning to the application!
- No queuing required
  - Removes a lot of processing of placing the message onto the queue
- Significantly reduces CPU cost and improves throughput
  - Lots of improvements on this area in newest releases



Guide MQ du 10/03/2015 WebSphere MQ Internals for Best Application Performance 29

Performance Implications: Syncpoint

- Do you need it?
  - Yes, when a set of work needs to either all be performed, or all not performed
- Maximum Size of UOW can be limited
  - QMGR MAXUMSGS parm
  - Set to sensible value to avoid runaway applications
- Make sure you keep the size of your UOWs small
  - Don't forget to end the UOW
- Cheaper to process in syncpoint for persistent messages
  - Up to a point, not huge UOWs
  - Log not forced after every MQPUT/MQGET
- Useful even when only a single message inside syncpoint
  - And running multiple parallel applications

Good Application Design - Summary

- Long-running transaction
- Open queues up-front
- Use syncpoint for persistent operations
- No message affinities so multiples instances can rue in parallel

Publish/Subscribe Implementation in V7

- MQOPEN, MQPUT, MQGET very similar to point-to-point
  - Includes cluster resolution
  - Need to find closest admin topic node
  - Internal subscribers may forward publication to another queue manager
- Durable Subscriptions held on SYSTEM.DURABLE.SUBSCRIBER.QUEUE
  - Multiple subscriptions consolidated into single message
  - Why is there no non-durable subscriber queue?
  - Retained publications also stored on a queue
- Handling application abend
  - V6 cleanup for non-durable subs was “automatic” for JMS, manual otherwise
  - Automatic for V7+
- Managed destinations
  - Agent creates queue in MQSUB - trace shows internal MQOPEN (kqiOpenModel)
- Parallel match-space access via shared memory set
  - Several applications can publish simultaneously on the same topic
  - Lock held during subscribe/unsubscribe processing

Message Processing in V7

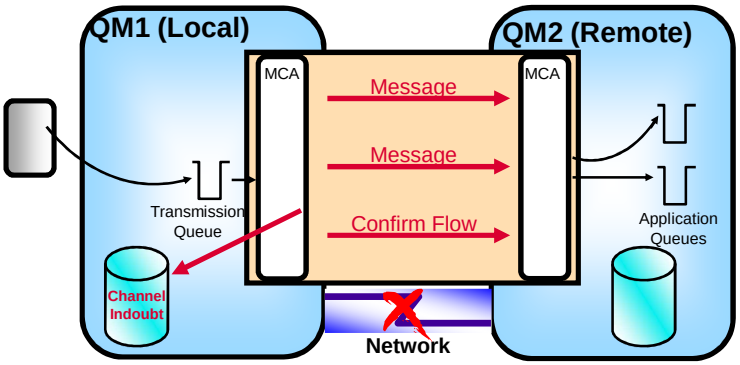
- Persistent pubs switch to non-persistent-ish for non-durable subscriptions
  - Does not change the reliability level
  - Messages are not logged, but they keep the “persistent” flag
  - Improves performance
- Properties stored as part of the message
  - Logged for persistence, rcdmqimg etc
  - Written to disk in either RFH2 or an “internal” format
  - Converted to application-required format during MQGET
- Selectors on queues can cause all messages to be browsed
  - Queue lock may be held during selection

Agenda

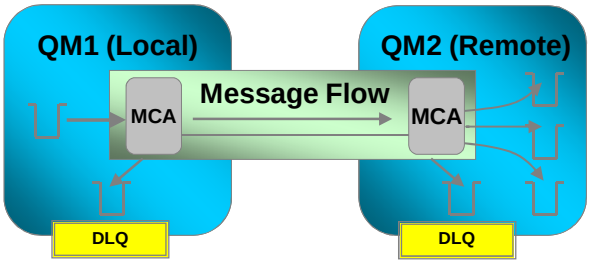
- What is distributed WebSphere MQ?
- Structure of the Queue Manager
- Function Walkthroughs
- Channels
- Logging and Recovery
- Multiple Installation Support
- Other ways to improve application performance

Assured Delivery

- Channel synchronisation uses ScratchPads
  - The SYNCQ was retained to hold channel status across restarts
  - A small area of data which can be part of 2-phase commit processing
  - Channel sync also uses file AMQRSYNA.DAT as an index into the scratchpads
  - Messages in an in-doubt batch cannot be reallocated by clustering algorithm

