Implementing Clusters for High Availability

A survey of the state of the art in the Linux space

James Bottomley
SteelEye Technology

2 July 2004

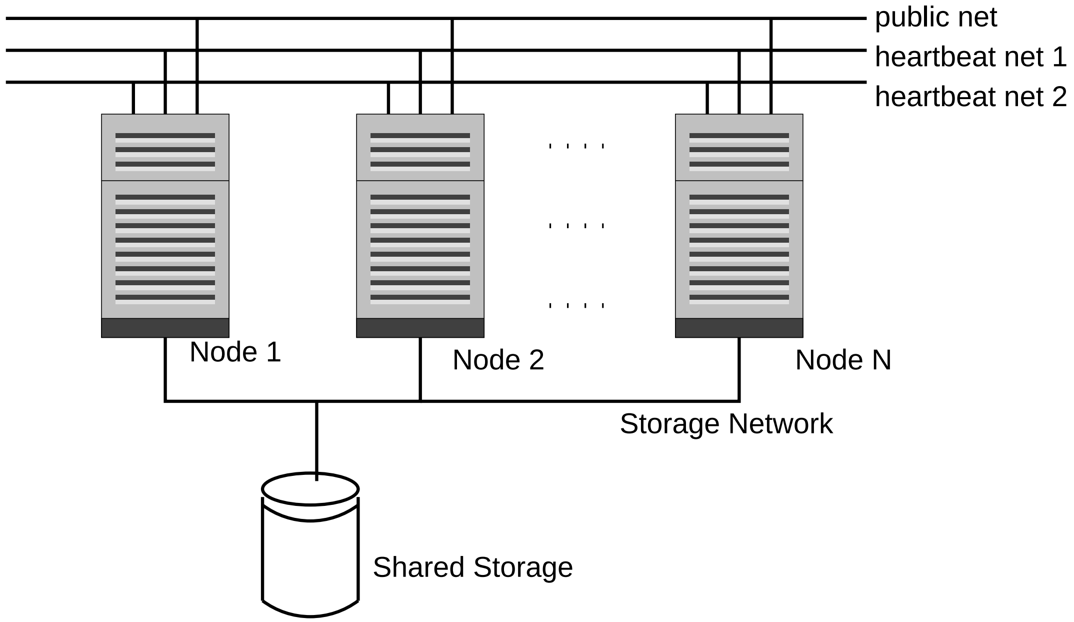
What Is Availability?

- Availability measures the ability of a given service to operate.
  - Defined as the fraction of time for which the service you are exporting is available for use.
  - So a service with 99.99% Availability must be down for no more than 52 minutes per year.
  - 99.999% is no more than 5 minutes and 15 seconds per year.
- Usually, in any system, availability decreases as the complexity increases.
- Any system which takes action to increase availability beyond what would ordinarily be possible may be termed Highly Available.

Class of Nines

- When the availability is shown as a percentage (or just as a decimal), the number of initial nines is called the availability class of the service
- Thus, $A = 0.9999$ or 99.99% is an availability class of four (or four nines)
- $A = 0.99999$ or 99.999% is an availability class of five (or five nines).
- and so on.

Paradigm for a HA cluster



- Consists of multiple local machines, LAN networked with some type of shared storage (either parallel SCSI, Fibre Channel or NAS).

Types of HA Clusters

- The rule of building HA clustering software is that anything beyond two nodes is hard.
- Thus, The world is essentially divided into three types of cluster
  - Two Node Only: Simplest type of cluster. Method of construction does *not* allow scaling beyond two nodes. Examples are Mission Critical, Heart Beat and old Red Hat Cluster Manager.
  - Quorate
  - Resource Driven

Quorate Clusters

- Centrally controlled: Cluster must form first before any action is taken
- Cluster membership *must* be well defined: This makes Membership services essential.
- Cluster must be defined in such a way that *no* other cluster may form from excluded nodes.
- Gets it's name from "Quorum" which means voting sufficiency.
- Quorum is often implemented via ownership of a single (or set of multiple) resources by the "controlling node" in the cluster.
- All actions governed by master entity.

