N-Way Web-Cluster for scalable web-platforms

ATIX GmbH

Marc Grimme

To appear at the LinuxTag 2003 as Paper No. 082
11. June 2003

Table of Contents

Introduction

In modern webserver environments it is getting more and more important to scale the application so that all web-servers are optimally utilized. Being able to do this one can build infrastructures that have an optimal price-performance ratio. This paper describes possibilities to build up such infrastructures and gives the technical background to understand them. The emphasis will be on the layer that seperates the computing power from the storage layer so that one can scale storage independently from the computing power. For such infrastructures *Cluster Filesystems* and SANs - *Storage Area Networks* - are the perfect combination.

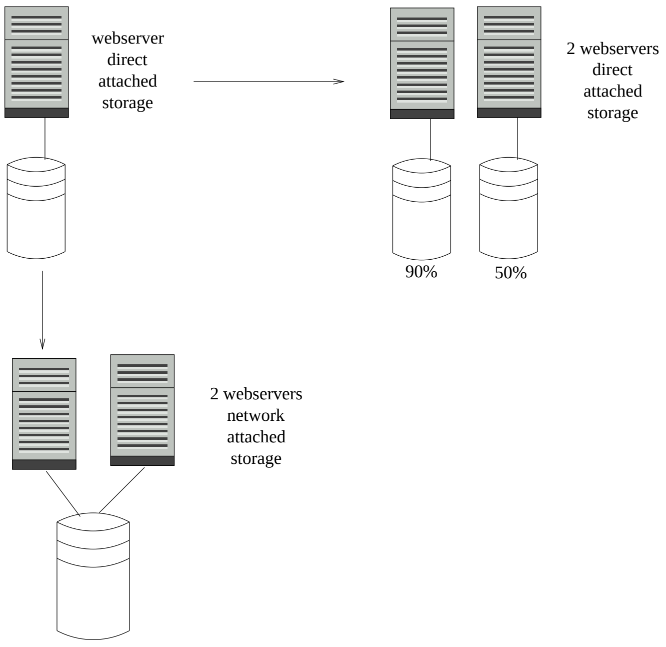
After describing the technologies used to build up the infrastructure, two different customer projects are depicted. One is a infrastructure for a classic *ISP - Internet Service Provider* and the other a *3-tier diskless web-application cluster* that is running two dutch insurance web portals.

The traditional approach: scalable webplattforms without *Cluster Filesystems*

Nearly everybody who is utilizing web-applications knows the problem of scaling the infrastructure optimal for current needs. Besides it is pretty easy to scale up a service like i.e. a web-service or more general a *stateless UDP-based Service*. You can just take a so called *Director*[1] to achieve this. All web-servers that are logically grouped together for the same purpose need to share the same data. If one server suits the needs a local attached harddisk does the job. But if more than one server is needed for whatever reasons - availability, performance or both - both servers have to access the same data concurrently. If you address this problem in the traditional manner you basically have two ways to achieve scalability (see Figure 1).

Figure 1 shows the two different approaches to scale up your web-server infrastructure. The middle shows one server with local attached harddisks. From this base you can traditionally do scaling in the two different ways.

Figure 1. Traditional approach for scaling webservers



NAS servers to share data

The most common way to allow different servers to concurrently access the same data is to use so called *Network Filesystems* like NFS or CIFS/SMB. *Network Filesystems* allow multiple clients concurrent access to the data over highlevel network protocols most often based on the local networks. All these protocols sit on top of the UPD/TCP layer and add some more layers of protocols to give the clients a transparent view of the data.

Such *Network Filesystems* are widely used with a lot of applications and have all advantages of a longterm proven often used software. But they also carry all the drawbacks of TCP/IP networking and put on top even some more protocol overhead. In particular if you thinking of accessing data and files with the lowest possible latency *Network Filesystems* are not always first choice.

Asyncronous replication to share data

In theory the best way to go the lowest latency is to give any server direct attached disks (in short DAS.

