OpenVMS Technical Journal

HP OpenVMS Clusters and IBM MQSeries Failover Sets

John Edelmann, Technology Consultant

Overview

This article focuses on the use of IBM MQSeries Failover sets within an OpenVMS Cluster environment. The goal of such a configuration is to maximize MQSeries (messaging middleware) availability.

IBM's product, MQSeries (currently re-branded WebSphere MQ) is a robust and guaranteed delivery messaging subsystem, which supports many diverse vendors' operating systems and hardware servers. This article explores and details high availability deployment on OpenVMS Clusters using a feature called Failover Sets. The combination of this IBM-developed failover capability coupled with OpenVMS Cluster availability, results in a very nearly fault-tolerant environment for customers relying on IBM's product for messaging middleware. It is, therefore, a unique solution.

This article assumes that the reader is familiar with OpenVMS Cluster concepts and describes, with examples, how the failover mechanism can be deployed.

MQSeries - Basics

MQSeries (recently renamed WebSphere MQ) is a messaging middleware product whose primary function is the transfer of a datagram from one application to another on one computer system, or from one application to an application running on another computer system.

There are many advantages to such a messaging model, with the primary ones being:

- Ease of use from the application standpoint.
- The benefit of secured, guaranteed, and trusted delivery of an individual message.
- The availability of a robust array of recovery features.

Unlike most other message or file transfer protocols (FTP for example), incorporation of MQSeries as a middleware component aids in decoupling message delivery details from other processing that application programs perform.

MQSeries, as with most such messaging products on the market today, is only as robust as the integrity of the underlying operating system and platform allows. Thus, it is both proper and fortuitous that IBM exploited the unique active-active clustering features of the OpenVMS operating system, when it designed the unique fail-over capabilities embodied in the MQSeries 5.1 release.

MQSeries Clusters

MQSeries includes a powerful clustering capability (not to be confused with OpenVMS Clusters), that provides for a number of queue managers, running on the same or different computer servers/platforms, to share the same named queue object to any application interested in writing a

message to the queue. An application writing a message from the context of a queue manager which actually hosts the queue in question will always put messages to the queue of the local queue manager. This is not the real intent behind clustering, however.

An application putting a message to a queue from a queue manager which is a member of the cluster, but which *does not have a local instance of the queue*, will put the message to the queue hosted by other members of the cluster, in a round robin methodology, and with a certain level of fail over provided. This assumes that processes reading the messages from the cluster queues are able to process them in a similar manner.

MQSeries Clusters are most useful for multi-site operations, where the various sites are connected by a WAN or LAN (thus, single or multiple customers put messages to a multi-site operation for workload distribution and high availability), or where multiple sites with many application queues and independent servers need to share queues among themselves. In both cases, the administration of sender/receiver channel pairs is drastically reduced, as well as the number and complexity of remote queue definitions and other standard MQSeries connectivity requirements.

In spite of the inherent availability and workload balancing MQ Clusters provide, there is nevertheless the opportunity for system crashes and other unplanned outages. When such events occur, applications reading the messages from the clustered queues hosted by the failing machine are no longer available for processing. Queue managers communicating with the failed machine likewise are no longer able to communicate with it.

Due to the reliance on standard inter-process communication heartbeat and keep-alive features that may or may not adequately stop a running channel to a failed system (and hence, its queue manager), messages may inadvertently be left uncommitted in a channel until the channel is activated again. This recovery may take a while, and thus, the messages in transit will become unavailable for a time.

For these reasons, an MQ cluster is a valuable asset, but not the most robust implementation that the industry has to offer.

Please refer to the MQ Clustering Manual or other basic IBM MQSeries product documentation for a more in-depth review of the product set.

OpenVMS Clusters and MQSeries Failover

In OpenVMS, in spite of the robustness of the cluster-locking mechanism, a given queue manager can run only on one node of an OpenVMS Cluster at a time. However, each system can run one or more *different* queue managers.

This becomes especially valuable with multiple queue managers running on separate nodes in the cluster, all being members of an MQSeries Cluster. In such a configuration, OpenVMS clustered applications can read messages being written to MQ Cluster shared queues, hosted by queue managers distributed throughout the OpenVMS cluster. This arrangement provides great processing power among applications requiring shared memory or other resources, while maximizing availability to members of the MQ Cluster on other distributed systems (whether they are running OpenVMS or not).

