Creating Linux Web Farms
(Linux High Availability and Scalability)

Horms (Simon Horman)
horms@verge.net.au

November 2000

http://verge.net.au/linux/has/

The world is full of questions, they fill me with fear
What of our beginnings? Why are we here?
Is there a purpose? Is there a God?
But one question troubles me
Far more than this lot

What nationality is Les Murray?

TISM, "What Nationality is Les Murray?"

Acknowledgement

I would like to acknowledge my employer VA Linux Systems, without whose help this paper would not have been possible. In particular, my colleague John Goebel has helped me enormously. I would like to acknowledge the support of the the people involved in Linux High Availability and The Linux Virtual Server Project, in particular Alan Robertson, Wensong Zhang, Julian Anastasov and Joseph Mack who have provided excellent software and support. Finally, I would like to thank Felicity for always being there.

Licence

Version Information

Version	Date	Description
0.0.0	2nd February 2000	Initial release
0.0.1	6th February 2000	Minor Corrections
0.0.2	3rd May 2000	Minor Changes
0.0.3	10th May 2000	Updated to include Ultra Monkey
0.0.4	24th May 2000	Minor Changes
0.0.5	13th August 2000	Updated for release of Linux Fail Safe
0.0.6	7th October 2000	Minor Changes
0.0.7	14th November 2000	Minor Changes

Presented

Linux Open Source Expo and Conference
7th – 10th March 2000
Sydney Convention and Exhibition Centre
Sydney, New South Wales, Australia

Linux Expo China
2nd – 5th June 2000
Shanghai Exhibition Centre
Shanghai, China

Silicon Valley Linux Users' Group
5th July 2000
Building 9, Cisco Systems, Inc.
San Jose, California, United States of America

O'Reilly Open Source Software Convention
17th – 20th July 2000
Monterey Convention Center
Monterey, California, United States of America

Linux World Conrerence and Expo
14th – 17th August 2000
San Jose Exhibition Center
San Jose, California, United States of America

Linux Expo Canada
30th October – 1st November 2000
Metro Tronto Convention Centre
Toronto, Ontario, Canada

New York Linux Users' Group
15th November 2000
IBM Building
New York, New York, United States of America

Abstract

Since the release of fake much work has been done to realise a High Availability solution under Linux. In this time significant gains have also been made towards scaling services using Linux. This session will look at the state of play of Linux in these areas and the problems that need to be addressed.

This paper will focus heavily on high availability and scalability technologies used to build a web farm. Web farms provide an interesting application of these technologies and is an area where Linux is becoming more and more accepted.

Contents

1 Introduction

In May 1998 a paper entitled "Creating Redundant Linux Servers" was presented at Linux Expo[1]. In November the IP Address takeover package detailed in the paper was released as "fake". This was arguably the first High Availability software released for Linux. During the same year Alan Robertson started a Linux High Availability page, focusing on the Linux High Availability HOWTO[6].

In the short time that has passed since then a myriad of closed and open source high availability solutions have become available for Linux. Fake has largely been superseded by Heartbeat. "Alan Robertson's HA Page", now known as "Linux High Availability" is still a focus for much of the activity around high availability for Linux and can be found at `www.linux-ha.org`. High availability under Linux is no longer limited to IP Address Takeover as both intelligent DNS[1] and Layer 4 Switching solutions become available.

The focus of this paper is to examine some of the technologies that are currently available under Linux and how these can be used to create Web Farms. Web farms provide an interesting application of these technologies and is an area where Linux is becoming more and more accepted. Beyond this the paper will examine some of the challenges facing Linux high availability and in particular the directions that open source efforts are taking in order to address these problems.

2 What is High Availability and Scalability?

High Availability: There are many definitions for high availability. Some make a distinction between high availability and fault tolerance. For the purposes of this paper these terms will be used interchangeably and will be taken to refer to the ability to provide some level of service during a situation where one or more components of a system have failed. The failure may be unscheduled as in the instance of a server crash or scheduled as in the case of maintenance.