But then you have the problem of sharing the data. If one needs the latency of DAS and i.e. has data that does not change very frequently - statical web-content - often tools like rsync (see [SAMBA]) or unison (see [UNISON]) are used to synchronize the data on very few servers every now and again.

Both concepts have different drawbacks but are quite useless if you think of large number scalability. The best solution would be very low latency networks for storage accessibility and a thin efficient protocol to concurrently share data. That is one stage where *Cluster Filesystems* can help out.

Using *Cluster Filesystems* for scalable web-platforms

Cluster - or *SAN Filesystems* - short CFS - are filesystems that the kernel sees as locally attached filesystem. All the data of the files is exchanged over a special Storage Area Network (short SAN). Such SANs are typically build on top of Fibre Channel Networks (short FCN).

Fibre Channel Networks

Such FCNs are high speed networks with a much lower latency than any available Ethernet. The nodes attached to the FCN are communicating with FCembedded SCSI commands. But the infrastructure is just build like typical networks. You have active components like switches or hubs and you can connect the nodes to the FCN mixed with copper or fiber cables. Right now data can be transfered at two speeds 100Megabyte and 200Megabyte per second over the same network.

Logically FC is a bidirectional Point-To-Point serial data channel for high performance throughput. That means only one initiator can establish a channel with one discrete target. At this moment no other initiator can connect to this target. With a switched topology other initiators can connect to other targets at the same time and many channels can be established in parallel. We talk of a network when a protocol allows multiple connections at the same time.

The *Sistina* Global-Filesystem-GFS

GFS a *Cluster Filesystem*. It gives the ability to share data over SCSI-based infrastructures. That can either be the traditional *parallel* SCSI Topology or SCSI based FCNs or any other SCSI talking infrastructure.

GFS was initially developed at the university of Minnesota by Matthew O’Keefe since 1997. Today the working group turned into a company called *Sistina*.The focus of the company is on storage-cluster with Linux. Besides GFS also LVM is developed by *Sistina* to give future storage-cluster the advantages of a better *volume management*.

GFS is also based on a *volume management* software called the *Network Storage Pool* that allows to pool together multiple storage devices accessible by all nodes in the *storage-cluster* and to distribute all data over these. That *storage-pool* builds the basis for the filesystem.

The *storage-cluster* itself is build as a symmetric cluster in which every member has the same rights. That means that locking, *i/o recovery* and data access can be done by any cluster-member. All nodes are connected via a *storage network* as described above and an Ethernet-based network that is used for locking and the so called *fencing*[2] mechanism. The locking and *fencing* actions themselves are delegated to a special purpose node that is not directly a cluster-member. This *lock-server* has to run both, an IP-based locking- and a *fencing* service.

With a software like this a so called *Storage Single System Image* can be build so that every cluster-member has the same view and concurrent access on data. From this point of view there is no difference between a Network Filesystem like NFS and a Cluster Filesystem like GFS. But the difference is that GFS works with much faster networks and is much thinner if you look at protocol overhead and so forth. GFS can be seen as local filesystem that is shareable over SCSI protocols.

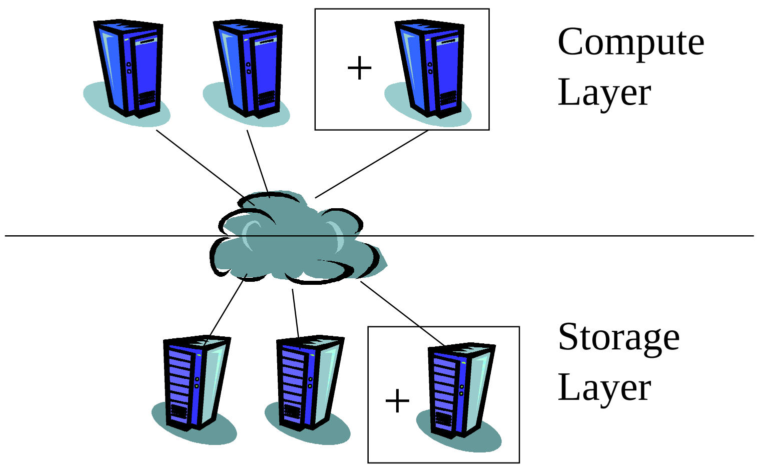
The only potential drawback is the locking and *fencing* instance that makes a potential *single point of failure* - short SPoF. To solve this the actual lock-server with version 5.1.1 can cluster itself over two nodes as a highly available service. The next version 5.2 (release date beginning of June 2003) can have the locking-service running on any node and failover to any other node if problems occur.

