Research Report
XEN based HA backup environment

Research Report for RP1
University of Amsterdam
MSc in System and Network Engineering

Class of 2006-2007

Peter Ruissen, Marju jalloh
{pruissen,mjalloh}@os3.nl

February 5, 2007

Abstract

In this paper we will *investigate the possibilities for High Availability (HA) failover mechanisms using the XEN virtualization technology and the requirements necessary for implementation on technical level.* Virtualization technology is becoming increasingly popular in server environments because it adds a layer of transparency and flexibility on top of a hardware layer, reduces recovery time and utilizes hardware resources more efficiently. Back in the 1960s, IBM developed virtualization support on a mainframe. Since then, many virtualization projects have become available for UNIX/Linux and other operating systems. The XEN project offers a novel technique known as paravirtualisation which brings a whole new range of possibilities to the table.

Our tests showed that it is possible to use XEN in combination with Hearbeat to provide a HA environment. Even though combining XEN virtualization technology and High Availability software is still in the beginning stages at this moment, our research showed that XEN can be used with Heartbeat to realize a flexible, reliable and efficient HA environment.5.1

Contents

Chapter 1

Project information

1.1 Assignment formulation

To research the possibilities for High Availability (HA) failover mechanisms using the XEN virtualization technology and the requirements necessary for implementation on technical level.

1.2 Project Description

By combining server virtualization with high-availability failover mechanism, organizations can realize the benefits of increased manageability and savings from server consolidation without risking increased downtime for business-critical applications. *The research covered in our project will determine if there is a scenario possible to implement HA on a technical level using Xen.*

1.3 Scope

This project scope is created during the initiation phase. In-scope is any work we will have to perform to address the problem or opportunity we are working on during the project. Out-of-scope is any work that is either not related to what we are working on or which is related, but not necessary to complete to achieve our project objectives. This is neccesary due to the limited time frame of our project.

In Scope:

- do an extensive research into how XEN works and virtualization technology in general
- investigate a possible HA failover scenario for services.
- investigate (dis)advantages and in particular the reason to use XEN.
- investigate XENs migration feature in failover scenarios.
- testing basic failover services: Apache as web service, BIND as DNS service, EXIM as mail service
- investigate the use of DRBD for data synchronization if necessary.
- investigate the use of using Heartbeat for monitoring XEN nodes.
- adding a second layer of HA using Heartbeat to control services of VM

Out of Scope

- Implementing other virtualization technologies like user mode linux (UML)

- doing multi platform tests; like installing XEN under BSD
- testing failover with exotic web services like AOL server, IIS, Tomcat. Same applies to other services.
- making quality assurance plans, user support guides and writing down resource needs and costs.
- security flaws of overall design
- doing scientific measurements
- investigate performance of services

Chapter 2

Virtualization technology

VT is used to fully utilize hardware resources in a server environment. The exponential growth of computing power has made it feasible to turn physical hosts into a multitude of virtual machnines. This slashes the majority of hardware acquisition and maintenance costs that can result in significant savings for any company or organization. VT is also used for disaster recovery, load balancing, development testing and in our research: for HA. Other advantages are:

- Multiple legacy system consolidation
- Flexibility to move one system to another
- Simplifies the application stack: largely removing hardware and drivers from the equation

There are also disadvantages:

- CPU and memory overhead
- Increased complexity

Virtualization is the perfect solution for small or medium-scale usage and should not be used for high-performance applications when one or more servers are clustered together to meet the requirements of a single application [1]. The applications and services running on virtual machines should not exceed a reasonable service level agreement.

Virtualization techniques allow to run multiple virtual machines on a single physical host and has existed for over 40 years. Since then many virtualization projects have become available for UNIX/Linux and other operating systems, including VMware, Qemu, MS Virtual PC and Solaris Zones. Today, virtualization is once again a hot topic in a server environment because several emerging virtualization technologies have the potential to improve resource utilization, efficiency, scalability and manageability for commodity hardware systems.

2.1 Forms of VT

There are six types of virtualization: emulation, full virtualization, para-virtualization, os-virtualization, application virtualization and hardware virtualization.

- Emulation is the most basic form and also known as simulation allowing an unmodified guest OS for a completely different CPU to be run. This approach has long been used to enable the creation of software for new processors before they were physically available. Examples include Bochs, PearPC, PPC version of Virtual PC, Qemu without acceleration, and the Hercules emulator. Emulation is implemented using a variety of techniques, from state machines to the use of dynamic recompilation on a full virtualization platform.