The key to achieving high availability is to eliminate single points of failure. If a web presence is hosted on a single Linux box running Apache, then this

[1]DNS: Domain Name System. Distributed database that is used to map hostnames to IP addresses and vice versa.

is a single point of failure. If a database is hosted on a lone PostgreSQL server, then this is a single point of failure. If a site's internet connectivity comes through a single core router then this is a single point of failure and if a site has only one link to the internet then this too is a single point of failure.

Elimination of single points of failure inevitably requires provisioning additional resources — more often than not hardware — that can be utilised when failure occurs. It is the role of high availability solutions to architect and manage these resources such that when a failure occurs users are still able to access the service. This may be a full service, this may be a degraded service, it may even be a service advising users to come back later, but it is still a service and is better than an "HTTP 404 server unreachable" error.

Scalability: In the context of this paper, scalability refers to the ability to grow services in a manner that is transparent to end users. Typically this involves growing services beyond a single chassis. There are a variety of methods for doing this and while it is very difficult to construct a generic, protocol-independent method for achieving this either DNS based or layer 4 switching based technologies often form the core of a solution.

The biggest problem with scaling across multiple machines is data replication. It is important that the data source is as reliable as possible. However, asynchronously replicating data between multiple machines is difficult at best.

3 Web Farms

When a service grows beyond the capabilities of a single machine, groups of machines are often employed to provide the service. In the case of HTTP[2] and HTTPS[3] servers, or web servers, this is often referred to a Web Farm. Web farms typically employ both high availability and scalability technologies in order to provide a highly available service spread across multiple machines, with a single point of contact for clients.

Web farms can take many forms, however, the three tiered approach is a

[2]HTTP: Hyper-Text Transfer Protocol. Protocol used to transfer web pages and related content such as images.

