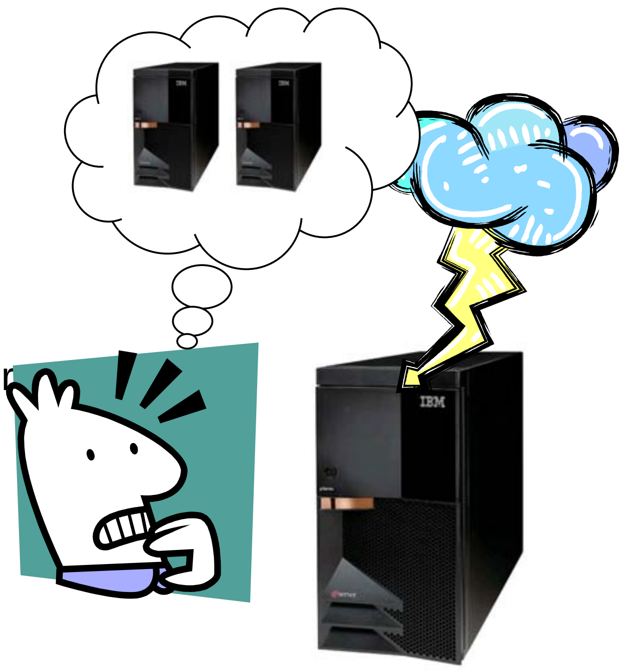
Highly Available Messaging Rock solid MQ

WebSphere MQ development
IBM Hursley

Morag Hughson hughson@uk.ibm.com
David Coles dcoles@uk.ibm.com

Overview

- Techniques and technologies to ensure availability of messaging
- WebSphere MQ technologies
  - Queue Manager Clusters
  - Multi-instance Queue Manager
  - Shared Queues
- Platform technologies
  - Failover with HA clusters
  - z/OS, Windows, Unix



Agenda

- Introduction to HA
- WebSphere MQ HA technologies
- Using MQ in an HA Cluster
- Application considerations

N O T E S

- This page intentionally left blank.

Introduction

- Availability is a very large subject
  - We won't be covering everything
- Not just HA technology – anything that can cause an outage is significant
  - This might be an overloaded system, etc
  - We will only be covering HA technology
- You can have the best HA technology in the world, but you have to manage it correctly
- HA technology is not a substitute for good planning and testing!

Introduction

N O T E S

- Availability is a very large subject. It is not simply a matter of selecting the correct hardware and software. That in itself doesn't provide an HA solution. It is necessary to consider anything that may cause an outage or loss of service. This might be an outage of a machine, but it could just be a slowdown in throughput which means you don't meet your SLAs.
- Even once you have tuned everything and set up your HA system, you can't just sit back and forget about it. You need to monitor the system to check it is working correctly and keep maintaining it. HA is also not a substitute for good planning and management.
- Just because you have implemented an HA environment doesn't mean you can avoid testing!

Some terms

- Downtime
  - A period when the system is unavailable
- Single Point of Failure (SPOF)
  - A part of a system whose failure prevents the entire system from working
- High Availability (HA)
  - Ensuring a certain (high) level of operational continuity for users of a system
- Redundancy
  - Additional instances of a critical component
- High availability cluster
  - A cluster of computer systems designed to give high availability. Typically, some degree of redundancy, heartbeating and automation are provided.
- Heartbeating
  - Regular verification that an active system is still available

Some terms

- Failover
  - Automatic switching of availability to a standby server
- Switchover
  - Controlled switching of availability to a standby server
- Failback
  - Restoration of a system back to its pre-failure state
- Disaster Recovery (DR)
  - Recovery of availability following major system or site disaster, sometimes with an amount of data loss
- Continuous Availability
  - Ensuring complete operational continuity for users of a system

What are you trying to achieve?

- The objective is to achieve 24x7 availability of messaging
- Not always achievable, but we can get close
  - 99.9% availability = 8.76 hours downtime/year
  - 99.999% = 5 minutes
  - 99.9999% = 30 seconds
- Potential outage types:
  - 80% scheduled downtime (new software release, upgrades, maintenance)
  - 20% unscheduled downtime (source: Gartner Group)
    - 40% operator error
    - 40% application error
    - 20% other (network failures, disk crashes, power outage etc.)
- Avoid application awareness of availability solutions

SHARE in Anaheim 2011

What are you trying to achieve?

N O T E S

- The objective is to achieve 24x7 availability of messaging. Applications should be processing messages continuously, regardless of any failures in any component. This presentation concentrates on the MQ and MB element, but they are not the only areas to think about.
- Availability is not necessarily the same as ensuring processing of each and every message. In some situations, some limited message loss is acceptable provided that availability is maximised. For example, a message might expire or be superseded during an outage. Here, the important thing is to ensure that messages are still getting through.
- Service Level Agreements (SLAs) should define what level of availability your applications and services should provide. The level of availability is often measured by the number of 9s.
- HA solutions should increase availability given scheduled or unscheduled downtime. Scheduled downtime is more common than unscheduled. Availability issues usually involve a multitude of hardware and software systems.
- Avoid application awareness of availability solutions and aim to have little or no code in the application managing the environment. That’s a task better left to systems administrators.

SHARE in Anaheim 2011

Single Points of Failure

- With no redundancy or fault tolerance, a failure of any component can lead to a loss of availability
- Every component is critical. The system relies on the:
  - Power supply, system unit, CPU, memory
  - Disk controller, disks, network adapter, network cable
  - ...and so on
- Various techniques have been developed to tolerate failures:
  - Uninterruptible Power Supply (UPS) or dual supplies for power loss
  - RAID for disk failure
  - Fault-tolerant architectures for CPU/memory failure
  - ...etc
- Elimination of SPOFs is important to achieve HA

Single Points of Failure

NOTES

- Any part of a system whose failure can make the system unavailable is single point of failure.
- There are several techniques and technologies used to eliminate SPOFs, such as deployment of redundant systems and hardware components.

WebSphere MQ HA technologies

- Queue manager clusters
- Queue-sharing groups
- Support for networked storage
- Multi-instance queue managers
- HA clusters
- Client reconnection

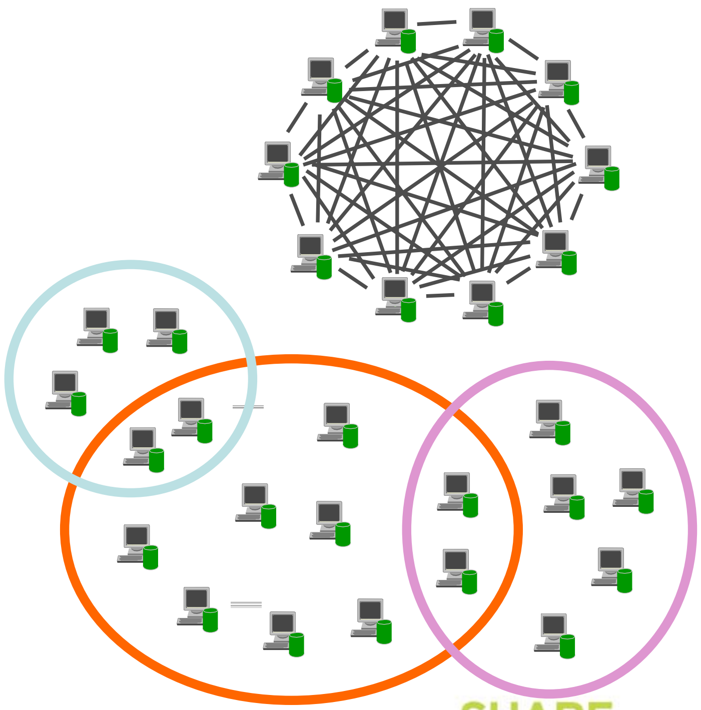
N O T E S

- This page intentionally left blank.

Queue Manager Clusters

- Sharing cluster queues on multiple queue managers prevents a queue from being a SPOF

- Cluster workload algorithm automatically routes traffic away from failed queue managers



SHARE in Anaheim 2011

Queue Manager Clusters

N O T E S

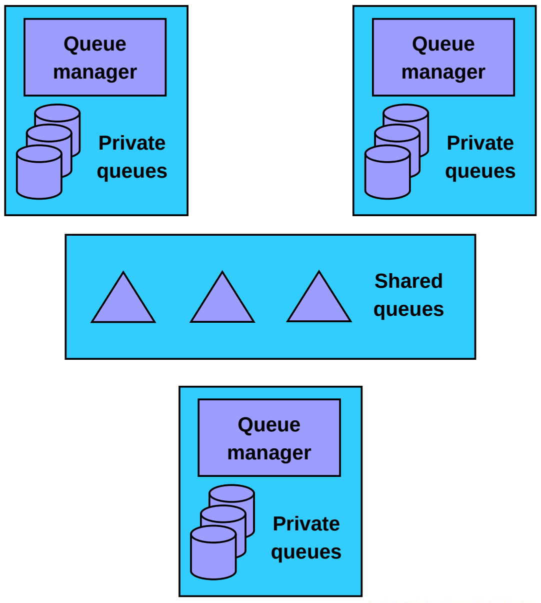
- Although queue manager clustering does provide some facilities useful in maintaining availability of messaging, it is primarily a parallel processing feature. It is simple to deploy extra processing power in the cluster to process more messages.

- If a queue manager in a cluster fails, the failure can be mitigated by other cluster queue managers hosting instances of the cluster queues. Messages are marooned on the failed queue manager until it restarts, but messaging through the cluster is still operational.

SHARE in Anaheim 2011

Queue-Sharing Groups

- On z/OS, queue managers can be members of a queue-sharing group
- Shared queues are held in a coupling facility
  - All queue managers in the QSG can access the messages
- Benefits:
  - Messages remain available even if a queue manager fails
  - Pull workload balancing
  - Apps can connect to the group



Queue-Sharing Groups

NOTES

- In the queue-sharing group environment, an application can connect to any of the queue managers within the queue-sharing group. Because all the queue managers in the queue-sharing group can access the same set of shared queues, the application does not depend on the availability of a particular queue manager; any queue manager in the queue-sharing group can service any queue. This gives greater availability if a queue manager stops because all the other queue managers in the queue-sharing group can continue processing the queue.