Cluster Filesystems for scalable web-platforms

Such *Cluster Filesystem* are completely transparent for the application. That means that any application like *webservers* or *webapplication-servers* or whatever application that is needed for the webservice makes transparent use of the SSI the *Cluster Filesystem* provides. As most webservices produce much i/o fast and low latency filesystems are the optimal solution. Besides most providers use loads of small webservers to provide their service. The optimal solution would be to use a common storage SSI for any node. That brings advantages in management, serviceability, availability and quality of service, which are key requirements for any service provider.

Figure 2 shows that you have two approaches depending on the way of scaling. If you need more computing performance you simply add more members to the *compute layer* if you need more storage you just add some more storage to the *storage layer*. That can be seen as two dimensional scalability (*2D Scalability*).

Figure 2. 2D Scalability



Advanced features with *Shared Root* and *diskless booting*

With CFS one could also assume that all storage cluster members don’t need nonsharable disks any more. One could achieve fully seperated and independently scalable infrastructures in terms of computing performance and storage - a 2 dimensional scalability (see Figure 2). But for such diskless nodes the storage SSI has to provide the ability to make special parts of the SSI hostdependent. Examples for such parts are different files and directories like i.e. `/etc/mtab` or `/var/lock` and so forth. A good overview on what has to be shareable and what has to be unsharable gives the *Filesystem Hierarchie Standard* (see [FHS]) down to the directories themselves.

Commonly used technics for hostdependencies on storage SSIs are the so called *context dependent symbolic links* - short CDSL. These are symbolic links that are evaluated at runtime by any storagenode itself. They can be dependent and evaluated based one the hostname, archicture and operating system. That means any storagenode for example accesses special data that is concurrently unique for all nodes with the same operating system, architecture or hostname - that should be only one. If for example `/etc/mtab` is only a symbolic link to `@hostname/etc/mtab` any clusternode sees it own `/etc/mtab` from `/<hostname>/etc/mtab`. With that feature and *hostbus adapters* that allow booting from the SAN all storagenodes can be equiped without any locally attached disk drives.

With a *shared root* all clusternodes are exchangeable and new ones can be easly integrated and an optimal scalability can be achieved.

Examples from the real world

“IP-Tech” one of the Top 5 ISPs in Switzerland

“IP-Tech” is one of the Top 5 *internet serive providers* in Switzerland. They can providing full services to their customer including *web*, *mail*, *VPN* and many more.

The problem at this company was that they basically had a special amount of nodes that used a NAS/NFS server for their storage single system image. In times of heavy usage the services could not make the i/os for all incoming request in time. They needed a more efficent i/o system that was able to handle more requests in real time.

Figure 3. New configuration of IP-Tech

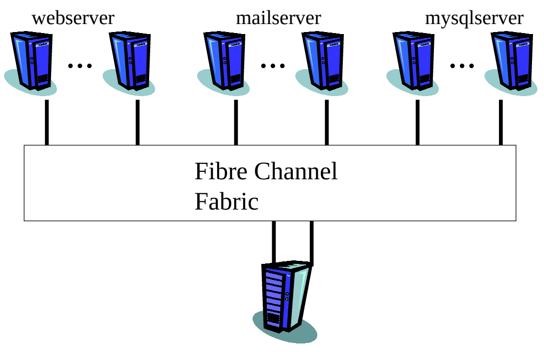


Figure 3 depicts the new configuration based on GFS. Right now 13 *storage-members* are connected to the filesystem. The servers are grouped in three logical groups: one group for web-serving (apache) one for mail-service (Sendmail, IMAP and POP3) and one for DBMS (Mysql).

