INTRODUCTION TO LINUX CLUSTERING

DOCUMENT RELEASE 1.1

If you publish this document anywhere, please do let me know via email, and if it is published in a physical medium, sending me a copy would be appreciated.

Email: jethro.carr@jethrocarr.com
Website: www.jethrocarr.com

Table of Contents

1 Introduction

One of the oldest problems of computing is designing failure proof computing systems. Over the years, many different methods have been developed.

Many of these you will be familiar with, including:

- Spare hardware – typically enough to mirror the production hardware.
- Fault tolerant hardware by using a number of spare components.
- Software failover features built into individual programs.

Hardware solutions are typically expensive (and not always 100% reliant). Application-specific failover methods can often add maintenance hassles as well as doing nothing to fix the problem that some of your programs may have no failover capabilities at all.

In recent years, a few new options have become available - in particular Linux clustering and virtualization.

This document presents the key factors in implementing effective Linux clusters and design. Please note the following:

- Clustering is a relatively new topic to me, and I have not had a lot of experience deploying and maintaining clusters.

  This document is the results of my research into the options available and looks at what solutions could be developed. If you have experience with clusters and have found technologies that do or do-not work well in practice, please supply me with feedback so I can extend this document and make it more useful.

- This document mentions Redhat Cluster Manager a bit, but doesn't go into details about the other major option, Linux-HA. However, most of the concepts and terminology is applicable for both solutions.

- This document does not cover the technical details on configuring clusters, it is more of a high level design view.

This document will also cover the use of clustering together with Xen virtualization for maximum advantage.

The aim of this document is to provide you with an understanding of cluster solutions so that you are empowered to identify applicable technologies and decide on the best approach to use them.

2 About clusters

There are three main reasons to use clustering:

- Better performance
- Fault tolerance by high availability services.
- Optimal usage of disk resources.

Often you will hear about high performance computing solutions using Linux clusters to create small super computers – such systems are usually referred to as “beowulf clusters”.

These systems typically run highly customised applications which are designed to run on multiple computer systems at once, and are beyond the scope of this document.

This document will cover the following:

- High-availability clusters.
- Shared storage solutions
- Other clustering considerations.

3 Advantages and reasons for clustering

Clustering provides a number of advantages over traditional standalone server configurations. First there are a number of obvious ones:

- High-Availability of system services.

  The cluster management software will handle service failures and will quickly bring up the service on alternate hardware.

- Better utilisation of system resources.

  Services can be spread around all the nodes – heavy nodes can be lightened by moving services away, lightly loaded nodes can have more services started on them.

  When you combine Xen with clustering, even more options become available, as you can split one node into many dozens of nodes.

- Optimisation of disk resources.

  Rather than having lots of small local disks, storage can be centralised or distributed across all machines making better use of the storage available.

There are also some other useful advantages:

- Usability of older hardware or whitebox hardware

  IT departments like to purchase new servers from a big name vendor like IBM or Dell, who will then provide 5 years of hardware replacement and service. However, once this 5 year period is over, the hardware is no longer supported and the customer either has to carry their own spare parts (which can be too expensive for a small organisation) or upgrade to newer hardware, which requires time, money and effort to migrate all the data and applications.

  With clustering, if the hardware fails, another node will take up the work – so companies can build their IT infrastructure around older hardware or custom built hardware saving a lot of money.

  Some hosting providers may even decide to leave old host in the cluster and to just keep adding new ones, and only pull out the old ones once they fail or once they reach an age where they are uneconomical to keep running.

- Faster more efficient system administration

  In server farms where all the servers are configured individually, it can often be quite hard to

migrate a service from one computer to another, which can sometimes be required due to security or performance reasons.

However, in a clustered environment, all the servers are typically identical. Plus, any service that has been setup in the cluster, can be moved from one host to another by the simple execution of a single command.

4 Clustering fundamentals

4.1 Basics

High-availability clustering is a complex topic, and it is important to fully understand key concepts behind it.

The basics are simple:

- Each computer is called a node.
- Two or more nodes form a cluster.
- In the event of a failure of any one of the nodes, the remaining nodes will take up the work being performed by the dead node.

What makes clustering complex, is how the cluster handles node failures, shared disk storage and situations such as split-brain.

A cluster typically works as follows:

- All the nodes run the cluster management software (eg: Linux-HA or Redhat Cluster Suite), which controls stuff such as heartbeats, application starting/stopping and keeping quorum (more on this later).

- One of the nodes runs an administration application, that allows you to manually add/remove nodes and provides the ability to manually move applications from one node to another.

- In the event of a failure with a node, the other nodes fence it, which involves using a hardware device like a managed power switch to physically turn the node off. This is done to prevent the node from writing to any of the storage devices and corrupting the data.

  The other nodes then decide which node should run the applications that were on the dead node, and one of the nodes will be chosen (depending on the config) and will start up the application.

- All the cluster nodes have access to a central storage array (eg: a SAN or network attached storage). This storage location runs a clustered filesystem which allows all the nodes to read and write at the same time.

4.2 Important clustering components

4.2.1 FAILOVER

There are 3 types of failover methods that exist.

1. Hot failover

   In a hot failover, the application is written specially for clustering and is able to continue running on another node, without any interruption to client services.

   Hot-failover is not often found in commonly used applications, and is usually found in specially written programs for banking or telco situations.

2. Warm failover

   Warm failover is what solutions like Linux-HA and Redhat Cluster Suite provide – the application doesn't have an instant recovery feature, but the cluster suite quickly restarts the application without minimal client disruption on another piece of running hardware.

   Using a cluster management solution, the application does not need to be written to support clustering, so you can provide redundancy for any service you desire.

   This can sometimes cause a small outage, as some applications can't tolerate the change of the server in the background. In other causes, the application is able to continue on, with no interruption to the client users with the exception of a bit of a delay (eg: NFS is good at not being affected by a server change behind the scenes).