[3]HTTPS: Hyper-Text Transfer Protocol Secure. An encrypted version of HTTP

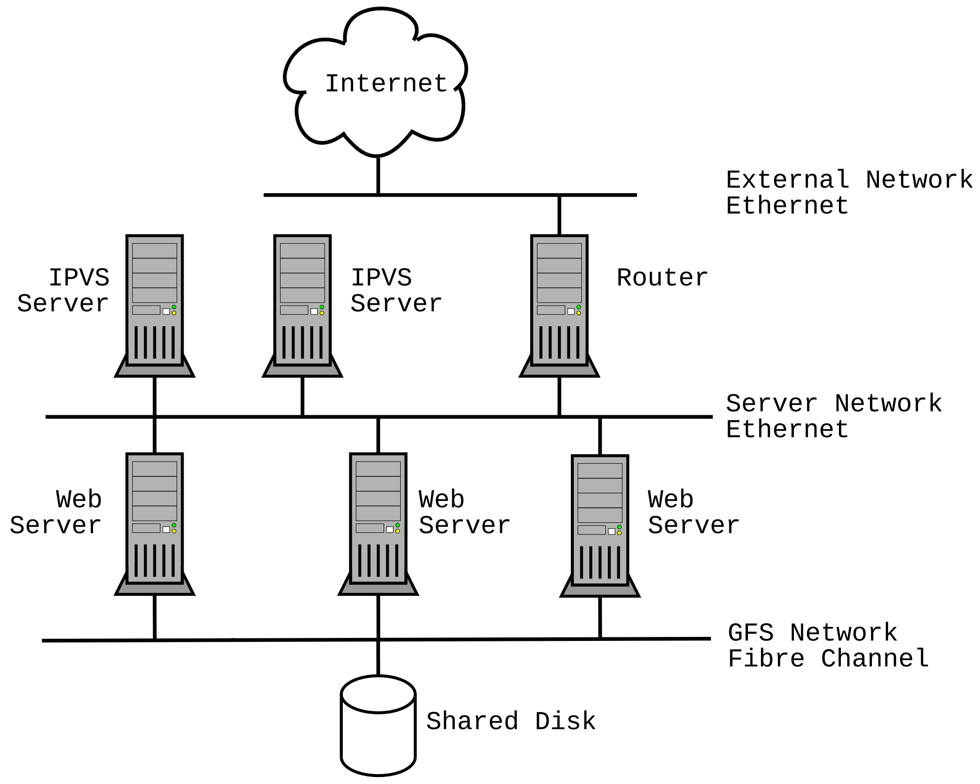

Figure 1: Sample Web Farm

useful model for explaining how a web farm works. A sample web farm is shown in figure 1. [2]

- The top layer of servers handles the multiplexing of incoming clients to web servers. In the case of the example given, incoming traffic travels through the router to the active IPVS Server — the other IPVS server is a hot stand-by. The IPVS server then routes the client to one of the back end web servers.

  A similar topology would be implemented for other host-based layer 4 switching technologies such as the Cisco LocalDirector or the F5 BIG/ip. It is possible to construct a topology whereby the layer 4 switching servers are also the gateway routers to the network. However, it is often desirable to separate routing and multiplexing functionality to provide greater flexibility in how traffic is handled. If a layer 4 switch is used then the IPVS servers are eliminated and this switch forms all or part of the switching fabric for the Server Network.

- The middle layer contains the web servers. Typically, this layer contains the largest number of servers and these servers contain no content and very little state. These servers can be thought of as shared disk or network file system or RDBMS[4] to HTTP or HTTPS converters. If any complex processing of requests is to be done then it should be done here as the compute power of this layer can easily be increased by adding more servers. Where possible state should be stored on clients by either using cookies or encoding the state into the URL[5]. This prevents the necessity for a client to repeatedly connect to the same server which makes for more flexible load-balancing, enabling a client session to continue even if a server fails. This is the layer where Linux servers are most likely to be found today.

- The bottom layer contains the data or truth source. There are many options here and common choices include servers for network file systems such as NFS[6] and AFS[7] or Database Servers for RDBMSs such as Oracle, MySQL, mSQL and PostgreSQL. In the future server independent storage such as that supported by GFS[8] are likely to be utilised in this layer.

If a geographically distributed system is required then intelligent DNS solutions such as Resonate and Eddieware can be employed that will return the IP address of one of the servers, based on some heuristic. Alternatively, a central web server can handle all incoming requests and distribute them using an HTTP redirect after making a decision on which server the client should be directed to. The rewrite module that ships with the Apache HTTP Server is a very useful method of achieving this. It is also possible, using EBGP4[9], to advertise the same network in more than once place and let the routing topology route customers to the most appropriate web server for them. Of

[4]RDBMS: Relational Database Management System

[5]URL: Universal Resource Locator

[6]NFS: Network File System. Ubiquitous network file system developed by Sun Microsystems

[7]AFS: Andrew File System. Network file system developed by IBM with data cached to local disk by clients.

[8]GFS: Global File System. Network file system that supports shared disks accessible via fibre channel.

[9]EBGP4: Exterior Border Gateway Protocol version 4. The routing protocol that is used to distribute external, inter-network routes on the internet.

course any instance of a web server in this discussion can be replaced with a web farm as per figure 1. A web farm of web farms if you will.

4 Technologies

There are several key technologies that are implemented in many Linux high availability solutions and are directly applicable to the implementation of a web farm. The names of these terms can be misleading and even be used to refer to different technologies in other contexts.

4.1 IP Address Takeover

If a machine, or service running on a machine, becomes unavailable, it is often useful to substitute another machine. The substitute machine is often referred to as a hot stand-by. In the simplest case, IP address takeover involves two machines, each with their own IP address that, are used for administrative access. In addition, there is a floating IP address that is accessed by end-users. The floating IP address will be assigned to one of the servers, the master.