But an even greater development was deployed in Version 5.1. This feature, called Failover Sets, enables a queue manager to be automatically restarted on another OpenVMS Cluster node if a failure occurs. While the feature can be used with or without MQ Clusters, it certainly enhances the overall efficiency and power of the solution, in either case.

A Failover Set is defined as a single queue manager and the nodes across which it can run. There can be multiple Failover Sets on a given OpenVMS Cluster or standalone system, but each Failover

Set can control only a single queue manager. The *set* component of the name refers to the set of OpenVMS Cluster nodes on which the queue manager can run (it can run only on one node in the cluster at a time).

One or more Failover Sets can be configured into an MQ Cluster, which provides the most robust configuration, mentioned previously.

There are several important issues with regard to MQ 5.1 on OpenVMS that must be considered in advance. At the time of this printing, ECO 2 was available (with support on OpenVMS 7.3-1) and ECO 3 was in process. The latter release is intended to support OpenVMS 7.3-2, as well as to address various important internal issues.

Manageability Tools on OpenVMS

As with most distributed platforms, MQSeries on OpenVMS can be managed locally via the **runmqsc** command utility, or, it can be managed remotely via the IBM MQSeries Explorer tool. This tool runs on Windows NT, 2000, and XP and uses the SYSTEM.ADMIN.SVRCONN channel to allow privileged access to the queue manager. Usually, it is necessary to create a *userid* within the UAF of the OpenVMS system identical to that of *userid* of the windows session and for the OpenVMS user to be granted the MQM identifier. It is also essential that the MQM identifier be granted with the resource attribute. Failure to do so will cause problems during certain remote management tasks (modifying, stopping, or starting channels, especially).

Considerations for MQ 5.1 on OpenVMS

In an OpenVMS Cluster, it is critical that a common SYSUAF and Rightslist be used across all nodes of the cluster. If this is not possible, the MQM identifier, *userid*, and UICs must be identical on all systems. This commonality must be in place before any MQSeries is installed on another node in the cluster. If this is not done, the new installation will not have the ownership it requires in the common MQS_ROOT directory.

When defining the MQS_ROOT as a device NOT on the system disk, the LIB subdirectory is not created in the proper location. This causes problems with the command server and other issues relative to data conversion. The LIB subdirectory tree must be placed (copied) under the MQS_ROOT:[MQM] directory tree. The location of the directory following the install is SYS$SYSROOT:[MQM].

It is recommended to use **runmqlsr** to initiate channel listening programs, specifically in the START_QM.COM procedure by uncommenting the provided commands after the queue manager starts. This creates a *<queue-manager_LS>* process. If this feature is used, the **endmqlsr** command must also be uncommented in the END_QM.COM procedure. A 20-second wait added just before the **endmqlsr** command may assist with some timing difficulties (to allow the queue manager to fully shutdown).

When configuring ports used for multiple queue managers (Failover Sets) running in a single OpenVMS Cluster environment, it is important to assign different ports to each Failover Set, especially if the same LAN is used for all such connections. During a failover condition, when the potential exists for having more than one queue manager running on a single system, different port assignments must be used to ensure separation between receiver channels using IP aliases to the same interface, regardless of the number of interfaces physically used on a given server.

Failover Set Details

The basic idea behind a Failover Set is to permit a given queue manager to be restarted on another node in the OpenVMS Cluster if a failure is encountered by the server on which it is originally running. This is accomplished from a data standpoint by the inherent cluster-wide access afforded by the lock manager in an OpenVMS cluster. It is accomplished from a network connectivity standpoint by incorporating alias IP addressing on a given Internet interface (details are provided in the manuals for varying TCP/IP product variants).

Thus, once a failure is detected, and a queue manager is moved to another node in the cluster, the queue manager is started, and the designated IP address for the Failover Set (which is specifically *not* the same as the IP address of the node itself) is added as an alias to one of the network interfaces on the machine.

This permits the re-connection (automatically with a 2-3 minute delay) by distributed MQ clients and servers using a fixed IP address to the same queue manager, on a different node in the cluster, with neither a client or distributed server being aware of the hardware hosting change. They experience only a channel interruption that will clear either by a manual restart or an automatic retry. It is critical that any such failover complete successfully; a residual alias to the wrong interface (regardless of server) can cause inbound channels to fail. On OpenVMS, the INET master process reports evidence of such a situation, via OPCOM.

