WebSphere MQ Disaster Recovery

Mark Taylor
marke_taylor@uk.ibm.com
IBM Hursley

Capitalware's MQ Technical Conference v2.0.1.3

Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Introduction

- Availability is a very large subject
- You can have the best technology in the world, but you have to manage it correctly
- Technology is not a substitute for good planning and testing!

What is DR – Wikipedia Version

- Disaster recovery is the process, policies and procedures related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster. Disaster recovery is a subset of business continuity. While business continuity involves planning for keeping all aspects of a business functioning in the midst of disruptive events, disaster recovery focuses on the IT or technology systems that support business functions.

What is DR

- Getting applications running after a major (often whole-site) failure or loss
- It is not about High Availability although often the two are related and share design and implementation choices
  - "HA is having 2, DR is having them a long way apart"
  - More seriously, HA is about keeping things running, while DR is about recovering when HA has failed.
- Requirements driven by business, and often by regulators
  - Data integrity, timescales, geography …
- One major decision point: cost
  - How much does DR cost you, even if it's never used?
  - How much are you prepared to lose

Disaster Recovery vs High Availability

- Designs for HA typically involve a single site for each component of the overall architecture
- Designs for DR typically involve separate sites
- Designs for HA (and CA) typically require no data loss
- Designs for DR typically can have limited data loss
- Designs for HA typically involve high-speed takeover
- Designs for DR typically can permit several hours down-time

Local Recovery

"MQ has failed"

- Don't restart queue manager until you know why it failed
  - You can probably do one restart attempt safely, but don't continually retry
  - When running under an HA coordinator, have retry counts
- At least ensure you take a backup of the queue manager data, log files, and any error logs/FDCs/dumps
  - So it can be investigated later
  - Might be possible for IBM to recover messages
  - Consider taking these copies as part of an HA failover procedure
- While trying restart/recovery procedures, consider a PMR
  - Often see "cold start" as first reaction at some customers
  - If you have a support contract, open PMR before trying cold start
    - IBM may have an alternative
    - IBM may ask you to start collecting documentation
  - Do not make everything a Sev1

First Manual Restart

- Restart your queue manager
  - Only clean the IPC (shared memory/semaphores) if IBM requests it
    - This should never be necessary
    - Remove calls to ipcrm or amqiclen from any startup/failover scripts
  - Start as simply as possible
    - strmqm –ns QM1
  - Monitor the restart, look for FDC's
  - If OK, then end the qmgr and restart normally

- What if the restart fails?
  - Option to escalate to cold start
  - Further escalation to rebuilding queue manager

Cold Starting WMQ

- Typical reason: hardware (most likely disk) failure or logs deleted by mistaken administrator
- Symptoms: Cannot start queue manager because logs unavailable or corrupt files

- "Cold start" is a technique to restart without needing logs

- What does a cold start cost you?
  - In-flight transactions will not be automatically rolled-back
  - In-doubt transactions will be forgotten
  - Ability to recover messages from the logs
  - Possible loss of messages
  - Possible duplication of already-processed messages

Cold Starts on Distributed Platforms

- These instructions are for Distributed platforms
  - Similar tasks are done on z/OS
- Basic idea is to replace your 'bad' logs with 'good' logs
  - By creating a dummy queue manager and using its logs instead
  - Logs do not contain queue manager name
- Other considerations
  - Is this queue manager part of a WMQ cluster?
    - This shouldn't matter, but may want to resynchronise repositories
    - In case any updates were in-flight when system failed
  - Is this queue manager under the control of an HA cluster?
    - Failover will not help if the shared disks/files are corrupt
    - Disable failover in the HA system until recovery complete

Cold Start – Procedure (1)

- Create a queue manager EXACTLY like the one that failed
  - Use qm.ini to work out parameters to crtmqm command

```
Log:
   LogPrimaryFiles=10
   LogSecondaryFiles=10
   LogFilePages=65535
   LogType=CIRCULAR
```

