Providing more processing power through clustered queue managers.

By Andrew Banks, WebSphere MQ Development
Providing more processing power through clustered queue managers.

Introduction

More is usually better. Having more queue managers can give your systems higher availability and more processing power. But it also brings complexity. With a large network, there is a greater chance of system malfunction somewhere along the line. Individual servers should be connected and mirrored to prevent catastrophic damage. But how can you link numerous systems in remote places while retaining access and control?

IBM WebSphere® MQ (formerly known as IBM MQSeries®) clusters are an effective way to deal with this complexity. WebSphere MQ clusters create the capability to collect queue managers to provide scalability and high availability that can help to reduce complexity and lower your administration costs. This white paper describes some situations that require information technology (IT) managers to use scalable systems with high availability, and how WebSphere MQ clusters can be used to meet this need.

The time-critical business environment

Messaging middleware, such as WebSphere MQ, gives you the capability to protect messages from loss. With asynchronous messaging, the producer, consumer and means of communication don’t have to be available at the same time, because the software can safely pass the message from one party to another when they are available. The messages will be kept safe, and eventually processed with assured delivery. However, in real life, eventually is usually not good enough. Messages are often time-critical and need to be processed sooner rather than later, so the systems that process them must be highly available. Overnight batch processing or spooling data that is ready for a future event is not time-critical. But messages are urgent and vitally important, in many situations.
Interactive work. Your customers don’t like being kept waiting. To keep users productive, computer response times and reliability need to be good. Your users need interfaces that help them know what the system is doing. After a few seconds of inactivity, users can begin to lose confidence and can become frustrated. Many customers are more comfortable when they get visible feedback, and see the results of their messages being processed. The requirement is to process messages in a fast and sure way.

High-value transactions. Some messages represent actions on a high-value commodity, with a rapidly changing price – for example, stock trading, where a message stuck in a dead system is a liability. Messages representing trade or other instructions in these commodities must be processed quickly or the trader risks loss.

Control instructions. Messages that contain instructions to close a valve in an oil pipeline, alter the action of machinery or change the movement of aircraft must be processed quickly. If the messages are delayed, it could lead to disaster. Assumptions about the length of delay in delivering and processing these messages are engineered into the plant or protocols that depend on them.

Considerations for time-critical messaging
There are some important issues to consider when designing systems to deliver time-critical messages. First, the systems must be continuously available. Second, they must scale to the power required to handle the throughput. And third, they must be configurable for ease of control.

High availability. It sounds self-evident, but to process time-critical messages, systems must be available and working at the time the message is sent. In the early days of message queueing, the novelty of the technology was that messages stuck in an offline system could be held safely until those systems became available again. The sending application didn’t have to check whether a message was processed – eventually it would be taken from the queue by the receiving application when the links were restored.
But with time-critical messages, that isn’t acceptable. WebSphere MQ clusters can choose from a number of different servers, so even if one is unavailable, there will be another to fill the gap. Depending on how you specify the servers for the cluster, you can create an ultra-high availability environment for messages that must be processed immediately.

**Scalability.** Powerful servers are needed to process messages quickly, so there’s less need to hold the messages in a queue waiting for processing resources to become available. The alternative is a *trickle feed*, where messages are sent slowly through an underpowered system. You can obviously build systems that are powerful enough to process messages on demand, with minimum queuing, by buying faster systems, or by buying more of the same slower systems, with WebSphere MQ clusters.

Having fewer but faster systems involves less administration but these systems often need more careful tuning to avoid contention and bottlenecks. With more systems and less sharing of resources, you have less contention for the resources within each system. Usually a balance exists where enough reasonably powerful systems are used, keeping administrative tasks at a minimum, yet giving enough parallel activity to complete all tasks in the required amount of time.

**Configuration.** The number of systems in even modest-sized networks can produce a daunting number of interconnections. The number of associated definitions, where parameters in one system must match or complement parameters in another, is also large. Controlling even a small number of multiple systems can produce too many combinations for administrators to track effectively.
Providing more processing power through clustered queue managers.

Adding required definitions, making appropriate alterations as requirements change and removing definitions that are no longer needed are error-prone tasks. Often the reasons why systems become unavailable are the mistakes introduced with these changes as the size and complexity of the systems environment increases. Using software like WebSphere MQ clusters eliminates the need to make associated changes across the network of queue managers, helping reduce potential for errors.

How WebSphere MQ clusters work

A new capability introduced in MQSeries, Version 5.1 and MQSeries for z/OS, Version 2.1, WebSphere MQ clusters allow a group of queue managers to advertise the existence of some of their channels and queues to each other. The knowledge of these resources is shared among the queue managers and forms the cluster by including the queues and channels belonging to various queue managers. Because no single controlling entity is in charge of the cluster, it is very robust, with no single point of failure. The collection of queue managers provides the service. As long as there are enough remaining queue managers and connections, the service is available.