Resource Driven Clusters

- Resources are grouped into independent sets called “hierarchies”.
- Each hierarchy must be separately “ownable”.
- Established ownership of the hierarchy is all a node needs to proceed with recovery.
- Full communications paths between every node in the cluster *not* required
- No centralised concepts of membership or quorum.
- Multiple independent sub clusters may form.

Comparisons

- Cardinal Rule is KISS, or in clustering terms "Complexity is the Enemy of Availability".
- Simplest cluster is 2 node only, followed by Resource Driven and then (quite a way behind) Quorate.
- Recovery in Resource Driven Clusters is much faster than in Quorate (nodes only need to obtain ownership to begin recovery; in Quorate, cluster must form, followed by membership, followed by directed recovery).
- The difference really shows up in the approaches to I/O fencing (see later).

Determining Availability

- This is actually one of the really hard things to do.
- Uptime U is defined as the average time to a failure
- Downtime D is defined as the time between experiencing the failure and getting the system working again.
- Obviously, the Availability A becomes

$$A = \frac{U}{U + D}$$

- But in order for this to be meaningful, you need to know what U and D are in your environment

Clustering and High Availability

- The simplest way to improve the Availability of a system is to have a duplicate waiting to take over if anything goes wrong.
- This duplication describes the simplest form of Active/Passive cluster.
- Here, the Down time of the Service is the Time it takes the Passive node to detect the failure plus the time it takes to recover the service.
- This is often termed the “Availability Equation” (but more accurately, it is the downtime equation)

$$D = T_{\text{detect}} + T_{\text{recover}}$$

Clustering and High Availability (2)

- So, if you have a Service Level requirement, what a cluster really does for you is quantify exactly the Downtime D.
- Thus, it eliminates a huge quantity of uncertainty from your enterprise.
- However, note that implementing cluster still **doesn't** give you any handle at all on your Uptime U.
- Therefore, you still cannot predict your Availability, even with a cluster, unless you know your Uptime.
  - All you've done is controlled your Downtime.

Clustering and High Availability (3)

- The reason clustering implementation is so important is precisely because the cluster cannot control Uptime.
- The only way to control uptime is by careful implementation and deployment of the cluster. This is why things like:
  - Hardware burn in,
  - Redundancy in communications and storage,
  - Multiple redundant power supplies,
  - All the traditional uptime lengtheners

  are still important in cluster deployment.

Users and Failure Tolerance

- Sometimes, Availability is a misleading measure, and Downtime is the true quantity users care about.
  - It's all about perception.
- In the web server example: a regular user who complains to the admin once a week to get the service restarted regards the service level as unacceptable.
- However, if the cluster can restart it in ten seconds, he only has time to notice the failure and click again to get the service restored.
- A similar service glitch could be caused by the Internet or DNS resolution or a host of other problems between the user and the service, so the user will tolerate this level of downtime.

Application Failures

- More often than not, it's the application that fails rather than the server.
- Failures fall into two categories:
  - **Non-Deterministic** internal application error like heap or stack overflow (or memory leak). Simply fixed by restarting application. Could also be caused by data corruption if device access not fenced properly.
  - **Deterministic** failure in *direct* response to data input. Restart application and redo input causes crash again.
- Local recovery is important for fast application restart on non-deterministic failure.

Monitoring

- Every cluster (without exception) provides the ability to monitor health at a node level.
  - so node failures may be spotted and corrected.
- some clusters also provide the ability to monitor individual applications and even restart them locally if they have failed.
  - this is essential, because applications can fail more often than the node (e.g. the web server crashes every week example)
  - Local recovery is important (because it can decrease downtime and minimise disruption).

