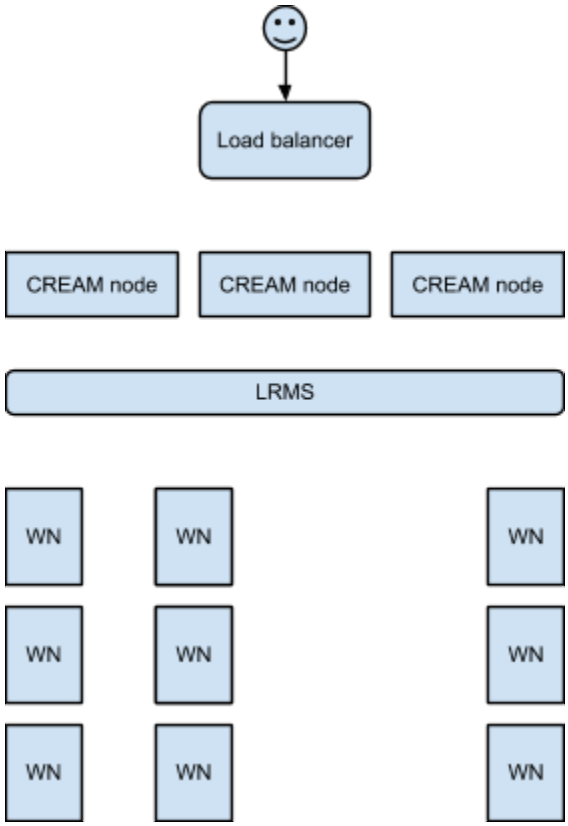
High Availability CREAM Guidelines

For any comment please send an email to the authors: Andrea Caltroni <andrea.caltroni@pd.infn.it> and Eric Frizziero <eric.frizziero@pd.infn.it> at INFN Padova.

v0.2 - 2013-05-14



Introduction

The critical aspect of an architecture designed for high availability is the lack of any single point of failure.
The goal is to have multiple nodes for any individual function, and to have automated failover mechanisms to route traffic away from any node that has become unavailable.

Load Balancing

These guidelines present a possible configuration for load balancing a CREAM cluster.

Problem Description

Minimize the time of interrupted service with the less expensive solution (in terms of money and secondly set up time) and reduce the service response time.
There are three possible ways to solve this problem:

1. Hardware with load balancing features. This is the most expensive solution but on the other side the most reliable one.
2. Load balancing software on a gate machine. This will likely interfere with the CREAM architecture. It's the most complex solution.
3. DNS load balancing and failover. DNS load balancing is the cheapest and simplest solution. It must be complemented with a good algorithm for selecting the best node

and failover management. With failover we mean removing a dead address when monitoring detects that that server is down.

We will provide guidelines for DNS load balancing plus failover management and selection of the next best node but nothing prevents a system administrator from trying to implement another solution deemed more appropriate to her site.
For example one solution we did not investigate is using IPTables rules. You can find a starting point on this reading rule #11 on this article:
http://www.thegeekstuff.com/2011/06/iptables-rules-examples/.

RR DNS Server Configuration

Round Robin is the simplest solution. It switches in turn to each host. CONs: no selection of the best node to switch to; not aware of unavailable nodes.
To configure the DNS server for delivering name resolution for a set of hosts in a Round Robin fashion, you need to add a line for each host of the cluster in the zone file:

```
; zone fragment for pd.infn.it
$TTL 60 ; zone default = 2 days or 172800 seconds
$ORIGIN pd.infn.it.
...
creamha    IN    A    192.168.0.1
creamha    IN    A    192.168.0.2
```

High TTLs are the default when the DNS server is used in a standard way (no load balancing). A low TTL (10-600) ensures clients do not store dead host values for too long. On the other hand setting it higher (1800-3600) may help your DNS propagation work reliably even if there is a short glitch in network traffic.

You might also want to update the reverse DNS configuration:

```
$TTL  60 ; 24 hours could have been written as 24h or 1d
$ORIGIN 0.168.192.IN-ADDR.ARPA.
...
; non server domain hosts
1       IN  PTR    creamha.pd.infn.it.
2       IN  PTR    creamha.pd.infn.it.
```

even if this is not strictly necessary.

