17890: Are z/OS & distributed MQ platforms like oil and water?

Mark Taylor
marke_taylor@uk.ibm.com
IBM Hursley

Lyn Elkins
elkinsc@us.ibm.com
IBM ATS

Introduction

- One objective of MQ is isolating apps from needing to understand platforms
  - There is a common API that can be expressed in many languages
- Another objective is to have (reasonably) common operational model
  - Much of admin is the same on all platforms

- But it's not all the same
  - One dichotomy has always been whether to be natural to MQ-ness or behave like other things on the platform
  - Some features don't make sense on some platforms
    - For example, .Net interface is only on Windows
  - Some features have not been implemented everywhere for other reasons

- So there are differences, and that is what this presentation will cover
- This is based on V8

Code Streams

- There are essentially two implementations of MQ from Hursley lab
  - z/OS
  - Distributed (Windows, Unix, Linux, i)
  - There are some further subspecies variants like VSE or NSS
- Within Distributed implementation, there are some platform unique features
  - But we won't discuss those here
  - Most platform-unique code abstracts OS facilities like locking or NLS or threads
- In the early days, some code was written for one and then "ported"
  - In particular, the channel code
  - Meant double-fixing, and re-porting for each release
- Internal architecture (eg tasks, threads) very different
  - But we won't discuss much of that. Understanding externals is more important
- Since V7.0, some code is truly common
  - Just one copy of the source part shared between both
  - New features generally use common code where feasible

Sections

- Setting up
- Application Programming
- Administration

How this presentation works

- Lyn will talk about z/OS in this color

- Mark will talk about Distributed in this colour

Setting Up

Getting started

- Lots of differences in initial installation and setup
- Getting the code on the box is part of the job
  - MQ uses native installation techniques for all platforms
  - Needs a suitably-authorised person to do that installation
    - SMPE for z/OS, installp for AIX, rpm for Linux etc
- But other differences primarily due to
  - Security
  - Storage
- Share philosophy of needing no more features than is found on the system
  - So no prereq software for core capabilities of MQ
  - Product ships components that are needed such as gskit
  - But can exploit things that we know are there
    - For example, on z/OS we use the system-provided SSL
  - Some extended capabilities may have additional prereqs
    - Shared Queues need DB2

Security

- On Distributed, MQ implements its own authorisation mechanism
  - There is no generally-accepted standard interface on these systems
- And relies on the existence of certain userids
  - There are differences even between individual platforms

- On z/OS, MQ exploits the common authorization interface, SAF
  - And so the z/OS security administrator has to be involved
  - Define the profiles etc.

- Will look more at security later on

Storage (Distributed)

- On Distributed, MQ uses directories such as /var/mqm/qmgrs and /var/mqm/logs
  - The system administrator will probably allocate filesystems and mount them
  - These days, may have separate SAN administrator
- Each queue has its own file within the filesystem
  - To store the message data
  - Each queue could hold 1TB
- Queues do not interfere with each other's storage requirements
  - Subject to max size of filesystem
- Logs can be LINEAR or CIRCULAR
  - Choice made when qmgr is created
  - With linear logging, you then need a job to remove old log files
  - MQ does not directly implement dual-logging; relies on RAID filesystems

Storage (z/OS)

- Queues are handled via pagesets and bufferpools
- Multiple queues may use the same pageset and bufferpool
  - Can lead to storage contention
  - V8 increases number of bufferpools to match number of pagesets
- No direct equivalent of circular logging but constraints can be applied to achieve a similar effect
  - Semi-circular?
  - Active logs are ‘almost like circular’, with offloading to archive logs
- Logs are managed via the BSDS
- MQ understands and implements Dual Logging
- Tool provided to format and extract messages from log

Shared Queues

- A z/OS-unique feature
  - Multiple queue managers can see the same queue
  - Continuous processing of messages from a queue even when one LPAR fails
- Relies on the Coupling Facility hardware
  - And relies on DB2 Data Sharing
- Results in several unique possibilities
  - Inter-qmgr communication without standard channels
  - Dynamic selection of which qmgr to connect to
- Effects appear in many places
  - For example, single MQSC command can be issued to multiple queue managers giving multiple responses

Extra Features

- MQ V7.5 on Distributed incorporated MFT (nee FTE) and AMS
  - "MQ Advanced" license covers all features

- On z/OS, these are available as separate products
  - V8 improved technical integration but still separate licenses
- On z/OS, use of inbound client connections was restricted
  - Restriction removed in V8 (no longer a CAF)

