T +44(0) 845 - 868 - 5041
F: +44(0) 870 - 471 - 2039
E: info@w3partnership.com
www.w3partnership.com

Creating an IBM MQ H/A Cluster using a Multi-Instance Queue Manager

W3Partnership have many years experience in the design development and implementation of Middleware technologies, in particular IBM Middleware. We have a series of documents that look at a particular aspect of a technology and how to implement the technology or improve a middleware design. To find more of our documentation visit W3Partnership.com

In this article we will explain how to achieve a High Availability IBM MQ environment utilising a multi-instance queue manager acting as a gateway to an MQ cluster, within a Unix-based (including Linux) or a Windows environment.

Introduction

When building a Production environment, one of the most important aspects for any client, is to ensure that the environment is resilient, efficient and long-lasting. At the same time the administration of the environment should be as easy as possible, as well as effective.

What is High Availability (H/A)

H/A clusters are groups of two or more computers and resources such as disks and networks, connected together and configured in such a way that, if one fails, a high availability manager performs a *failover*. The failover transfers the state data of applications from the failing server to another server in the cluster and re-initiates the operation there. This provides high availability of services running within the H/A cluster. The relationship between IBM MQ clusters and H/A clusters is described below.

What is IBM MQ Clustering

IBM MQ queue manager clusters reduce administration and provide load balancing of messages across multiple queue managers. They can be configured to be highly available, so despite a failure of a queue manager within the MQ cluster, messaging applications can still access surviving instances of a queue manager cluster queue. However, queue manager clusters alone do not provide automatic detection of queue manager failure and automatic triggering of queue manager restarts or failover. H/A clusters provide these features. The two types of clustering – hardware clustering and IBM MQ clustering - can be used together to good effect.

Cluster repository

A repository is a collection of information about the queue managers that are members of a cluster.

The repository information includes queue manager names, their locations, their channels, which queues they host, and other information. The information is stored in the form of messages on a queue called SYSTEM.CLUSTER. REPOSITORY.QUEUE. This queue is one of the default objects which is defined when an IBM MQ queue manager is created.

Typically, two queue managers in a cluster hold a full repository; the remaining queue managers all hold a partial repository.

w3Partnership
EXPERTISE TO INSPIRE CONFIDENCE

T +44(0) 845 - 868 - 5041
F: +44(0) 870 - 471 - 2039
E: info@w3partnership.com
www.w3partnership.com

Full repository and partial repository

A queue manager that hosts a complete set of information about every queue manager in the cluster has a full repository. Other queue managers in the cluster have partial repositories containing a subset of the information in the full repositories.

A partial repository contains information about only those queue managers with which the queue manager needs to exchange messages. A queue manager will automatically request updates to the information they need, so that if it changes, the full repository queue manager sends them the new information. For much of the time, a partial repository contains all the information a queue manager needs to perform within the cluster. When a queue manager needs some additional information, it makes inquiries of the full repository and updates its own partial repository. The queue managers use a queue called SYSTEM.CLUSTER.COMMAND.QUEUE to request and receive updates to the repositories. This queue is one of the default objects and is defined when you create an IBM MQ queue manager.

What are IBM MQ Multi-Instance Queue Managers?

IBM MQ multi-instance queue managers are instances of the same queue manager configured on different servers. One instance of the queue manager is defined as the active instance; another is defined as the standby instance. If the active instance fails, the multi-instance queue manager restarts automatically on the standby server.

A multi-instance queue manager is one part of a high availability solution. Additional components are required in order to build a useful high availability solution.

- Client and channel reconnection to transfer IBM MQ connections to the computer that takes over running the active queue manager instance.
- A high performance shared network file system (i.e. NFS v4) that manages locks correctly and provides protection against media and file server failure.
- Resilient networks and power supplies to eliminate single points of failure in the basic infrastructure.
- Applications that tolerate failover. In particular you need to pay close attention to the behaviour of transactional applications, and to applications that browse IBM MQ queues.
- Monitoring and management of the active and standby instances to ensure that they are running, and to restart active instances that have failed. Although multi-instance queue managers restart automatically, you need to be sure that your standby instances are running, ready to take over, and that failed instances are brought back online as new standby instances.

Note: when multi-instance queue managers are used as a gateway for multiple instances of IBM IIB, the queue managers associated with IBM IIB should be used as the full repositories.

Queue Manager Logging

IBM MQ logs enable recovery of persistent messages from various types of failure. When the system is running properly, the logging process is an overhead that reduces the peak messaging capacity of the system in return for increased reliability. **Circular logging** enables the queue manager to reconcile the status of any outstanding transactions on restart. **Linear logging** enables recovery from this and more drastic outages such as loss of the queue file.