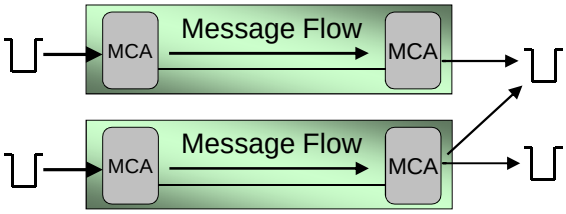


Channel Protocol



- Send from XmitQ until Batchsize/limit or Empty & Batchint expired.
- Store 'indoubt' record and send 'End-of-Batch' to Remote MCA.
- Remote MCA updates Batch Sequence, MQCMIT, sends ack
- Local MCA updates Batch Sequence number and issues MQCMIT.
- Pipeline Length =2 provides additional thread that will start processing next batch after 'End-of-Batch' sent to Remote MCA

Performance Implications: One or Multiple Channels

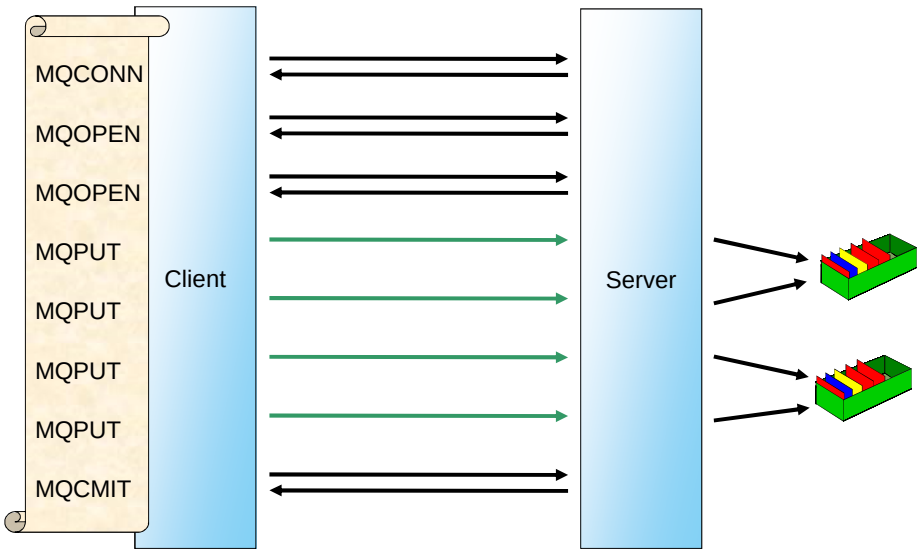


- Use one channel if it can handle the throughput
  - Monitor depth of XMIT queue
  - One high-use channel is more efficient than two low-use channels
    - The actual batch size will be larger
  - Multiple channels can yield some performance benefit
    - Depends on network and arrival rate
- Multiple channels for separate classes of work
  - Large messages only delay large message
  - Encryption cost on taken on worthwhile messages
  - Small interactive messages do not get delayed

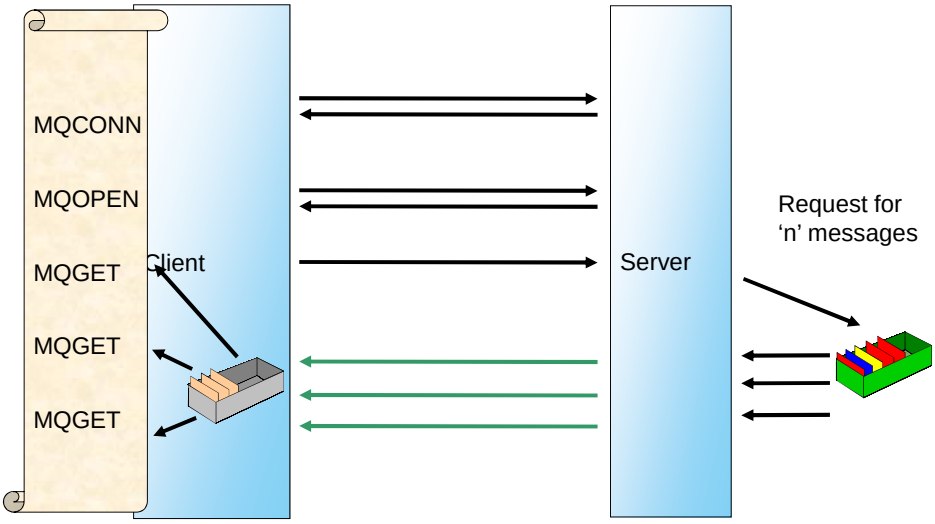
Clients in V7

- Many changes to the client protocols in V7
  - All can improve performance
- Multiplexing or Shared Conversations
  - For multi-threaded applications
  - Several connections use the same socket
- Asynchronous Put
  - For sending applications
- Read-ahead
  - For receiving applications
- New threads inside application for full-duplex comms
  - Sharecnv(0) – May be fast but no full-duplex so miss good functionality
  - Sharecnv(1) – One socket per connection, may be faster than Sharecnv(10)!
  - Sharecvn(10) – Shared socket for multiple conversations

