Cluster Configuration Subsystem

A “under the bonnet” guide for advanced users and developers.

Doc version 0.1
Christine Caulfield (caulfie@redhat.com)
Fabio M. Di Nitto (fdinitto@redhat.com)

Disclaimer

The following slides will explain how the Cluster Configuration Subsystem (CCS in short) works. They do not contain explanations on how to perform basic cluster configurations.

This is document is still a work in progress. Please report any discrepancies or errors.

Bits of history (again?)

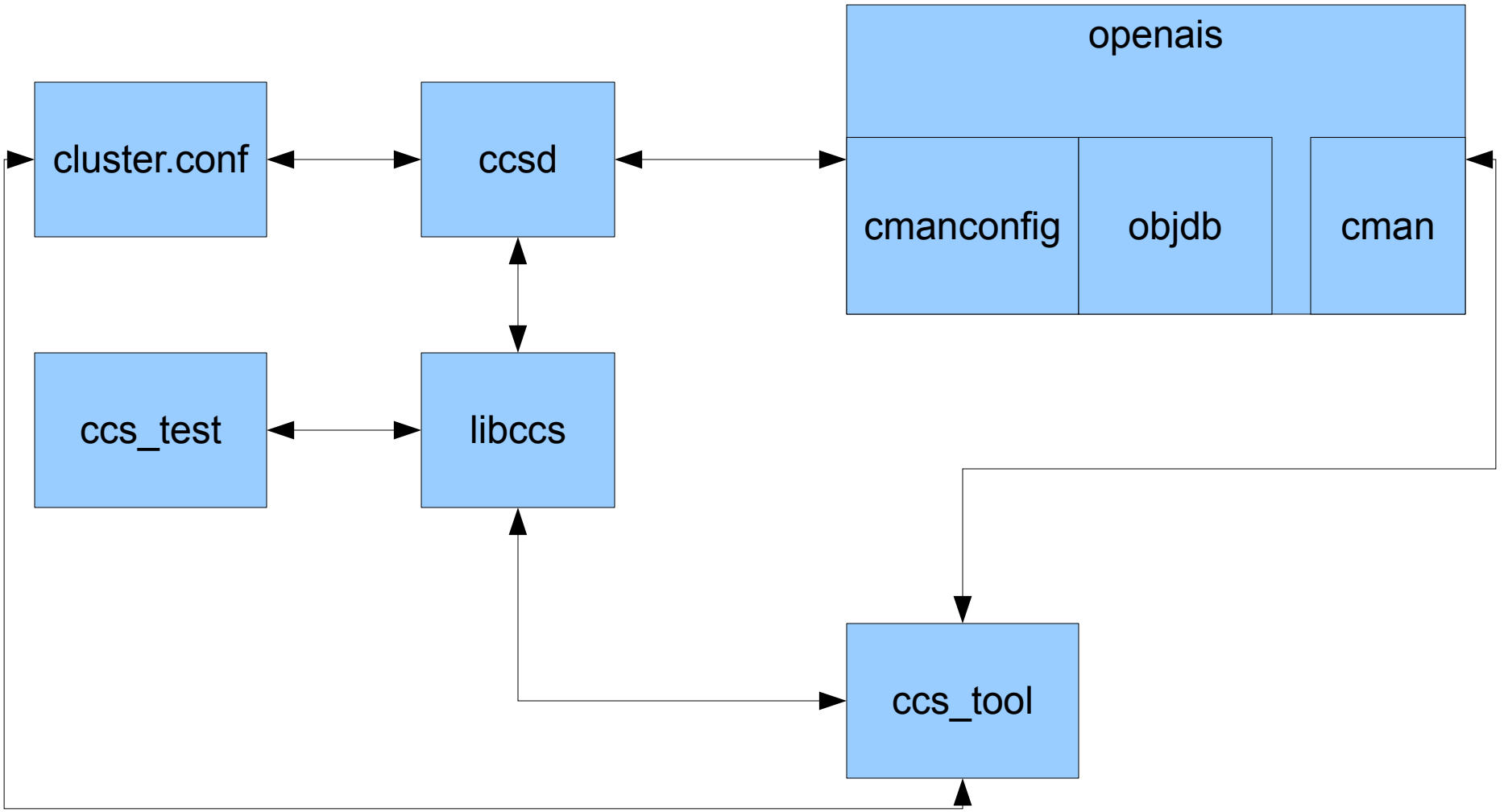
Yes, in order to understand where we are now, we need to make a slight digression in the past to look at how the old configuration system used to work and what motivations drove the rewrite into the new one.