- To further enhance the availability of messages in a queue-sharing group, WebSphere MQ detects if another queue manager in the group disconnects from the Coupling Facility abnormally, and completes units of work for that queue manager that are still pending, where possible. This is known as peer recovery.

Support for networked storage

New in 7.0.1 on distributed

- Support has been added for queue manager data in networked storage
  - NAS so that data is available to multiple machines concurrently
    - Already have SAN support
  - Added protection against concurrent starting two instances of a queue manager using the same queue manager data
  - On Windows, support for Windows network drives (SMB)
  - On Unix variants, support for Posix-compliant filesystems with leased file locking
    - NFS v4 has been tested by IBM
    - Currently, Solaris has some issues with some network-attached storage systems.
- Some customers have a “no local disk” policy for queue manager data
  - This is an enabler for some virtualized deployments
  - Allows simple switching of queue manager to another server following a hardware failure

SHARE in Anaheim 2011

Support for networked storage

SHARE Technology • Connections • Results

N O T E S

- While not directly an HA technology, this is an enabler for customers who want to place all of the data remote from their servers such that it becomes possible to replace one server with another in the event of a failure.
- Support has been added for networked (NAS) storage for queue manager data and logs. Previously, it’s been supported for error and trace directories, and for installation binaries.
- On Unix platforms, we support Posix-compliant filesystems which supports lease-based file locking. The lease-based locking ensures that files unlock when the server running a queue manager fails. This rules out NFS v3 for use in an HA environment because the file locks are not released automatically for some failures and this will prevent failover.
- See this page for the latest support statement (include current Solaris limitations): http://www.ibm.com/support/docview.wss?uid=swg21433474
- On Unix, we have provided a test program (amqmfsck) which checks out the filesystem’s behavior. If the tests do not pass, a queue manager using the filesystem will not behave correctly. Output from this program can be used to diagnose a failure.
- On Windows, we support Windows network drives (SMB – server message block).

SHARE in Anaheim 2011

Introduction to Failover and MQ

- Failover is the automatic switching of availability of a service
  - For MQ, the "service" is a queue manager
- Traditionally the preserve of an HA cluster, such as HACMP
- Requires:
  - Data accessible on all servers
  - Equivalent or at least compatible servers
    - Common software levels and environment
  - Sufficient capacity to handle workload after failure
    - Workload may be rebalanced after failover requiring spare capacity
  - Startup processing of queue manager following the failure
- MQ offers two ways of configuring for failover:
  - Multi-instance queue managers
  - HA clusters

Introduction to Failover and MQ

N O T E S

- Requirement to access data
  - Shared disks – for an HA cluster, usually "switchable" between the servers
  - Mirrored disks – must be truly synchronized mirror for integrity
- Requirement for client connectivity
  - IP address takeover (IPAT) is generally a feature of failover environments
  - If a queue manager changes IP address, intelligent routers can hide this or MQ network configuration can be defined with alternative addresses
- Servers must be equivalent
  - Common software levels – or at least compatible, to allow for progressive upgrade of the servers
  - Common environments – paths, userids, security
- Sufficient capacity to handle workload
  - Often, workload will be redistributed following a failover. Often, the systems are configured for mutual takeover where the workload following failover is doubled since the surviving servers must handle the traffic intended for both.

Failover considerations

- Failover times are made up of three parts:
  - Time taken to notice the failure
    - Heartbeat missed
    - Bad result from status query
  - Time taken to establish the environment before activating the service
    - Switching IP addresses and disks, and so on
  - Time taken to activate the service
    - This is queue manager restart
- Failover involves a queue manager restart
  - Nonpersistent messages, nondurable subscriptions discarded
- For fastest times, ensure that queue manager restart is fast
  - No long running transactions, for example

N O T E S

- This page intentionally left blank.

Multi-instance Queue Managers

New in 7.0.1 on distributed

- Basic failover support without HA cluster
- Two instances of a queue manager on different machines
  - One is the “active” instance, other is the “standby” instance
  - Active instance “owns” the queue manager’s files
    - Accepts connections from applications
  - Standby instance monitors the active instance
    - Applications cannot connect to the standby instance
    - If active instance fails, standby performs queue manager restart and becomes active
- Instances are the SAME queue manager – only one set of queue manager data
  - Queue manager data is held in networked storage

SHARE in Anaheim 2011

Multi-instance Queue Managers

NOTES

- “Basic failover”: no coordination with other resources like disks, IP addresses, databases, user applications. There is also no sophisticated control over where the queue managers run and move to (like a 3-node HACMP cluster, for example). Finally, once failover has occurred, it is necessary to manually start a new standby instance.
- Only one standby instance is supported.
- Architecturally, this is essentially the same as an existing HACMP/VCS setup, with the data shared between systems. It does not give anything “stronger” in terms of availability – but we do expect the typical takeover time to be significantly less. And it is much simpler to administer.
- Just as with a configuration using an HA cluster, the takeover is in essence a restart of the queue manager, so nonpersistent messages are discarded, queue manager channels go into retry, and so on.

SHARE in Anaheim 2011

Setting up a Multi-instance Queue Manager

1. Set up shared filesystems for QM data and logs
2. Create the queue manager on machine1
   - `crtmqm –md /shared/qmdata –ld /shared/qmlog QM1`
3. Define the queue manager on machine2 (or edit mqs.ini)
   - `addmqinf –vName=QM1 –vDirectory=QM1 –vPrefix=/var/mqm -vDataPath=/shared/qmdata/QM1`
4. Start an instance on machine1 – it becomes active
   - `strmqm –x QM1`
5. Start another instance on machine2 – it becomes standby
   - `strmqm –x QM1`

- That's it. If the queue manager instance on machine1 fails, the standby instance on machine2 takes over and becomes active
- On Windows, all instances must run on domain controllers

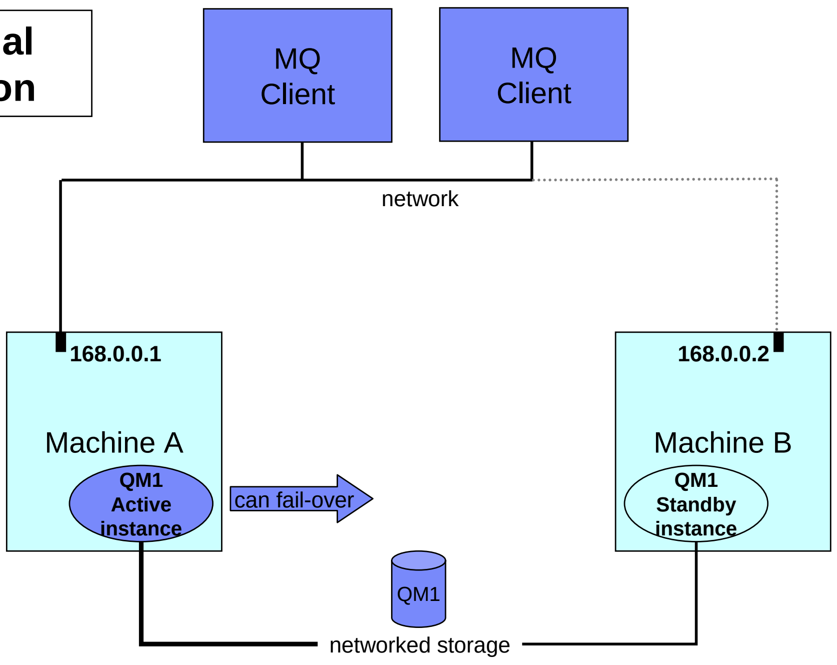
Setting up a Multi-instance Queue Manager

N O T E S

- Create the filesystems in networked storage and mount on the machines to run the queue manager instances. The networked storage must be mounted at the same location on the machines appropriate file permissions so that MQ can create the queue manager. Typically, a pair of filesystems would be used, one for the queue manager data and one for the logs.
- Create the queue manager on one machine specifying the DataPath (-md) and LogPath (-ld) parameters as directories on the networked storage.
- Define the queue manager on the other machine. This involves creating a QueueManager stanza in mqs.ini. The new addmqinf command can be used to do that if you don't want to edit the file directly. On Windows, addmqinf creates the equivalent entries in the registry.
- Start the first instance. You need to use '-x' to indicate that you want to start an instance of a multi-instance queue manager. If you omit '-x', the queue manager starts but it will not "permit" a standby instance – it's not a multi-instance queue manager.
- Start the second instance. You also need to use '-x' to indicates that you want to start an instance of a multi-instance queue manager. If you omit '-x', the command fails because the queue manager is already running. If you specify '-x', the command starts a standby instance.