NSCD

The Name Server Cache Daemon (NSCD) should be switched off. Caching IPs goes against the core of our problem which is a balanced answer.

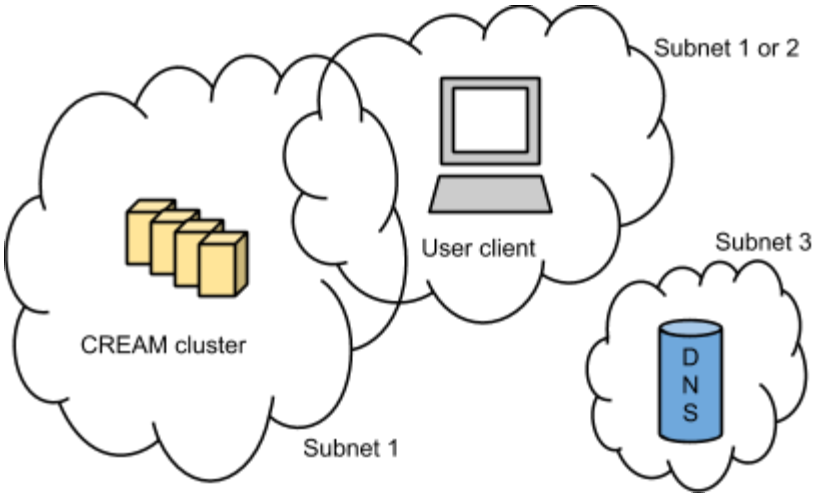
Topology

The user client generating the request should not be on the same subnet of the DNS server.

This is a rare event but we observed responses coming in blocks of identical IPs in this case, which might break the round robin behavior.

All hosts of the CREAM cluster should be on the same subnet. If this is not true, for some yet unknown reason the round robin nature of the answers is lost.

A user client making the request to the alias hostname can be on the same subnet or on another subnet. We verified a correct behavior under both conditions.



Failover Management

RR DNS load balancing alone does not provide failover management. Dead nodes must be excluded from the set of nodes to choose from.
Implementing failover management requires an additional component to monitor the cluster and scripts using information from the monitoring tool to update the DNS records adding new nodes or nodes which are back from a downtime and remove those which have become unresponsive.

[TODO] Add Eric's contribution here

There is no general purpose solution because it depends on the site size, traffic or other parameters.

You can base your solution on the scripts you can find here:
http://www.freebsdwiki.net/index.php/BIND,_dynamic_DNS,_failover_A_records

Other Selection Algorithms

In place of the simple Round Robin you could implement a smarter solution. The script used to implement failover management could additionally collect monitoring data to select the best node based on different algorithms and update the master DNS. The RR configuration can then be discarded.

Selection Algorithms:

- Least connections. Selects the server with the least number of active connections to ensure that the load of the active requests is balanced on the servers. When there is more than one server with the least number of connections, the algorithm selects the server in a Round Robin manner.

Acknowledgements

We would like to thank Alvise Dorigo at INFN Padova for the first tests, Vania Boccia, Luisa Carracciuolo and Davide Bottalico at the University of Naples and Stefano Dal Pra at INFN CNAF for the conclusive tests on their networks.

Redundant MySQL

You can choose from open source and commercial solutions to implement a high availability solution for MySQL. Below you can find a few examples.
Please note that we provide a link to these tools for your convenience but they are general purpose tools and have no connection with CREAM and we did not test them.
You can find specialized configurations, tips and opinions browsing the Web.

Open Source

MySQL Cluster

MySQL Cluster is a write-scalable, real-time, ACID-compliant transactional database, designed to deliver 99.999% availability. With a distributed, multi-master architecture and no single point of failure, MySQL Cluster scales horizontally on commodity hardware with auto-sharding (partitioning) to serve read and write intensive workloads, accessed via SQL and NoSQL interfaces.
http://dev.mysql.com/downloads/cluster/.
Reference documentation: Look for the MySQL Cluster chapter in the MySQL reference documentation. the exact title depends on the MySQL version.
For example for version 5.5 it can be found here:
http://dev.mysql.com/doc/refman/5.5/en/mysql-cluster.html.

