OpenVMS Technical Journal V11

WebSphere MQ and OpenVMS Failover Sets

John Edelmann, Technology Consultant

Overview

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, the primary ones being related to ease of use from the application standpoint; the benefit of secured, guaranteed, and trusted delivery of an individual message; and the availability of a robust array of recovery features.

As with most such messaging products on the market today, MQ is only as robust as the underlying operating system and platform integrity permits. Thus, it is both proper and fortunate that IBM exploited the unique active-active clustering features of the OpenVMS operating system, when it designed the unique failover capabilities embodied in the 5.3 release.

Because the features of OpenVMS Clusters are well known to the OpenVMS technical community, this document focuses on the WebSphere MQ capabilities designed in the OpenVMS implementation.

WebSphere MQ Clusters

This report is not intended to replace the MQ Clustering Manual or other basic IBM WebSphere MQ product documentation. Please refer to these documents for a more in depth review of the product set.

WebSphere MQ includes a clustering capability (not to be confused with OpenVMS Clusters) that allows a number of queue managers, running on the same or different computer servers or platforms, to share the same named queue object to any application that must write a message to the queue. An application writing a message from the context of a queue manager that actually "hosts" the queue in question always puts 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 that is a member of the cluster, but that *does not have a local instance of the queue*, puts the message to the queue hosted by other members of the cluster, in a round-robin methodology and with a certain level of failover provided. This assumes that processes reading the messages from the cluster queues can process them in a similar round-robin fashion. If messages need to be processed in a certain order (called message affinity), other clustering features (such as bind options) need to be incorporated in the application model.

WebSphere MQ Clusters are most useful for multiple-site operations, where the various sites are connected by a WAN or LAN (thus, single or multiple "customers" put messages to a multiple-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 are the number and complexity of remote queue definitions and other, more typical WebSphere MQ connectivity requirements.

In spite of the inherent availability and workload balancing that 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 failed machine are no longer available for processing. Queue managers that were connected to the failed machine likewise are no longer able to communicate with it.

Because the reliance on standard interprocess communication heartbeat and keep-alive features, which may or may not adequately stop a running channel to a failed system (and hence, its queue manager), messages can inadvertently be left uncommitted in a channel until the channel is activated

again. This recovery can take a while, and thus the messages in transit will become unavailable for a time.

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

OpenVMS Clusters and WebSphere MQ Failover

In OpenVMS, in spite of the robustness of the cluster locking mechanism, the WebSphere MQ product requires that a given queue manager can run on only one node of an OpenVMS Cluster at a time. (A typical nonclustered server environment has only one instance of a given queue manager running as well on a single server.) However, as with other WebSphere MQ implementations, each system can run one or more *different* queue managers.

This feature becomes especially valuable with multiple queue managers running on separate nodes in the cluster, all being members of a WebSphere MQ 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 that require shared memory or other resources, while maximizing availability to members of the MQ Cluster on other distributed systems, whether or not they are running OpenVMS.

An even greater development was deployed in WebSphere MQ version 5.3. This new feature, called failover sets, enables a queue manager to be restarted automatically on another OpenVMS Cluster node if a node failure occurs. Although the feature can be used with or without MQ Clusters, it certainly enhances the overall efficiency and power of the solution.

A failover set is 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 names refers to the set of OpenVMS Cluster nodes on which the queue manager can run. However, one or more failover sets can be configured into an MQ Cluster, which, in my opinion, provides the most robust configuration.

There are several important issues with regard to MQ 5.3 on OpenVMS that must be considered in advance.

Failover Set Details

Failover sets are intended to permit a given queue manager to be restarted on another node in the OpenVMS Cluster if a failure occurs on the server on which it is running. From a data standpoint, this is accomplished by virtue of the inherent clusterwide access afforded by the lock manager in a cluster. From a network connectivity standpoint, it is accomplished by incorporating alias IP addressing on a given internet interface. (For more information, see the manuals for various TCP/IP product variants.)

Each failover set has a failover *monitor* that keeps track of the status of the failover set to which it is assigned. As long as this monitor process is active, only the provided FAILOVER commands can be used to stop, start, or inquire about the status of the queue manager failover set. (The use of standard MQ commands like `strmqm`, `endmqm`, and so on, is not permitted.)

After 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/server itself) is added as an alias to one of the network interfaces on the machine using standard IP `ifconfig` commands.

Migrating the IP address for the failover set permits reconnection by distributed MQ clients and servers using a fixed IP address to the same queue manager channels, on a different node in the cluster. Distributed MQ connections that were active experience only a channel interruption that clears either by manual restart or automatic retry.

A failover set consists of an initialization file and three associated command procedures (START_QM.COM, TIDY_QM.COM, and END_QM.COM). To establish a failover set, the template FAILOVER.TEMPLATE in MQS_EXAMPLES: is copied to FAILOVER.INI and is then modified for the required changes. Required logicals are defined and the other template procedures are customized.

After the systemwide MQS_STARTUP.COM procedure runs, the `runmqfm` command is issued to start the failover monitor on the correct node for the designated failover set. Then the `failover` command starts the queue manager, the associated listener, and any dependent applications (runs the START_QM.COM procedure).

Changes to any of the template files coded prior to starting the failover monitor requires the monitor to be restarted. Thus, the failover monitor also monitors the integrity of the failover set command environment.

A Simple Example

The following example will illustrate the failover operation using a two node OpenVMS cluster supporting two WMQ queue managers, with a distributed HP-UX queue manager in the same WMQ cluster.

In Figure 1, Node A and Node B are OpenVMS Cluster Members. Node A and B each support a distinct queue manager failover set. Each set 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 to reduce unnecessary linkage between a failover set and the node on which it runs.