Multi-instance Queue Managers

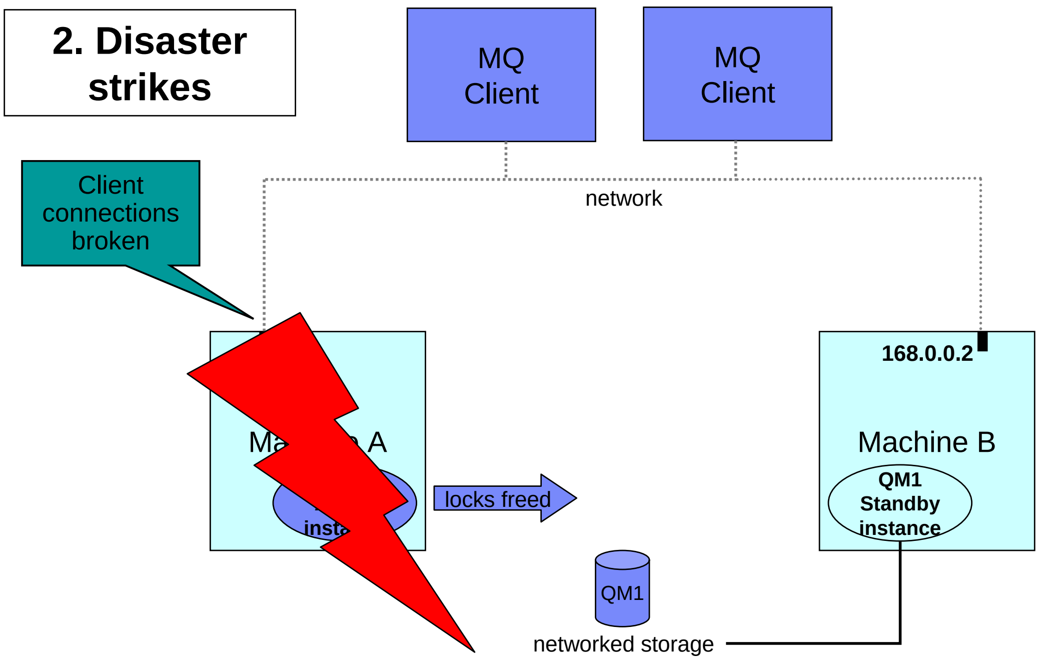
1. Normal execution



Owns the queue manager data

Multi-instance Queue Managers

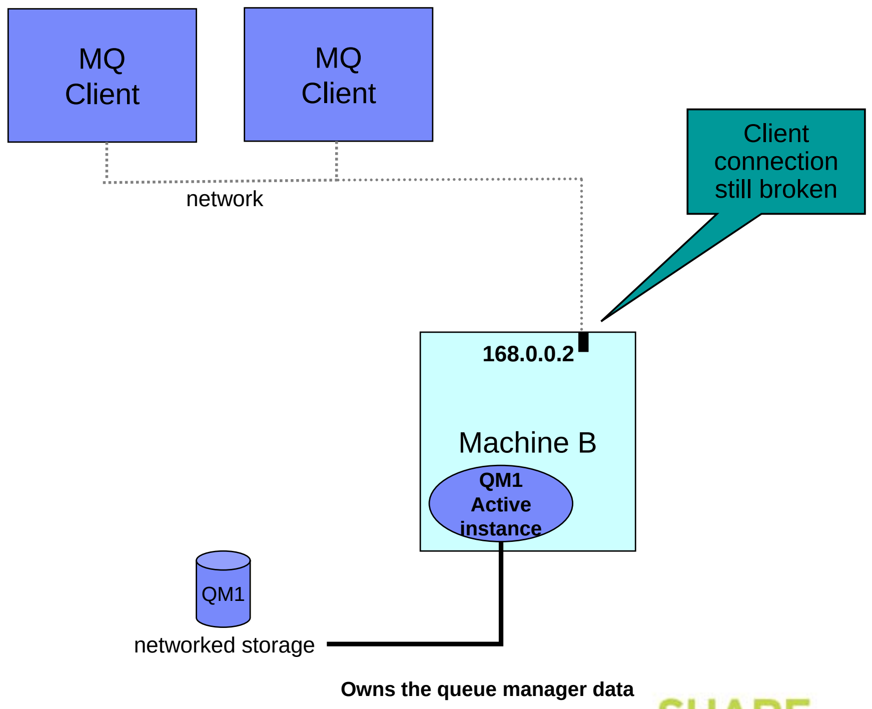
2. Disaster strikes



Multi-instance Queue Managers

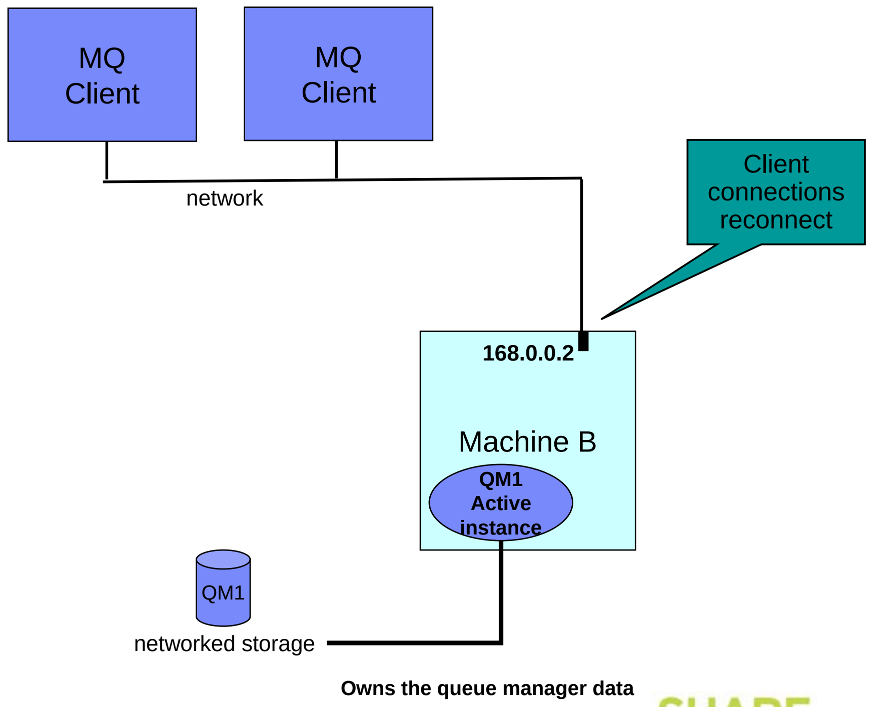
3. FAILOVER

Standby becomes active



Multi-instance Queue Managers

4. Recovery complete



Multi-instance Queue Managers

- MQ is NOT becoming an HA cluster
  - If other resources need to be coordinated, you need an HA cluster
  - WebSphere Message Broker and WebSphere MQ File Transfer Edition will integrate with multi-instance QM
  - Queue manager services can be automatically started, but with limited control
- The IP address of the queue manager changes when it moves
  - MQ channel configuration needs list of addresses unless you use external IPAT or an intelligent router
  - Connection name syntax extended to a comma-separated list
    - CONNAME(‘168.0.0.1,168.0.0.2’)
- System administrator is responsible for restarting another standby instance when failover has occurred

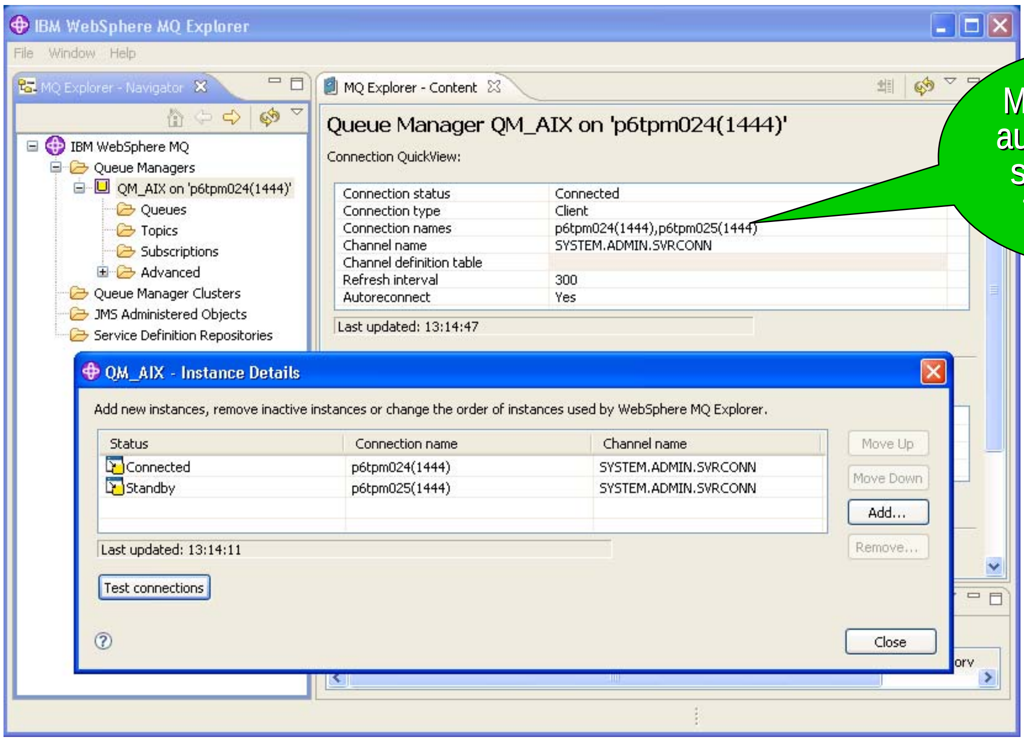
Administering a Multi-instance Queue Manager

- All queue manager administration must be performed on the active instance
- dspmq enhanced to display instance information

```
$ hostname
staravia
$ dspmq -x
QMNAME(MIQM)      STATUS(Running as standby)
    INSTANCE(starly)     MODE(Active)
    INSTANCE(staravia)   MODE(Standby)
```

  - dspmq issued on “staravia”
  - On “staravia”, there’s a standby instance
  - The active instance is on “starly”

Multi-instance Queue Manager in MQ explorer



Queue Manager QM_AIX on 'p6tpm024(1444)'

Connection QuickView:

Connection status	Connected
Connection type	Client
Connection names	p6tpm024(1444),p6tpm025(1444)
Channel name	SYSTEM.ADMIN.SVRCONN
Channel definition table	
Refresh interval	300
Autoreconnect	Yes

Last updated: 13:14:47

QM_AIX - Instance Details

Add new instances, remove inactive instances or change the order of instances used by WebSphere MQ Explorer.

Status	Connection name	Channel name
Connected	p6tpm024(1444)	SYSTEM.ADMIN.SVRCONN
Standby	p6tpm025(1444)	SYSTEM.ADMIN.SVRCONN

Last updated: 13:14:11

MQ Explorer automatically switches to the active instance

N O T E S

- This page intentionally left blank.