Protection against application, software, or power failure can be achieved with circular logs. Linear logs provide the same functionality, plus protection against media failure (a damaged queue file). Circular logging requires minimum

T +44(0) 845 - 868 - 5041
F: +44(0) 870 - 471 - 2039
E: info@w3partnership.com
www.w3partnership.com

human intervention because the queue manager automatically cycles through log extents, reusing them as needed. Linear logs are never reused and must be deleted or archived periodically. Circular logs also provide faster throughput. The additional performance cost of linear logs is from creating and formatting new extents and, if the logs are saved rather than deleted, moving the log extent to long term storage.

Category	Circular logs	Linear logs
Recovery	Circular logs are used to reconcile units of work that were outstanding at the time of failure. No provision to recover from damaged queue files.	Linear logs contain a copy of all persistent messages that are queued. In a normal restart, linear logs perform the same function as circular logs -- recovery of outstanding units of work. In addition, linear logs support recovery of data when queue files are damaged.
Performance	Circular logs are allocated once and then reused. Therefore no time is required to allocate and format new log extents or to delete or archive them.	New linear logs must be allocated periodically which degrades performance. In addition, the logs must be deleted or moved to prevent filling the underlying file system. Drive head contention during archive operations reduces performance.
Overhead	No administrative overhead is required during normal operations.	Administrators must provide for management of the log files. In addition, the file system must be monitored to prevent the log files from consuming all available space. Human processes touch the administration, operations and support teams.
Operational risk	The loss of a queue file results in loss of all messages on that queue. Loss of a disk partition under the queue files results in loss of all messages on that queue manager.	A normally running queue manager will eventually fill all available disk space if log files are not managed regularly. This will result in an outage of the queue manager if allowed to happen.

This high level comparison should give an idea of which mode to use, but in order to make a sound decision it is important that the costs and the probability of the risk that are involved are understood.

Queue Manager Naming

A Queue Manager name, on all distributed platforms, can be up to 48-characters in length. A recommended structure is for it to be 8-characters in length, upper case, and of the following format:

QMEEPPNN e.g. QMPRGW01

Where:

Mask	Element	Description
QM	Queue Manager	Fixed text to indicate an MQ Queue Manager
EE	Environment indicator	Two-character identifier for the environment. For example: DV – Development ST – System Integration Test (SIT)

w3Partnership
EXPERTISE TO INSPIRE CONFIDENCE

T +44(0) 845 - 868 - 5041
F: +44(0) 870 - 471 - 2039
E: info@w3partnership.com
www.w3partnership.com

		UA – User Acceptance PR – Production
PP	Queue manager type	Identifier to indicate the intended use of the queue manager. For example: GW – Gateway queue manager IB – IIB instance queue manager
NN	Queue manager instance	Instance number of the queue manager, e.g. 01, 02, 03, etc.

Typical IBM MQ Topology

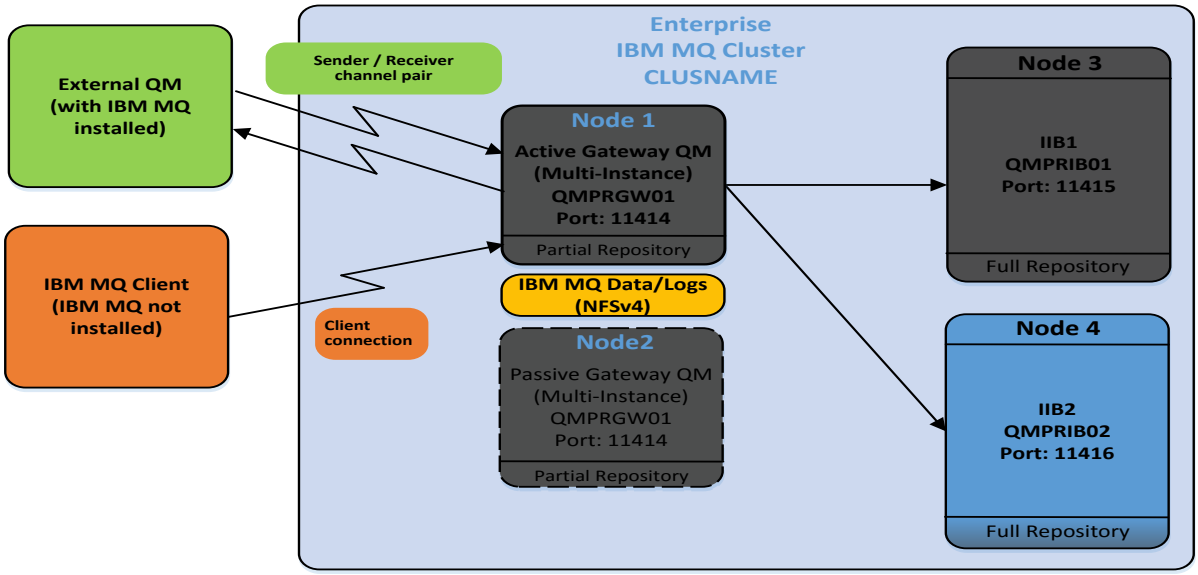

Figure 1 – Typical IBM MQ H/A Cluster Configuration

Assuming that the hardware and environments have been created and updated along with appropriate network connectivity between servers and the shared file system, the following are the steps required in the IBM MQ clustered environment.