3. Cold failover

   Cold fail-over is commonly used as a solution for redundancy where a cluster was not able to be setup. In a cold failover, the dead computer need to be powered down, and a spare computer started up. This is usually a manual process.

4.2.2 FENCING

When a node crashes or becomes unresponsive, it MUST be quickly powered off or blocked from the storage device (fencing).

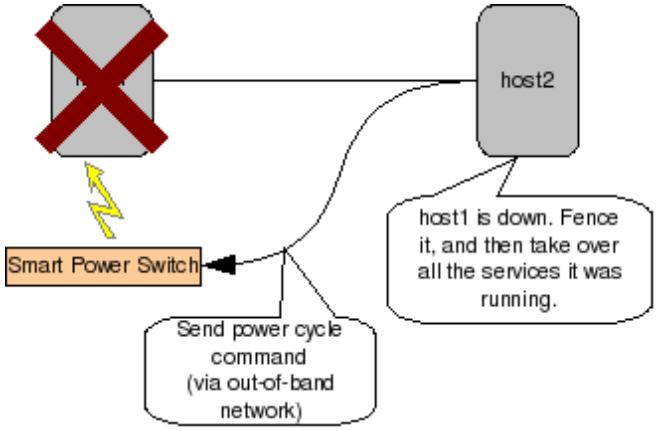
Fencing is required, because if the cluster assumes the node has crashed, and reallocates it's services and IO, if the server was to wake up, it could cause havoc and possibly disk corruption.

Therefore a failover device must be available so that the cluster can do STONITH – Shoot The Other Node In The Head – by doing an instant power off of the node.

Various devices exist – smart power switches are usually used and scripts exist in Redhat Cluster Suite that can connect to a number of commonly available devices to shutdown

servers.

Other fencing devices include fencing at the SAN level as well as Xen VM fencing, however it is recommended that power fencing be used rather than SAN fencing as it will guarantee that the node is completely killed and not doing anything unwanted.



4.2.3 SPLIT-BRAIN

Split-brain is a nasty problem found in clustering, and requires careful thought to prevent. Consider the following scenario:

1. A two node cluster exists – one server in city A, one server in city B.
2. The internet link between the two cities falls over. Neither server can contact each other.
3. Each server assumes the other one is down, and both resume activities as the master.
4. When the link comes back online, data corruption occurs.

Other nasty problems can occur if the two nodes are still able to fence each other via the out-of-band management system, as you may end up with each node repeatedly powering off the other node.

To prevent this from happening, we have a solution called “Quorum”.

4.2.4 QUORUM

Quorum is effectively a scoring method, where each node in the cluster has a number of votes (by default one). Each on-line cluster node adds it's votes to the quorum count, and as long as the quorum count is larger than 50% of the combined votes, the cluster is intact.

If the cluster falls below quorum, the cluster has “lost quorum” and all services will shut-down and become unavailable.

This is actually a desired feature. Consider a cluster with 10 servers and to maintain quorum, a score of 6 is needed. In the event of the network suffering a failure and causing the cluster to spit into two, the smaller half will shut-down and the larger half will continue on. This prevents split-brain in clusters.

What about situations where there are an even number of nodes, such as 10 nodes? It would be possible for the network to split into two equally sized clusters. Therefore, any cluster with an even number of nodes requires at one of the nodes to have an additional vote in the

quorum to unbalance the quorum voting.

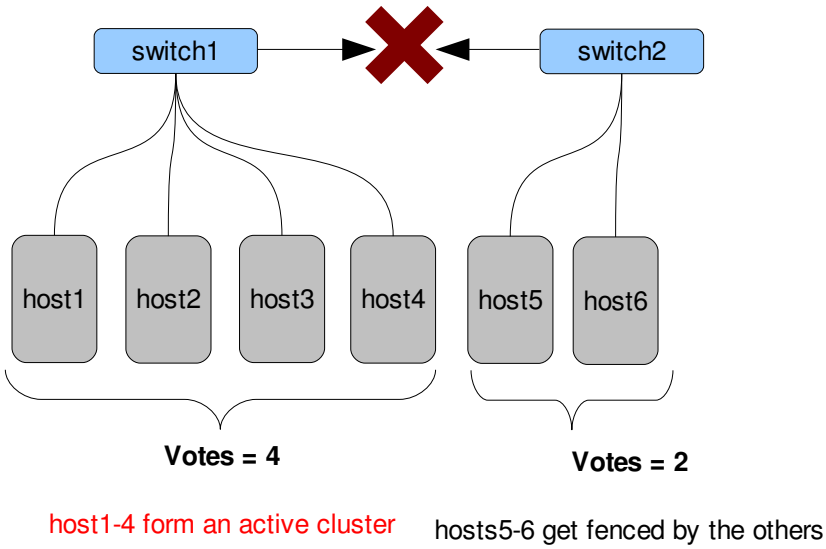
Effectively, quorum allows you to make an even cluster, uneven when it comes to failovers, so that split brain will not occur.

But what happens when you only have a two-node cluster? A failure of either node would cause the cluster to lose quorum, as both machines have equal votes. In this case, you need a tie-breaker vote to determine the master.

There are two ways this can be done:

1. If using a SAN/NAS, the central storage device can be used to provide the tie-breaker vote by using a custom filesystem called a quorum disk.

2. Setup a heuristics test. This can be any program (typically something like ping) that can provide another vote if the program completes successfully. In a two-node cluster, the most likely candidate would be to ping the network gateway, or a remote



5 Cluster management software

To control the cluster and the movement of services, a cluster management application is required – Linux has two main options available - Linux-HA and Redhat Cluster Suite.

Redhat cluster suite is only found on Redhat's distributions- Fedora, RHEL and other derivatives such as CentOS, whereas Linux-HA is found in a wider range of distributions.

Most of the concepts and ideas between these two solutions are the same, so the knowledge gained using one is likely to make it easy to use another.