HA clusters

- MQ traditionally made highly available using an HA cluster
  - IBM PowerHA for AIX (formerly HACMP), Veritas Cluster Server, Microsoft Cluster Server, HP Serviceguard, …
- HA clusters can:
  - Coordinate multiple resources such as application server, database
  - Consist of more than two machines
  - Failover more than once without operator intervention
  - Takeover IP address as part of failover
  - Likely to be more resilient in cases of MQ and OS defects
- SupportPac MC91 has been withdrawn
  - Existing MC91 will work, but is not really appropriate any more
  - New commands allow for simpler taker over, will be described in the info center.

N O T E S

- This page intentionally left blank.

HA clusters

- In an HA cluster, queue manager data and logs are placed on a shared disk
  - Disk is switched between machines during failover

- The queue manager has its own “service” IP address
  - IP address is switched between machines during failover
  - Queue manager’s IP address remains the same after failover

- The queue manager is defined to the HA cluster as a resource dependent on the shared disk and the IP address
  - During failover, the HA cluster will switch the disk, take over the IP address and then start the queue manager

HA clusters

N O T E S

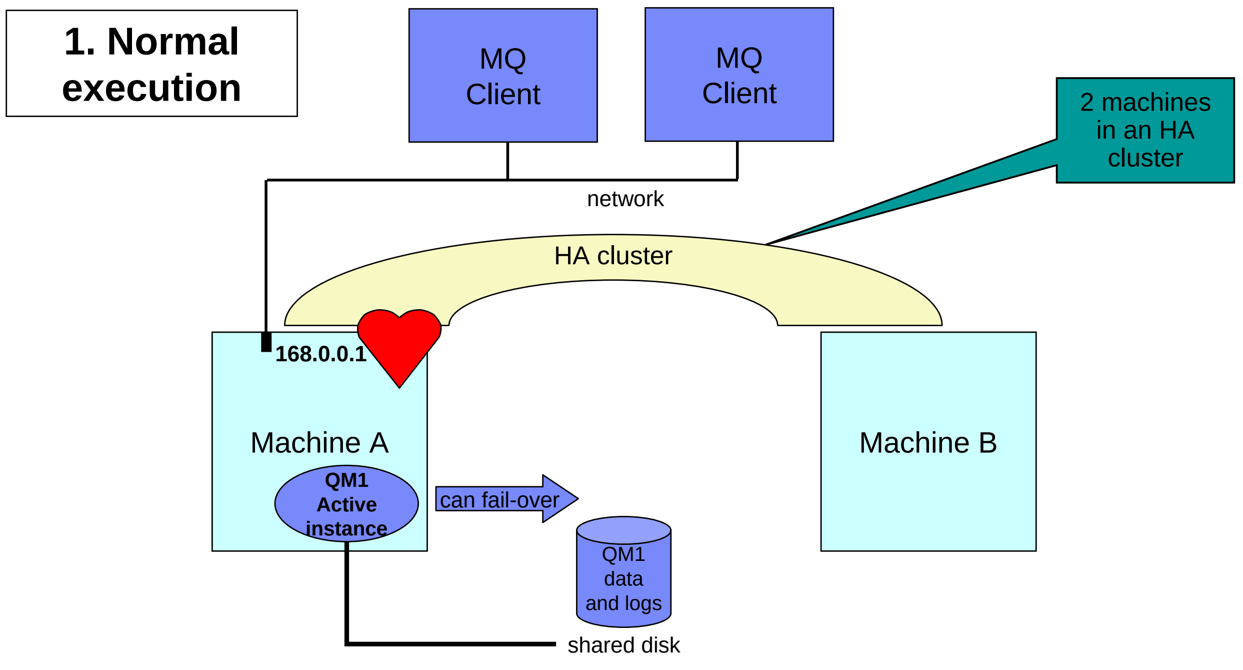
- The collection of servers that makes up a failover environment is known as a cluster. The servers are typically referred to as nodes.

- One node runs an application or service such as a queue manager, while the HA cluster monitors its health. The following example is called a cold standby setup because the other nodes are not running workload of their own. The standby node is ready to accept the workload being performed by the active node should it fail.

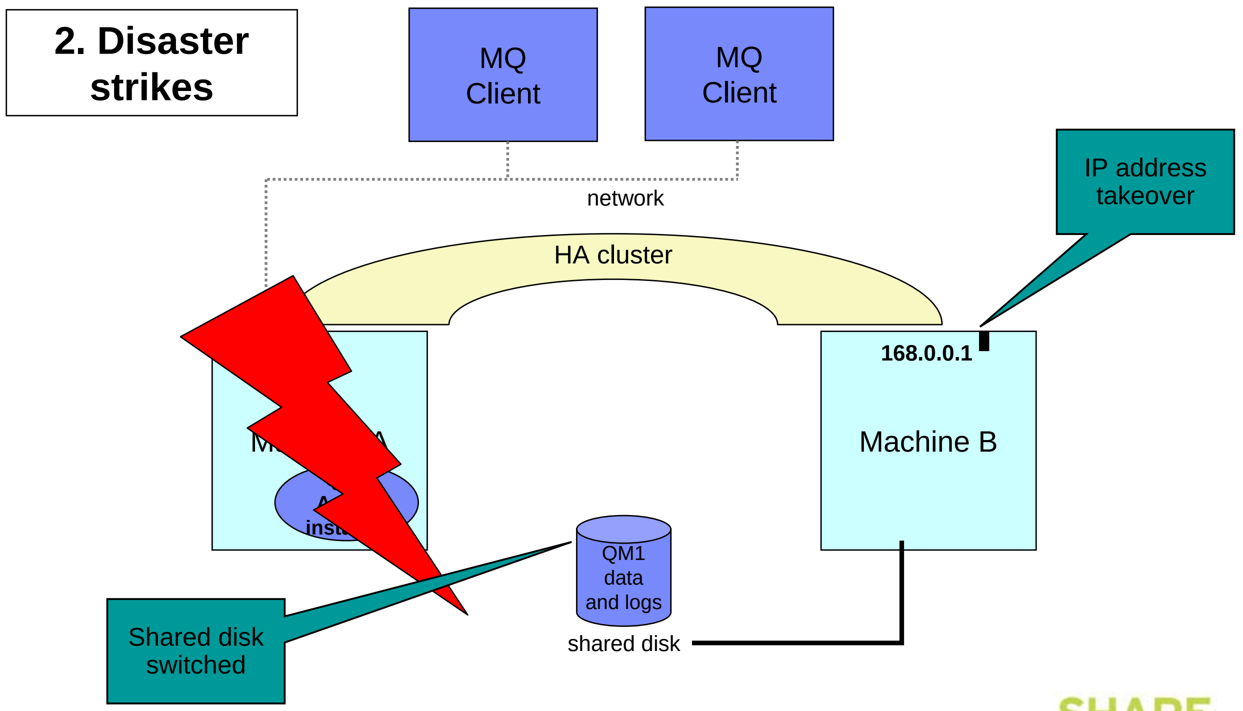
- A shared disk is a common approach to transferring state information about the application from one node to another, but is not the only solution. In most systems, the disks are not accessed concurrently by both nodes, but are accessible from either node, which take turns to "own" each disk or set of disks. In other systems the disks are concurrently visible to both (all) nodes, and lock management software is used to arbitrate read or write access.

- Alternatively, disk mirroring can be used instead of shared disk. An advantage of this is increased geographical separation, but latency limits the distance that can be achieved. But for reliability, any transaction logs must be the same - which means any synchronous disk writes must also be sent down the wire before being confirmed.

