Cluster Failover Modes

Qpid cluster failure modes.

This section describes failure modes and techniques to deal with them, the following section provides configuration details for the techniques mentioned here.

Broker process terminated

E.g. broker killed.

Clients: disconnected immediately, can fail over to another broker in the cluster.

Multicast group: broker is automatically removed from the multicast group.

The broker needs to be manually restarted.

Broker host crash

E.g. power failure, hardware failure.

Clients: may not detect loss of connection until a long TCP timeout is reached. Use heartbeats to reduce the time to detect loss of connection.

Multicast group: broker is automatically removed from the multicast group after the configurable totem token timeout value.

Broker freeze -e .g. kill -STOP

E.g. using kill -STOP.

Clients: disconnected after TCP timeout, use heartbeats to disconnect quicker.

Multicast group: Broker is not automatically can eventually hold up all cluster trafic. Use the watchdog plugin to kill a broker that is unresponsive for a configured period of time.

Broker needs to be manually restarted.

Client-broker network failure

Clients: disconnected after TCP timeout, use heartbeats to disconnect quicker.

Broker: clean up client resources (e.g. auto-delete queues) when client disconnect is detected after TCP timeout. Use heartbeats to disconnect quicker.

Broker-broker multicast network failure

A failure in the multicast network creates a "partition" creating two or more sub-clusters that are unable to communicate. This creates inconsistent state in the sub clusters that cannot be reconciled correctly if they are re-connected, and will result in unpredictable behaviour.

To deal with this situation, you need cman's quorum service. In the event of split-brain only one of the sub clusters will have a "quorum". Brokers in the other sub-clusters will automatically shut down, allowing clients to fail over to a broker in the quorum.

Alternatively to avoid partitions entirely you can use the openais/corosync Redundant Ring Protocol which uses two physically separate networks for cluster communication. This enables the multicast group to survive the loss of either of the networks (but not both.)

Brokers that shut down need to be manually restarted.

Broker-broker update network failure.

New brokers joining the cluster receive an initial state snapshot from an established member of the cluster via TCP. A network failure at this point will cause the joining broker to exit.

Broker must be manually restarted.

Note as of qpid 0.6 the update connections are made using the same URL that clients uses to connect, its not possible to restrict broker-broker update connections to a different network from client connections.

Client crash

Broker: client resources such as auto-delete queues are reclaimed immediately.

Client host crash

Broker: client resources such as auto-delete queues are reclaimed after the TCP time-out. To have resources reclaimed more quickly use heartbeats.

Configuration

Separate client/multicast networks

For best performance use a separate network for clients and the multicast group. If possible the multicast group network should be

openais.conf/corosync.conf

totem.token: timeout in milliseconds until host crash or network disconnect is detected by the multicast group. Defaults to 1000ms.

Redundant ring protocol (RRP), uses two physically separate networks for cluster communication. To use RRP, you must choose a replication mode for your environment. RRP has 3 modes:
Modes

active: Active replication can offer slightly lower latency in faulty network environments, however it can reduce throughput.
passive: Passive replication can nearly double the speed from transmit to delivery, but also carries the potential for the protocol to become bound to a single CPU.
none: Disables redundant ring.

To enable RRP make the following changes to corosync.conf (for RHEL6) or openais.conf (for RHEL5):

1. In the totem section, add rrp_mode=active or rrp_mode=passive
2. Add a second interface section with a different bindnetaddr for your second network.

qpidd configuration options

cluster-url: specify addresses that clients will use to connect. Can be used to ensure clients connect on a different network from the multicast network.

Note a future release will provide cluster-update-url to allow updates to be restricted to a different network from client connections.

watchdog plugin

The watchdog plug-in will kill the qpidd broker process if it becomes stuck for longer than a configured interval.

If the watchdog plugin is loaded and the --watchdog-interval=N option is set then the broker starts a watchdog process and signals it every N/2 seconds.

The watchdog process runs a very simple program that starts a timer for N seconds, and resets the timer to N seconds whenever it is signalled by the broker. If the timer ever reaches 0 the watchdog kills the broker process (with kill -9) and exits.

This is useful in a cluster setting because in some instances (e.g. while resolving an error) it's possible for a stuck process to hang other cluster members that are waiting for it to send a message. Using the watchdog, the stuck process is terminated and removed from the cluster allowing other members to continue and clients of the stuck process to fail over to other members.

cman configuration

Note: when using cman, do not start the openais/corosync service. It will be started automatically by the cman service.

Only basic cman configuration (cluster.conf) is required. Other cluster suite services (GFS, DLM, fencing etc.) do not need to be configured.

Enabling heartbeats

In C++ clients, heartbeat is disabled by default. You can enable heartbeat by specifying a heartbeat interval (in seconds) for the connection:

```
ConnectionSettings settings;
settings.heartbeat = 1;
FailoverManager fmgr(settings);
```

In a JMS client, heartbeat is set using the idle_timeout property of the connection URL. For instance, the following line from a JNDI properties file sets the heartbeat time out to 3 seconds:

```
pconnectionfactory.qpidConnectionfactory = amqp://guest:guest@clientid/test?brokerlist='tcp://localhost:5672',idle_timeout=3
```

Heartbeats are enabled in both directions, the connection can be closed at either end if the heartbeat interval is missed.

References

Cman configuration: See chapters 3 & 5 of http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.4/html/Cluster_Administration/index.html