5.1 Redhat Cluster Suite

Cluster Suite is designed for creating High-Availability clusters and the default steps to configure a service will result in a HA-service depending on the number of nodes you put into the failover domain.

5.1.1 MANAGEMENT AND CONFIGURATION

Configuration of the cluster is controlled by the /etc/cluster/cluster.xml file, which is an XML format file. Once changed, you can run a command to redistribute the file to all the other nodes in the cluster.

However, most people do configuration using either system-config-cluster (GTK GUI application) or luci which is a web-based utility for cluster configuration.

The configuration of the cluster is broken in the same components regardless of the configuration method chosen:

- Resources

  Resources are anything that makes up a service. For example, a resource may be:
  1. An IP address.
  2. A mount point.
  3. A system service (eg: httpd)

- Service

  A service is a group of resources that have been given a name. The service configuration allows you to set what order the resources are started/stopped and gives you a name that you can use to control the service.

- Failover domains

When a cluster node fails, the services that are running on it need to be migrated to other machines. The cluster management software will look at a list of other nodes called the “failover domain”, and will select one from the list to run the service.

The list can also be prioritised if desired – both the order of what servers should be used in a failure, as well as whether or not the service should be moved back to a higher priority node when one becomes avaliable.

In the event of the service running out of online hosts in the failover domain, the service will be stopped until a node comes back online.

- Fencing devices

  A fencing device is used to power off or reset unresponsive/crashed cluster nodes. This is typically something like a network controlled power strip, or a out-of-band management card in the server.

5.1.2 LUCI AND RICCI

Luci does not have to be run on the cluster itself, although that is the recommended method as you can cluster luci and thus be able to always administrate the cluster.

The ricci daemon runs on all the nodes and allows Luci to communicate with the nodes to configure them.

Luci is smart enough to install the packages it requires via yum on the nodes when you add them to the cluster which makes setup easier.

5.1.3 SYSTEM-CONFIG-CLUSTER

Luci seems to be replacing system-config-cluster as the favourite program to use, but at this stage system-config-cluster is a capable GTK GUI application for cluster configuration.

You can run it on any cluster node, once you save your changes you can then click a button to send out the new configuration to all the cluster nodes.

5.1.4 LOAD BALANCING

Redhat Cluster Suite is focused on providing HA servers and doesn't provide any special features for doing load balancing.

However, you can setup load balancing by the following method:

- Setup a device on the network (eg: a reverse-proxy or session balancing application) that passes the session traffic to one of the cluster nodes. This device may be a hardware device or a standalone PC – maybe even a two-node cluster to ensure HA of the load balancer!

- Setup multiple service in the cluster suite for the number of nodes you want to load balance.

- Configure the services to belong to various failover domains – for example, you may not want to failover some services as long as you run one instance at a minimum – so you setup one service with a failover domain and setup the rest of the services to only run on the one node.

6 Combining Xen with clusters

Virtualization technology is becoming increasingly popular due to the reduced costs and better utilisation of hardware resources.

Linux has various solutions for virtualization – one popular option is Xen which comes bundled with a number of distributions. So the next step, is to combine the advantages of Xen, with the advantages of clustering.

Whilst there are numerous ways this can be done (and the best solution will depend on what you need). One basic method is to configure all the physical machines as Xen host servers and place them into a cluster. (Ideally, the host servers should also be in the same Xen domain to allow live migration of VMs from one server to another.)

The cluster management software can then be configured to treat virtual machines as services – moving them between cluster nodes and restarting dead VMs on other machines.

However, this only provides basic failover services – a VM will only be moved to another host if the whole VM becomes unavailable, or if the host node crashes.

For more fine-grained control, there are two options:

6.1 VMs as part of main cluster

Add all the virtual machines to the cluster as nodes, along side the physical servers. The services can be configured using failover domains to only failover to virtual machines.

This means you only have one cluster, but it will introduce a lot more complexity into the cluster configuration and administration.

6.2 VMs runs as a separate cluster.

The other option is to run the cluster software on the Xen VMs themselves. This can be useful in that it allows you to configure multiple clusters on top of the main cluster which may be appealing to hosting providers who can offer customer their own private two-node clusters.

This method is also useful for systems that intend to run large number of services on top of a single VM and provides the ability to migrate individual services from one VM to another.

The disadvantage is that more overhead is introduced by running the additional clustering software on all the nodes and it may become more time consuming to manage.

7 Storage Management

Storage management may appear to be a separate topic, but it is in fact a very important part of a cluster's design.

For a cluster, it is very important that data remains intact and accessible by all the nodes. One major topic is the use of a cluster-capable filesystem such as GFS.

It is also important to choose the correct storage media for the cluster, taking future growth into consideration. Is performance or reliability more important? Are all the nodes in the same premises, or do you require a distributed storage solution that will work across the internet? Does the data need to be replicated in real time between the nodes?

There is a vararity of solutions available (such as SANs), however all solutions fit into one of the following three categories:

- Centralised storage.
- Distributed storage.
- Replicated storage (this can sometimes be a feature of either of the two categories above)

7.1 Centralised storage

Centralised storage involves having one or more devices providing storage to all the other nodes. A typical example is a single array of disks such as SAN or NAS, which all the nodes connect to for storage.

Centralised storage solutions are often found in enterprise server installations, with many medium to large organisations using something like a SAN for their data storage needs.

Centralised storage is popular for a number of reasons:

- More cost effective to purchase a single array of disks than purchasing disks for each server.
- Central location allows for easier backups and mirroring.
- Easy to configure, easy to maintain – if you need to add more storage, there's only one device to upgrade.

However, there is a common problem with centralised storage - Often there will be just a single device providing the storage. (often due to the cost of purchasing redundant hardware being too high, devices such as SANs are not cheap).

This introduces a single point of failure – if a hardware fault occurs in the device, it could cripple the entire cluster, since all nodes rely on it. To prevent this, you either need to be prepared for the possibility (and cost due to downtime) of a device failure or invest in redundant hardware.