- Distributed MQ has the MQXR service for mobile (MQTT) clients
  - Not available on z/OS
  - Expected that mobile clients connect via front-end qmgr before hitting z/OS apps

Application Programming

General

- Default codepages and encoding differ by platform
- Always use the header files for your platform
  - Don't be tempted to cross-compile
- Maximum lengths of fields may vary

- Language support may vary
  - Assembler only on z/OS
  - And some APIs: MQAI only on Distributed
- MQI return codes may be different
  - Often because underlying storage mechanisms have different error conditions
  - For example, Coupling Facility errors on shared queues
- z/OS does not have MQ clients
  - Some parameters to some verbs only apply in client environments
  - For example, the MQCD passed during MQCONNX

API - Connections

- MQCONN/MQCONNX
  - Verbs not required for CICS transactions
    - MQHC_DEF_HCONN can be used for subsequent verbs in applications
  - ConnTag is available to control serialization
    - An application (especially an MCA) can tell if another instance of itself is already running
    - On either the same local qmgr or any other in the QSG
  - Group connection to QSG

  - Lots of client-only options for connection
    - MQCD can be specified
    - Reconnect options
  - MQCNO_SHARED options for multi-threaded applications
    - Controls whether an hConn can be (serially) used by other threads in the same process
  - Fastpath binding
  - Control of accounting
    - When accounting information is being collected, some apps may request exclusion

API - Disconnections

- MQDISC
  - Always recommended
  - Rollback when application abends
    - Although definition of "abend" is not clear in every case
    - CICS and IMS do make it clear!
    - A JVM has been known to return OK to the operating system even when the user's code has caused a fatal exception
  - Rollback when not used and application ends

API - Objects

- MQOPEN
  - Default dynamic queue names begin with CSQ.* or AMQ.*
  - Distributed can open multiple queues simultaneously via Distribution List
    - Publish/Subscribe preferred cross-platform model
- MQCLOSE
  - No platform differerences in practice
- MQSET
  - Follows the same rules as MQSC attributes for platforms
- MQINQ
  - Follows the same rules as MQSC attributes for platforms

API - Messages

- MQPUT/MQPUT1
  - Messages can be automatically segmented
    - But Message groups are cross-platform
  - Distributed supports "Reference messages" which can avoid putting large amounts of data on a queue
- MQGET
  - z/OS has "get with signal" to asyncronously notify app when messages appear
    - MQCB is now preferred cross-platform model
  - z/OS has MARK_SKIP_BACKOUT for simpler processing of poison messages
    - Bad messages can be moved to an application-specific DLQ while backing out other resource changes
  - Distributed can get portions of messages via segmentation
- MQSUB
  - No platform differences
- MQSUBRQ
  - No platform differences

API – Flow control

- MQCB
  - Definition of the callback function in MQCBD varies by environment
  - eg C function pointer, CICS program name

- MQCTL
  - Not in IMS adapter
  - In CICS, cannot use MQOP_START – use MQOP_START_WAIT
  - On z/OS, apps must be authorized to use USS to use MQOP_START

- MQSTAT
  - Client applications only
  - But usable regardless of server platform

API - Properties

- MQDLTMP
- MQBUFMH
- MQCRTMH
- MQDLTMH
- MQMHBUF
- MQSETMP
- MQINQMP

- No platform differences

API - Transactions

- MQBEGIN
  - Only available on Distributed
  - z/OS always has a transaction manager available
- MQCMIT
  - On all platforms when not running under external TM
- MQBACK
  - On all platforms when not running under external TM

- Default for MQ transactional behaviour is different
  - MQI on Distributed assumes NO_SYNCPOINT
  - MQI on z/OS assumes SYNCPOINT
  - Always specify syncpoint options on MQI calls
- Environments for two-phase transactions differ
  - On z/OS, RRS CICS and IMS are all available for transaction management
  - On Distributed, XA is available as the standard interface
    - And MQ can act as a transaction manager

Exits

- z/OS has API-Crossing exit for CICS
  - But no other environments
- Distributed has API exit for all environments
  - With a very different interface
- Installable Services on Distributed
  - But very few people write these so not too interesting
  - Primarily used for the OAM security module
- Channel send exit – ExitSpace field
  - Used to reserve space in network transmission buffers for send exits
  - Always zero on z/OS
- No publish exit on z/OS
- z/OS exits have MQXWAIT
  - Necessary because process/thread model for channels is different

Administration

Object Definitions

- Attributes and ini files
  - Some items are queue manager attributes on one platform but not other
  - z/OS has lots related to its storage

- Some unique object types
  - z/OS has STGCLASS and CFSTRUCT
  - Distributed has SERVICES and COMMINFO