IP address takeover begins with the hot stand-by bringing up an interface for the floating IP address. This is most conveniently done by using an IP alias, that is, setting up a second logical interface on an existing physical interface. Once the interface is up, the hot stand-by is able to accept traffic, and answer ARP requests, for the floating IP address. This does not, however, ensure that all traffic for the floating IP address will be received by the hot stand-by.

Though the master host may be inaccessible, it may still be capable of answering ARP[10] requests for the hardware address[11] of the floating IP address. If this occurs then each time a host on the LAN[12] sends out an ARP request there will be a race condition, and potentially packets will be sent to the

[10]ARP: Address Resolution Protocol. The protocol used on ethernet networks to map IP addresses to hardware addresses

[11]Hardware Address: On an ethernet network each network card has a unique address that is used designate a frame on the segment to a host. This is known as the hardware address. There is a broadcast hardware address that designates that a frame should be received by all hosts.

[12]LAN: Local Area Network. Network used to connect machines in close physical proximity. Typically high bandwidth and low latency

master which has been determined to have failed in some way. In addition, even if the master host does not issue ARP replies, traffic will continue to be sent to the interface on the master host. This will continue until the ARP cache entries of the other hosts and routers on the network expire.

To expediate fail-over and ensure all traffic goes to the the hot stand-by, a technique known as gratuitous ARP is used. Usually ARP works as follows. Host A sends out an ARP request for the hardware address of an IP address on host B. Host B sees the request and sends an ARP reply containing the hardware address for the interface with the IP address in question. Host A then records the hardware address in its ARP cache so it doesn't have to do an ARP request and wait for a reply each time it wants to send a packet. Entries in an ARP cache typically expire after about two minutes. A gratuitous ARP is an ARP reply when there was no ARP request. If the ARP reply is addressed to the broadcast hardware address then all hosts on the LAN will receive the ARP reply and refresh their ARP cache. If gratuitous ARPs are sent often enough then no host's ARP entry for the IP address in question should expire, so no ARP requests will be sent out, so there is no opportunity for a rouge ARP reply from the failed master to be sent out.

To relinquish an address obtained through IP address takeover the interface for the floating address should be taken down. Furthermore, to ensure a rapid transition, gratuitous ARP should be issued with the hardware address of the interface on the master host with the floating address. Depending on the service, it may be better to reverse the roles of the hot stand-by and master once the failed master comes back on line, rather than undoing fail-over. To do this effectively the hosts will need to negotiate ownership of the floating IP address, ideally using a heartbeat protocol.

Gratuitous ARP can be used to maliciously take over the IP address of a machine. Because of this, some routers and switches ignore, or can be configured to ignore gratuitous ARP. On a given network, this may or may not be an issue, but for IP address takeover to be successful, the equipment must be configured to accept gratuitous ARP or flush the ARP caches as necessary. Other than this there are no know problems with using gratuitous ARP and, hence, IP address takeover on both switched and non-switched ethernet networks.

4.2 Layer 4 Switching

Layer 4 switching is a term that has almost as many meanings as it has people using the term. In the context of this paper it refers to the ability to multiplex connections received from end-users to back-end servers. This can be implemented in an ethernet switch such as the Alteon Networks ACESwitch. It can also be done in a host such as the Linux Virtual Server, Cisco LocalDirector, F5 BIG/ip and an element of IBM WebSphere[13].

A Virtual Service is the point of contact for by end-users and is typically advertised through DNS. A virtual server is defined by: the IP address that clients will use to access the service; the port that clients will connect to and a protocol, either UDP/IP or TCP/IP. The virtual service is assigned a scheduling algorithm which allocates incoming connections to the back-end servers. The scheduling algorithms available will depend on the implementation. In the case of TCP/IP all packets for the life of the connection will be forwarded to the same back-end server so the integrity of the connection between the client and the back-end server is maintained. Many implementations have a feature that allows subsequent TCP/IP connections or UDP/IP datagrams from a host or network to be forwarded to the same back-end server. This is useful for applications such as HTTPS where the encryption used relies on the integrity of a handshake made between the client and a server, hence, clients need to consistently hit the same back-end server.