Centralised storage systems typically export the disk space as a block device which appears on the nodes as a local disk, which then needs to be partitioned and formatted – so you can run whatever filesystem you wish on top of it.

This differers from network filesystems like NFS which appear as a mountable filesystems and can not be partitioned or have other filesystems ontop of them.

7.1.1 SAN – STORAGE AREA NETWORK

A SAN is a hardware device consisting of a number of hard drives in RAID. The SAN is then attached to each cluster node by fibre channel.

Advantages

- High-speed performance.
- Directly attached, so no issues due to network loss, congestion, etc.
- Tried and tested technology.

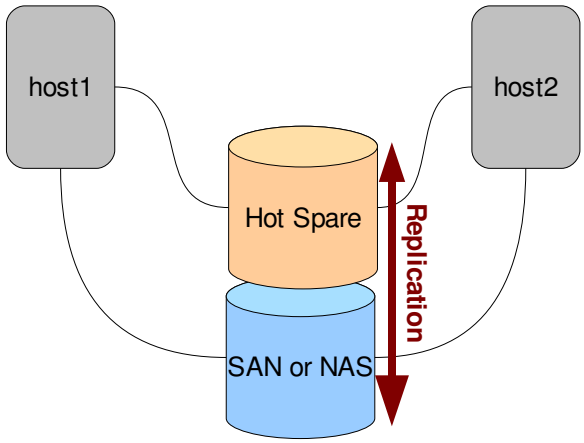
Disadvantages

- Expensive – every node needs to have a fibre channel communication card installed, both the SAN and the fibre channel hardware is expensive.
- Limited scalability – the number of cluster nodes possible are limited by the number of interfaces on the SAN.

Redundancy

For proper redundancy to prevent an outage in the event of a hardware failure, it is required to purchase two SANs which are capable of mirroring each other, and having fibre channel cards in the servers capable of talking to both SANs.

Without this redundancy all the work and resources put into developing a high-availability cluster will be wasted when the SAN dies.



Ideal Use

Suitable for use in clusters where all the nodes are located on the same physical site, as well as being suitable for use in clusters requiring maximum I/O performance.

However for budget concious organisations, a NAS may be a better option.

7.1.2 NAS – NETWORK ATTACHED STORAGE

A NAS is a hardware device consisting of a RAID array of hard drives (like a SAN). However, instead of using fibre channel, it connects to a standard ethernet network and supports protocols like iSCSI or ATA over ethernet.

It still appears as a block device which is configured to appear as a real disk on the server

Advantages

- Can be a lot cheaper than a SAN
- No physical limit to the number of nodes possible. (instead, limited by the number and speed of the ethernet interfaces and network)
- Commodity servers work out of the box.
- With high-speed networks like 10gig ethernet, performance can surpass fibre channel SANs.

Disadvantages

- Network outage can cause a failure which can cause outages preventing access to the storage. However, the risk of this can be reduced, by running the NAS traffic on a separate network to the normal network. (dedicated hardware, etc.)
- Some network storage protocols like iSCSI introduce overheads – for example iSCSI is a TCP/IP protocol, which adds overhead of TCP windowing, headers, etc. Other protocols like ATA over Ethernet add less overhead and should definitely be a factor to consider when evaluating NAS.

Redundancy

For proper redundancy, a second NAS should be purchased and made to mirror the primary NAS. They can either be set to have a floating IP addrress, or both of them can have iSCSI exports which are then multipathed on the nodes (so the node can choose which one to use).

If you require additional redundancy for the nodes, a second ethernet card can be installed into the nodes and both cards can be put into bonded mode.

In bonded mode, both interfaces work together to provide one virtual interface – if an interface fails, the other one will continue to work.

With many servers coming with dual ethernet interfaces out of the box, there is usually little need for further hardware investment for the nodes.

Ideal Use

From a high-level view, a SAN and a NAS is a very similar device, so make your choices based on what will give you the best value for your investment based on your needs.

A budget minded organisation may find a NAS over 100mbit/1000mbit ethernet provides the

best result, whereas others may find the performance of a SAN to be better.

Also consider future expandability – if you decide to grow the cluster in the future, a SAN will probably be less expandable than a NAS. It's also easier to upgrade the speed of an ethernet network with a faster switch, or more ethernet cards in the servers.

7.1.3 COMMODITY NETWORK FILE SHARE

Another option for centralised storage is the use of a commodity server with a large number of hard drives using RAID.

This can be used in two different ways:

1. Using a block sharing software solution such as iSCSI, ATAoE or GNBD, to effectively create a cheap NAS using standard computer components.
2. Running a network file system such as CIFS or NFS.

Advantages

- Cheap and simple. You can use standard computer off the shelf from a local store to built this.

Disadvantage

- Does not provide full redundancy in the event of failure of components such as motherboard or CPU. However, this can be off-set using something DRBD which is covered further on.
- Usually less performance than a SAN or NAS, which is tuned to allow maximum IO.
- iSCSI software targets might not be supported by your Linux vendor and may suffer performance issues.

Redundancy

This tends to be limited to hardware redundancy of server. Typically a server will only have disk redundancy (and ethernet via use of bonded interfaces), although some more expensive models offer PSU and even CPU redundancy.

Once solution that allows good redundancy is to run two identical servers with DRBD and replicate the filesystems between the servers – effectively both servers will have exactly the same data on them.

Ideal use

An iSCSI software target running on a server can provide a cheap NAS emulator for use in development environments.

This solution could be used anywhere a NAS is, however it will require careful tuning and smart hardware purchases to get optimal performance. One example that will make a difference is where or not your ethernet cards have TCP off-load engines, which will

increase performance when using a TCP-based storage service like iSCSI.

7.2 Access methods for centralised storage

Whilst centralised storage can sometimes provide the data via a network filesystem, it is more common to use iSCSI or a SAN.