Failure of a network or failure of the destination queue manager manifests itself as a channel failure, with the communications channel performing its normal retry cycle. At the same time any messages waiting on the SYSTEM.CLUSTER.TRANSMIT.QUEUE intended to move along the failed channel will be reprocessed and another destination found, if this is possible and appropriate. The communications channel between each pair of queue managers in the cluster is protected independently, without any global monitoring of the cluster as a whole.
Providing more processing power through clustered queue managers.

The ability to define equivalent queues, with identical names on several of the queue managers in the cluster, leads to increased system availability. Each queue manager runs equivalent instances of the applications that process the messages. Figure 1 shows how a message put on one queue manager anywhere in the cluster is moved to other clustered queues and processed. If one of the queue managers fails, or the communication to it is suspended, it is temporarily excluded from the choice of destinations for the messages. This approach allows redundant servers to be completely independent of each other, thousands of miles apart if necessary. A problem in one server is less likely to affect the others.

![Figure 1. Built-in WebSphere MQ cluster support](image)

Advantages of clustering

Several advantages of clustering can bring major benefits to your operation:

Disaster recovery. Successful disaster management requires planning and testing—before a disaster happens. Clustering is a valuable part of that process. If the servers are the subjects of the disaster, and they are well separated, the disaster could amount to the loss of just one of them, allowing normal operation with the other servers. Disaster testing is a good practice that can often be combined with regularly scheduled maintenance or upgrade activities, when some of the system or network connection clusters are routinely taken out of service.
Providing more processing power through clustered queue managers.

Scalability. Deploying redundant queue managers in a cluster can give you extra capacity on the spot. And these queue managers don’t have to share resources. At first sight, shared memory multiprocessor systems seem to have the advantages of high availability and simple system administration, but they have to share resources. This means you have overhead with nonlinear scalability. Queue clustering gives you scalability more economically.

Administration. Aside from increased availability and throughput, WebSphere MQ clustering can help the amount of administration needed to maintain definitions because each channel is only administered on the queue manager that hosts it. The effects of any changes are advertised to the other members of the cluster. The amount of administration increases linearly, in proportion to the number of queue managers on the cluster, and avoids the combinatorial explosion that would otherwise happen as more queue managers are added. This is a major benefit of WebSphere MQ clustering, even if you have no immediate need to exploit it for availability and scalability reasons.

Scope. You experience almost no performance overhead with WebSphere MQ clustering. The system messages used to create and maintain the cluster usually represent a tiny fraction of the overall message flows. Cluster information itself is transmitted over the same channels as application messages. Each queue manager in the cluster detects its individual ability to reach the others by attempting to send messages. Therefore, the clusters can be very large, if necessary, consisting of thousands of queue managers. This capability sets WebSphere MQ software apart from many other clustering technologies. For example, the cluster can contain the many small branch systems that produce and consume the messages, as well as the large servers that process them. Changes are reflected throughout the whole collection, not just the central servers. The entire collection can be made highly available rather than just the central hub.
Preparing to use queue manager clusters

You need to review your applications carefully to fully exploit WebSphere MQ clustering. You can experience easier transition with a little preparation and the right tools. Several key factors should be addressed to assure a smooth transition:

*Catering for message affinity.* Message affinity is the first area that needs to be examined to gain the most benefit. Message affinity means that a sequence of messages must be processed in the same queue manager, usually because the application depends on a transient state created by messages earlier in the sequence. WebSphere MQ clustering enables your administrator or application writer to specify that all messages using an OPEN handle must go to the same queue instance. The application writer does this by specifying BIND_ON_OPEN when the queue is opened. This solves the affinity problem with the knowledge that any failure that occurs after the OPEN handle has been created will delay the processing of later messages until the failure is corrected.

*Tuning recovery time.* By tuning the frequency of heartbeats (HBINT) or keepalive (KAINT) on the CLUSRCVR channels, the detection time for channel failures can be short. The delay in rerouting a message to an already active alternate queue manager is also short, making it feasible to detect and recover from failures without being noticed, even by interactive users. Making the channel failure detection times too small can lead to false failures, where a slowdown or a delay in network traffic is treated like a failure.
Providing more processing power through clustered queue managers.

Marooned messages. If a request message reaches its destination queue manager, or is in route to the destination on the channel, and the destination queue manager fails, the system will not be able to recover the message. This occurs because the possibility of duplicating messages exists without recovering the destination queue manager itself. If it is important to protect your systems against this type of marooned message problem, you can use the form of hardware failover shown in Figure 2. Another alternative is to use the IBM z/OS™ shared queue facility in conjunction with WebSphere MQ clustering, as discussed later in this white paper.