When a packet is to be forwarded to a back-end server several mechanisms are commonly employed. As a guide the mechanisms implemented by the Linux Virtual Server Project are detailed here.

- **Direct Routing:** Packets from clients are forwarded directly to the back-end server. The IP packet is not modified, so the back-end servers must be configured to accept traffic for the virtual server's IP address. This can be done using a dummy interface, or packet filtering to redirect traffic addressed to the virtual server's IP address to a local port. The back-end server may send replies directly back to the client. That is if a host based layer 4 switch is used, it may not be in the return path.
- **IP-IP Encapsulation:** IP-IP Encapsulation or Tunnelling enables packets addressed to an IP address to be redirected to another address,

[13]Web Sphere is a collection of tools including layer 4 switching technology. It is probably best know for providing the infrastructure for hosting the web presence for the 1996 Atlanta Olympic Games.

possibly on a different network. In the context of layer 4 switching the behaviour is very similar to that of direct routing, except that when packets are forwarded they are encapsulated in an IP packet, rather than just manipulating the ethernet frame. The main advantage of using tunnelling is that back-end servers can be on a different networks.

- **Network Address Translation:** Network Address Translation or NAT is a method of manipulating the source and/or destination port and/or address of a packet to map networks. The most common use of this is IP Masquerading that is often used to enable RFC 1918[8] private networks to access the internet. In the context of layer 4 switching, packets are received from clients and the destination port and IP address are changed to that of the chosen back-end server. Return packets pass through the layer 4 switching device at which time the mapping is undone so the client sees replies from the expected source.

4.3 DNS Methods

One of the simplest ways to effect fail-over is to manually modify the DNS records for a host. If a server fails then a DNS lookup for the host can return the IP address of another machine. DNS can also be used to implement scalability by assigning multiple IP addresses to a single hostname in DNS. Modern DNS servers such as BIND[14] 8.x will deterministically issue the different IP addresses assigned to mail.bigisp.com in a round-robin fashion[5]. Unfortunately this has a number of fundamental problems [1], [11].

1. The time to live (TTL) on the zone files needs to be turned down severely to to reduce the time for which results are cached. The longer the TTL, the less control there is over which IP addresses that end-users are accessing. The shorter that TTL, the greater the potential for congestion on the DNS server.

2. Users may access servers using an IP address rather than a host name.

3. Users may use non-DNS methods such as an /etc/hosts file to map server host names to IP addresses.

[14]BIND: Berkeley Internet Name Domain. A reference, and arguably the most widely used DNS daemon.

4. An additional problem with round-robin DNS is that the DNS daemon cannot differentiate between a request for a one-off hit, and a request that will result in many hits. That is, it is hard to control the granularity of scheduling.

5. When using round-robin DNS there is no way to assign weights to servers, all servers will theoretically receive the same number of requests, regardless of their resources and current load.

Problems 2 and 3 are more likely to be a problem when dealing with a site with a very large number of end users, such as a large corporate network or an Internet Services Provider (ISP) – users tend to make more assumptions about the static nature of a network that they are connected to than other networks. In particular, these are unlikely to be problems on large internet sites that are not provided primarily for the users of a particular ISP.

Problems 1 and 4 are inherent problems with a DNS based solution. While there is no good way to get around the granularity problem, both this and the TTL problem are generally helped by lowering the TTL. Provided that there are well linked, powerful DNS servers to handle the higher than otherwise required number of DNS requests.

Problem 5 can be aided by a more intelligent DNS server that takes into account feedback from servers about their load, availability and other metrics. This enables load to be distributed to servers compatible with their ability to serve customers.