- Issue the crtmqm command
  - crtmqm -lc -lf 65535 -lp 10 -ls 10 –ld /tmp/mqlogs TEMP.QMGR
  - Make sure there is enough space for the new log files in that directory
- Name of the dummy queue manager is irrelevant
  - Only care about getting the log files

Cold Start – Procedure (2)

- Don't start this dummy queue manager, just create it
- Replace old logs and amqhlctl.lfh with the new ones
  ```
  cd /var/mqm/log
  mv QM1 QM1.SAVE
  mv /tmp/mqlogs/TEMP!QMGR QM1
  ```
  - Note the "mangled" directory name … this is normal
- Data in the queues is preserved if messages are persistent
- Object definitions are also preserved
  - Objects contain their own definitions in their files
  - Mapping between files and object names held in QMQMOBJCAT

Rebuilding a Queue Manager

- A rebuild creates a replacement queue manager
  - Same object definitions
  - But loss of message data and channel sequence numbers
- Replacement queue manager has a new QMID
  - MQ Explorer saves QMID in its list of known queue managers
  - Will allow you to connect, but requires confirmation that the new qmid is expected
- Recommend issuing RESET CLUSTER at full repository to remove the old QMID before bringing the replacement online

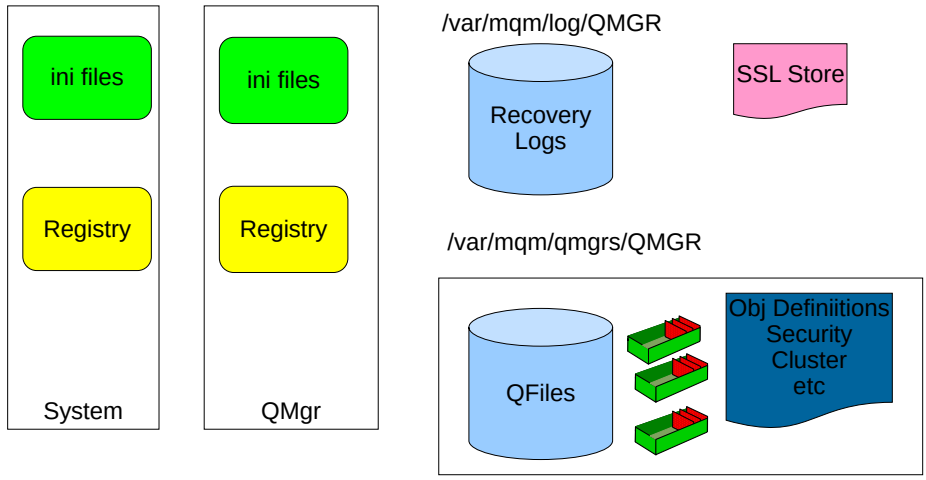
Rebuilding a Queue Manager

- Make sure you have a backup of the definitions
  - Either through a tool such as Omegamon Configuration Manager
  - Or by manually creating the backup – MAKEDEF, dumpmqcfg or MS03
- Make sure you know which version of WMQ is installed
  - And you have the install images for the code
- Make sure you've got the security configuration
  - Windows SIDs?
- Also any customisation in the qm.ini file (or registry)
- And sometimes exits might have external configuration
  - Don't forget to have the binaries available - often separately installed

Recovering Messages

- It might be possible to recover messages after rebuilding a queue manager
- While queue manager is stopped, copy the qfile from the damaged system
- No guarantees, and transactional operations may be inconsistent
  - But it might be good enough