- Startup
  - CSQZPARM is assembled/linked and other inputs run during startup
    - Reset configuration, define default objects etc
  - On Distributed, standard objects are created by qmgr creation and updated during migration

Object Attributes – Queue Manager

- Apart from shared queue and storage-related attributes …
  - Various events differ as shown in other charts
  - Some z/OS attributes are in qm.ini for Distributed
- z/OS only
  - ACTCHL, MAXCHL
  - ADOPTCHK/ADOPTMCA
  - CHIDISPS, CHIADAPS
  - DNSGROUP, DNSWLM
  - EXPIRYINT
  - GROUPUR
  - IGQ, IGQAUT, IGQUSER
  - LSTRTMR,LU62ARM, LU62CHL, OPORTMIN, OPORTMAX, RCVTMIN, RCVTTYPE, TCPNAME, TCPKEEP, TCPSTACK
  - SCYCASE
  - SSLTASKS
  - TRAXSTR, TRAXTBL
- Distributed only
  - ACCTCONO, ACCTINT, ACCTMQI, ACCTQ
  - ACTIVREC
  - CCSID
  - CERTVPOL
  - CHAD
  - SCHINIT, SCMDSERV

Object Attributes – Queues and Channels

- Apart from shared queue and storage-related attributes …
- Queue
  - DEFTYPE(SHAREDYN) z/OS
  - HARDENBO only effective on z/OS
  - INDXTYPE z/OS
  - DISTL Distributed
  - NPMCLASS Distributed
  - SCOPE Distributed only, but obsolete
  - STATQ Distributed
  - TriggerData for transmission queues can be blank on Distributed, names channel on z/OS
- Channel
  - CONNAME 48 characters on z/OS, 264 elsewhere
  - Format of exit names and exit data is platform-specific
  - STATCHL Distributed only before V8
  - PUTAUT ONLYMCA/ALTMCA
  - KAINT only effective on z/OS

Queue Manager operations

- Message Expiry
  - z/OS has explicit config for timing of task to remove expired messages
  - Distributed has a similar task but no documented configuration
- Security Cache Scavenger
  - z/OS has parameters to control authority cache lifetime
  - No equivalent on Distributed; use REFRESH SECURITY explicit command
- Storage Scavengers
  - z/OS has tasks to release bufferpool and pageset storage
  - Distributed will release queue file storage at intervals
- Queue Indexing
  - z/OS has explicit indexes on queues to assist with retrieval patterns
  - Distributed has hashing to perform similar role but no documented configuration

Intercommunication and Clusters

- Channels are the same
- Clustering is essentially the same across all platforms

- MQ 7.5 introduced concept of multiple cluster transmission queues
  - Made common in V8

Security (1)

- Authentication feature added in V8
  - Distributed supports userid validation in operating system and explicit LDAP
  - z/OS supports userid validation in operating system
  - Both support LDAP when transparently used by OS

- SSL/TLS configuration
  - Distributed (mostly) qmgr uses gskit toolkit
  - z/OS qmgr uses System SSL
  - Can lead to discrepancies in crypto algorithms supported
  - Different versions support different algorithms
  - And there may also be client-related discrepancies
    - Java programs rely on JSSE implementations
    - .Net programs (from V8) can use Microsoft inbuilt implementations
    - NSS client uses OpenSSL
  - There is a good common overlap between all of these, but not identical sets
  - Current versions have significantly reduced the sets supported on all platforms

Security (2) – Access Control

- z/OS
  - Uses system-provided interface for authorization
    - SAF is common API to RACF, Top Secret, ACF2
  - Has to work with the 4 permissions available in SAF
  - No distinction between PUT and GET
    - Often alias queues are used to isolate permissions
  - Granular control of "impersonation" (setting context, alt-user)
  - One operation may result in several authorization queries

- Distributed
  - MQ-provided authorisation interface
    - Implemented in the OAM – OS or LDAP-based
  - Many permissions on objects
  - Global controls on impersonation
    - If you have authority to use alt-user, there are no constraints on which user
  - Well-known "mqm" id for full authority

Commands

- Basic OS-level commands are different
  - Create, start, stop, delete queue manager procedures
  - Distributed has command-line interface
  - z/OS has JCL
- Issuing configuration commands like ALTER QLOCAL
  - Distributed has runmqsc shell
  - z/OS has ISPF panels for most commands
  - And the +cpf commands for runmqsc equivalence
  - MQ Explorer is product-provided common GUI
- Common programming interface (PCF) for configuration commands
  - z/OS requires an "extended" format which may have multiple sets of responses
    - Supporting a Queue Sharing Group environment
  - Distributed supports the same format but not the default
  - Differences are hidden in the Java PCF classes