Figure 2. Hardware failover based on a shared disk.

Controlling multiple networks. It’s also possible to define multiple channels over separate networks to each queue manager so that a failure of a network does not disable large numbers of queue managers. You can use the NETPRTY channel parameter to define which of several network paths should be used in order of preference. If your server cluster is confined inside a single campus, you can use this technique to protect against a campus network failure.
Providing more processing power through clustered queue managers.

Realizing the benefits of clustering
The benefits of clustering are wide-reaching across many different industries. WebSphere MQ clustering technology enables e-businesses to streamline their back-office systems by linking the servers within their networks. The following sections discuss some of the types of systems that can benefit from WebSphere MQ clusters.

Simple Web front-end to a back-end messaging system
A Web browser can connect to a Web server with a queue manager running in the same hardware. This forwards messages to back-end servers. Queues servicing the requests are defined in the back-end systems and their existence is advertised as part of the cluster. This is typical of a small cluster setup. You can adopt this approach to provide high availability, but as demand grows, it is easy to add more front-end servers and more back-end servers to provide greater capacity as well.

In retail networks or retail banking, it is common to find a structure where a large number of WebSphere MQ clients or queue managers in branches connect to a set of concentrator queue managers in a cluster. These concentrator queue managers forward messages to back-end systems for processing. Adding a new service or branch does not require you to make changes on the other systems. So long as there are enough front-end and back-end systems, failure just means that the service is maintained by the remaining systems.

For example, a national bank has thousands of banking terminals and ATMs which make requests into large UNIX® servers, which are part of the bank cluster. The banking transactions are handled by a number of large z/OS systems running as part of the same cluster. The systems are built and configured to handle many failure modes, including failure of the network itself. They are also able to detect and recover failures in times which are short enough to be invisible to the bank’s customers.
Providing more processing power through clustered queue managers.

Branch networks
In Figure 3, the cluster encompasses the whole branch network and its servers. The main issue here is the scale. With many hundreds or thousands of branches involved, a single back-end server is unlikely to handle the required load, or provide the availability needed. By including the branch systems in the cluster, they can be made to tolerate a network failure to a server, as well as a failure of the server itself. If the servers are located in two different places, the configuration can be virtually disaster-proof.

For example, a regional bank with a network of hundreds of branches grouped in a single cluster with a number of central server systems sends messages to generic services that run on one or more servers. This means that when a server fails or is taken offline, the others assume the load. New servers and branches can be added with relatively little disruption.

Multiple overlapping clusters
Several clusters may be needed to partition a large number of branches, and to mitigate the risks in managing a single large configuration. These are the most complex configurations, often used to manage very large-scale operations.
Using clusters with z/OS shared queue support

The z/OS shared queue support was first introduced as a standard part of IBM MQSeries for OS/390®, Version 5.2. It exploits the coupling facility hardware running under z/OS. With WebSphere MQ, Version 5.3, messages can be persistent or nonpersistent but must be less than 63 kilobytes in size.

Messages reside in queues stored in the shared coupling facility. The queue managers that are in a queue-sharing group and have access to the coupling facility can get and put messages to these queues, as shown in Figure 4. Availability and scalability derive from the ability of any of the group of queue managers to process a message.

Queue managers in a queue-sharing group can be administered as a group, with queues being defined as shared and the definition applied to the group by including the CMDSCOPE keyword. Each of the queue managers in the queue-sharing group can be part of a wider cluster. The WebSphere MQ cluster protects the ability of the distributed queue managers to get their messages to a z/OS server. Once in the server, the shared queue facility makes the messages available for processing by any of the members of the queue-sharing group.

The capability to have several clusters with their own names is an advanced feature offered by WebSphere MQ clusters. The queues and channels that form the cluster can each belong to one or more of the clusters.
Used as an organizational tool, the advanced feature can present one set of queues and interconnections to one set of users, and another set to others. Each user of the systems can have a different view of the services these systems offer. For example, the payroll department might be quite small with some of its resources not made widely available. Other payroll resources may be used by the rest of the organization.

Need help with high availability?
You can implement a high-availability solution with IBM WebSphere MQ software. A services specialist can review and assess your current WebSphere MQ environment and make recommendations to help you realize that your high-availability requirements are realized. This includes recommendations about WebSphere MQ software versions and patches, security, resilience, improving IT infrastructure high availability, clustering, application affinities and systems management.

For more information
An educational manual about designing and architecting clustering solutions is available at:

ibm.com/software/ts/mqseries/library/manuals

Use the pull-down menu to find the latest MQSeries messaging cross-product manuals and select “MQSeries Queue Managers Clusters”.

For more information about WebSphere MQ clusters, visit:

ibm.com/services/its/us/drmklm11.html