Both iSCSI and SAN provide access to the storage as if it was a local disk. It is then necessary to run a cluster-capable filesystem on top of them such as GFS.

Note of interest: It is possible to have a non-clustered filesystem on a shared drive, but seriously bad issues would occur if you accidentally mounted it in two places at one time.

7.2.1 ACCESSING SAN DIRECTLY

A decent SANs allows the administrator to split the SAN into a number of logical hard drives, and then export only the desired drives to each node.

The drives appear on the node just like any locally connected SCSI/SATA drive.

7.2.2 ACCESSING NAS WITH ISCSI

Like a SAN, many NASes can be configured to split the storage into a number of logical drives.

Because the NAS is not connected locally, it uses a TCP/IP protocol called iSCSI. This means iSCSI can be routed, and even transferred over the internet (although the performance on this would be terrible without a high-speed, low-latency link).

iSCSI is used by attaching an iSCSI target. Once connected the iSCSI export appears just like a local SCSI hard drive.

It is important to note that the naming of the drives may be change, thus it is important to use udev to ensure stable naming.

Further information about how to identify and name iSCSI devices using udev can be found in the scsi_id man page.

7.2.3 ACCESSING NAS WITH OTHER PROTOCOLS

If you are using some other protocol, such as ATA over Ethernet, you will need to run software on the nodes to make the NAS shares appear as block devices on the server, which is similar to iSCSI in concept.

7.3 Distributed Storage

Distributed storage takes another approach to the storage mechanism, and instead of having a central location of data, the data is spread across all the nodes in the cluster, often including some form of redundancy in order to be able to cope with the failure of a node.

Unfortunately, this redundancy comes at a cost – distributed storage solutions are complex, and have to be able to handle issues such as the failure of nodes, delays in the network linking the machines and locking issues.

Note: You may have come across DRBD, which is a two-node block replication solution. This is covered in the next section of this document, under “Replicated Storage”.

7.3.1 AFS – ANDREW FILE SYSTEM

AFS is a distributed filesystem which caches data locally on machines. Currently there are two different implementations of AFS for Linux:

- OpenAFS (IBM Public License)
- AFS implementation in vanilla kernel (under development)

The caching provides increased speed and limited off-line access in the event of network failures, but the servers do not replicate themselves (although that could be achieved with DRBD).

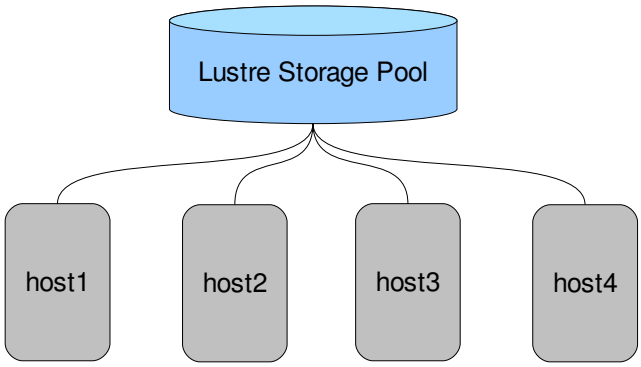
However, due to the types of file locking used, it is not suitable for large shared databases, and can not handle a single file being updated by multiple clients.

AFS was designed to run services such as mail servers using maildir where each email is stored as an individual file.

7.3.2 LUSTRE

Lustre is a distributed filesystem suitable at creating massive (many thousands of nodes) distributed filesystems.

Lustre is quite a complex technology to setup and unfortunately does not provide it's own data replication system. If data replication is required, then another technology like DRBD is needed to perform the replication between individual nodes, which does limit the scalability.



7.3.3 CODA

Coda is an interesting filesystem with features for allowing offline data caching for client computers, as well as server replication.

Unfortunately, Coda has only really been deployed in research situations and is therefore not suitable for running in a production environment, but is worth a mention here.

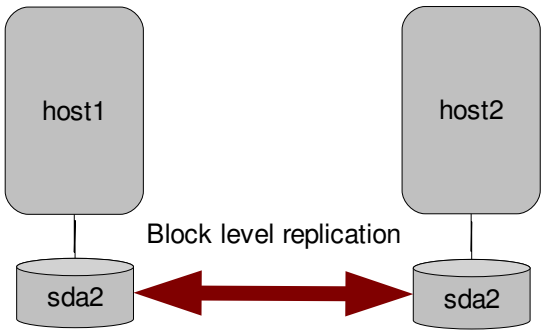
7.4 Replicated Storage

Some distributed and some centralised storage systems have inbuilt methods for data mirroring (eg: two SANs with hardware mirroring enabled). However, there are also software solutions that run at the block level and which can mirror any filesystem on top of them, the most popular one being Distributed Replicated Block Device (DRBD)

7.4.1 DISTRIBUTED REPLICATED BLOCK DEVICE (DRBD)

DRBD is a commonly used to provided block-level disk replication on two-node clusters, by mirroring the disks between the servers ensuring they have the identical data on them.

Unfortunately, DRBD suffers from the limitation of only supporting up to two-nodes, although there is a commercial closed-source version released by the developers that allows the addition of a third node.



This makes DRBD very useful for creating HA two-node servers, but not useful for creating a large shared storage area for large multi-node clusters.

DRBD is an ideal solution for a two-node cluster that is geographically separated – such as mail or web servers.

DRBD can be configured to work in one of two ways:

- Primary/Secondary – The storage device can only be mounted on one node (primary) at any time, the two nodes simply mirror the storage. In the event of the primary server going offline, the secondary server can become the primary. This is controlled by the cluster management software.
- Primary/Primary – In recent versions of DRBD, it is now possible for both nodes to run as primary – so both nodes can read/write. This requires use of a cluster-capable filesystem such as GFS to run on top of DRBD.

The method of writing can be configured – the default is to only count a write as complete once both nodes have been written to, but other options can be chosen in order to improve performance at the cost of reliability.