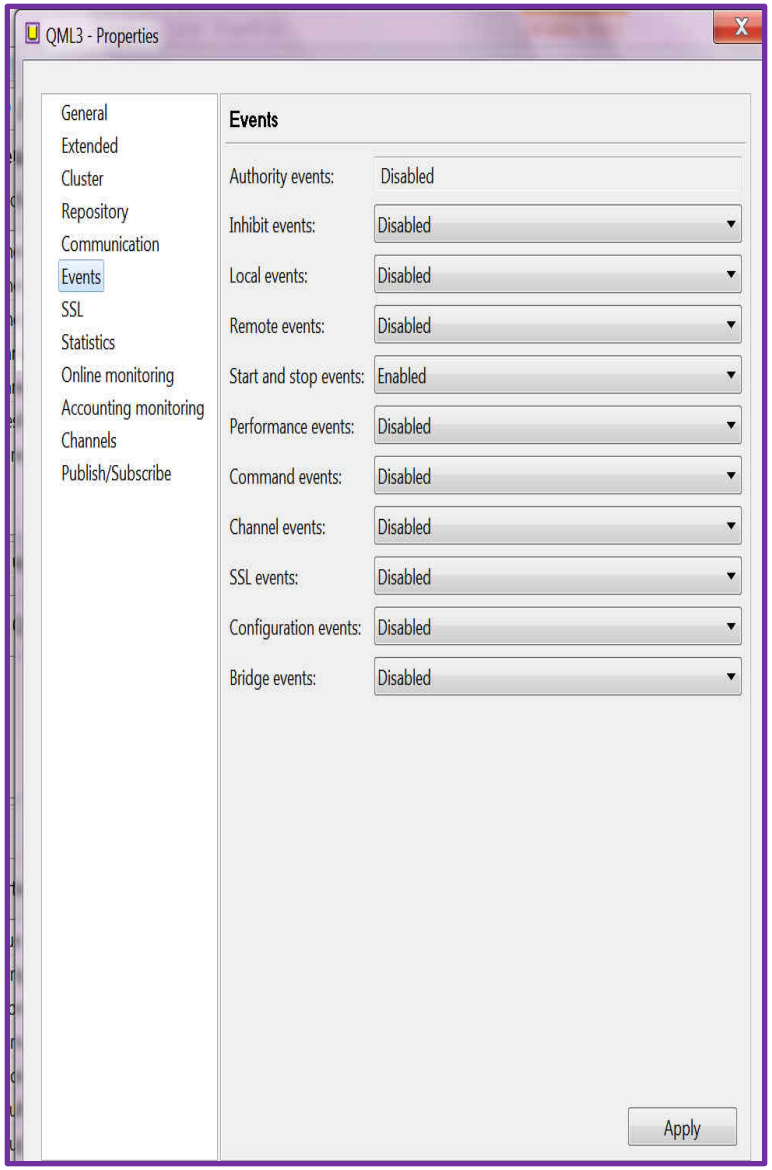
MQSC Commands

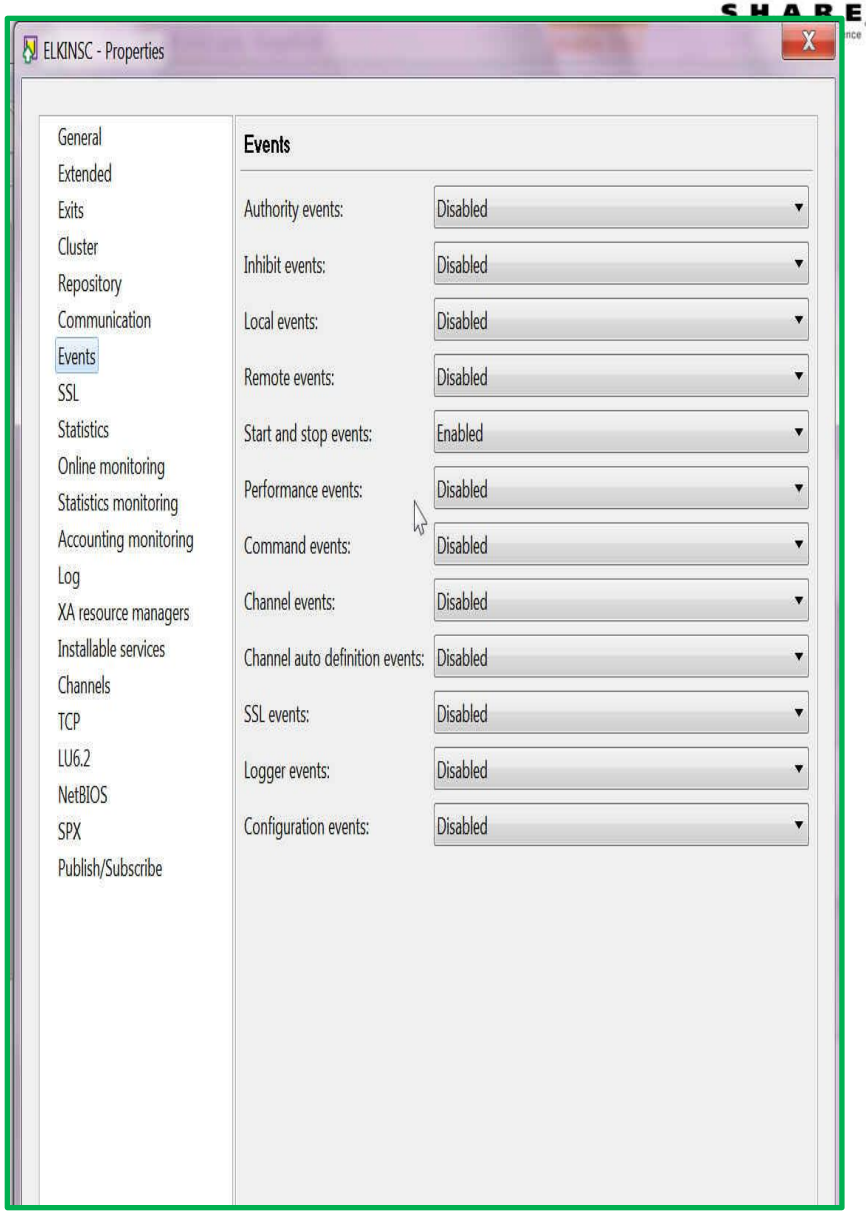
- Some commands not available in all platforms
- RESET QSTATS is only on z/OS
  - PCF available on all platforms
- ARCHIVE LOG is only on z/OS even though Distributed also have logs
- DISPLAY SYSTEM
- MOVE QLOCAL
  - V8 has dmpmqmsg
- DISPLAY QMSTATUS only on Distributed
  - Some items have z/OS specific commands like DISPLAY CMDSERV
- STOP CONN only on Distributed
  - All platforms have STOP CHL to kill client connections
- Client application name shows differently in DISPLAY CONN/QSTATUS
  - Program name shown on Distributed, "CHIN" on z/OS
- Some z/OS MQSC have command-line equivalents on Distributed
  - STOP QMGR == endmqm

Monitoring

- Many queue manager event messages are common
  - For example, queue full
- But not every event is on every platform
  - Authorisation, Logging, and Channel auto-definition events are Distributed only
  - IMS Bridge events are only on z/OS
- Recording queue manager and application activity is very different
  - z/OS has SMF 115 and 116 records
  - Distributed has accounting, statistics and application activity events
- Distributed accounting and stats events are analogous to SMF 116
  - No equivalent to 115 records

Events Images





Problem Determination