A Failover Set consists of an initialization file (FAILOVER.INI), and 3 command procedures, (START_QM.COM, TIDY_QM.COM, END_QM.COM). A Failover Set requires a failover monitor process, which runs each node in the OpenVMS Cluster designated to run the queue manager in question. The resulting process monitors the status of the queue manager to which it applies. To establish a Failover Set, the template FAILOVER.TEMPLATE in MQS_EXAMPLES: is copied to:

`MQS_ROOT:[MQM.QMGRS.qmgrname]FAILOVER.INI`

It is customized for the Failover Set in question. These instructions are explained in Chapter 16 in the *MQSeries for Compaq OpenVMS Alpha, Version 5 Release 1 System Administration Guide*.

The FAILOVER.INI file contains all symbol definitions used to establish the Failover Set. These include log directories for errors (which includes the startup and shutdown logs), IP addresses for the interfaces that will support the queue manager, and any other startup information required for automated startup and shutdown. An example FAILOVER.INI file is appended to the end of this article.

The following figures show simple Failover Set configurations. A normal configuration (Figure 1) and a failed configuration (Figure 2) are provided.

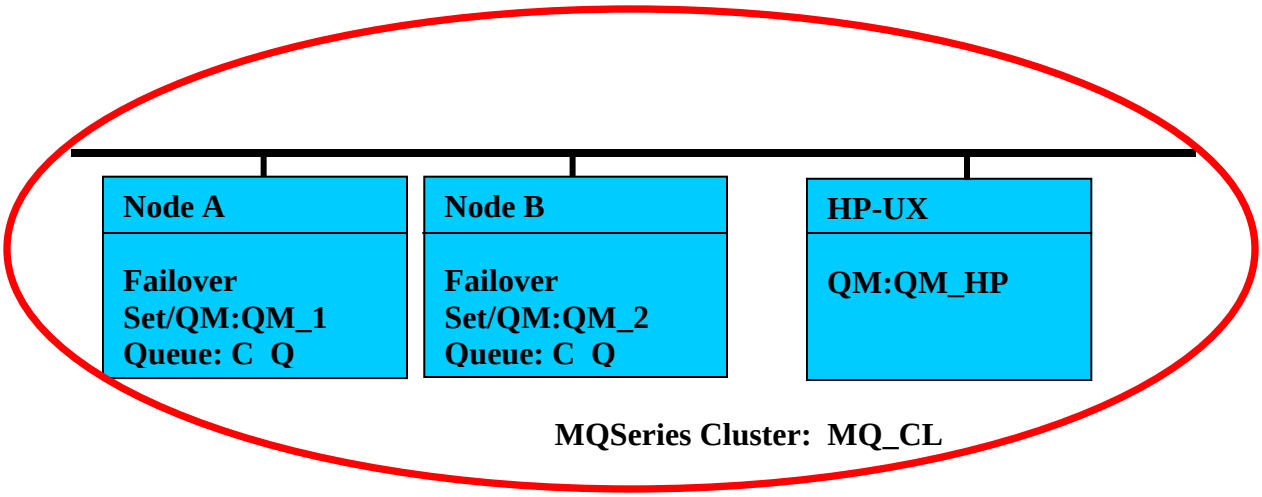

Figure 1 – Normal Configuration

Node A and B are OpenVMS Cluster members. Node A and B each support a distinct Queue Manager Failover Set, each of which is a member of an MQ Cluster (MQ_CL), which includes an HP-UX server, running queue manager QM_HP.

Applications running on QM_HP put messages to the cluster queues, C_Q, hosted by QM_1 and QM_2, in a round robin manner. The naming convention used for queue managers and node names is specifically decoupled, so as to reduce unnecessary linkage between a Failover Set and the node on which it runs.