Reducing Down Time

- As we learnt previously: a cluster helps you minimise Downtime. It cannot help you with uptime.
- However, uptime is extremely important to availability.
- Thus, as well as implementing clustering to improve Downtime, you should assess your cluster hardware for ways to improve uptime.
- Key to this is eliminating Single Points of Failure (SPOF).
  - Cluster wide SPOFs must be eliminated entirely
  - Individual Node SPOFs should be assessed to see if eliminating them would improve uptime.

Cluster Single Points of Failure

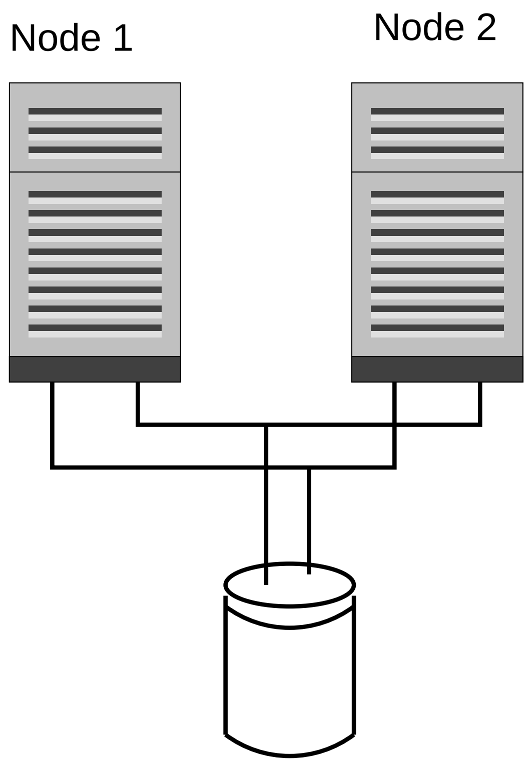
- In a shared storage cluster, the real SPOF is the storage.
- Make sure that the external array is configured as a RAID
- Not only that, but make sure it's RAID 1 (mirroring)
  - RAID 5 is cheaper, but in a double fault situation it may end taking the array offline **and corrupting your data**.
  - RAID 1 preserves data integrity (but still takes the array offline) in the double fault case.
- replication provides a cheap method of eliminating the storage SPOF (separate copies of the data in each node).

Node Single Points Of Failure

- In the shared storage cluster, the first and most obvious Node SPOF is the connection to storage.
- The next most common is the power supply (the non-silicon components often burn out or fail).
- Almost equally common is the failure of mechanical devices like Fans
  - Particularly nasty in todays world of hotter, faster and actively cooled CPUs
  - For example, a top of the line P4 will overheat and burn out in less than a second if its heat-sink fan fails.

Multi Path

- In the standard shared storage cluster, if a link to storage fails, the application loses contact with the data and the cluster must fail to another node that can still access it.



- This happens surprisingly often (cables get trodden on, dust gets into transceivers etc.)
- Can obviously eliminate this by having more than one connection to the storage per node (called Multi-path).

The Costs

- Replication is essentially free.
- External Storage arrays cost about \$3,000+ (FC arrays begin at about \$5,000)
- Multi Path, starts at about \$10,000 and the sky is the limit for truly Rolls-Royce solutions.
- Redundant Power Supplies and Redundant Fans only found in higher end servers (not as add on items to low cost servers), will drive server costs up by \$3-5,000.

Managing your Systems

- Systems management is integral to SPOF elimination
- It is no use at all to buy a fully redundant system and then keep it in a cupboard and never monitor it.
  - redundancy will protect you when the first failure occurs.
  - the second failure will take down your server (or cluster if it's in the shared array).
  - maintaining and replacing failed redundant components is essential to preserving uptime.
- if you have no way of monitoring your server's redundant components, you may just as well opt for cheaper hardware and allow the cluster to manage the Downtime instead.