Note: IBM MQ should be installed on each node of the enterprise.

Note: Minimum requirement for a complete environment is IBM MQ v7.0.1.3. Previous versions for client / application connections can be used but this may cause issues on the multi-instance connectivity.

Note: All ports and configuration settings are for example only

w3Partnership
EXPERTISE TO INSPIRE CONFIDENCE

T +44(0) 845 - 868 - 5041
F: +44(0) 870 - 471 - 2039
E: info@w3partnership.com
www.w3partnership.com

Create Shared File Systems

Firstly, the shared file systems required for the multi-instance queue manager should be created, assuming a gateway queue manager name of QMPRGW01 in this instance. This task differs between Windows & Unix-based systems:

For Windows & Unix-based systems:

On the NFS server create the following directories for the queue manager data and logs:

/MQHA/QMPRGW01/data	- for the queue manager data
/MQHA/ QMPRGW01/log	- for the queue manager logs

Now carry out the following instructions on Nodes 1 & 2 - this is Windows-centric but the same applies to Unix.

Node 1	Node 2
Log in with user who is a member of the local group mqm.	
Create the following log and data directories in a folder on an NFS drive, C:\MQHA, making sure that the owner is a member of mqm, and mqm has full-control authority to the folders. C:\MQHA\data C:\MQHA\log	
Create a share called MQHA for C:\MQHA. The UNC names are used to refer to the data and log folders – i.e. ***hostname*\MQHA\data** and ***hostname*\MQHA\log**.	Connect to *hostname*\MQHA

Create Multi-instance Queue Manager, QMPRGW01

For Unix-based systems:

Node 1: crtmqm[1] -u SYSTEM.DEAD.LETTER.QUEUE[2] -lc -lp[3] 3 -ls 2 –lf 4096 –md /MQHA/QMPRGW01/data –ld /MQHA/QMPRGW01/log QMPRGW01

Node 2: addmqinf –s QueueManager –v Name=QMPRGW01 –v Prefix=/var/mqm –v DataPath=/MQHA/ QMPRGW01/data/QMPRGW01 –v Directory=QMPRGW01

[1] Once the queue manager has been created, issue the following command: ***dspmqinf –o command QMPRGW01***. Copy the results for use in the subsequent ***addmqinf*** command to be executed on Node 2.
[2] SYSTEM.DEAD.LETTER.QUEUE is used as the "dead-letter queue"; this should be defined for the queue manager as required by the installation
[3] The logging parameters must be calculated according to the specific installation. The default values are shown in these commands.

w3Partnership
EXPERTISE TO INSPIRE CONFIDENCE

T +44(0) 845 - 868 - 5041
F: +44(0) 870 - 471 - 2039
E: info@w3partnership.com
www.w3partnership.com

For Windows-based systems:

Node 1: crtmqm –sa -u SYSTEM.DEAD.LETTER.QUEUE -lc -lp 3 -ls 2 –lf 4096 –md \MQHA\data\QMPRGW01\ –ld \MQHA\QMPRGW01\log QMPRGW01

Node 2: addmqinf –s QueueManager –v Name=QMPRGW01 –v Prefix=\var\mqm –v DataPath=\MQHA\QMPRGW01\data\QMPRGW01 –v Directory=QMPRGW01

Create IIB Queue Managers, QMPRIB01/02

For Unix-based systems:

Node 3: crtmqm -u SYSTEM.DEAD.LETTER.QUEUE -lc -lp 3 -ls 2 –lf 4096 QMPRIB01

Node 4: crtmqm -u SYSTEM.DEAD.LETTER.QUEUE -lc -lp 3 -ls 2 -lf 4096 QMPRIB02

For Windows-based systems:

Node 3: crtmqm –sa -u SYSTEM.DEAD.LETTER.QUEUE -lc -lp 3 -ls 2 –lf 4096 QMPRIB01

Node 4: crtmqm –sa -u SYSTEM.DEAD.LETTER.QUEUE -lc -lp 3 -ls 2 -lf 4096 QMPRIB02

-sa Automatic queue manager start-up - for **Windows** systems only. The queue manager is configured to start automatically when the machine starts up, or more precisely, when the AMQMSRVN process starts up

-u The name of the local queue that is to be used as the dead-letter (undelivered-message) queue.