DRBD is commonly used with the Linux-HA cluster management software, however it is possible to make it work with Redhat Cluster Suite by preparing a start/stop script for it.

Because DRBD only supports two nodes, in the event of requiring three or more nodes, there are some methods that can be used to work around this limitation:

1. Setup two DRBD nodes that handle all the storage and all the other nodes connect to the two storage node using a network filesystem like NFS or some other protocol like ATA over ethernet or GNBD. (effectively creating your own replicated NAS device)

2. Setup all the nodes in pairs – each pair mirrors with DRBD and then runs a distributed filesystem such as Lustre or AFS on top of them. This will always have the weakness that the failure of both nodes in a pair would cause failure of the entire array, but otherwise the failure of any one node in any of the pairs will not disrupt the filesystem services.

3. Run DRBD on top of DRBD – this is not a recommendation, it is mentioned here because sometimes people do this. DON'T DO THIS. It introduces a huge number of problems and limitations as well as the unknown stability of running DRBD on top of DRBD yet again.

4. Modify the DRBD code base in order to add support for additional nodes. There does not appear to be any obvious theoretical reasons why this wouldn't be possible, it should just be a case of adding additional nodes and perhaps applying modifications to the distributed lock manager to make it suitable for three nodes.

   There would obviously be more of a performance impact due to increased amount of overhead for each node added, however as technology advances, this should become less of a problem.

7.5 Clustered Filesystems

When using a block-level storage system like iSCSI, SAN, GNBD or DRBD a cluster capable filesystem should be used, in order to allow multiple nodes to read and write at the same time.

A clustered filesystem differs from a conventional filesystem by including features to handle file-locking and journalling from multiple nodes.

7.5.1 CLVM

It is possible to run LVM on top of a centralised block storage device, by enabling clustered locking in the LVM configuration and running the CLVM service together with CMAN for clustering.

Once clustered LVM is enabled, it can be used in the exact same way as conventional LVM.

7.5.2 GFS – GLOBAL FILE SYSTEM

GFS is a clustered filesystem developed and supported by Redhat, and available on RHEL. It is fully open source, and Redhat are currently working on getting it merged with the mainstream kernel.

GFS has a number of powerful features that make it ideal for use in production clusters:

- Tried and tested technology, fully supported for customers of Redhat.
- Scales up to hundreds of cluster nodes.
- Supports extended access control lists.
- Supports user quotas.
- Dynamic symlinks (known as “Context Dependent Path Names”) which allow the symlink to point to different locations depending on various variable of the node using it. Ideal for node-dependent configuration.

7.5.3 EXT4 (UNDER DEVELOPMENT)

Most people are familiar with ext3 which is the default Linux filesystem for almost all distributions.

In 2007 development started on ext4, which will fix the limitations of ext3. One of the new features that is being developed with this release is support for clustered filesystems.

However, it is likely that ext4 will not be ready for production use for a number of years and is only mentioned here for the reader's interest.

8 Cluster Examples and Challenges

There are numerous ways you can configure a cluster, which will depend on your requirements and budgets. There are also some complex requirements if you wish to have virtualization with the cluster as well as building geographically separated clusters.

This document details a number of examples of cluster designs that may be suitable for you and to just give you an understanding of what is possible as well as discussions of the problems and limitations with each design.

8.1 Two node HA cluster with DRBD

A common high-availability requirement is to make a particular server survive any hardware failure. The solution to this, is to add a second identical server and setup a HA cluster between them.

To make a two-node high-availability cluster work, we have the following set of requirements:

- Data must be accessible by both servers, with both servers being able to read/write at the same time.
- If an individual service die, it should resume on the secondary server.
- If one server suffers a complete failure, the secondary server should resume all tasks.

Solution:

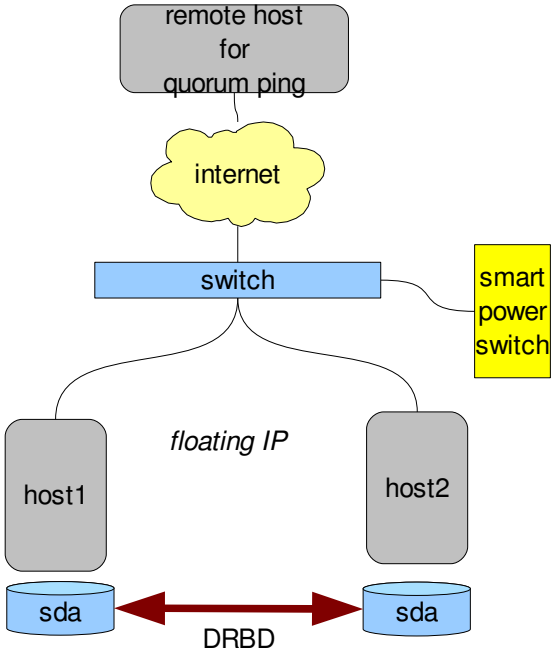
- Both servers identical hardware, running Redhat Cluster Suite.
- Local root and swap filesystems, remaining disk space turned into DRBD block device setup with both nodes running as primary.
- DRBD device running GFS with journal space for 2 nodes and both nodes mounted at the same time.
- Each service configured in cluster suite with a floating IP address. Any service that fails will relocate to the second node.
- In the event of a full server failure, the second node will resume all services.

Notes:

- DRBD is used primary/primary with GFS so that both servers can be running services at once. This may be undesirable or unwanted if you only have one floating IP address, in this scenario you need to group all the services using that IP together.

  In that case, it would also be okay to run DRBD as primary/secondary with a traditional filesystem like ext3 or xfs and have all the services configured as a single resource, to all failover together.

In the event of any node failing, Redhat Cluster Suite will move all the services to the alternate node and switch DRBD from secondary to primary on the new master node.