STABLE2

CCS, in the old times, was composed by:

- ccsd – a daemon in charge to read cluster.conf (XML format) and keep it synchronised across the cluster nodes
- ccs_tool – a simple CLI tool to edit and manage cluster.conf
- ccs_test – an over-abused debugging tool that become de-facto standard to poll CCS information from CLI
- libccs – a C written library to access CCS from other applications
- cman_tool version – in charge of driving configuration updates at runtime

STABLE2


openais
cluster.conf
ccsd
cmanconfig
objdb
cman
ccs_test
libccs
ccs_tool

STABLE2

Cons:
• Code is old and buggy, difficult to maintain and duplicated in ricci/luci/conga.
• Cluster bootstrap issues: ccsd required quorum to operate, quorum required config from ccsd.
• Configuration updates are not dynamic and required too much interaction between tools.
• Configuration bits were duplicated between ccsd and openais/objdb.
• Lots of hacks in place to allow ccs_test to work from command line.
• Libxml2 was not used properly with lots of overhead in xpath.
• Not modular (does not allow sources other than cluster.conf)
• Applications did not have configuration change notifications.

Pros:
• Automatic config synchronisation at startup (when/if does work).

STABLE2

The typical workflow was:
• ccsd starts up, reads cluster.conf, loops waiting for quorum and offering cluster.conf unverified (available only in force mode, other apps would block on validation). If cluster.conf is not available, an attempt to gather the configuration from the network is done.
• openais starts, invoke cmanconfig to read the config from ccsd, fixes it for openais operations and writes it into the objdb, joins the cluster, tell ccsd that quorum is available.
• ccsd then verifies the configuration with the other nodes and takes actions (reject, update, etc).

• User updates the config, asks ccsd to reload the config via ccs_tool update
• User informs cman of the new config via cman_tool version

Other applications such as rgmanager would regularly poll ccsd to learn of configuration changes.

STABLE2 to STABLE3

Clearly the number of limitations in STABLE2, required a complete redesign of the configuration system.

The major drivers behind the new systems were:

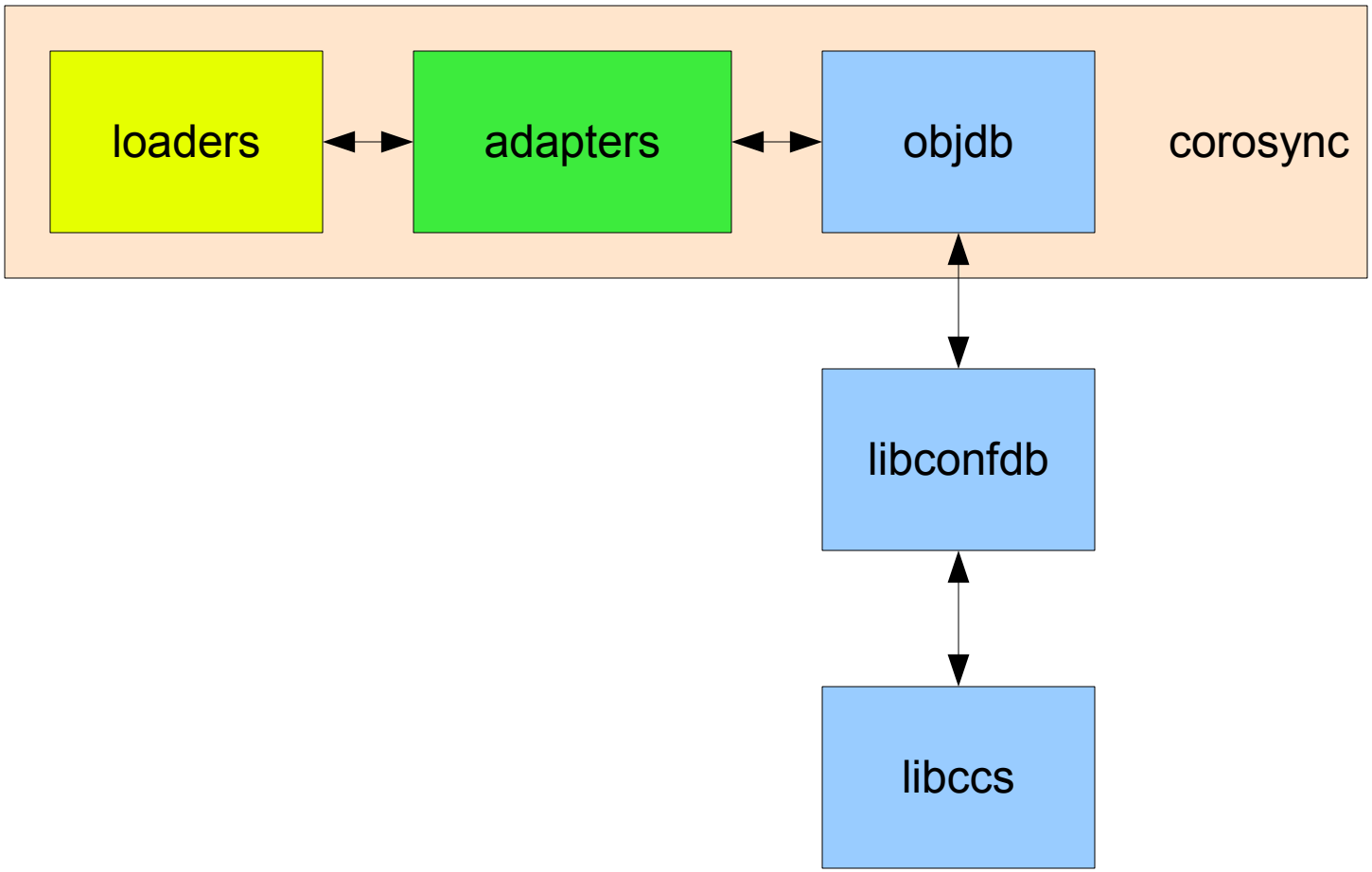
- decouple ccsd functionality of configuration provider and synchroniser
- allow the configuration to be loaded from sources other than cluster.conf