“IP-Tech” sums up the following experiences since their change to GFS. The cluster system is running more stable because they could get rid of problems coming from the implementation and the concept of NFS. With the use of a lower latency storage network and a thinner and a conceptual *local* filesystem their servers are running on 15-20 percent load compared to NFS.

“IP-Tech” is hosting currently 30000 web-domains, 10000 mail-domains and 1000 databases.

Table 1. Requests per day for “IP-Tech”

HTTP	1.5 mio
SMTP	150 k
POP	250 k
IMAP	20 k
Webmail	1 k
FTP	6 k
DB-Connections	100 k

Dutch bank with 3 tier diskless *web-application* cluster

The dutch bank wanted to use evalute modern infrastructures for actual and future applications. They had to build a 3 tier web-application-service based on Oracle as DBMS, Websphere as application-service and Apache as web-server or frontend. For this application a CFS in combination with diskless booting was the preferred and most forward-looking solution.

Figure 4. 3 tier *web-application* cluster (Hardware-configuration)

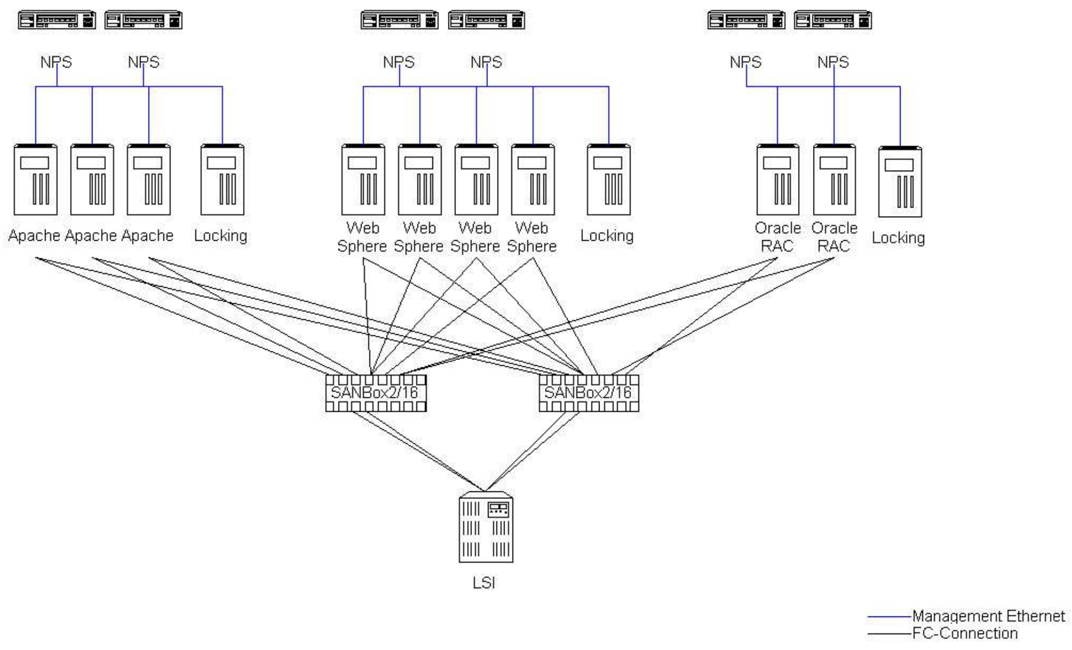
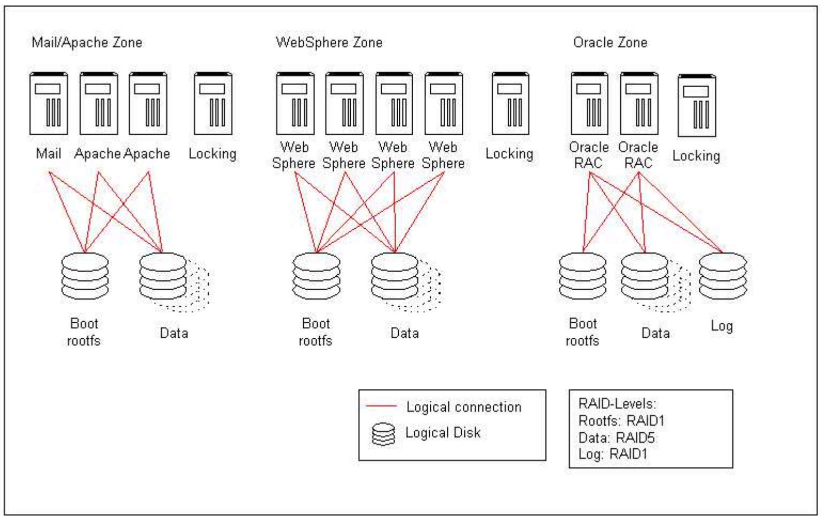