- A network failure runs the risk of a split-brain situation – if neither node can see the other, they will both try and become master. To fix this, run ping to a remote server to provide a tie-breaking third vote (Quorum heuristics).

- If you have a SAN or NAS, instead of storing the data locally and replicating with DRBD, the attached storage device could be used instead.

  The SAN could be setup with a non-clustered or a clustered filesystem, the difference being that a clustered filesystem is required if you require both servers to be able to run services at once

  If using a SAN, the cluster is scalable to more than two nodes, but the SAN could become a single point of failure for the cluster and are a lot more expensive than software solutions like DRBD.

Suitable Environments:

- Making any mission critical server HA.
- Any business or organisation that can not tolerate hardware downtime of their production sever.
- Ecommerce servers that need to provide mail/websites/databases.
- Small hosting organisations. (larger ones should use designs like the one below)

8.2 Five node HA cluster with DRBD

One of the problems with DRBD is that it only works for two-node clusters. It is possible to add a third node if the commercial version is purchased, but no DRBD solution exists which can work more than three nodes at max.

When building clusters, it is more economical to have a single multi-node cluster rather than many two-node clusters, as only one computer needs to be set aside for spare resources.

Some installations use SANs which limits the cluster size by the number of interfaces on the SAN. However, SANs are very expensive and require special hardware.

A cheaper solution, is to build two computers with plenty of storage in them using off-the-shelf parts and then to use DRBD to create what is effectively a HA NAS. These two storage nodes mirror each other and can transparently tolerate either one of the two nodes failing.

These two storage nodes can then export the available storage using a network filesystem like NFS or a block-level service like GNBD, which the rest of the nodes can use. Depending on your applications there may be no need to have local disks in any of the servers and they can all run directly off the network.

Here is an example for a five node cluster using DRBD for storage, providing a range of services such as HTTP and MAIL.

Solution:

- Two nodes with RAID 5 hard drive storage in each node - “storage nodes”.
- Three nodes with no disks - “production nodes”
- Storage node are setup in Primary/Secondary mode,with LVM and ext3 ontop of the DRBD layer with NFS exports of the data.
- Storage nodes provide user authentication via NIS/LDAP/Kerberos.
- Storage nodes provide pxelinux and DHCP for network booting.
- Quroum votes are setup in such a way that failure of both storage nodes will cause a cluster failure resulting in all services stopping.
- Production nodes boot off the active storage node using netboot and mount the root filesystem using NFS. All the production nodes run the same software build.
- Services are spread across the three production nodes – if any node fails, the services are resumed on another one.

Notes:

- The above design can also be used with a small two-interface SAN. The SAN can be connected to both storage nodes instead of using local disks and the data then exported via NFS.

- To increase redundancy, two SANs could be used, with one connected to each storage server and mirroring done either between the SANs themselves or using DRBD on the storage nodes. However, standard hard drives will usually be cheaper and thus will probably be a better solution.

Suitable Environments:

- Ideal for hosting providers – in particular, shared webhosting and email.
- Ideal for large companies to increase server availability and to centralise storage.

8.3 Five node HA cluster with DRBD + Xen

The five-node DRBD cluster design above can be extended to become a HA Xen cluster. The three production nodes can be configured as Xen hosts, with the Xen guests being booted from the network and using NFS for storage (just like the hosts themselves).

Cluster Suite is running on the host node, and in the event of a Xen VM failing or the entire host itself suffering a failure, Cluster Suite can start each Xen VM on an alternate host, with each Xen VM being a cluster node and running a single or multiple number of services.

The Xen hosts can be configured to be part of the same domain, which also allows live migration of Xen VMs – so if one host server is being heavily loaded, some VMs can be moved to another host whilst they are still running with no downtime at all.

Notes:

- Optionally, instead of having the Xen VMs belong to the same cluster as the hosts, the Xen VMs could be setup in their own cluster with each Xen host running their own cluster, leaving the host cluster to only deal with the Xen VM as a whole. See the Xen section of this document for details on advantages/disadvantages with this method.

Suitable Environments:

- Ideal for hosting providers
- Ideal for large companies to increase server utilisation and availability.
- Ideal for IT companies that need large number of servers for various application and development needs.

8.4 Geographically distributed clusters

All the cluster designs show have been for use in one physical location – the cluster nodes are all sitting in a rack, connected via ethernet, and is able to support floating IP addresses because there is only one route into the cluster.

However, a common desire is to have geographically distributed clusters to prevent failure of a single site taking the whole cluster offline.

Typical uses for this might be:

- A company with offices in two cities would like to have one server at each office with the data mirrored between them.
- An ecommerce website wanting replicated email, website and database services between two sites to ensure availability.

However, there are some big issues with a geographically separated cluster that need to be

solved:

- Internet connections are slow – data needs to be mirrored at both sites in a way that is bandwidth friendly and transferred data when changes are made needs to be minimal.
- Internet connections fail/have outages fairly frequently. Any solution must be able to handle this without split brain issues.
- DRBD is an ideal candidate for distributed clusters, but is only able to scale to two nodes (or three using the commercial version). This causes a complication for organisations with more than three offices that they want mirrored servers in.
- You can't float an IP addresses around a country unless you're an ISP. But even they can't float an IP address to a server on the other side of the world.
- Fencing is more complicated – a secondary (independent) management network is useful in order to be able to communicate to fencing devices in other locations.

  Without a secondary connection, a crashed server can not be fenced if the production network goes down, although a running server will stop the clustering if it loses quorum so split-brain is not an issue.

How can we solve this? There are a number of solutions:

- Use a distributed, locally caching filesystem like AFS, which will cache commonly accessed data.
- Have two main offices running DRBD and all other offices must connect using a network file system, perhaps assisted by a proxy/caching device for improving performance at the sites.

  Devices such as WAN accelerators can be used to make optimal use of network performance and some models have internal hard drives that can cache some data like HTTP or SMB traffic.
- Float DNS names rather than IP addresses. When a node goes down which is providing public services (eg: websites) have the new node providing the service connect to the DNS server and change the A records.