Despite all of these problems an intelligent DNS server arguably one of the most robust, easy to implement and transparent to end users method of distributing traffic to multiple servers. This is particularly true when the servers are geographically distributed, remembering that each geographically separated server could be a point of contact to a web farm using a technology such as layer 4 switching to manage traffic within a point of presence.

4.4 Heartbeat

A heartbeat is a message sent between machines at a regular interval of the order of seconds. If a heartbeat isn't received for a time — usually a few heartbeat intervals — the machine that should have sent the heartbeat is assumed to have failed. A heartbeat protocol is generally used to negotiate and monitor the availability of a resource, such as a floating IP address. Typically when a heartbeat starts on a machine it will perform an election process with other machines on the heartbeat network to determine which, if any machine owns the resource. On heartbeat networks of more than two machines it is important to take into account partitioning, where two halves of the network could be functioning but not able to communicate with each other. In a situation such as this it is important that the resource is only owned by one machine, not one machine in each partition.

As a heartbeat is intended to be used to indicate the health of a machine it is important that the heartbeat protocol and the transport that it runs on is as reliable as possible. Effecting a fail-over because of a false alarm may, depending on the resource, be highly undesirable. It is also important to react quickly to an actual failure, so again it is important that the heartbeat is reliable. For this reason it is often desirable to have heartbeat running over more than one transport, for instance an ethernet segment using UDP/IP, and a serial link.

5 Existing Solutions

This section will briefly outline some of the solutions, both open and closed source, that are currently available for Linux. This is by no means a comprehensive list as the number of solutions available makes construction such a list difficult. However, the list does make mention of solutions that utilise the technologies outlined in section 4.

5.1 Heartbeat

Author	Alan Robertson
Site	http://www.linux-ha.org/download/
Licence	GNU General Public Licence

Heartbeat implements a heartbeat as per section 4.4 over raw serial, PPP over serial and UDP/IP over ethernet. In the case of fail-over, heartbeat effects IP address takeover. When a fail-over occurs heartbeat can activate or deactivate resources. A resource is a programme that is executed by heartbeat and hence, heartbeat can be used to produce arbitrary system changes on fail-over.

5.2 Linux Virtual Server Project

Project Lead	Wensong Zhang
Site	http://www.LinuxVirtualServer.org/
Licence	GNU General Public Licence

The Linux Virtual Server or Internet Protocol Virtual Server (IPVS) is an implementation, in the Linux kernel of layer 4 switching as described in section 4.2. The implementation supports direct routing, tunnelling and network address translation as forwarding methods. Least Connected, Weighted Least Connected, Round Robin and Weighted Round Robin scheduling algorithms are provided.

A daemon, ldirectord that ships as part of the Linux Virtual Server can be used to monitor the health of back-end servers if they are being used as web servers. ldirectord periodically requests a known page, checking that the response contains an expected string. If a web server fails then the server is taken out of the pool of real servers and will be reinserted once it comes back on line. If all the web servers are down then a fall-back server is inserted into the pool, which will be removed once one of the back-end web servers comes back on line. Typically the fall-back server will be localhost, running an Apache HTTP server that returns a page indicating that the service is temporarily inaccessible, for all URLs requested.

5.3 Eddieware

Vendor	Ericsson
	The Royal Melbourne Institute of Technology
Site	http://www.eddieware.org/
Licence	Erlang Public Licence

Eddieware is designed for creating both non-geographically and geographically distributed internet services. At this stage Eddieware only supports web servers, through the underlying infrastructure should be easily extendible to other services. Eddieware contains two key components, the "Intelligent HTTP Gateway" and and the "Enhanced DNS Server".

- **Intelligent HTTP Gateway** runs on front-end servers for a site, that is servers that are directly contacted by end-users and advertised through DNS. The front-end servers receive information about the load an availability of back-end servers and uses this information along with quality of service metrics to load-balance incoming connections. If a connection is to be forwarded to a back-end server then a connection is made from the front-end server to the back-end server and any data received from the end-user is sent to the back-end server and vice versa.