MQ in an HA cluster
Cold standby

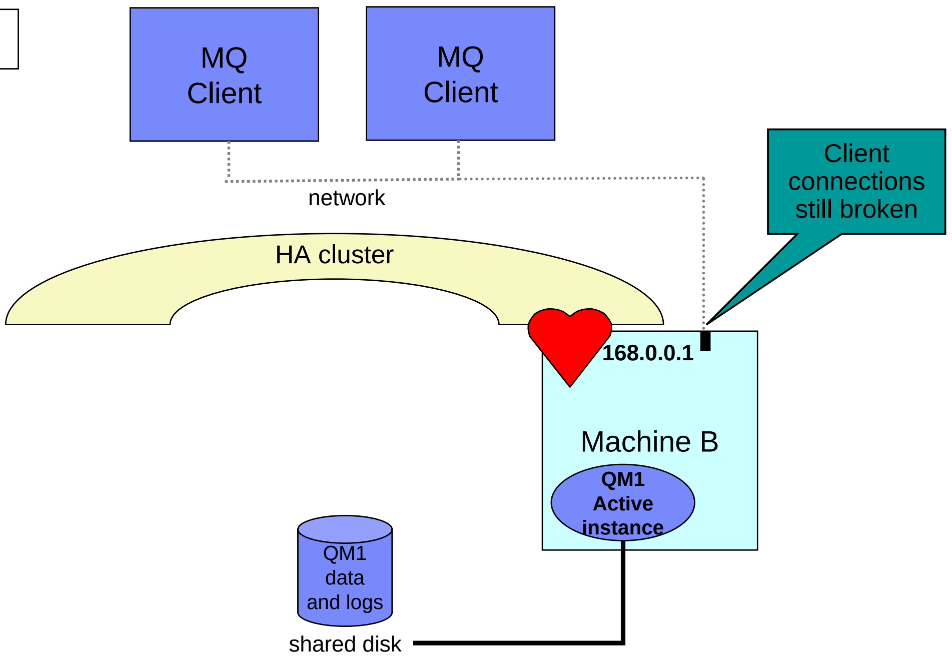


MQ in an HA cluster
Cold standby



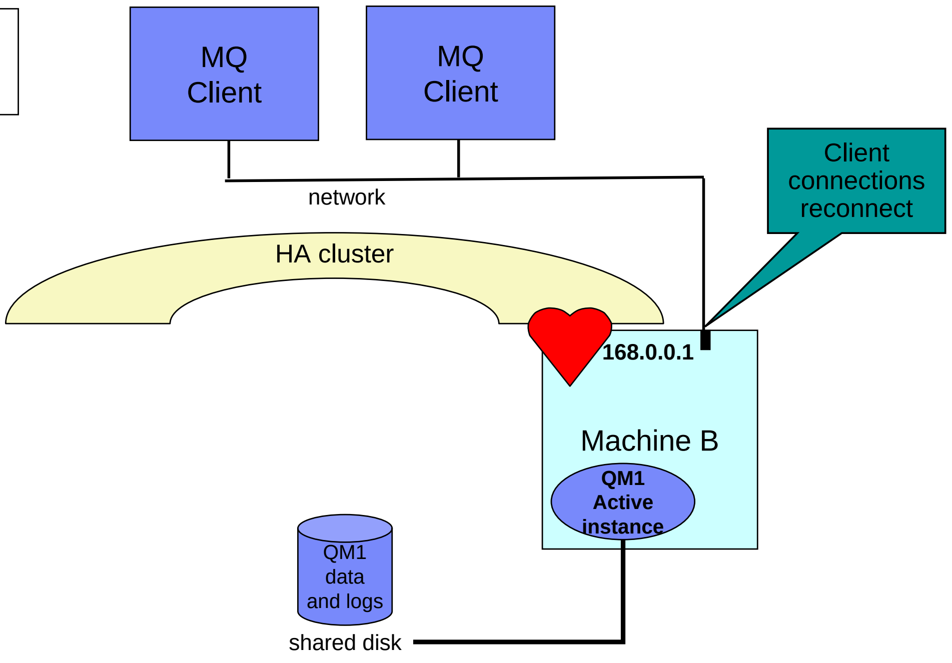
MQ in an HA cluster
Cold standby

3. FAILOVER



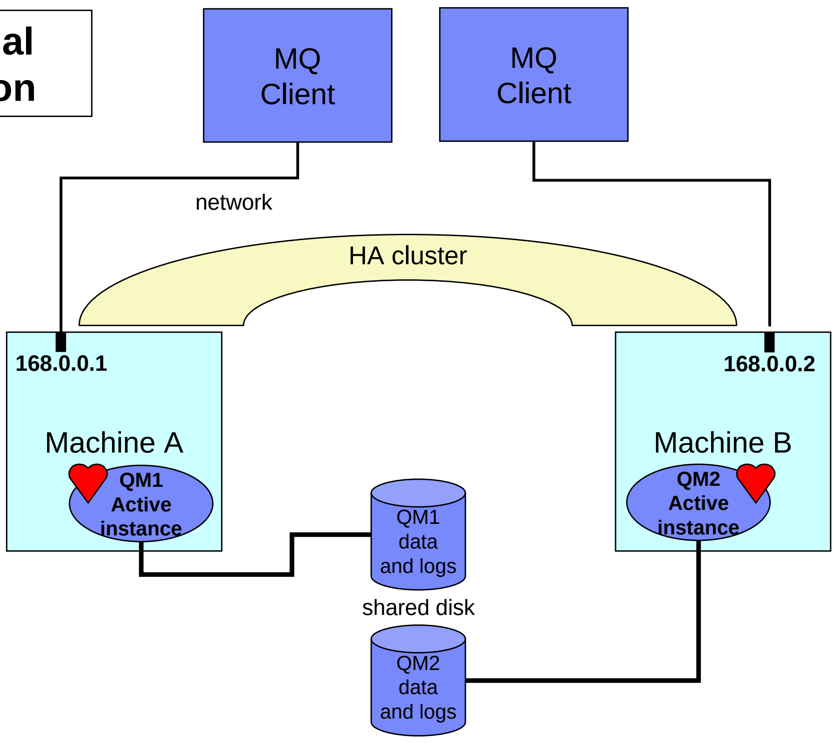
MQ in an HA cluster
Cold standby

4. Recovery complete



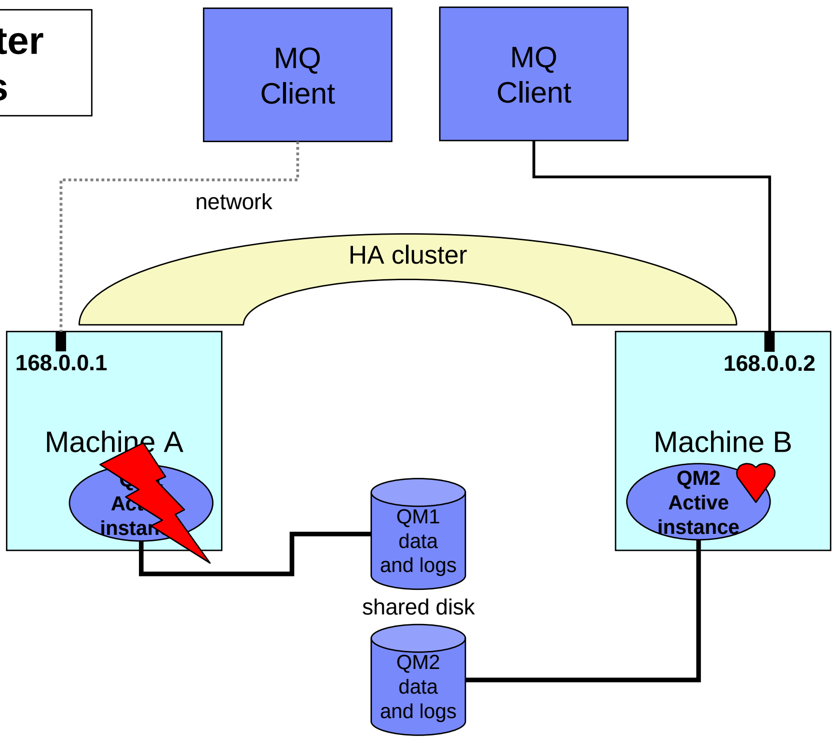
MQ in an HA cluster
Active/active

1. Normal execution

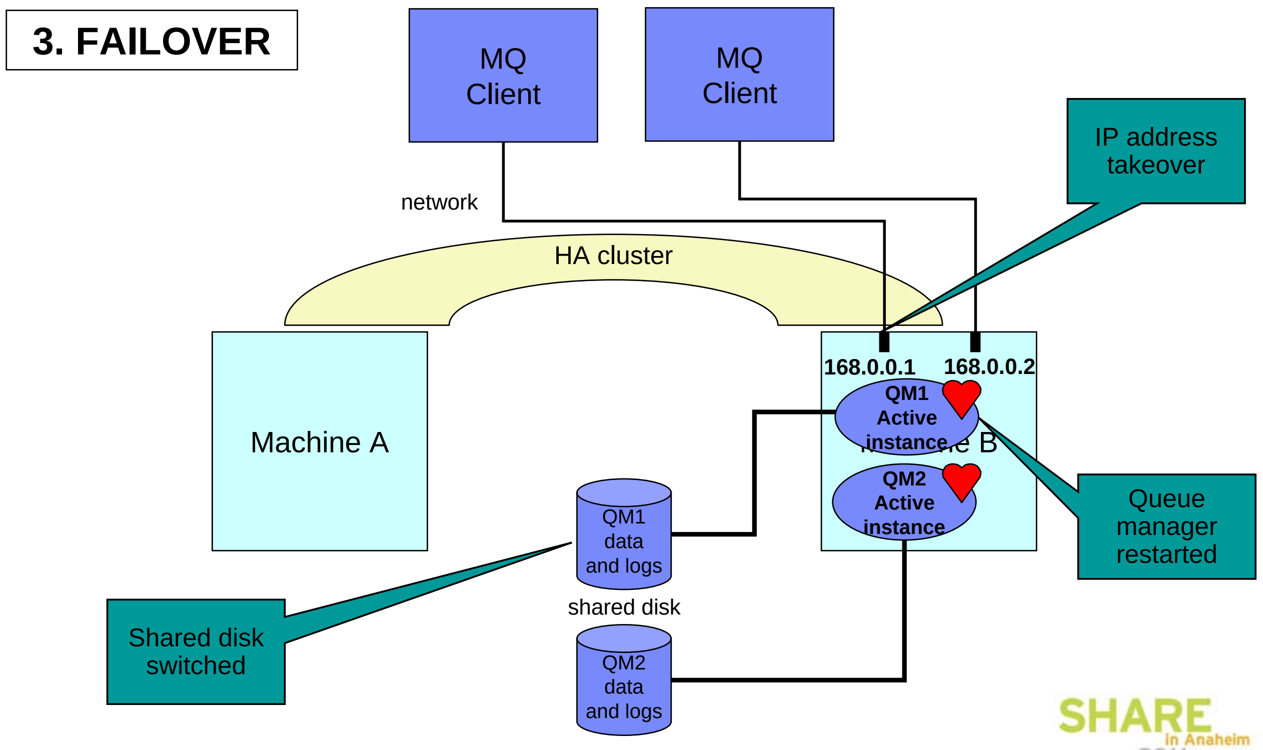


MQ in an HA cluster
Active/active

2. Disaster strikes



MQ in an HA cluster
Active/active



MQ in an HA cluster
Active/active

- This configuration is also sometimes called a "mutual takeover" system
- In normal operation, both machines are running independent queue managers. If one of the systems fails, then this configuration can migrate the failed queue manager to the working machine. So it still appears to applications outside the cluster that you have 2 queue managers. The throughput of each queue manager may degrade (depending on how heavily loaded they run) but at least the work is still getting done.
- With this kind of setup, you probably have a failback capability so that the queue manager can be sent back to its original node when the failure has been corrected. Whether the failback is automatic or not may be your choice, but I'd strongly recommend that it's done manually so that applications which have already connected to the running queue manager do not have their connections broken arbitrarily. You probably want to monitor the workload and only failback when there's not too much work that's going to be disrupted by the failback.
- A number of variations on the themes of cold and hot standby are also possible, for example having 3 nodes to host 2 queue managers (an "N+1" configuration). The availability of these options will depend on the facilities available in your cluster software.
- In this configuration, we've shown the IP address associated with each queue manager being migrated. You will also need to keep a port number reserved for each queue manager (the same number on both machines in the cluster), and have appropriate setup for runmqlsr or a queue-manager listener object.