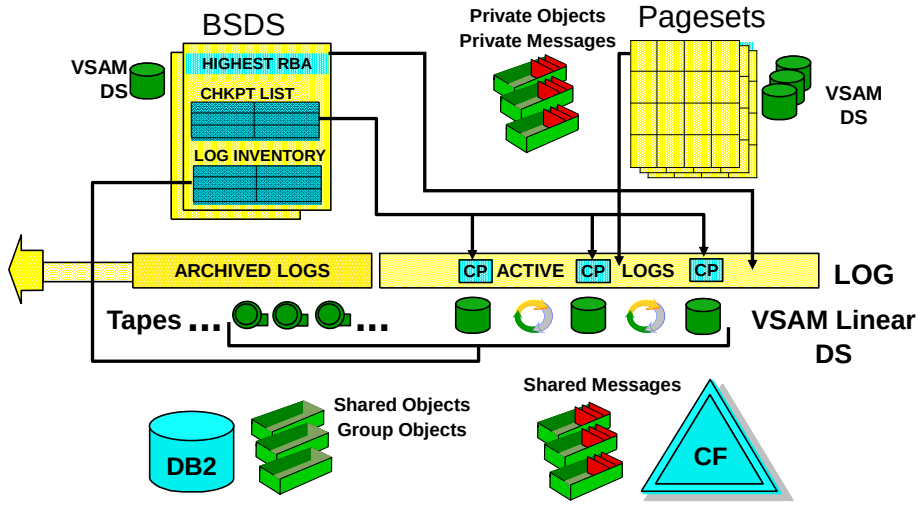
What makes a Queue Manager on Dist?



Backups

- At minimum, backup definitions at regular intervals
  - Include ini files and security settings
- One view is there is no point to backing up messages
  - They will be obsolete if they ever need to be restored
  - Distributed platforms – data backup only possible when qmgr stopped
- Use rcdmqimg on Distributed platforms to take images
  - Channel sync information is recovered even for circular logs
- Backup everything before upgrading code levels
  - On Distributed, you cannot go back
- Exclude queue manager data from normal system backups
  - Some backup products interfere with WMQ processing

What makes a Queue Manager on z/OS?



What makes up a Queue Manager?

- Queue manager started task procedure
  - Specifies MQ libraries to use, location of BSDS and pagesets and INP1, INP2 members start up processing
- System Parameter Module – zParm
  - Configuration settings for logging, trace and connection environments for MQ
- BSDS: Vital for Queue Manager start up
  - Contains info about log RBAs, checkpoint information and log dataset names
- Active and Archive Logs: Vital for Queue Manager start up
  - Contain records of all recoverable activity performed by the Queue Manager
- Pagesets
  - Updates made “lazily” and brought “up to date” from logs during restart
  - Start up with an old pageset (restored backup) is not really any different from start up after queue manager failure
  - Backup needs to copy page 0 of pageset first (don’t do volume backup!)
- DB2 Configuration information & Group Object Definitions
- Coupling Facility Structures
  - Hold QSG control information and MQ messages

Backing Up a z/OS Queue Manager

- Keep copies of ZPARM, MSTR procedure, product datasets and INP1/INP2 members
- Use dual BSDS, dual active and dual archive logs
- Take backups of your pagesets
  - This can be done while the queue manager is running (fuzzy backups)
  - Make sure you backup Page 0 first, REPRO or ADRDSSU logical copy
- DB2 data should be backed up as part of the DB2 backup procedures
- CF application structures should be backed up on a regular basis
  - These are made in the logs of the queue manager where the backup was issued

Remote Recovery

Topologies

- Sometimes a data centre is kept PURELY as the DR site
- Sometimes 2 data centres are in daily use; back each other up for disasters
  - Normal workload distributed to the 2 sites
  - These sites are probably geographically distant
- Another variation has 2 data centres “near” each other
  - Often synchronous replication
  - With a 3rd site providing a long-distance backup
- And of course further variations and combinations of these