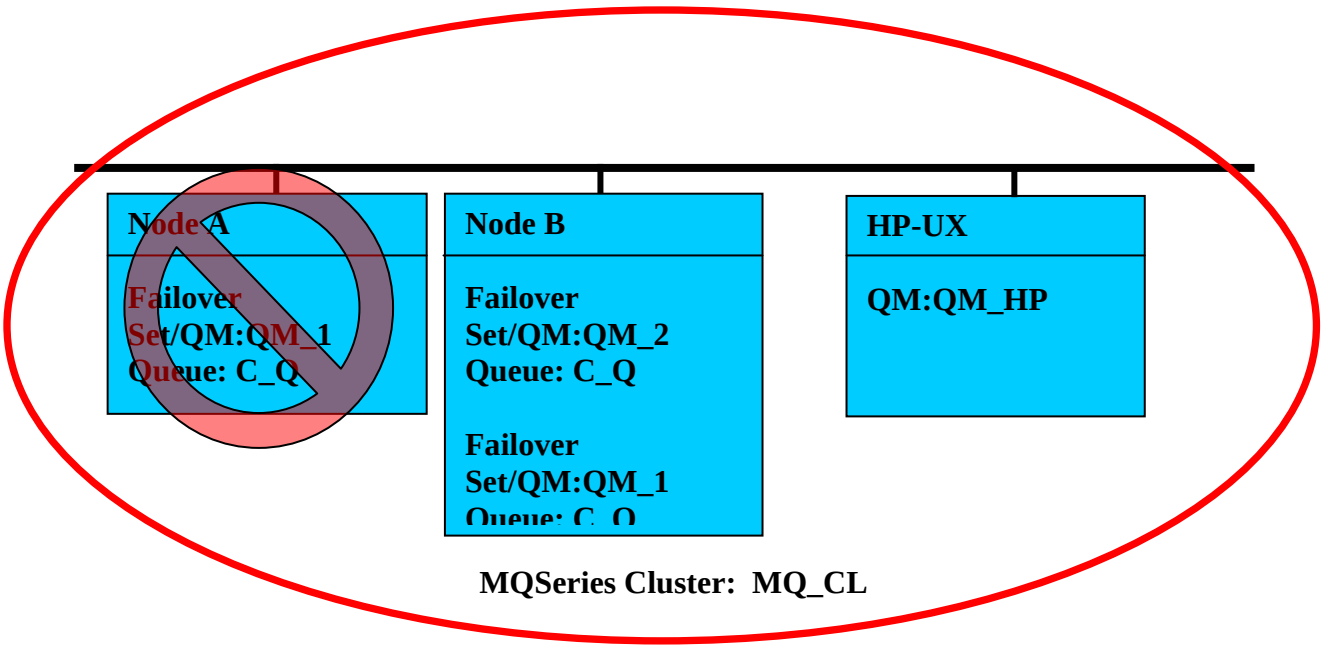

Figure 2 – Failover Configuration

Node A has crashed or is otherwise unavailable. The Failover monitor for QM_1 running on Node B has determined that Failover Set QM_1 is no longer running on Node A. Because the Failover Set was configured to run on Node A at priority 1 and Node B at priority 2, the queue manager QM_1 will now start on Node B, using the START_QM procedure specified for QM_1. This procedure ensures that the proper IP address alias is established so that listener programs respond to channel start requests appropriately. Thus, both OpenVMS MQSeries Cluster queue managers are now running on Node B. Data that was in process on QM_1 is now accessible to processes running on Node B, and queue manager availability is sustained throughout the failure of Node A. Any links that existed between QM_HP and QM_1 when it was running on Node A are re-established automatically to Node B, where the queue manager is presently running.

Summary

In a distributed MQSeries Clustered environment, one might be led to believe that the round-robin and failover capability inherent in that configuration would be adequate for high availability solutions. However, should a processing latency exist for messages delivered to a queue on a node in the OpenVMS Cluster that subsequently fails, those messages would be unavailable until the node rejoins the cluster. With the Failover Set feature, such messages can be accessible as soon as the queue manager on the failed server is restarted on another node in the OpenVMS Cluster.

This is a unique feature that is available with OpenVMS Clusters, owing to their Active-Active clustering design.

Again, much more detailed concept descriptions and configuration guides are available in Chapter 16 in the *MQSeries for Compaq OpenVMS Alpha, Version 5 Release 1 System Administration Guide*.

For more information

For more information about this topic, please contact John Edelmann at john.edelmann@hp.com.