Figure 5. 3 tier *webapplication* cluster (Logicalview)



The *web-application* infrastructure as depicted in Figure 4 and Figure 5 consists of three fully independent GFS filesystems so that any two members of a different filesystem cannot access files of each other. Every cluster is managed by one locking and fencing server.

All three clusters are grouped by their service. These are a two node Oracle 9i Real Application Cluster a four node IBM Websphere applicationserver and the 3 node Apache webserver and Sendmail mail-server. All three clusters are diskless booting and provide a maximum of flexibility in management and scalability.

On this *web-application* cluster are running two portals of insurance companies since November 2002 without downtime.

Conclusion

The usage of these scalable *N-Way Web Clusters* can satisfy the challenges of the world wide web. *N-Way Web Cluster* simplify the management of the services and the storage also performance, availability and redundancy can be brought in because it is part of the concept. Such infrastructures can be scaled independently for performance and storage, theoretically infinite and linear. For important applications redundancy and high availability can be added as needed.

With Sistina GFS and ATIX Comoonics Cluster solutions forward-looking *N-Way Web clusters* are possible today.

Important: If you have any questions on this whitepaper or technical details, feel free to contact me at grimme@atix.de (mailto:grimme@atix.de), ATIX GmbH.

Bibliography

Websites

[LVS] *The Linux-Virtual-Server Project (http://www.linuxvirtualserver.org/).*

[SAMBA] *SAMBA - opening windows to a wider world (http://de.samba.org/samba/samba.html).*

[UNISON] *Unison - Filesyncronizer (http://www.cis.upenn.edu/~bcpierce/unison/).*

Whitepapers

[O'KeefeWPSC02] Ph.D.. Matthew O'Keefe, 2002, *Edge Serving with Sistina's GFS-Based Storage Clusters*.

[Preslan99] *Proceedings of the Fourth Annual Linux Showcase and Conference*, October 2000, K. Preslan and et al., 1999, "Scalability and Failure Recovery in a Linux Clustered File System".

[Grimme00] *Storage Area Network - State of the Art*, November 2000, Marc Grimme.

[Grimme01] *The Com.oonics System*, http://www.atix.de/www/index_c.html, 2001, Marc Grimme.

Notes

1. A *Director* directs all incoming requests on predefined port and ipaddress combinations to one of the productive webservers that he knows of. If one of them fails it justs updates its tables without the one. There are also a lot of advanced configurations which are using more statefull services (like TCP-services or SSL) but the configuration of the *Director* is not the focus of this document and it is assumed that there is a Director that forwards the requests to the right servers (for more information see [LVS]).
2. *Fencing* means that if any cluster-member does not act as expected one of the other nodes recognizes this (via continuous heartbeat messages) and disconnects the faulty member from the cluster.