cman and corosync
the Yin & Yang of cluster.conf

Christine Caulfield, Red Hat Ltd.
Christine.Caulfield@redhat.com

Revision History

1.0 3rd September 2010 Christine Caulfield Initial version

This document describes the strange relationship between cman and corosync as regards configuration. It assumes a working knowledge of cluster.conf and, to a lesser extent, corosync configuration. It can also be annoyingly frivolous.

Corosync.conf

The first, and most important thing to know is that when using corosync in a cman environment, the file /etc/corosync/corosync.conf is not used. You can put whatever you like in there, up to and including a recipe for chorizo tartlets, and it will have absolutely no effect on the operation of corosync.

All configuration of corosync comes from the /etc/cluster/cluster.conf file. This is an XML file that should be the same on all nodes. The usual way to distribute this file is using ccs_sync, but other methods will also be fine, and might even be better. I shall say no more about that bit as blood has already been shed.

Overview

Those who know will realise that cluster.conf is the same on all nodes of a cluster, yet corosync.conf will usually be different for each node. So, how does cman manage this amazing magic trick?

Well, those who *really* know will also know that cluster.conf contains information for all the nodes in the cluster, rather than just the local node. This information is used by many parts of a RHEL6 cluster, not just cman. So, from this list of nodes cman can locate the local node name then get the IP address and node ID information.

The rules for determining the IP address to use for cman are documented elsewhere but I'll include them here to save both you and Google the effort:

1. It looks up $HOSTNAME in cluster.conf
2. If this fails it strips the domain name from $HOSTNAME and looks up that in cluster.conf
3. If this fails it looks in cluster.conf for a fully-qualified name whose short version matches the short version of $HOSTNAME
4. If all this fails then it will search the interfaces list for an (ipv4 only) address that matches a name in cluster.conf

As well as filling in the IP address, cman also provides defaults for the multicast address, port name and several totem parameters too. All of these can be overridden if you need to. We hope that most people won't though as it just usually generates support calls.

The gory details

I'll start with the ring parameters that cman sets, as those are the simplest. If you want to see these in action then start up a cman node and type in the command:

```
# corosync-objctl -a|grep ^totem
```

This is how I've obtained the examples shown below - I don't like to hide things from you, dear readers. The examples here are for a very basic cluster.conf with no parameter overrides. I'll explain later which parameters can be overridden. I might even explain why, but don't get your hopes up too high.

The first thing that cman does is try to determine the local node name using the algorithm above. It then writes 'interface' sections for all the addresses of a node. Multi-ring isn't really supported yet, but cman doesn't exclude its presence. The ring sections have the following parameters set:

```
totem.interface.ringnumber=0
totem.interface.bindnetaddr=192.168.1.29
totem.interface.mcastaddr=239.192.209.5
totem.interface.mcastport=5405
```

Where bindnetaddr is the address of the node, mcastport defaults to 5405 and the mcastaddr is calculated using the cluster ID or a hash of the cluster name. If you specify an altname for the node then there will be more of these, the second having a ringnumber of 1, the third 2, etc. I'm sure you can spot the sequence there.

Next cman sets totem.vsftype to “none” and totem.version to “2”. These are hard-coded and should not be changed as they could break things. Actually, scrub that ... they ***will*** break things and not in an interesting “ooo, I wonder what happened?” sort of way either. It'll just stop. So leave them alone. OK ?

Then cman sets some corosync totem values to slightly higher defaults than a normal corosync-only system. This is to allow for failover on busy clusters or networks.

```
totem.token=10000
totem.join=60
totem.fail_recv_const=2500
totem.consensus=12000
```

Lastly cman enables data encryption and sets the key to be the cluster name truncated to a multiple of 4 bytes.

```
totem.rrp_mode=none
totem.secauth=1
totem.key=composers
```

rrp_mode will be set to “none” for the usual case where the node has only one ring configured, or “active” if two or more rings are configured.

Lastly cman sets the quorum provider to be itself, so that the main cman plugin gets loaded and interfaces with the quorum features of corosync. This is the bit that cman uses to stop the cluster working if it gets lonely.

```
quorum.provider=quorum_cman
```

Broadcast

As I write this broadcast is not supported, but the code is there and is known to work in lab conditions. By “lab” I mean, of course, a small room crammed with too many computers, too little air-conditioning and a strange smell of fried cockroaches.

If you add the stanza:

```
<cman broadcast=”yes”/>
```

in cluster.conf then it will trigger some code in cman to switch corosync into broadcast mode rather than multicast mode. Do ***not*** set corosync to broadcast yourself using a totem stanza, it will confuse cman and not work as you expect.

When cman sees its broadcast flag, it will set totem.interface.broadcast to “yes” and unset totem.interface.mcastaddr. Internally cman also sets the multicast 'name' to “255.255.255.255”, but corosync never sees this, it's just for cman_tool's benefit, so that when you do “cman_tool status” it has something to tell you rather than going “nnnngg ... I don't really know”.

Overriding parameters

As we've seen, cman doesn't change very many totem parameters, mostly it's just using the whole-cluster nature of cluster.conf to locate its networking information. The few totem parameters that it does mess with are there for the convenience of clusters that run DLM applications and GFS and to generally reduce the likelihood of problems (and support calls).

There are two main reasons you might want to change these. If you're just using a simple failover cluster on an enclosed network you might like to reduce them to shorten the recovery time. Or, if you're using qdisk and/or multipathed disks you might need to increase the values to allow for disk failover time. Some multipath disks can take a *long* time to failover ... I think they use elves or something. See the documentation for those components for details on what needs to be done for them, I try not to get involved.

Do not attempt to override anything in the totem.interface section, it will be rewritten either wholly or partially by cman and the results will rarely be what you want. If you need to change the multicast address, port number or broadcast flag, then always use the cman parameters instead.

Most of the rest of the totem parameters can be changed in cluster.conf though. And it's pretty easy to do, though I say so myself. One nice feature of doing it in cluster.conf is that all nodes will always have the same parameters. This ensures that stability of your cluster. No, really.

So, to some examples. To increase the token timeout for a cluster to 30 seconds add the following to cluster.conf:

```
<totem token=”30000”/>
```

If you also want to change the consenus timeout to 45 seconds it would look like this

```
<totem token=”30000” consensus=”45000”/>
```

When cman sees these it will not overwrite them with its own default values. I don't recommend changing these in a running cluster, but it can be done if you're either brave, foolish or a corosync developer (which probably encompasses the first two to be honest).

Be aware that corosync has rules that apply to certain combinations of parameters, to prevent impossible or stupid combinations. These still apply when reading the configuration from cluster.conf. So, if cman doesn't start with your modified parameters check syslog carefully for errors – it *will* be your fault. Or, at least, that's what we'll tell you.

Logging

cman also adds some defaults for logging which I will document here but not say anything about as they should be self-explanatory – if they're not then go and gaze in wonder at the corosync.conf(5) man page. Again, they can be overridden if they don't match what you need:

```
logging.timestamp=on
logging.to_logfile=yes
logging.logfile=/var/log/cluster/corosync.log
logging.logfile_priority=info
logging.to_syslog=yes
logging.syslog_facility=local4
logging.syslog_priority=info
```