- **Enhanced DNS Server** can be used to distribute sites geographically. It should be run as the name daemon on the authoritative name servers for the domain or domains in which the service or services to be distributed reside. The Enhanced DNS server receives load information from the front-end servers at each site and based on availability and load returns the IP address of the front-end servers that are in the cluster to end-user requests.

5.4 TurboCluster

Vendor	TurboLinux, Inc.
Site	http://www.turbocluster.com/
Licence	Kernel Modules: GNU General Public Licence
	User Level Applications: Closed Source

TurboCluster supports layer 4 switching and fail-over in much the same way as IPVS and Heartbeat combined. It is shipped as a modified TurboLinux distribution and comes with several GUI configuration utilities. To set up TurboCluster, routers and servers are defined. For each service a single router is active at any given time and accepts traffic for the floating IP address of the service. Other routers are swapped in using IP address takeover in the case of failure. The servers are the back-end servers to which connections are forwarded. These servers are monitored by the routers and will not be forwarded traffic if they become inaccessible. TurboCluster supports tunnelling and direct routing to forward packets to the back-end servers and both weighted and non-weighted round-robin scheduling algorithms are available. [9]

5.5 Resonate

Vendor	Resonate, Inc.
Site	http://www.resonate.com/
Licence	Closed Source

Resonate, Inc, provides several products for network management and providing scalable, highly available internet sites. Of most relevance to this paper are Central Dispatch[3] and Global Dispatch[7].

- **Central Dispatch** is somewhat analogous to the intelligent HTTP gateway component of Eddieware. Incoming packets from clients are examined and using either round robin or resource based scheduling, allocated to a back-end server. Resource based scheduling requires the data in the IP packet be examined and enables user-defined rules based on content and resource availability. Packets are forwarded using TCP Connection Hop[4] a method of forwarding packets from user space that is quite similar to IPVS's direct routing, done outside of the kernel.
- **Global Dispatch** is similar in operation to the enhanced DNS server of Eddieware. Global dispatch is an intelligent DNS server that replies to client requests according to the load and availability of geographically distributed sites. Global dispatch also takes into account network latency between the client's local DNS server and the available sites. It is interesting that latency is used as a metric when EBGP — the routing protocol used to distribute inter-network routes on the internet — does not use latency as a metric in selecting least-cost routes.

5.6 Piranha

Vendor	Red Hat, Inc.
Project Lead	Keith Barrett
Site	http://www.redhat.com/
Licence	GNU General Public Licence

Piranha is a suite of tools to configure and manage an Linux Virtual Server (LVS) based service. Older versions of Piranha support a GTK+ front end which is being phased out in favour of an HTML-based configuration tool. Piranha comes with its own heartbeat tool to monitor the LVS servers and effect IP address takeover in the case of failure. Piranha also has its own tool for monitoring the health of back-end servers and manipulating the real servers for each virtual server configured for LVS appropriately.

5.7 Ultra Monkey

Vendor	VA Linux Systems, Inc.
Project Lead	Horms
Site	http://www.ultramonkey.org/
Licence	GNU General Public Licence

Ultra Monkey uses the Linux Virtual Virtual Server to provide load balancing, a heartbaeat protocol to provide high availability and ldirectord for service lever monitoring of real servers. Ultra Monkey offers a single point of contact for software and documentation for network engineers to deploy high availability and/or scalability.

6 Developing Technologies

6.1 The Global File System

Creating highly available data stores remains a significant problem. It is desirable that data should be highly available but – other than in the case of static, read only data – asynchronous changes make this very difficult to achieve. The advent of fibre channel and shared storage often referred to as a SAN[15] has led to the development of The Global File System that effectively eliminates much of the need for file servers and single point of failure data storage.