Multi-instance QM or HA cluster?

- Multi-instance queue manager
  - ✓ Integrated into the WebSphere MQ product
  - ✓ Faster failover than HA cluster and MC91
    - Delay before queue manager restart is much shorter
  - ✗ Runtime performance of networked storage
  - ✗ More susceptible to MQ and OS defects
  - ✗ Only fails over MQ Queue Manager (+related)
- HA cluster
  - ✓ Capable of handling a wider range of failures
  - ✓ Multiple resource managers supported
  - ✓ Failover historically rather slow, but some HA clusters are improving
  - ✗ Some customers frustrated by unnecessary failovers
  - ✗ Require MC91 SupportPac or equivalent configuration
  - ✗ Extra product purchase and skills required

NOTES

- Page intentionally left blank

Comparison of Technologies

	Access to existing messages	Access for new messages
Shared Queues, HP NonStop Server	continuous	continuous
MQ Clusters	none	continuous
	automatic	continuous
HA Clustering, Multi-instance	automatic	automatic
No special support	none	none

Comparison of Technologies

N O T E S

- This picture shows one view of the different capabilities. However you also need to consider factors such as total hardware/software price, the requirement for nonpersistent message availability (remember that they are discarded by a failover-restart), and the requirement for persistent message availability (not if you're using the shared queue support)

How to make your own applications HA

- Application environment
  - What does the application depend on
- Application design
  - Affinities implicit in the application design
  - Message loss
- MQ connectivity
  - What happens when the application loses connectivity

SHARE in Anaheim 2011

N O T E S

- Page intentionally left blank

SHARE in Anaheim 2011

HA applications
Application environment

- Simple applications only need a queue manager connection
- Many business applications have dependencies, such as:
  - Database instance, message broker, application server
  - Configuration information
  - Some data is machine-specific, other data is server-specific
  - Get the ordering of dependencies and timing correct
- How can you tell if it's working
  - Such as PING QMGR
  - Remember that restart might take a little while
- Start/stop operations need to be robust
  - Don't rely on anything!
  - Remember that a 'stop' command might erroneously block
- If you want to put your app in an HA cluster, you'll need to answer these

HA applications
Application environment

N O T E S

- Having a highly available queue manager is not a lot of use unless there are applications which are going to be dealing with the messages. So you need to also think about how you are going to make your applications highly available.
- If you take an education class on something like HACMP then you will be taught how to do this. However, this slide gives some high-level critical design steps you should look at, based on our experiences of making the HA SupportPacs for MQ and the message brokers.
- First off, you need to work out the unit of failover. This will probably be an instance of the "application program" if it is a single executable process or perhaps a related group of processes. This program is often tied to a specific queue manager or broker (perhaps it's a command line option).
- What configuration information needs to be moved around with the application? Is there some that is machine-specific (and hence should not move) and is there some that is instance-specific. Is there any synchronization required between 2 machines? Where is the config information stored? In a file or in a system registry?
- If you are going to be actively monitoring the health of your application so you can recover if it should fail, what is the actual test you will apply? Checking for a process running? Looking to see if a file exists? Remember that if a program is updating a file with its status, you might not be able to rely on that information if the program abends.
- Start and Stop commands need to be able to recover from just about any state the application is in. While you might try a 'graceful' stop, this may need to be repeated in a more forceful way if the operation does not complete in a reasonable time. You might need to do some kind of synchronization between Start and the initial Monitoring - to ensure you don't report a failure just because there's a slow restart.
- Doing operations asynchronously within the HA framework is often a good idea, so that other components in the system can also be monitored or restarted simultaneously. Again though, you might have some dependencies (eg a database) to wait for.

HA applications
Application affinities

- Affinities can be introduced by:
  - The need to continue using the same instance of a service
  - Multiple messages sent to the same application process
  - Conversational-style applications
- Try to avoid Affinities!
  - they cause problems with availability and WLM
  - also cause difficulties using generic channels or ports on z/OS
- Carry any transient state in the message
  - And replicate frequently-read data
- Let other components handle partitioning or sharing of data
  - e.g. store state in a parallel database
- MQ clusters can handle application affinity requirements
  - Use BIND_ON_OPEN option
  - Could program a workload exit to remember previous messages

HA applications
Message loss

- Does your application really need every message delivered?
  - If so, it's quite fragile to failures
  - Queue manager restart will lose nonpersistent messages
  - Message expiry discards old messages
  - Typically, disaster recovery (DR) situations involve message loss
- By careful design of the messages and applications, it is often possible to keep messaging even without failover
  - Some customers use workload balancing and application redundancy instead

HA applications
MQ connectivity

- If an application loses its 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
  - Use MQ automatic client reconnection

Automatic client reconnection

New in 7.0.1 on distributed

- MQ client automatically reconnects when connection broken
  - MQI C clients and JMS clients
- Reconnection includes reopening queues, remaking subscriptions
  - All MQI handles keep their original values
- Can connect back to the 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)

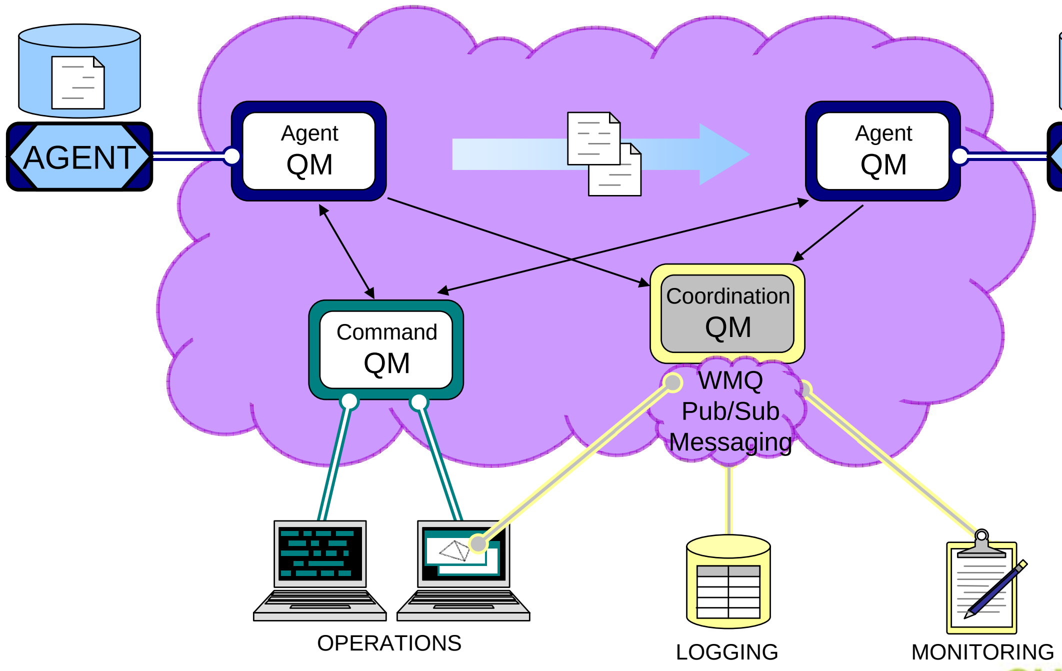
Automatic client reconnection

- 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 or ini file
  - MQI: MQCNO_RECONNECT, MQCNO_RECONNECT_Q_MGR
  - JMS: Connection factories/activation specification properties
- Plenty of opportunity for configuration
  - Reconnection timeout
  - Frequency of reconnection attempts
- Requires:
  - Threaded client
  - 7.0.1 server (distributed or z/OS)
  - full-duplex client communications (SHARECNV >= 1)

WMQFTE Overview for HA



WMQFTE Overview for HA

N O T E S

- Each agent is associated with exactly one “Agent QM”
  - However one QM can host multiple agents
  - FTE Agents can connect as an MQ Client or using bindings to connect to a local QM.
- Connectivity is required between “Command QM” and “Agent QM”
- Transfer results and audit information are sent point to point via “Agents QM” to the “Coordination QM” where it is published to interested parties.
- Some commands connect to the “coordination QM”
  - Eg. Receive information about the MQFTE topology.
- Messages are point-to-point except for the Coordination QM.

Making file transfers highly available

- Since version 7.0.2 WMQFTE has had support for MQ Multi-Instance queue managers
- Configurable for coordination queue managers and agent queue managers.
- Done through the properties file
- If the multi-instance queue manager fails over to the standby then the agent will attempt to connect to the standby instance.
- Also works for the database logger tool

http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/high_availability_qms.htm

SHARE in Anaheim 2011

Making file transfers highly available

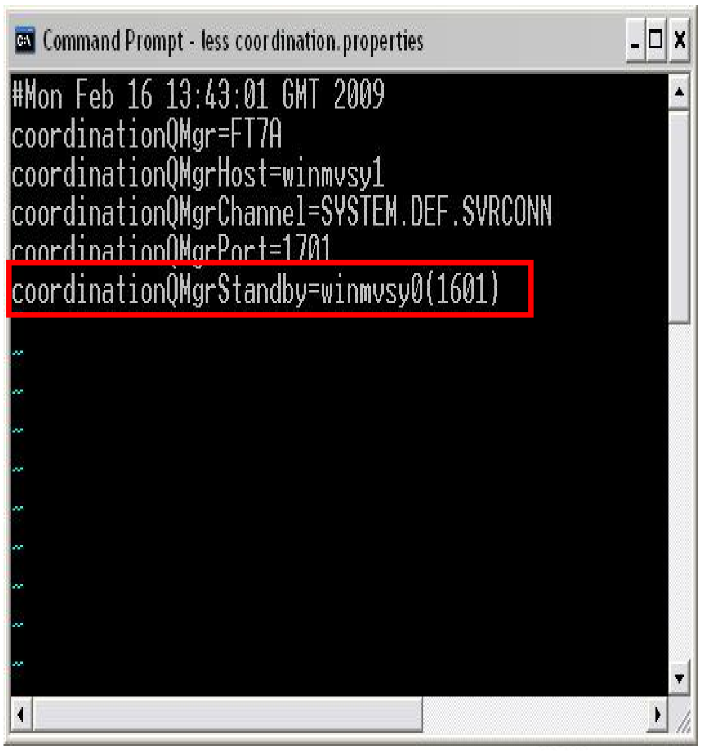
N O T E S

- WebSphere MQ File transfer Edition multi-instance config
  - (7.0.2) Coordination Queue Manager
  - (7.0.2) Agent Queue Manager
  - (7.0.3) Command Queue Manager
- The database logger feature can also use an MQ multi-instance queue manager
- This is fully documented in the online Information center here: http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/topic/com.ibm.wmqfte.admin.doc/high_availability_qms.htm
- Other WMQ HA Reference material
  - HA Solutions Redbook
    - http://www.redbooks.ibm.com/abstracts/sg247839.html?Open
  - Microsoft Cluster
    - http://www.ibm.com/developerworks/websphere/library/techarticles/1101_bareham/1101_bareham.html?ca=drs-