Asynchronous Put Response



Read-ahead of messages



Agenda

- What is distributed WebSphere MQ?
- Structure of the Queue Manager
- Function Walkthroughs
- Channels
- Logging and Recovery
- Other ways to improve application performance

WebSphere MQ Objects

- Recoverable entities known by the LQM
  - Queue, Process, Queue Manager, Channel etc definitions
  - Scratch Pads
- Objects have security control information
  - Attempts to access them are mediated by the OAM
- Information is stored in Object files
  - May be part of other data in same file
  - Queue File contains messages and attributes
- Topics are objects, but Subscriptions are not
- Object Catalog points at object files
  - dspmqfls

What's the point of logging?

- A log record is written for each persistent update
  - The log record describes the update
- Optimisations to minimise serialisation points
- Write-Ahead Logging
  - The log is always more up-to-date than the actual data
- Log is a sequential file
  - Sequential I/O is much quicker than random
  - Single point of writing rather than to individual object files
- Log and actual data are reconciled during strmqm
  - Progress information displayed
- Point of consistency – Checkpoint
  - Log control file: amqhlctl.lfh – in log directory
  - Checkpoint amqalchk.fil – qmgr directory
  - Backup queue managers with WMQ V6

Looking at logger performance

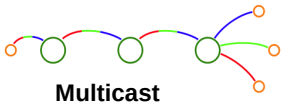
- Can extract internal information from a service tool
  - Lots of MQ performance PMRs turn out to be disk-related. So recording was added to the internal state
    - amqldmpa –c H –m <qmgr> -d 8 –n <count> -s <interval> -f <file>
  - The amqldmpa program can dump lots of other internal information too
- Includes logger I/O activity
  - WriteTimeMax shows maximum time (microseconds) to complete I/O
  - WriteSizeMax shows largest (bytes) I/O
  - Since qmgr started
- Maintains averages
  - WriteSizeShort is short-term (64-sample) weighted average of recent writes
  - WriteSizeLong is longer-term (1024-sample) weighted average
  - Similarly for WriteTimeShort/Long
- From one PMR:
  - WriteTimeMax = 59102377, WriteSizeMax=2097152
  - So it has taken nearly 60 seconds to write 2MB
  - Implies they need to talk to disk support team!

Agenda

- What is distributed WebSphere MQ?
- Structure of the Queue Manager
- Function Walkthroughs
- Channels
- Logging and Recovery
- Other ways to improve application performance

Multicast – publish to multiple receivers

- Single message is duplicated in the network
  - Receivers register interest on specific IP addresses
  - Sender send datagrams to the multicast address
- Network cards/ routers make copies of data and send to receivers who have registered for an address
- Uses normal MQI with some restrictions
  - No persistence nor transactionality
  - No durable subscribers
  - No message segmentation nor grouping
  - Pub/Sub only



Multicast

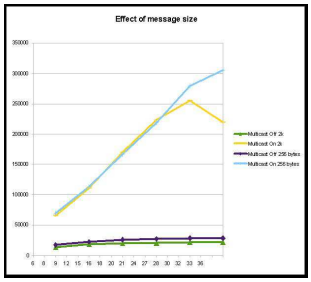
Multicast – One message cloned

Unicast

Unicast – Multiple messages throughout

Multicast - What are the benefits

- Low latency
  - Much higher volumes than standard non-persistent messaging
  - Messages do not pass through qmgrs, and peer to peer communication
- High Scalability
  - Additional subscribers cause no slow down
  - Reduced network traffic
- 'Fair delivery' of data
  - Each subscriber 'sees' the data at the same time
  - Multicast offers near simultaneous delivery
- High availability
  - Multicast uses the network so no pub/sub engine to fan-out data
  - Reduces load on Queue managers servers



MQTT (Telemetry, Extended Reach, Mobile)

- Provides support for the MQTT protocol
  - Ideal for small or embedded devices
    - mobile devices, smart meters, set top boxes, remote telemetry units
  - Typically used for infrequent, small
  - Does not use the MQI
- Supports 3 Quality of Services
  - 0 - At most once (fast but unreliable)
  - 1 - At least once (duplicates possible)
  - 2 - Exactly once (slower but assured)
- Ideal for large numbers of connections with low message rates
  - Tested with up to 100,000 clients on Linux, 64,000 on Windows

Summary

- Common code for multi-platform delivery
- Process isolation for integrity
- Persistent information safely stored on disk
- High Performance through Concurrency
- Newer capabilities significantly improve specific scenarios