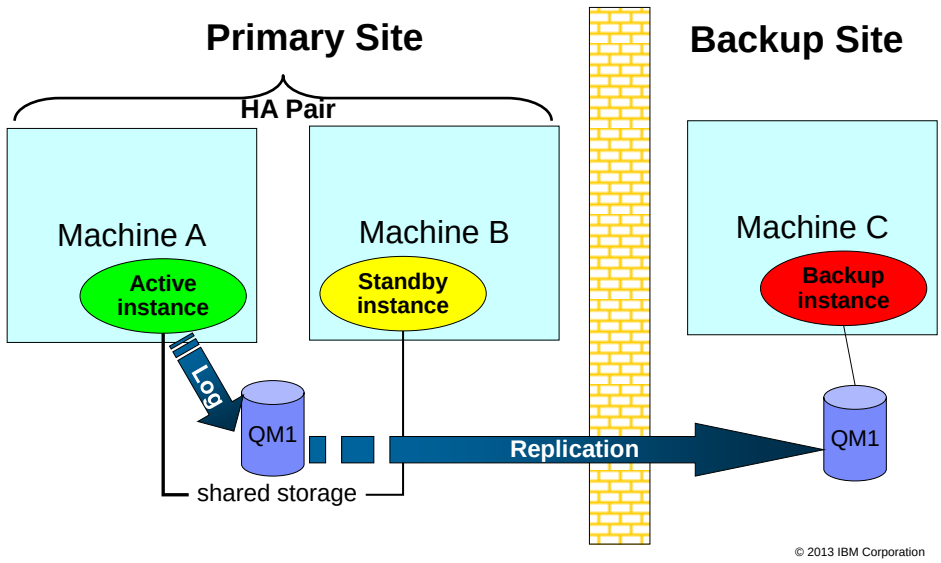
Queue Manager Connections

- DR topologies have little difference for individual queue managers
- But they do affect overall design
  - Where do applications connect to
  - How are messages routed
- Clients need ClntConn definitions that reach any machine
- Will be affected by how you manage network
  - Do DNS names move with the site?
  - Do IP addresses move with the site?
- Some sites always put IP addresses in CONNAME; others use hostname
  - No rule on which is better

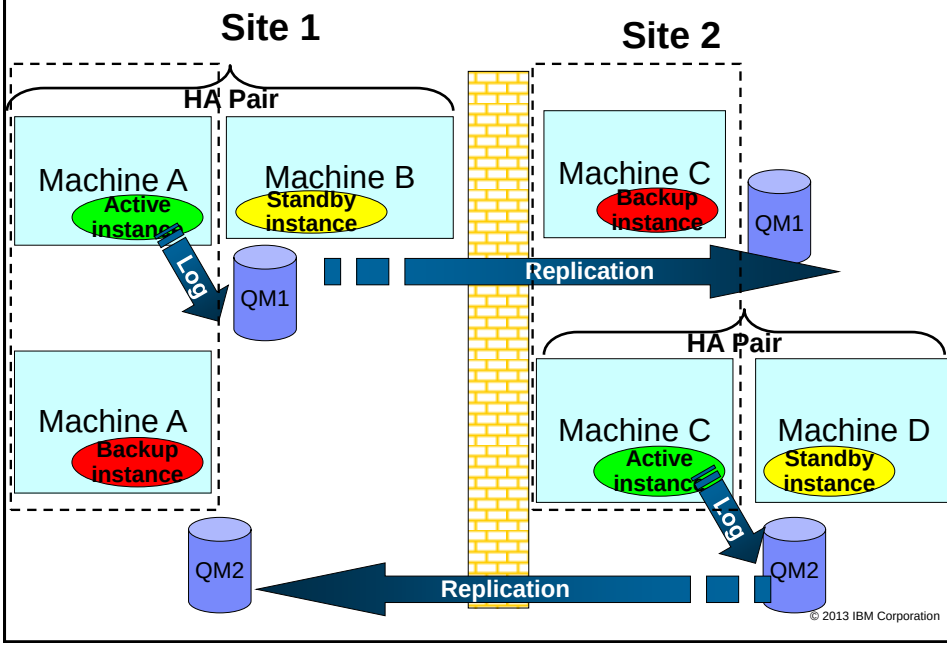
Disk replication

- Disk replication can be used for WMQ disaster recovery
- Either synchronous or asynchronous disk replication is OK
  - Synchronous:
    - No data loss if disaster occurs
    - Performance is impacted by replication delay
    - Limited by distance (eg 100km)
  - Asynchronous:
    - Some limited data loss if disaster occurs
    - It is critical that queue manager data and logs are replicated in the same consistency group if replicating both
- Disk replication cannot be used between the active and standby instances of a multi-instance queue manager
  - Could be used to replicate to a DR site in addition though