  The problem with this method, is the DNS changes can take some time to synchronise across the web – this can be reduced by setting your DNS Time To Live (TTL) to a low value – eg: 5mins – but it may not be honoured by all DNS caching servers. (although as a general rule it is)
- In addition, it is possible to setup Round Robin DNS servers, which can allow you to

load balance between your geographically separated servers – this is good for services such as HTTP or read-only database requests.

8.4.1 TWO NODE DISTRIBUTED CLUSTER

As discussed above, a two-node distributed cluster is able to use DRBD data replication which solves the major challenge of mirroring the data.

This two-node setup is an ideal solution for organisations which want the redundancy of an off-site mirror suitable for providing both redundancy and load-balancing. It is also suitable for organisations which have two offices and wants identical servers at both locations.

This example assumes the following environment:

- Two geographically separated servers, identical hardware.
- Network between both servers controlled by an outside party (therefore floating IP addresses are not possible)
- No plans to scale beyond two nodes.

Requirements:

- Have a server at both locations.
- Both servers need to offer file sharing to the local networks at each office with Samba.
- Mail, DB and HTTP services required to be HA.

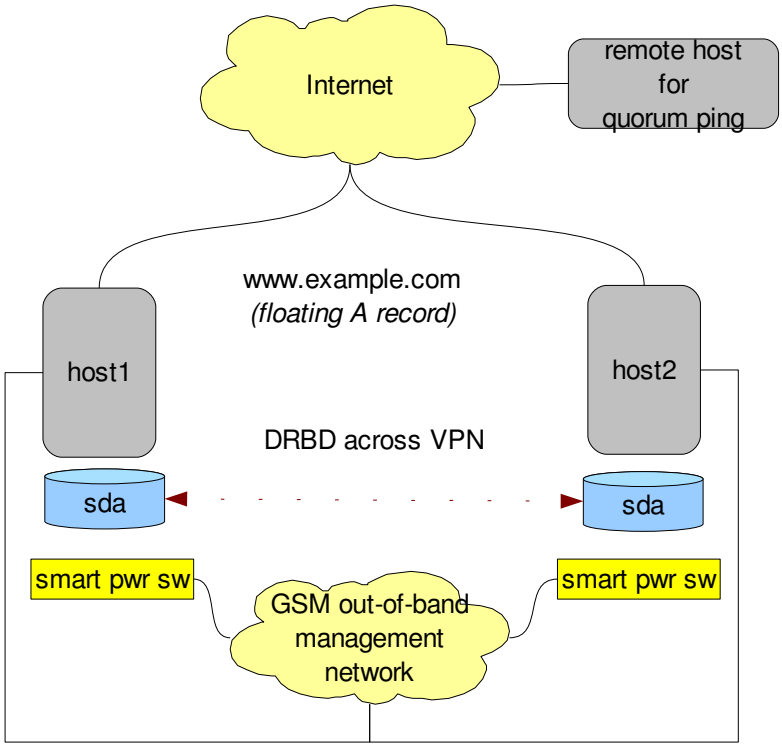
Solution:

- Servers setup with DRBD in primary/primary setup (both servers can read/write).
- GFS filesystem to allow both nodes simultaneous access to the data.
- Use ping to a remote server to determine tie-breaker third vote. (Quorum heuristics)
- Floating DNS names for services.
- Cluster suite capable of migrating individual services.

  MAIL, DB and HTTP services will only run on one server at any time (otherwise interesting lock file issues would occur).

  In the event of either server suffering a failure, the services will be relocated to the alternate server and the DNS records will be adjusted to redirect traffic.

  Samba runs on both servers providing access for the local network. In the event of a server failure, the DNS name for the local network server will be changed to be the alternate server.



8.4.2 THREE+ NODE DISTRIBUTED CLUSTER

The two-node distributed cluster detailed above will work fine with two-nodes, but what about when three or more nodes are required?

We want data replication between the servers, but DRBD will only work with two-nodes in primary/primary modes. This gives us two options:

1. Run DRBD between the two main servers, and the other servers can connect to one of the two servers via a network filesystem like NFS.

   This is quite simple, but has the obvious flaw of disk I/O being limited to the speed of the WAN connection, as well as the transfers causing data cap usage on non-flatrate connections.

   Depending on your usage and requirements, this may or may not be a problem.

2. Run a distributed file system with replication support on all the nodes.

   Unfortunately there are currently no distributed filesystems available which can provide both replication and distributed data.

   Therefore, the next best solution is a using a distributed filesystem, with DRBD underneath it to provide the redundancy.

This is covered in more details in the DRBD filesystem section earlier in this document, but basically you divide the cluster nodes into pairs.

Each pair runs DRBD primary/primary to ensure data replication. On top of that, you run a distributed filesystem such as Lustre of AFS. However, this can be quite complex and has the weakness of being limited by having replication only on 2 nodes.

Three+ distributed clusters can be quite complex and require a lot of planning to make sure they will work reliably and have speedy access to storage. The best solution will depend greatly upon the applications you need to run.

9 Further Reference

The following resources are good further reading for information on setting up cluster solutions:

Cluster Management:

Redhat clustering guides (includes info on GFS)
http://www.redhat.com/docs/manuals/csgfs/index-master.html

Linux-HA documentation
http://www.linux-ha.org/

Storage :

DRBD (block-layer replication)
http://www.drbd.org/documentation.html

AFS (distributed filesystem)
http://en.wikipedia.org/wiki/Andrew_file_system

Lustre (distributed filesystem)
http://www.lustre.org

Training/Courses:

Additionally, if you are an RHCE, Redhat's RH436 training is a good course that teaches you how to configure clusters and shared storage on RHEL with Redhat Cluster Suite.
https://www.redhat.com/courses/rh436_red_hat_enterprise_clustering_and_storage_management/