-lc Use circular logging[4], (or **-ll** for linear logging, often preferable in Production environment)

-lp LogPrimaryFiles - the number of preallocated primary log files

-ls LogSecondaryFiles - the number of secondary log files that can be created for use when the primary log files are full

-lf LogFilePages[5] - the number of 4K pages for each primary and secondary log file

-md Command option for the location on the shared file system for the queue manager configuration data

-ld Command option for the location on the shared file system for the queue manager logs

Configuring the IBM MQ Installation

For each IBM MQ queue manager within the environment the commands required to create and configure the cluster are listed below. The IBM MQ objects required to (a) connect an external queue manager to the cluster via the gateway, and, (b) define further MQ objects (i.e. queues) to the IIB instances are not shown.

Queue Manager QMPRIB01

DEFINE LISTENER (TCP.LISTENER) TRPTYPE (TCP) CONTROL (QMGR) PORT (11415) REPLACE

START LISTENER (TCP.LISTENER)

DEFINE CHANNEL (CLUSNAME.QMPRIB02) CHLTYPE (CLUSSDR) CONNAME ('Node 4 (11416)') CLUSTER (CLUSNAME) REPLACE

DEFINE CHANNEL (CLUSNAME.QMPRIB01) CHLTYPE (CLUSRCVR) CONNAME ('Node 3 (11415)') CLUSTER (CLUSNAME) REPLACE

[4] The decision to have circular or linear logging is made during design and is dependent on the environment and system involved.

[5] The number of LogFilePages differs between environments and between systems. Calculations should be based on the logging requirements of each implemented system. Details of calculations can be found here LogFilePages Calculations

T +44(0) 845 - 868 - 5041
F: +44(0) 870 - 471 - 2039
E: info@w3partnership.com
www.w3partnership.com

DEFINE CHANNEL (CLUSNAME.QMPRGW01) CHLTYPE (CLUSSDR) CONNAME ('Node 1 (11414)') CLUSTER (CLUSNAME) REPLACE

ALTER QMGR REPOS ('CLUSNAME') REPOSNL (' ') MAXMSGL (104857600) PERFMEV (ENABLED) DEADQ ('SYSTEM.DEAD.LETTER.QUEUE')

Queue Manager QMPRIB02

DEFINE LISTENER (TCP.LISTENER) TRPTYPE (TCP) CONTROL (QMGR) PORT (11416) REPLACE

START LISTENER (TCP.LISTENER)

DEFINE CHANNEL (CLUSNAME.QMPRIB01) CHLTYPE (CLUSSDR) CONNAME ('Node 3 (11415)') CLUSTER (CLUSNAME) REPLACE

DEFINE CHANNEL (CLUSNAME.QMPRIB02) CHLTYPE (CLUSRCVR) CONNAME ('Node 4 (11416)') CLUSTER (CLUSNAME) REPLACE

ALTER QMGR REPOS ('CLUSNAME') REPOSNL (' ') MAXMSGL (104857600) PERFMEV (ENABLED) DEADQ ('SYSTEM.DEAD.LETTER.QUEUE')

DEFINE QLOCAL (Q_IN) CLUSTER (CLUSNAME) DEFBIND (NOTFIXED) DEFPSIST (YES) REPLACE

Queue Manager QMPRGW01

DEFINE LISTENER (TCP.LISTENER) TRPTYPE (TCP) CONTROL (QMGR) PORT (11414) REPLACE

START LISTENER (TCP.LISTENER)

DEFINE CHANNEL (CLUSNAME.QMPRGW01) CHLTYPE (CLUSRCVR) CONNAME ('Node 1 (11414)') CLUSTER (CLUSNAME) REPLACE

DEFINE QREMOTE (QAPRGW01) RNAME (' ') RQMNAME (' ') REPLACE

Where:

CLUSNAME Example name for the IBM MQ Cluster

QMPRGW01 Example name for the IBM MQ Multi-instance queue manager

QMPRIB01/02 Example name for the IBM IIB queue managers

Conclusion

The above IBM MQ topology will provide a resilient, efficient and effective highly available clustered environment and will allow for ease of administration and monitoring. The addition of the multi-instance queue managers will allow for client reconnection assuming that connection details have been configured correctly and clients switch between instances once the standby instance takes over.

References and Resources

IBM Redbook IBM MQ V8 Features and Enhancements: IBM MQ V8 Features and Enhancements

IBM MQ V8 Knowledge Centre: IBM MQ V8 Knowledge Centre

Cluster Workload Algorithm: Cluster Workload Algorithm

Testing statement for IBM MQ multi-instance queue manager: IBM MQ Multi-instance Queue Manager

Log File Pages Calculations: LogFilePages Calculations