Introduction to High Availability

This chapter covers the following topics:

- Purpose of Clustering
- Why is High Availability Important
- Clustering for High Availability, Load Balancing and Fault Tolerance
- Advantages of Network-based Clustering
- Virtual IP Adressing
- Client Considerations

Purpose of Clustering

When determining how to increase availability and to decrease downtime for important applications, there are many different clustering solutions to consider. It is important to start any availability improvement discussion however, with a preemptive understanding of what it is you want to improve. Begin by looking at the history of your application failures, downtime scenarios, and the underlying causes of availability issues. Map out your application topology and network infrastructure such that you can ascertain potential weak spots or single points of failure and prioritize component redundancy based on exposed risk to the overall system availability. Ask yourself the following questions:

- What is it that I want to accomplish from clustering?
- Who are the stakeholders?
- How will I measure availability improvement?

Once you have a prioritized list of availability improvement objectives, look for solutions that address these points of failure and look to provision an implementation plan that ensures a prearranged level of operational performance will be met during a contractual measurement period. Improvement must be measurable in both the technical and business perspectives.

Why is High Availability Important

Many of the world’s largest organizations including financial institutions, manufacturing, transportation, and communication companies along with large government agencies, rely on the availability and reliability of applications to deliver their most important business transactions and data. These large-scale information systems consist of several hardware and software components, each of which performs a particular function and is a critical-path to successful daily operations. If any of these components fail however, the outcome could vary drastically - from a single user experiencing a slow-down in their order response time, to thousands of retail bank customers not being able to access any of their cash assets. Creating a highly available system topology removes any single points of failure from a large system, enabling another redundant component to effectively take over the workload of the failed component. Improving availability ultimately leads to a reduction in downtime, improved business performance, and a better user experience.

Another IT concern is how to apply maintenance and upgrades to these important systems without affecting users. In a highly available world, a system in need of maintenance can be taken out of the workload pool and updated while the rest of the system continues to process service requests.

Clustering for High Availability, Load Balancing and Fault Tolerance

As previously mentioned, there is a wide variety of clustering techniques that are designed to accomplish specific improvement in application availability based on planned or unplanned events. Planned events are typically scheduled maintenance activities associated with specific fixes or general upgrades. In this case, clustering can be utilized to maintain processing workloads while certain instances or services of the cluster are brought down, updated, and rejoined to the cluster.

Unplanned events require a means of automated failover whereby work is picked up by a pooled resource. Cluster architectures vary in how they handle fault tolerance, recovery, and guarantee delivery. Each high availability solution has a different architecture or technique in which work is redirected to or picked up by an available process. There are shared memory solutions (e.g. Terracotta Server Array), shared data store (e.g. Integration Server cluster), shared message queue (e.g. Universal Messaging), and shared virtual IP (e.g. EntireX Broker) among the list of possible solutions. Each technique provisions a group or cluster of common processes working on in-flight data or messages that may or may not be persisted and coordinated by the state of the application endpoints.

Advantages of Network-based Clustering

While a loosely coupled system such as network clustering cannot recover or coordinate distributed work, it does protect against a wide range of failures up and down the stack including hardware, OS, and application failures. Network-based HA solutions are relatively easy to configure and work transparently with stateless applications.

Another advantage is to address system availability during planned events such as applying maintenance patches or upgrades. For example, when a major or minor update is required to be performed to a broker, it is important that the system remains operational during this planned event. In this case, individual broker instances are taken out of the cluster without impacting the overall operation of the system. As updates are completed, Brokers can individually be added back into the cluster independent of their version.

Virtual IP Adressing

Traditionally, an IP address is associated with each end of a physical link (or each point of access to a shared-medium LAN), and the IP addresses are unique across the entire visible network, which can be the Internet or a closed intranet. The majority of IP hosts have a single point of attachment to the network, but some hosts (particularly large server hosts) have more than one link into the network.

A TCP/IP host with multiple points of attachment also has multiple IP addresses, one for each link. Within the IP routing network, failure of any intermediate link or adapter disrupts end user service only if there is not an alternate path through the routing network. Routers can route IP traffic around failures of intermediate links in such a way that the failures are not visible to the end applications or IP hosts. However, because an IP packet is routed based on ultimate destination IP address, if the adapter or link associated with the destination IP address fails, there is no way for the IP routing network to provide an

alternate path to the stack and application.

Endpoint (source or destination) IP adapters and links thus constitute single points of failure. While this might be acceptable for a client host, where only a single user will be cut off from service, a server IP link might serve hundreds or thousands of clients, all of whose services would be disrupted by a failure of the server link.

The virtual IP address (VIPA) removes the adapter as a single point of failure by providing an IP address that is associated with a stack without associating it with a specific physical network attachment. Because the virtual device exists only in software, it is always active and never experiences a physical failure. A VIPA has no single physical network attachment associated with it.

Client Considerations

Only synchronous, non-conversational application scenarios are supported. Additional prerequisites apply to client applications:

- No Persistent Sockets (Socket Pooling)
- Socket Reconnect
- Security Handling
- Matrix of Supported Features

No Persistent Sockets (Socket Pooling)

Socket pooling needs to be explicitly disabled for all EntireX clients, except webMethods EntireX Adapter for Integration Server. See *Matrix of Supported Features*.

Socket Reconnect

Client applications connected to a broker instance may need to react when this broker instance becomes unavailable and the cluster system establishes connection to a different broker instance.

Most EntireX clients support some reconnect logic on socket disconnect if the cluster system routes the connection to a different broker instance. However, the Java RPC, EJB and XML/SOAP client Java APIs do not support automatic reconnect. This needs to be handled by the client application logic. The XML/SOAP Listener does not support socket reconnect.

Security Handling

If the Brokers in the cluster have security enabled, client applications need to re-authenticate with a new Broker instance on reconnect.

For Java RPC, EJB and XML/SOAP client Java API, re-authentication has to be completely handled by the client application. The XML/SOAP Listener does not support automatic re-authentication.

Matrix of Supported Features

In the table below, "yes" means the feature is handled automatically and no user action or configuration is required; "no" means the feature is not supported; and "application" means that the client application must be adapted accordingly.

Note:
This table assumes you are using the latest version of the components listed.

RPC Client	No Persistent Sockets	Socket Reconnect	Security Handling
EntireX Adapter for IS	yes	yes	yes
RPC-ACI Bridge	Specify `socketpoolsize=0` as part of the broker ID. See *Socket Parameters for TCP and SSL Communication* under *Writing Advanced Applications - EntireX Java ACI*.	yes	yes
WebSphere MQ Listener	Specify `socketpoolsize=0`.	yes	yes
SAP XI Adapter	Specify `socketpoolsize=0`.	yes	yes
Java RPC	Specify `socketpoolsize=0`.	application	application
EJB	Specify `socketpoolsize=0`.	application	application
XML/SOAP client API	Specify `socketpoolsize=0`.	application	application
XML/SOAP Listener	Specify `socketpoolsize=0`.	no	no
Natural RPC	Specify `ETB_SOCKETPOOL=OFF`. See *Support of Clustering in a High Availability Scenario* under z/OS | UNIX | Windows.	yes	application
C RPC	Specify `ETB_SOCKETPOOL=OFF`.	yes	application
COBOL RPC	Specify `ETB_SOCKETPOOL=OFF`.	yes	application
PL/I RPC	Specify `ETB_SOCKETPOOL=OFF`.	yes	application
.NET Wrapper	Specify `ETB_SOCKETPOOL=OFF`.	yes	application
DCOM Wrapper	Specify `ETB_SOCKETPOOL=OFF`.	yes	application