The Global File System facilitates access to shared fibre channel disks without the need for a master node for mediation of resources. This is achieved by storing meta-data on the disks using device locks or dlocks. As there is no master node, there is no host acting as a single point of failure or bottleneck. While the disks themselves still represent single points of failure this can be eliminated by using a fibre channel RAID[16] device. The failure of a fibre channel switch should, at worst, prevent access to part of the fabric.

GFS is currently functional and a network with in excess of 1Tb of storage was demonstrated at Linux World[2], New York in February 2000. Unfortunately

[15]SAN: Storage Area Network. Host-independent disks, connected to each other and hosts by a network.

[16]RAID: Reliable Array of Inexpensive Disks. A method of creating fault tolerant storage by striping and/or mirroring data across multiple disks.

journaling has not been implemented; while an array of n machines with m disks will work fine across the switched fabric, if a node fails then there is a risk that the file system will become corrupted. Work is currently in progress to resolve this problem and it is hopped that this will be completed before the end of the year.

GFS is developed by The GFS Group at the University of Minnesota, led by Dr. Matthew O'Keefe. The code is released under the GNU General Public Licence. More information can be found on the project web site[17].

6.2 Generic Framework

There has been a lot of discussion about the need for a generic high availability framework for Linux. To date the foremost proponent of this has been Stephen Tweedie of Red Hat. The idea is that instead of providing specific applications, kernel modifications and hardware to solve specific problems a more generic API should be defined to enable the components to be integrated together [10]. This work is a long term project.

SGI and SuSE have recently released SGI's FailSafe, ported to Linux. Fail Safe has been released under the GNU Genereal Public Licence and Lesser Public Licence and is available for download[18]. FailSafe offers a generic API for high availability, though not as extensive as that proposed by Stephen Tweedie et al.

7 Conclusion

The support for high availability and scaling services under Linux continues to grow. At this stage, Linux is well accepted as the middle layer of a scaled service, for example the stateless compute power for a web farm. Technologies such as layer 4 switching and intelligent DNS implemented under Linux are helping to push Linux towards the front-end of such services. In the long term, emerging technologies will help Linux to fulfill all the requirements for a highly available, scaled service.

While high availability and scalability is much more than building big web

[17]`http://www.globalfilesystem.org/`
[18]`http://oss.sgi.com/projects/failsafe/`

farms, these technologies to do this provide a good starting point to build more advanced solutions. Generic frameworks to implement high availability upon are beginning to take shape and should enable much more powerful solutions to be developed in the future.

References

[1] Simon Horman. Creating redundant linux servers. In *Proceedings of the 4th Annual Linux Expo*, 1998.

[2] Simon Horman and John Goebel. Complete open source linux web farm solution. http://ultramonkey.sourceforge.net/, January 2000.

[3] Glen Kosaka. Resonate, techinical overview, resonate central dispatch: Powerful distributed traffic management for ip-based networks. http://resonate.com/.

[4] Glen Kosaka. Resonate, techinical overview, resonate central dispatch tcp connection hop. http://resonate.com/.

[5] Cricket Liu and Paul Albitz. *DNS and BIND*. O'Reilly and Associates, 3 edition, September 1998.

[6] Harald Milz. Linux high availability howto. http://metalab.unc.edu/pub/Linux/ALPHA/linux-ha/High-Availability-HOWTO.html, December 1998.

[7] Andrew Reback. Resonate, techinical overview, resonate global dispatch: Optimized traffic management across geographically-dispersed points of presence. http://resonate.com/.

[8] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Request for comments 1918: Address allocation for private internets, February 1996.

[9] TurboLinux. Turbocluster server 4.0 administrator guide, version 4.0, October 1999.

[10] Stephen Tweedie and Peter Braam. Ha/clusters development symposium, January 2000.

[11] Wensong Zhang, Shiyao Jin, and Quanyuan Wu. Creating linux virtual servers. http://linux-vs.org/linuxexpo.html, February 1999.