STABLE3

In order to address the different problems, without gigantic transitions, we had to:

- Maintain libccs API compatibility (to avoid changing all the applications on top and disrupting rolling upgrades).
- Offer modules to load the configurations without teaching each application about the different sources.
- Find a common location and format in which to store the runtime configuration (that is/or could be slightly different from on-disk configuration).
- Allow the new subsystem to notify applications of configuration changes (remove polling, more reliable configuration distributions to applications local to the node).

STABLE3



STABLE3

OBJDB
the perfect “format-agnostic” storage

OBJDB is a very simple, fast and efficent service to store information. The API is rich and provides very useful features such as notifications with callbacks, and global locking.

The unique feature of loading OBJDB without running corosync is the cherry on top of the pie.

It become clear, very quickly, that OBJDB would be our source-independent configuration storage.

STABLE3

Loaders

The task of a loader is to read the configuration from a specific format and load it into the OBJDB in the expected format.

A loader is nothing more than a format2objdb translator.

There are 2 constraints a loader has to respect.

A loader must:
- support both load and reload operations (write is optional)
- **NOT** change the configuration (1:1 translation)

STABLE3

Loaders

At the time of writing, there are 2 loaders available:

- xmlconfig
- ldapconfig

STABLE3

xmlconfig

Allows to load any xml file into the OBJDB. Default is set to /etc/cluster/cluster.conf (or equivalent set at build time).

It is possible to override the default config file by setting COROSYNC_CLUSTER_CONFIG_FILE environment variable.

This is currently the default loader.

STABLE3

ldapconfig

Allows the configuration to be loaded from an LDAP server into the OBJDB.

A detailed document on setting up ldapconfig can be found here:

http://people.redhat.com/ccaulfie/docs/ClusterLDAP.pdf

STABLE3

Adapters

The tasks of adapters is to make sure that the configuration that has been loaded into the OBJDB is valid and suitable for operations.

Adapters are allowed to modify the runtime configuration applying sane defaults where none are provided and should not care what the configuration source is. It will perform all the operations within the OBJDB.

It is important to understand that adapters will **NOT** change on-disk configuration.

At the time of writing, only the cman-preconfig adapter exists.

STABLE3

libconfdb

libconfdb, in short, provides access to the full OBJDB API for applications.

Cluster applications don´t use it directly, but it is worth mentioning it because it is the gatekeeper for accessing the OBJDB without running corosync.

STABLE3

libccs

is the only authoritative method to access runtime cluster configuration.

Generally speaking libccs is an XML view of the OBJDB.

It has been heavily rewritten to increase performance, error checks/reports.

STABLE3

libccs

has 2 operational mode:

• xpathlite (default), is a fast xpath emulator that supports only a very small subset of xpath features. It has been written to avoid the libxml2 overhead and fulfill all the current cluster xpath requirements.

• fullxpath, uses the full xpath engine as provided by libxml2. It has a heavier memory and cpu footprint because of the internal communication with libxml2.

The operational mode has to be set before initialising the library.

STABLE3

Chaining loaders and adapters

The schema in slide 10 only shows the current default configuration subsystem setting, by limiting the view to one Loader (xmlconfig) and one Adapter (cman-preconfig) but in reality it is possible to chain an endless number of Loaders and Adapters.

The env. variable COROSYNC_DEFAULT_CONFIG_IFACE can be set with a colon separated list of loaders/adapters to be executed at startup time.

The order of the list is respected both at load and reload operation time.

STABLE3

Command line tools

- ccs_tool – replaces ccs_test with query functionalities. Other features remain unchanged.
- ccs_sync (conga tool) – provides a single command to synchronise the configuration across cluster nodes.
- ccs_config_dump – allows configuration dump in XML format, independantly from the source.
- ccs_config_validate – validate the configuration against a RelaxNG schema.
- ccs_test is now a symlink to ccs_tool and emulates the behaviour of the old one for backward compatibility.

STABLE3

Configuration reload

“cman_tool version -r $version” is now the only tool required to perform the operation.

A lot of changes have been done for this operation. The most noticeable ones are:

- configuration is now validated before reload operations.
- configuration is automatically synchronised, via ccs_sync, before operation takes place.
- all cluster applications are now notified that the configuration has been changed (most applications will reload as much as possible. Some parameters cannot be changed at runtime).