Examples

Below is a display of the processes across the clusters that are related to the queue managers running on a two-node cluster. The queue managers are QMD0VMS1 and QMD0VMS2 respectively. The _FM processes are the Failover Set monitors. Note that a monitor for each Failover Set runs on each node of the cluster, whether or not that node is presently running the queue manager at the time or not.

```
DAYEDI> sho sys/proc=q*/cluster

OpenVMS V7.3  on node DAYEDI  20-MAY-2002 13:23:26.06  Uptime  10 02:31:49

  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
21600405 QIO$CONFIGURE   HIB      9       67   0 00:00:00.01       156     64 M
21600411 QUEUE_MANAGER   HIB     10      185   0 00:00:00.04       133    170
21600454 QMD0VMS1_FM     HIB      7      211   0 00:00:02.09       362    327
21600455 QMD0VMS2_FM     HIB      7      132   0 00:00:00.18       360    325
21600462 QMD0VMS1_EC     HIB      6    77241   0 00:00:00.43     54686    160
21600463 QMD0VMS1_LG     HIB      6      299   0 00:00:00.03       449    480  S
21600464 QMD0VMS1_CP     HIB      6      494   0 00:00:00.06       412    443  S
21600465 QMD0VMS1_RM     HIB      6    76189   0 00:00:00.22       477    496 MS
21600466 QMD0VMS1_CI     HIB      6    75981   0 00:00:00.08       569    437 MS
21600467 QMD0VMS1_AG     HIB      6   253029   0 00:00:25.82      1046    887 MS
21600468 QMD0VMS1_LS     HIB      6      735   0 00:00:00.16       580    499 M
21600469 QMD0VMS1_CS     HIB      6   142587   0 00:00:00.08       617    403

OpenVMS V7.3  on node DAYVMS  20-MAY-2002 13:23:26.07  Uptime  10 02:44:22

  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
21400105 QIO$CONFIGURE   HIB      8       67   0 00:00:00.03       230     62 M
21400111 QUEUE_MANAGER   HIB     10      354   0 00:00:00.38       165    196
2140015D QMD0VMS1_FM     HIB      6      104   0 00:00:02.97       476    249
2140015E QMD0VMS2_FM     HIB      6      133   0 00:00:03.35       447    249
21400160 QMD0VMS2_EC     HIB      6    77381   0 00:01:18.78     26676    552
21400161 QMD0VMS2_LG     HIB      6      277   0 00:00:00.63       588    306  S
21400162 QMD0VMS2_CP     HIB      6      496   0 00:00:00.61       598    304  S
21400163 QMD0VMS2_RM     HIB      5    76217   0 00:00:26.04       827    350 MS
21400164 QMD0VMS2_CI     HIB      6    76048   0 00:00:26.08       977    311 MS
21400165 QMD0VMS2_AG     HIB      5   250657   0 00:02:59.54      1942    566 MS
21400166 QMD0VMS2_LS     HIB      6      230   0 00:00:00.54       670    310 M
21400167 QMD0VMS2_CS     HIB      6   161593   0 00:00:37.77       764    304
```

Example Failover Set initialization file for Queue Manager QMD0VMS1:

```
# OpenVMS Cluster Failover Set Configuration information – QMD0VMS1
# ------------------------------------------------------
#
# The TCP/IP address used by the OpenVMS Cluster Failover Set
#
IpAddress=nnn.nnn.nnn.nnn
#
# The TCP/IP port number used by the MQSeries Queue Manager
#
PortNumber=1415
#
# The timeout used by the EndCommand command procedure
#
TimeOut=30
#
LogDirectory=mqs_root:[mqm.QMGRS.QMD0VMS1.errors]
#
# The number of nodes in the OpenVMS Cluster Failover Set. The
# number of nodes defined below must agree with this number
#
NodeCount=2
#
# The Name of the OpenVMS node
#
NodeName=DAYEDI
#
# The TCP/IP interface name for the node
#
Interface=ie0
#
# The priority of the node
#
Priority=1
#
# The Name of the OpenVMS node
#
NodeName=DAYVMS
#
# The TCP/IP interface name for the node
#
Interface=we1
#
# The priority of the node
#
Priority=2
```

Example Failover Set initialization file for Queue Manager QMD0VMS2:

```
# OpenVMS Cluster Failover Set Configuration information – QMD0VMS2
# ------------------------------------------------------
#
# The TCP/IP address used by the OpenVMS Cluster Failover Set
#
IpAddress=nnn.nnn.nnn.nnn
#
# The TCP/IP port number used by the MQSeries Queue Manager
#
PortNumber=1416
#
# The timeout used by the EndCommand command procedure
#
TimeOut=30
#
LogDirectory=mqs_root:[mqm.QMGRS.QMD0VMS2.errors]
#
# The number of nodes in the OpenVMS Cluster Failover Set. The
# number of nodes defined below must agree with this number
#
NodeCount=2
#
# The Name of the OpenVMS node
#
NodeName=DAYVMS
#
# The TCP/IP interface name for the node
#
Interface=we1
#
# The priority of the node
#
Priority=1
#
# The Name of the OpenVMS node
#
NodeName=DAYEDI
#
# The TCP/IP interface name for the node
#
Interface=ie0
#
# The priority of the node
#
Priority=2
```