Combining HA and DR



Combining HA and DR – “Active/Active”



Backup Queue Manager - Objective

- Feature introduced in WMQ V6
- Prepares a queue manager for restart/recovery
  - Without needing to replay all logs at a critical time
  - For Windows, Unix and System i
- “Backup” queue manager takes the place of original
  - New QMID but contains original definitions and messages

Backup Queue Manager - Procedure

- Configure queue manager with linear logging
- Create a queue manager at the primary site
  - Create an identical one at the DR site – the backup queue manager
- Ship full, inactive log files from active QM to the DR site
  - Can use disk replication to do this
  - Or modify SupportPac or sample programs for log management to copy files at the same time as deleting/archiving local logs
- Replay log files on the backup QM to bring it up to date
  - Do this at regular intervals
  - strmqm -r
- If disaster occurs, activate the backup queue manager
  - strmqm -a
- For more control, can force filling of current log file
  - MQSC RESET QMGR TYPE(ADVANCELOG)

Integration with other products

- May want to have consistency with other data resources
  - For example, databases and app servers
- Only way for guaranteed consistency is disk replication where all logs are in same group
  - Otherwise transactional state might be out of sync
- DB2 can use WMQ as part of its own replication strategy
  - InfoSphere Replication Server

Planning and Testing

Planning for Recovery

- Write a DR plan
  - Document everything – to tedious levels of detail
  - Include actual commands, not just a description of the operation
    - Not “Stop MQ”, but “as mqm, run /usr/local/bin/stopmq.sh US.PROD.01”
- And test it frequently
  - Recommend twice a year
  - Record time taken for each task
- Remember that the person executing the plan in a real emergency might be under-skilled and over-pressured
  - Plan for no access to phones, email, online docs …
- Each test is likely to show something you’ve forgotten
  - Update the plan to match
  - You’re likely to have new applications, hardware, software …
- May have different plans for different disaster scenarios

Example Exercises from MQ Development

- Different groups have different activities that must continue
  - Realistic scenarios can help show what might not be available
- From the WMQ development lab …
- Most of the change team were told there was a virulent disease and they had to work from home
  - Could they continue to support customers
- If Hursley machine room was taken out by a plane missing its landing at Southampton airport
  - Could we carry on developing the WMQ product
  - Source code libraries, build machines, test machines …
  - Could fixes be produced
- (A common one) Someone hit emergency power-off button

Networking Considerations

- DNS - You will probably redirect hostnames to a new site
  - But will you also keep the same IP addresses?
  - Including NAT when routing to external partners?
  - Affects CONNAME
- Include external organisations in your testing
  - 3rd parties may have firewalls that do not recognize your DR servers
- LOCLADDR configuration
  - Not normally used by MQ, but firewalls, IPT and channel exits may inspect it
  - May need modification if a machine changes address
- Clustering needs special consideration
  - Easy to accidentally join the real cluster and start stealing messages
  - Ideally keep network separated, but can help by:
    - Not giving backup 'live' security certs
    - Not starting chinit address space (z/OS)
    - Not allowing channel initiators to start (distributed)
    - Use CHLAUTH rules
- Backup will be out of sync with the cluster
  - REFRESH CLUSTER() resolves updates

A Real MQ Network Story

- Customer did an IP move during a DR test
- Forgot to do the IP move back when they returned to prime systems
- Didn't have monitoring in place that picked this up until users complained about lack of response

APPLICATIONS AND AUTO-RECONNECTION

HA applications – MQ connectivity

- If an application loses connection to a queue manager, what does it do?
  - End abnormally
  - Handle the failure and retry the connection
  - Reconnect automatically thanks to application container
    - WebSphere Application Server contains logic to reconnect JMS clients
  - Use MQ automatic client reconnection