SHARE in Anaheim 2011

Using an multi-instance coordination queue manager

- Configured using the coordination.properties file
- coordinationQMgrStandby property
- Follows the standard MQ notation of <hostname>(<port_number>)



Command Prompt - less coordination.properties

```
#Mon Feb 16 13:43:01 GMT 2009
coordinationQMgr=FT7A
coordinationQMgrHost=winmvsy1
coordinationQMgrChannel=SYSTEM.DEF.SVRCONN
coordinationQMgrPort=1701
coordinationQMgrStandby=winmvsy0(1601)
~
~
~
~
~
~
~
~
~
```

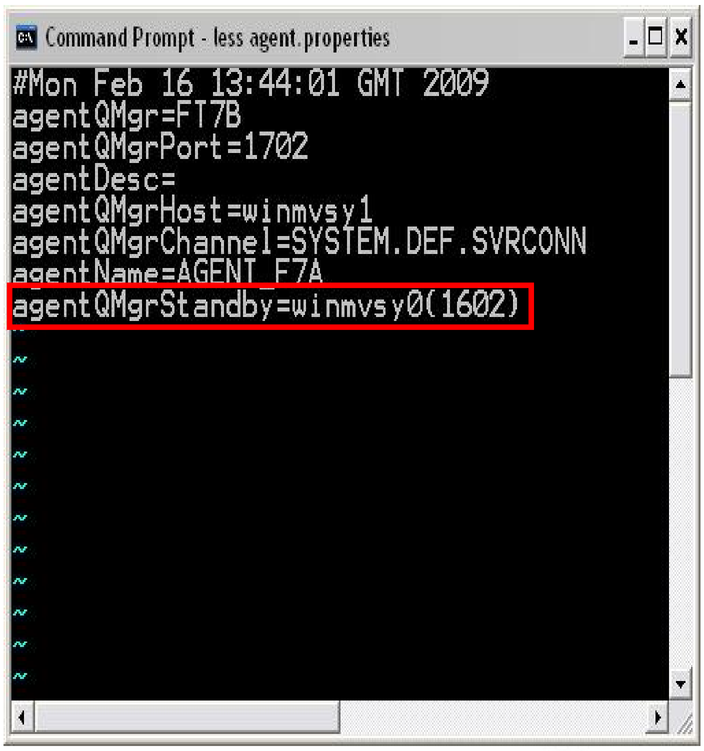
Using an multi-instance coordination queue manager

N O T E S

- To configure the coordination queue manager to use an MQ multi-instance queue manager the coordinationQMgrStandby property needs to be set in the coordination.properties file. The value for this property follows the standard MQ notation of <host_name>(<port_number>)
- The example shows the coordination.properties file for the coordination queue manager FT7A:
- The coordination queue manager is on host winmvsy1, and is available on port 1701.
- The standby instance is on host winmvsy0 and is available on port 1601
- The properties file should contain:
  - coordinationQMgr=FT7A
  - coordinationQMgrHost=winmvsy1
  - coordinationQMgrChannel=SYSTEM.DEF.SVRCONN
  - coordinationQMgrPort=1701
  - coordinationQMgrStandby=winmvsy0(1601)

Using an multi-instance agent queue manager

- Similar to the coordination queue manager setup
- Configured using the agent.properties file
- agentQMgrStandby property
- Follows the standard MQ notation of <hostname>(<port_number>)



```
#Mon Feb 16 13:44:01 GMT 2009
agentQMgr=FT7B
agentQMgrPort=1702
agentDesc=
agentQMgrHost=winmvsy1
agentQMgrChannel=SYSTEM.DEF.SVRCONN
agentName=AGENT_F7A
agentQMgrStandby=winmvsy0(1602)
~
~
~
~
~
~
~
~
~
~
~
```

Making file transfers highly available

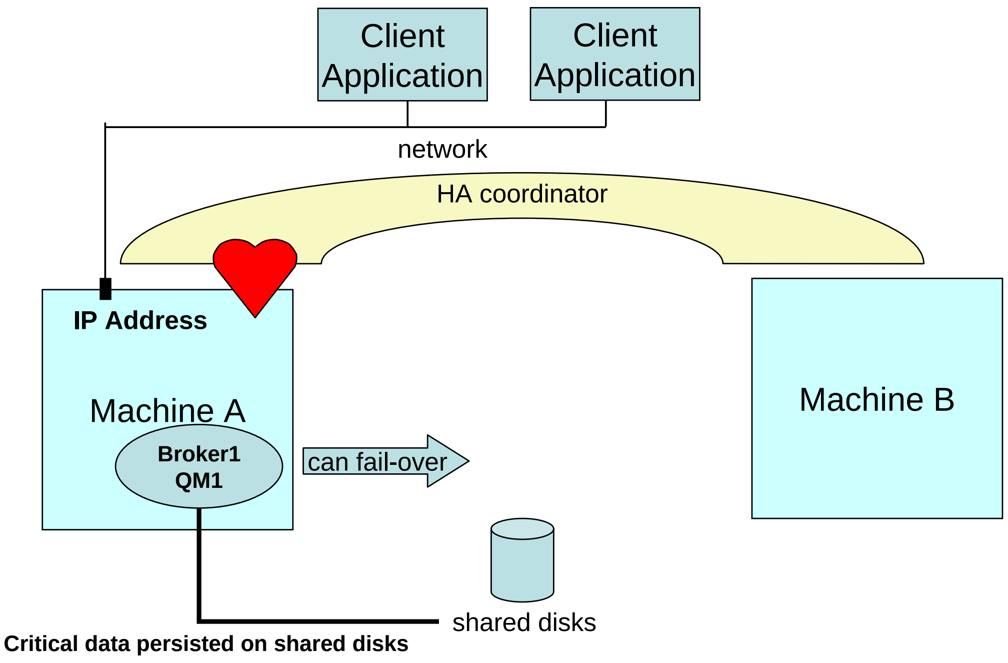
N O T E S

- To configure the agent queue manager to use an MQ multi-instance queue manager the agentQMgrStandby property needs to be set in the agent.properties file. The value for this property follows the standard MQ notation of <host_name>(<port_number>)
- The example shows the agent.properties file for the coordination queue manager FT7B:
- The agent queue manager is on host winmvsy1, and is available on port 1702.
- The standby instance is on host winmvsy0 and is available on port 1602
- The properties file should contain:
  - agentQMgr=FT7B
  - agentQMgrPort=1702
  - agentDesc=
  - agentQMgrHost=winmvsy1
  - agentQMgrChannel=SYSTEM.DEF.SVRCONN
  - agentName=AGENT_F7A
  - agentQMgrStandby=winmvsy0(1602)

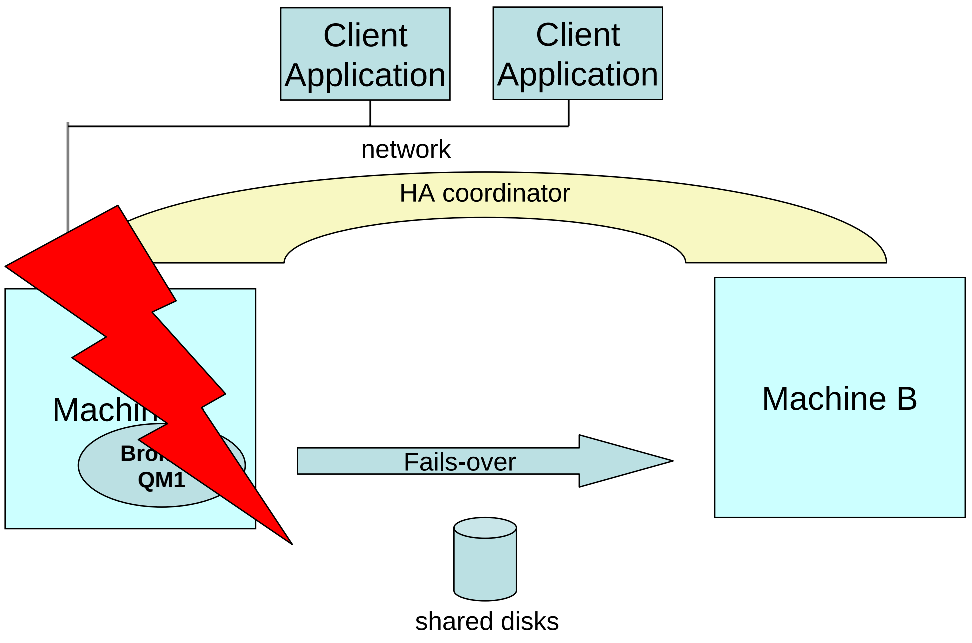
Use of MQ HA by Message Broker

- How is it achieved on Broker.
  - Broker “Global” Data is retained on shared network storage
    - Broker Registry
    - Components directory – Configuration Store
    - Logs, errors and shared-classes remain on local machine.
  - Software HA Support for Broker uses Multi-Instance Queue Managers
    - Pre-requisite : MQ v7.0.1.0 which delivered :
      - *Multi-Instance queue manager*
      - *Automatic client reconnect*