- On Distributed, there are several places to look for PD information
  - Error logs written to /var/mqm/errors and /var/mqm/qmgrs/<qmgr>/errors
  - FFST written to /var/mqm/errors for serious errors
  - Trace provided by MQ commands and written to /var/mqm/trace

- On z/OS, also numerous places to follow the clues:
  - The MSTR and CHIN JES log
    - Should always be the first place to look
  - MQ API trace (aka user parameter trace) – a GTF trace
  - SMF 115 statistical information
  - SMF 116 class(3) accounting (task related) data
  - A dump for serious problems

Backup

- MAKEDEF and dmpmqcfg are tools to backup configuration
  - With V8, have dmpmqmsg to backup messages

- On Distributed, backup of log files is done by stopping qmgr and copying /var/mqm/log directory
  - rcdmqing takes images of queues into logs

- On z/OS, full and fuzzy backups of pagesets are supported
- CFSTRUCT backup required for QSG
  - takes image of shared queue into logs

High Availability and Disaster Recovery

- Shared queues on z/OS for continuous processing
- On Distributed, MQ provides multi-instance
  - Not on z/OS because ARM is provided

- Cross-site DR will usually use disk replication for any platform

Summary

- Title asks about oil and water

- Perhaps (olive) oil and (balsamic) vinegar is better description
  - Blending together