Galera Cluster for MySQL

MySQL/Galera is synchronous multi-master cluster for MySQL/InnoDB database:
http://codership.com/content/using-galera-cluster.

mysql-master-ha

A primary objective of MHA is automating master failover and slave promotion within short (usually 10-30 seconds) downtime, without suffering from replication consistency problems.
http://code.google.com/p/mysql-master-ha/.

Tungsten Replicator

Tungsten Replicator is a high performance, open source, data replication engine for MySQL. http://code.google.com/p/tungsten-replicator/.

Commercial

MySQL Enterprise HA

A commercially supported version which offers its own support for high availability. More details on this page on the official MySQL Enterprise web site: http://www.mysql.com/products/enterprise/high_availability.html

MySQL Cluster CGE

A commercially supported version which offers many high availability features.
More details on the official page: http://www.mysql.com/products/cluster/.

Books

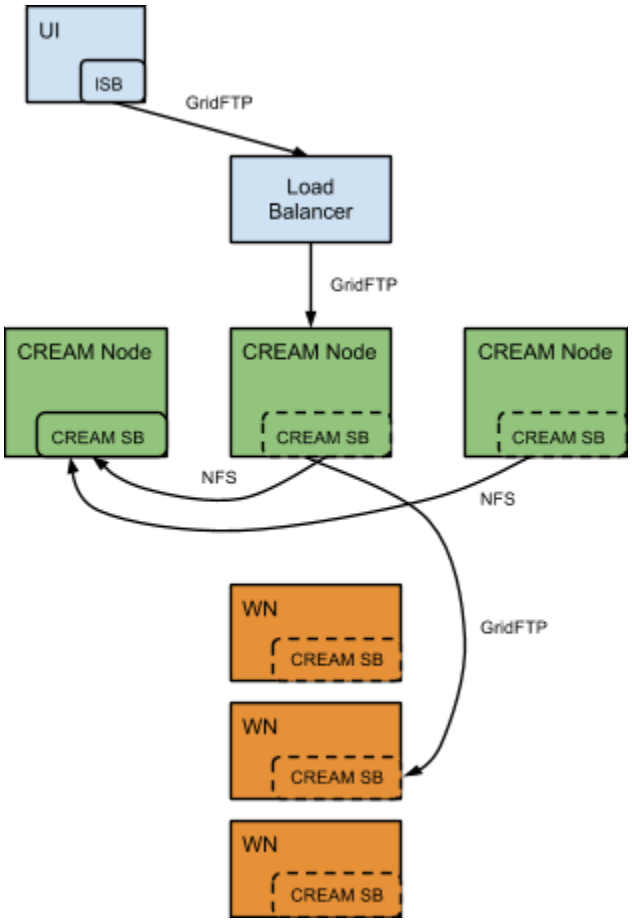
1. *MySQL High Availability. Tools for Building Robust Data Centers*; Charles Bell, Mats Kindahl, Lars Thalmann; O'Reilly Media; June 2010; 624 pages.

Shared Components

A few directories and files of the system need to be shared. Sometimes it's necessary for the correct functioning of the cluster, other times it's a matter of opportunity.

Sharing the Input SandBox

Sharing the input sandbox is necessary because [TODO].
The input sandbox is a directory on the user interface--one for each job--where the files needed by the related job are stored. This directory will be copied to the CREAM sandbox on the CE.
The CREAM sandbox (usually `/var/cream_sandbox`) contains a directory for each VO, user and finally job. Inside we can find the jobwrapper script, the input sandbox (/ISB) from the user interface, the output sandbox (/OSB) the StandardError and StandardOutput files. The CREAM sandbox is copied to the worker node where the job execution will be started by the jobwrapper.



In a clustered CREAM environment the CREAM sandbox on the CE needs to be a shared directory among all CEs of the cluster.
Each node will share the sandbox from one server node chosen at installation time. All operations from then on can proceed with no party knowing it is in fact a shared directory.