- Full virtualization or Native virtualization simulates all hardware needed to allow multiple unmodified guest OS to be run in isolation. The guest OS must be designed for the same CPU. Examples include Virtual Iron, VMware Workstation, VMware Server (formerly GSX Server), Parallels Desktop, Adeos, Mac-on-Linux, Win4BSD, Win4Lin Pro, and z/VM.
- Para-virtualization offers a special API instead (or in addition) of simulating hardware. It requires the guest OS to be explicitly ported to run on top of the virtual machine monitor (VMM) or additional hardware support.This enables the VMM to be simpler and to achieve higher performance. Examples of para-virtualization are XEN, Denali, User Mode Linux (UML),VMware ESX Server, Win4Lin 9x, and z/VM.
- OS virtualization uses the same kernel for both the host as guest OS on a physical server. Examples are Linux-VServer, Virtuozzo, OpenVZ, SUN Solaris Zones and FreeBSD Jails. The technology is also referred as Single Kernel Image (SKI). It avoids performance problems found with pure emulation, but it does so at the expense of flexibility and security.
- Hardware virtualization allows hardware functionality that runs together with compatible VMM software to increase supportability of the overall virtual machine solution. Xen supports the use of hardware virtualization capabilities of Intels VT and AMDs Pacifica processors. These technologies, while differing quite substantially in their implementation and instruction sets, are managed by a common abstraction layer in Xen and enable unmodified guest operating systems to run within Xen virtual machines, starting with Xen 3.0. The list of supported unmodified guests is limited to certain versions of Windows (incl. XP) and Linux.
- Application Virtualization creates a small virtual environment containing the components needed to execute applications. These include registry entries, files, environment variables, interfaces and global objects. This virtual environment acts as a layer between the application and the operating system, and eliminates application conflicts and application-OS conflicts. Examples include the Sun Java Virtual Machine, Softricity, Thinstall, Altiris, and Trigence.

Chapter 3

High availability concepts

High availability is the system management strategy of quickly restoring essential services in the event of system, component, or application failure. The goal is minimal service interruption rather than fault tolerance. High availability can be achieved by detecting node or daemon failures and reconfiguring the system appropriately, so that the workload can be taken over by the remaining nodes in a cluster.

The term cluster has different meanings within the computing industry. Throughout this report, unless noted otherwise, cluster describes a collection of nodes and resources (CPU time, storage and memory) that cooperate to provide high availability of services running within the cluster. If one of those machines should fail, the resources required to maintain business operations are transferred to another available machine in the cluster.

HA is not the same as fault tolerant, in which fault tolerant components are designed for continuous processing. [1]

The two main cluster configurations are:

- Standby configuration: The most basic cluster configuration, in which one node performs work while the other node acts only as standby. The standby node does not perform work and is referred to as idle; this configuration is sometimes called cold standby. Such a configuration requires a high degree of hardware redundancy.
- Takeover configuration: A more advanced configuration in which all nodes perform some kind of work, and critical work can be taken over in the event of a node failure. In a one-sided takeover configuration, a standby node performs some additional, non-critical, non-movable work. In a mutual takeover configuration, all nodes are performing highly available (movable) work.

You must plan for several key items when setting up an HA cluster:

- The disks used to store the data must be connected by a private interconnect (serial cable) or LAN to the servers that make up the cluster.
- There must be a method for automatic detection of a failed resource. This is done by a software component referred to as a heartbeat monitor.
- There must be automatic transfer of resource ownership to one or more surviving cluster members upon failure.

The availability of a service is determined in a service level agreement of a company (SLA). SLA are increasingly being used in enterprise networks and are parts of contracts that specify the performance parameters within which a network service is provided.[2]

[1]It should be noted that uptime and availability are not synonymous. A system can be up, but not available, as in the case of a network outage.

[2]article Managing Service Level Agreements By Nathan J. Muller

3.1 Service availability

In general there are two causes of service unavailability:

- Controlled outages can be controlled by the system administrator. Examples are a system reboot/shutdown or restarting service after updating config files, downtime due to hardware maintenance and software updates. In general, planned downtime is usually the result of some logical, management-initiated event. In this scenario, downtime can be avoided by using a cluster manager together with the live migration feature of the XEN virtual machine.

- Uncontrolled outages can be caused by network failures, power failures and hardware failures. In this scenario downtime cannot be avoided but can be minimized. Live migration cannot be used. Cold failover by starting a new virtual machine is preferred. The reason for this is that the memory dump of the failed node most likely already contains the reason for the failure (memory leaks, corrupted pointers, stale processes). A interesting approach is to make periodic snapshots of a current running VM without disturbing it much. The advantage is that you can restore it quickly with the most actual status. Unfortunately the on-disk and in-memory state has to match, else you will badly corrupt your filesystem. Resuming a suspended machine after an uncontrolled outage is therefore out of the question.