Linux Specific Problems

- Originally, in the 2.2 and early 2.4 days this was as basic as shared storage didn't quite work.
- Now, primary problem for Cluster manufacturers with binary modules is simply keeping up with kernels as they turn.
- Biggest unsolved problem is the dreaded oops:
  - If the kernel of a machine fails, you'd like it to fail hard so that the heartbeat notices and takes over the services.
  - Oopses simply kill processes because of kernel errors and then try to continue.
  - if the kernel was in a critical section at oops time, system may become hung and unusable.

The 2.6 Kernel and HA

- General features that help clusters (in the enterprise)
  - Large Block Device (LBD) support; 2.4 is limited to 2TB.
  - Large File and Filesystem support which takes advantage of LBD.
- Multi-path:
  - Every vendor has a separate multi-path solution for 2.4
  - Trying to unify the architecture on Device Mapper for 2.6
  - Still little buy in from Hardware Manufacturers.

I/O Fencing

- Basic problem is "split brain". All communications fail and all nodes in cluster try to take over service.
- For Quorate clusters answer is often a Stonith Device
  - once cluster is formed, kill power to all non-members
  - problems: stonith doesn't protect against accidental access; path to stonith often a SPOF.
- Resource Driven Clusters often use SCSI Reservations:
  - Storage owned exclusively by one node, ideal for Resource Driven ownership model.
  - Even accidental access is prevented.
  - Reservations fairly universal, but storage usually has to be "qualified" to make sure they work properly.

Customising your Cluster Environment

- Cluster vendors try to provide off the shelf recovery tools for typical applications
  - web servers, databases, file exports (NFS or SMB/CIFS) etc.
- However, in a complex environment you often have custom applications that the cluster vendor won't support out of the box.
- In this case you need to know what options are available to you to support your application
  - does the cluster provide an easy way to protect and monitor arbitrary applications?
  - Does this come as an extra, or is it available with the base product.

Open Source Linux Cluster Products

- **Failsafe** (ex SGI).
  - Multi node (to 32) quorate cluster
  - includes full monitoring and local recovery.
  - Uses Stonith for node fencing.
  - No support for replication or Host Based Raid.
- **HeartBeat**.
  - Currently Two Node Only (expansion planned).
  - Uses other available components for active monitoring and local recovery.
  - Supports replication using drbd and Host Based Raid.
  - Uses stonith for node fencing.

Open Source Linux Cluster Products

- **Red Hat Cluster Manager**. Based on Mission Critical cluster product.
  - Up to six node, Quorate.
  - Limited active monitoring and no local recovery.
  - Uses stonith for Node fencing.
  - Requires kernel extensions for NFS (in Red Hat kernel).
  - No support for Replication or Host Based Raid.

Closed Source Linux Cluster Product

- **Veritas Cluster Server**
  - Multi Node (to 32) Quorate cluster.
  - includes full monitoring and local recovery
  - no support for replication (in Linux).
  - no support for Host Based Raid.
  - Uses SCSI Reservations (or Veritas Volume Manager) for I/O fencing.
  - Requires three proprietary (closed source) kernel module for operation (for reservations, heartbeat and cluster communications).

Closed Source Linux Cluster Products

- **SteelEye LifeKeeper**
  - Multi Node (to 32) Resource Driven Cluster.
  - includes full monitoring and local recovery.
  - integrated support for replication using md/nbd.
  - support for most common Host Based Raid systems.
  - Uses SCSI reservations for I/O fencing.
  - Uses open sourced kernel additions for NFS on 2.4; will require no kernel changes for NFS on 2.6
  - Also has support for Stonith devices.

Conclusions

- You need to know your requirements based on the type of services you are trying to protect:
  - Control Down Time
  - Lengthen Up time
  - Or both.
- Monitoring is vital. First failure usually eliminates redundancy, second one will take down your service.
- Knowing the right questions to ask when choosing HA often more important that the choice itself.
  - It gives you (the implementor) a better understanding of the limitations and trade-offs within your system