Configuring MRG messaging with Cluster Manager

Overview

Note: openais in RHEL 5 is called corosync in RHEL 6. Wherever openais is mentioned, substitute corosync on RHEL 6.

Brokers in an MRG cluster communicate using the reliable multicast transport provided by openais. It is recommended that they also use the Cluster Manager (cman) for quorum. Without cman, a network partition may cause lost and/or duplicated messages and unpredictable shutdown of MRG brokers in the cluster.

When a network partition occurs, the cluster may be split in to into two or more sub-clusters that cannot communicate with each other. Each of the sub-clusters acts without knowledge of the others, resulting in an inconsistent cluster state. This condition is called a split-brain.

Cluster manager (cman) avoids split-brain conditions by using a simple-majority quorum algorithm. A group of more than half of the total cluster nodes has quorum and can provide service. If a node loses contact with the rest of the cluster, it loses quorum and the broker on that node shuts down to avoid inconsistency. Clients then reconnect to a quorate broker. This ensures that only one group continues processing in the event of a network partition. For ideal operation, a cluster should have an odd number of nodes.

While it is possible to create a cluster of 2 nodes, this requires additional hardware (power fencing) and additional configuration that is not covered in this article.

MRG brokers do not use shared storage; each MRG broker has its own independent store which is kept up to date with the other brokers via multicast. This means that MRG brokers do not require hardware fencing to prevent corruption of shared storage. However, if your cluster runs any other services that use shared storage, then you need fencing.

If a broker process crashes or shuts down, it can be re-started automatically by the Resource Group Manager (rgmanager). Configuration is explained below.

MRG broker clusters are active/active, meaning that clients can connect to any node at any time. This is different from "cold standby" services, where only instance is active a time.

See the High Availability Add-On Overview and Cluster Administration in the Red Hat Enterprise Linux documentation for more information on cman and rgmanager.

Configuring MRG brokers to use cman

To enable CMAN integration add this to /etc/qpidd.conf:

```
cluster-cman=yes
```

When cluster-cman is enabled, the MRG broker will wait until it belongs to a quorate cluster before accepting client connections. It continually monitors the quorum status and shuts down immediately if it the node it runs on loses touch with the quorum. This avoids inconsistencies and allows clients to fail over to a quorate broker.

Notes on using openais and cman

When using cman, you should not start the openais service, as it is started automatically by cman, i.e.

```
# service openais stop chkconfig openais off
```

The configuration for cman is in /etc/cluster/cluster.conf. You can use the tools conga to edit this file, see Cluster Administration. An example cluster.conf file is shown at the end of this article.

When openais is started by cman, the openais.conf file is not used. Many of the configuration parameters listed in openais.conf can be set in cluster.conf instead. See the cman man page for details.

Create a cluster configuration listing each of your nodes using conga as described in Cluster Administration

You should enable the cman and rgmanager services:

```
# service cman start
# chkconfig cman on
# service rgmanager start
# chkconfig rgmanager on
```

Restarting failed broker processes.

You can use the Resource Group Manager (rgmanager) to restart crashed broker process automatically. There is an example configuration at the end of the article.

Using conga, edit the cluster configuration as follows:

1. Create a restricted failover domain for each node in the cluster; add only one node to each failover domain.
2. Create a script resource that points to /etc/init.d/qpidd
3. Create one qpidd service for each failover comain with "autostart" checked and the recovery policy set to "restart". Ensure you link in the script resource you created in step 2.

Rgmanager only allows one instance of a service to be active at a time. Since qpidd brokers are all active at the same time, we simply reference the same script from multiple services. Placing each service in a restricted failover domain of exactly one node allows rgmangaer to restart any one of them automatically on a designated cluster node.

Example configuration

Note: if you are not using fencing, you should disable the fence daemon by adding the following line to /etc/sysconfig/cman:

```
FENCE_JOIN="no"
```

Example cluster.conf for 3 node cluster with hosts mrg33, mrg34, mrg35 (without fencing):

```
<?xml version="1.0" ?>
<cluster alias="test" config_version="37" name="test">
  <clusternodes>
    <clusternode name="mrg33" nodeid="1" votes="1"/>
    <clusternode name="mrg34" nodeid="2" votes="1"/>
    <clusternode name="mrg35" nodeid="3" votes="1"/>
  </clusternodes>
  <cman/>
  <rm log_level="7">
    <failoverdomains>
      <failoverdomain name="only_broker1" restricted="1">
        <failoverdomainnode name="mrg33"/>
      </failoverdomain>
      <failoverdomain name="only_broker2" restricted="1">
        <failoverdomainnode name="mrg34"/>
      </failoverdomain>
      <failoverdomain name="only_broker3" restricted="1">
        <failoverdomainnode name="mrg35"/>
      </failoverdomain>
    </failoverdomains>
    <resources>
      <script name="qpidd" file="/etc/init.d/qpidd" />
    </resources>
    <service name="qpidd_broker1" domain="only_broker1">
      <script ref="qpidd" />
    </service>
    <service name="qpidd_broker2" domain="only_broker2">
      <script ref="qpidd" />
    </service>
    <service name="qpidd_broker3" domain="only_broker3">
      <script ref="qpidd" />
    </service>
  </rm>
</cluster>
```