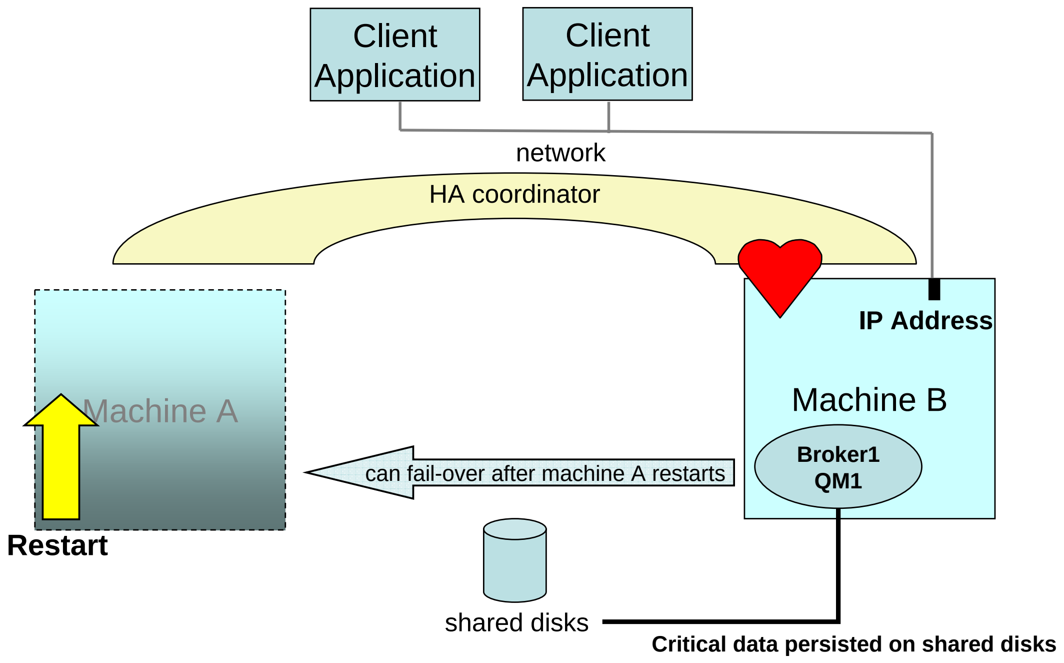
High Availability Cluster Coordination
- active / passive machines



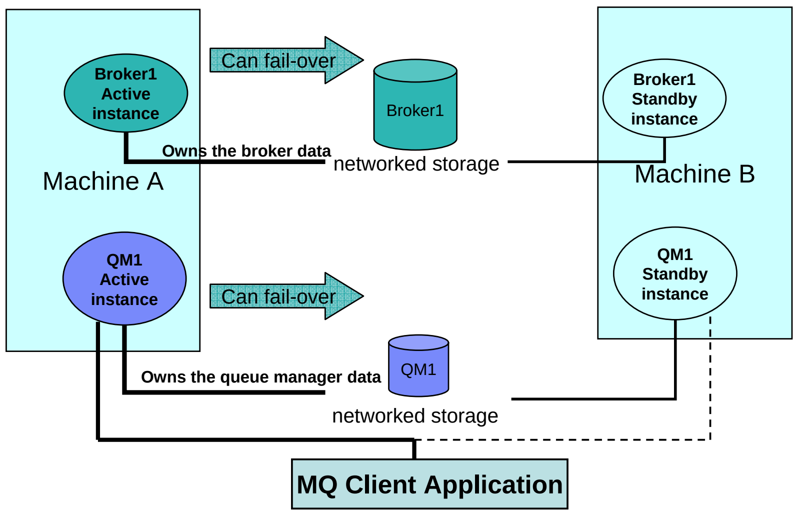
High Availability Cluster Coordination - a failure occurs



High Availability Cluster Coordination - broker fails over to machine B

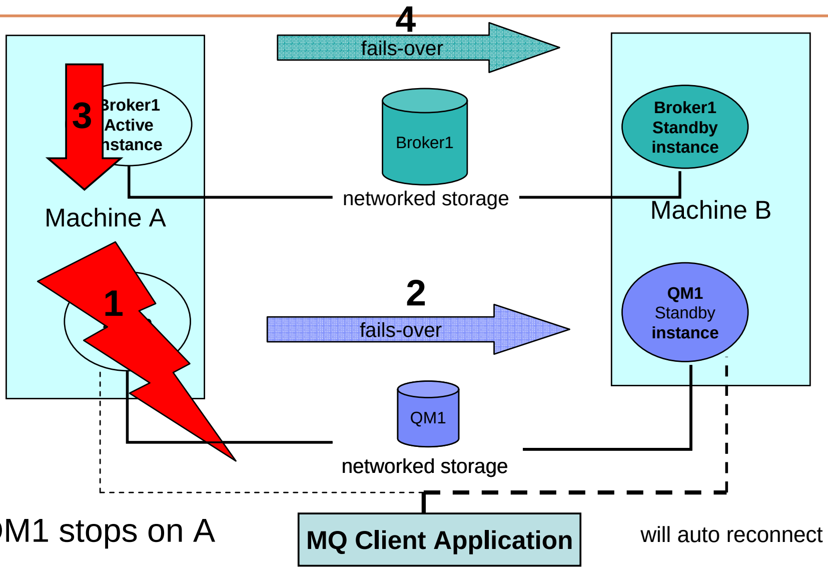


Software HA Support - multi-Instance broker Using multi-instance queue manager



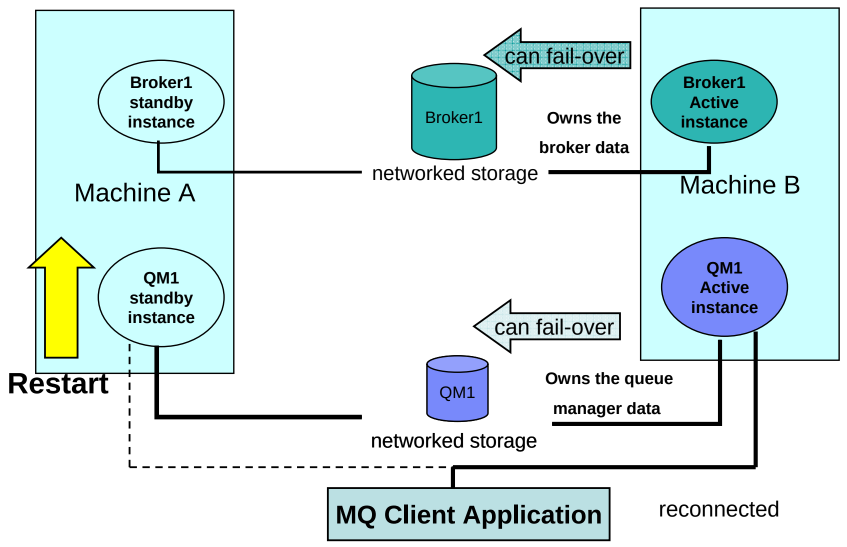
- No need for external HA co-ordinator
- Broker relies on queue manager fail-over capability
- Failover will occur when active queue manager terminates / stops

Software HA Support
- a failure occurs



- 1. QM1 stops on A
- 2. QM1 fails-over to machine B
- 3. Broker1 detects QM1 has stopped . Moves to Standby on machine A
- 4. Broker1 fails-over to machine B

Software HA Support
- queue manager and broker fail over to machine B



- Recovery is complete on machine B. Broker1 and QM1 are active
- Restart QM1 on machine B in standby mode. BK1 still in standby mode

Configuration – The Broker

- **Create multi-instance broker “Broker1” on node A**

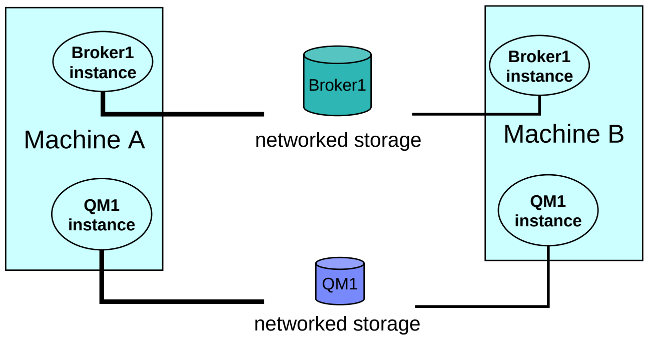
```
-> mqsicreatebroker Broker1 -q QM1 –e <shared network directory>
```

  - creates shared registry and configuration data on shared network storage
  - creates local registry reference to shared network path
  - logs ,error data and shared-classes remain on local machine

- **Create broker instance “Broker1” on node B**

```
-> mqsiaddbrokerinstance Broker1 –e <shared network directory>
```

  - References the broker configuration on the shared network directory
  - Creates local registry reference to shared network path
  - logs, error data and shared-classes remain on local machine



Runtime
- Displaying status of multi-instance broker

- Run mqsilist on Node A

```
->mqsilist
BIP1295I: Broker ‘Broker1' is a multi-instance broker
 running in active mode on multi-instance queue manager
 'QM1'.
BIP8071I: Successful command completion.
```

- Run mqsilist on Node B

```
->mqsilist
BIP1294I: Broker ‘Broker1' is a multi-instance broker
 running in standby mode on multi-instance queue manager
 'QM1'. More information will be available when the broker
 instance is active.
BIP8071I: Successful command completion.
```

Summary

- MQ and operating system products provide lots of options to assist with availability
  - Many interact and can work well in conjunction with one another

- But it's the whole stack which is important ...
  - Think of your application designs
  - Ensure your application works in these environments

- Decide which failures you need to protect against
  - And the potential effects of those failures

- Also look for RedBooks and read the MQ HA whitepaper
  - www.ibm.com/developerworks/websphere/library/techarticles/0505_hiscock/0505_hiscock.html
  - http://www.redbooks.ibm.com/abstracts/sg247839.html