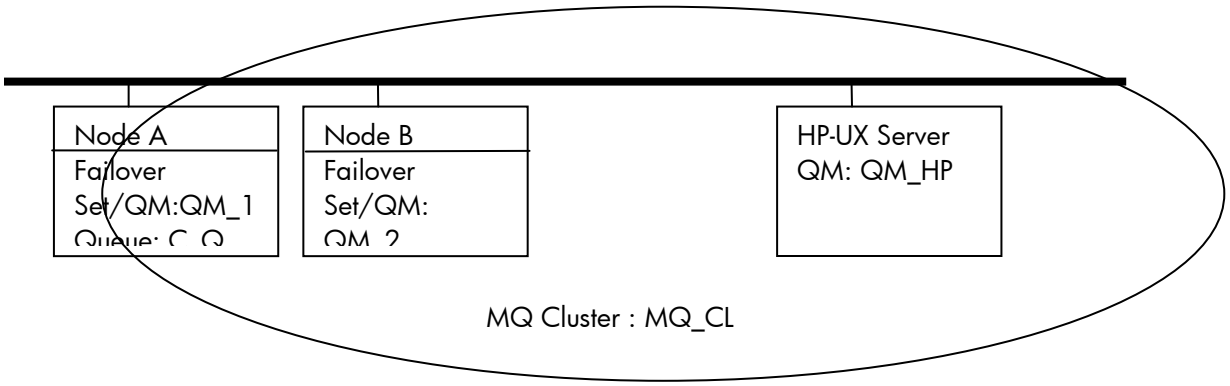

Figure 1 – Normal Configuration

In Figure 2, Node A has failed or is otherwise unavailable. The failover monitor running on Node B determines that failover set QM_1 is no longer running. Because the failover set is configured to run on Node A at priority 1 and on Node B at priority 2, the queue manager QM_1 now starts 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 request appropriately. Thus, both OpenVMS WebSphere MQ Cluster queue managers are now running on Node B.

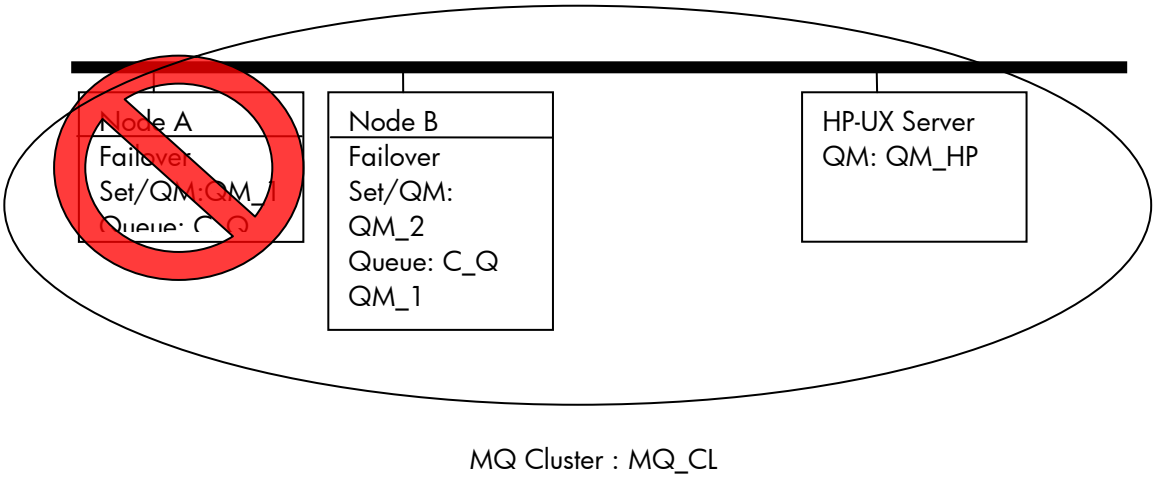

Figure 2 – Failover Configuration

The following aspects of WebSphere MQ 5.3 require more explanation than what is provided in the product documentation:

It is critical that the latest WebSphere MQ 5.3 patch be applied.

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 and user name, the MQS_SERVER user name, and their respective UICs must be identical across all systems. This commonality must be in place before any WebSphere MQ 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:[MQM] directory.

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. In version 5.3, MQ on OpenVMS has adopted the same syntax as other distributed platforms, namely the use of the ampersand (&) to initiate the listener processes as detached processes. When IBM releases WebSphere MQ Version 6.0, listener objects will likely be supported. At that time, the starting and stopping of listeners will be coordinated with the corresponding actions by the queue manager, and use of these commands in the start and end scripts will no longer be necessary.

When configuring the port 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.

Summary

In a distributed WebSphere MQ clustered environment, one might be led to believe that the round-robin and failover capability inherent in that configuration is adequate for high-availability solutions. However, if a processing latency exists for messages delivered to a queue on a node in the OpenVMS Cluster that subsequently fails, those messages are unavailable until the node rejoins the cluster. With failover sets, such messages can be accessible as soon as the queue manager on the failed server is restarted on another node in the OpenVMS Cluster. In such an arrangement, the application processes that utilize such messages must be cluster aware and able to migrate to the node where the queue manager is being restarted.

For this reason, it is best to start such applications from within the START_QM.COM procedure itself (and end them in the accompanying END_QM.COM).

Again, much more detailed concept descriptions and configuration guides are available in the *WebSphere MQ for HP OpenVMS System Administration Guide, Version 5 Release 3*, but it is my hope that this paper is a useful starting point for those new to the failover capabilities of WebSphere MQ on OpenVMS.

For more information

John Edelmann can be reached via email at john.edelmann@hp.com.

For additional information, see the following website and select the Platform Specific section, for HP OpenVMS:

http://www-306.ibm.com/software/integration/wmq/library/library6x.html