Automatic client reconnection

- MQ client automatically reconnects when connection broken
  - MQI C clients and standalone JMS clients
  - JMS in app servers (EJB, MDB) does not need auto-reconnect
- Reconnection includes reopening queues, remaking subscriptions
  - All MQI handles keep their original values
- Can reconnect to same queue manager or another, equivalent queue manager
- MQI or JMS calls block until connection is remade
  - By default, will wait for up to 30 minutes
  - Long enough for a queue manager failover (even a really *slow* one)

Automatic client reconnection

- Can register event handler to observe reconnection

- Not all MQI is seamless, but majority repaired transparently
  - Browse cursors revert to the top of the queue
  - Nonpersistent messages are discarded during restart
  - Nondurable subscriptions are remade and may miss some messages
  - In-flight transactions backed out

- Tries to keep dynamic queues with same name
  - If queue manager doesn't restart, reconnecting client's TDQs are kept for a while in case it reconnects
  - If queue manager does restart, TDQs are recreated when it reconnects

Automatic client reconnection

- Enabled in application code, ini file or CLNTCONN definition
  - MQI: MQCNO_RECONNECT, MQCNO_RECONNECT_Q_MGR
  - JMS: Connection factory properties

- Plenty of opportunity for configuration
  - Reconnection timeout
  - Frequency of reconnection attempts

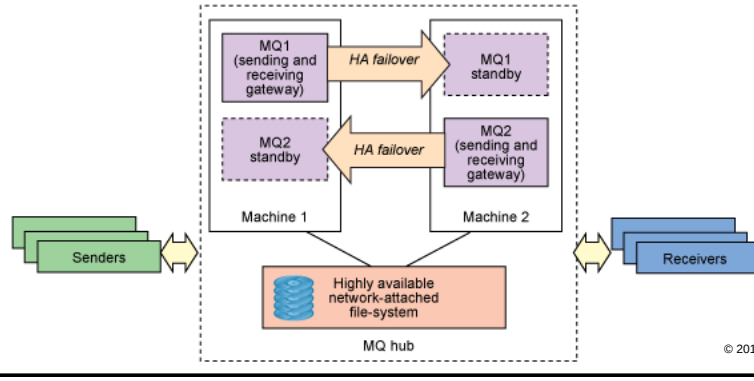
- Requires:
  - Threaded client
  - 7.0.1 server – including z/OS
  - Full-duplex client communications (SHARECNV >= 1)

Client Configurations for Availability

- Use wildcarded queue manager names in CCDT
  - Gets weighted distribution of connections
  - Selects a “random” queue manager from an equivalent set
- Use multiple addresses in a CONNAME
  - Could potentially point at different queue managers
  - More likely pointing at the same queue manager in a multi-instance setup
- Use automatic reconnection
- Pre-connect Exit from V7.0.1.4
- Use IP routers to select address from a list
  - Based on workload or anything else known to the router

Application Patterns for availability

- Article describing examples of how to build a hub topology supporting:
  - Continuous availability to send MQ messages, with no single point of failure
  - Linear horizontal scale of throughput, for both MQ and the attaching applications
  - Exactly once delivery, with high availability of individual persistent messages
  - Three messaging styles: Request/response, fire-and-forget, and pub/sub
- http://www.ibm.com/developerworks/websphere/library/techarticles/1303_broadhurst/1303_broadhurst.html



Other Resources

- Applications may need to deal with replay or loss of data.
  - Decide whether to clear queues down to a known state, or enough information elsewhere to manage replays
- Order of recovery may change with different product releases
  - Every time you install a new version of a product revisit your DR plan
- What do you really need to recover
  - DR site might be lower-power than primary site
  - Some apps might not be critical to the business
  - But some might be unrecognised prereqs

If a Real Disaster Hits

- Hopefully you never need it. But if the worst happens:
- Follow your tested plan
  - Don't try shortcuts
- But also, if possible:
  - Get someone to take notes and keep track of the time tasks took
  - Prepare to attend post mortem meetings on steps you took to recover
  - Accept all offers of assistance
- And afterwards:
  - Update your plan for the next time

Summary

- Various ways of recovering queue managers
- Plan what you need to recover for WMQ
- Plan the relationship with other resources
- Test your plan