Server Setup

1. You need to install the NFS server package. On RedHat-based distributions both the server and the clients can be found in the package `nfs-utils`.

2. Uncomment the Domain line in `/etc/idmapd.conf` and set it to your domain.

3.
`/etc/hosts.allow`
`/etc/hosts.deny`
These files control access to services on your machine. When the server gets a request from a machine is does the following:
A. It checks `hosts.allow`, if the request matches a description there, the remote request is granted access.
B. It checks `host.deny`, fie the request matches a description there, the remote request is denied access.
C. If the request matches nothing in either files, it is granted access.

The order of this check is important

4. Add to the `/etc/exports` file the directories that need to be shared (exported):
`/var/cream_sandbox 10.0.0.0/24(rw,sync,no_root_squash,no_all_squash)`

Notes:
`/home` → shared directory
`10.0.0.0/24` → range of networks NFS permits accesses
`rw` → writable
`sync` → synchronize
`no_root_squash` → enable root privilege
`no_all_squash` → enable users' authority

5. Service Startup
```
> /etc/rc.d/init.d/rpcbind start
> /etc/rc.d/init.d/nfslock start
> /etc/rc.d/init.d/nfs start
```

6. Configuration
```
> chkconfig rpcbind on
> chkconfig nfslock on
> chkconfig nfs on
```

7. Mount Point
Check that the root of the mount directory (/var/cream_sandbox) exists and has the appropriate permissions (775) and owner (tomcat:tomcat). They will override the permissions of the mount point on the clients.

Client Setup

1. You need to install the NFS client. As said above, on RedHat-based distributions both the server and the clients can be found in the same package `nfs-utils`.

2. Uncomment the Domain line in `/etc/idmapd.conf` and set it to your domain.

3. Service Startup
```
> /etc/rc.d/init.d/rpcbind start
> /etc/rc.d/init.d/rpcidmapd start
> /etc/rc.d/init.d/nfslock start
> /etc/rc.d/init.d/netfs start
```

4. Configuration
```
> chkconfig rpcbind on
> chkconfig rpcidmapd on
> chkconfig nfslock on
> chkconfig netfs on
```

5. Manually mount the remote directory:

```
> mount -t nfs creamha.pd.infn.it:/var/cream_sandbox /var/cream_sandbox
```

Check if it correctly mounts with `df -h`.

6. Now you can automatically mount it at boot time adding a line to `/etc/fstab`:

```
creamha.pd.infn.it:/var/cream_sandbox /var/cream_sandbox nfs defaults  1 1

# 131.154.193.104: hace-02.cr.cnaf.infn.it
# 131.154.193.5: xenserver5.cr.cnaf.infn.it
xenserver5:/cream_shared /cream_shared nfs
rw,vers=4,addr=131.154.193.5,clientaddr=131.154.193.104 0 0
```

Sharing the Configuration Directory

Sharing the configuration directory is a matter of opportunity. The configuration files (the most important are `cream-config.xml` and `log4j.properties` in the directory `/etc/glite-ce-dbtool/`) could be kept duplicated on each host since once installed they're seldom modified.
On the other side if you don't install CREAM on each host using an automated configuration tool (like Yaim) you have to configure each host separately. Plus having the configuration in a shared place keeps the file synchronized without effort and saves you from errors once you need to modify them.
The same node chosen to host the shared job sandbox could share the configuration directory too (see section Sharing the Input SandBox).

Server Setup

Follow points 1-7 from section Sharing the Input SandBox - Server Setup.
Modify point 4 and 7 as follows:

4. Add to the `/etc/exports` file the directories that need to be shared (exported):

```
/etc/glite-ce-cream 10.0.0.0/24(rw,sync,no_root_squash,no_all_squash)
```

7. Mount Point
Permissions for the root of the mount directory (`/etc/glite-ce-cream`) should be 755 and ownership tomcat:tomcat.

Client Setup

Follow points 1-6 from section Sharing the Input SandBox - Server Setup.
Modify point 5 as follows:

5. Manually mount the remote directory:

```
> mount -t nfs creamha.pd.infn.it:/etc/glite-ce-cream /etc/glite-ce-cream
```