3.2 Linux High Availability projects

Linux Virtual Server (LVS) is an advanced load balancing solution for Linux systems. It is an open source project started by Wensong Zhang in May 1998. LVS uses a router known as the Linux Director to forward packets from end users to the nodes running services. When packets are received for a virtual service by the Linux Director, the scheduling algorithm decides which real-server to send the packet to. Once this decision is made subsequent packets to for the same connection will be sent to the same real server. Thus the integrity of the connection is maintained. HA solutions can be achieved by combining LVS with one of the following tools:

- Keepalived provides a strong and robust health checking for LVS clusters. It implements a framework of health checking on multiple layers for server failover, and VRRPv2 stack to handle director failover.

- Piranha is one of the clustering products from Red Hat Inc that can check the heartbeat between active and backup load balancers and the availability of the services on each of the nodes.

- The ldirectord (Linux Director Daemon) is a simple stand-alone daemon to monitor services of real servers, currently http and https service. It is specially written for LVS and it works with heartbeat V1

Heartbeat version 2 is a high-availability system for Linux and provides monitoring of cluster nodes, applications, and provides a sophisticated dependency model with a rule-based resource placement scheme. It monitors services and restarts them on errors. When managing a cluster, it will also monitor the members of the cluster and begin recovery of lost services in a short time period. It runs over serial ports and UDP broadcast/multicast, as well as OpenAIS multicast. When used in a cluster, it can operate using shared disks, data replication, or no data sharing. The design is different in contrast to LVS because the nodes in the cluster acts failover nodes for each other by performing IP failover. In V2, the main components are as follows:

- HeartbeatProgram implements the heartbeat-protocol that perform dead-of-node detection. Heartbeat can send heartbeat messages over both serial links and ethernet interfaces. Heartbeat is also able to use qourum devices [3] to help determine which nodes should be active, this is done using the ipfail plugin.

[3]a special shared logical device/drive used during clustering

- LocalResourceManager (LRM) has responsibility for performing operations on resources by using ResourceAgent scripts. A resource agent is an encapsulation of information about a particular kind of cluster resource. This resource refers to a script to start, stop and reload a certain service like a webservice.
- Cluster Resource Manager tries to make sure that every resource is made available to users by making sure it is running somewhere in the cluster.
- Cluster Information Base is a replicated store of (primarily) two types of cluster related information. The Cluster Information Base is stored and processed as XML and its layout and mandatory information is contained in a DTD.
- StonithDaemon is a cluster-wide abstraction for STONITH topology. STONITH is a technique for NodeFencing, when an HA system declares a node as dead, it is merely speculating that it is dead. STONITH takes that speculation and makes it reality.

3.3 High Available Storage

To maintain data integrity and allow all virtual machines to have access to the same data pool there is a need for a HA storage facility. Currently there are several approaches.

- Distributed filesystem (network file system) or network attached storage is a shared medium where the data isnt locally attached to a host. The most common examples are NFS, CIFS or SMB. Although this is the simplest way to provide storage its considered to be a rather slow solution and is not recommended in a production environment.
- Global File System refers to the namespace, so that all files have the same name and path name when viewed from all hosts. This makes it easy to share data across machines and users in different parts of the organization. It allows a cluster of computers to simultaneously use a block device that is shared between them. GFS reads and writes to the block device like a local filesystem, but also uses a lock module to allow the computers coordinate their I/O so filesystem consistency is maintained. [4]
- SAN Filesystem provide a way for hosts to share Fibre Channel storage, which is traditionally carved into private chunks bound to different hosts. To provide sharing, a block-level metadata manager controls access to different SAN devices. A SAN file system mounts storage natively in only one node, but connects all nodes to that storage and distributes block addresses to other nodes.

[4] GFS info at http://freshmeat.net/projects/theglobalfilesystem/

Chapter 4

The XEN project

XEN is a virtual machine monitor (VMM) for x86-compatible computers. Xen can securely execute multiple virtual machines, each running its own OS, on a single physical system with close-to-native performance. It uses paravirtualization techniques to achieve this without sacrificing either performance or functionality. An overview of XEN is given in the paper XEN and the art of Virtualization [2]

4.1 How XEN works

On a standard Intel compatible OS, the OS is designed to gain exclusive access to the underlying hardware. Therefore virtualization of the hardware is needed to allowing more than one operating system to access the hardware. XEN, hypervisor [1] allows multiple operating systems to access a single hardware at the same time without undesirably influencing each other.

The protection model of the Intel x386 CPU is built from four rings: ring 0 (kernel space) and ring 3 (user space) . Rings 1 and 2 are not used except in rare case like OS/2. [3] Processes in Ring 0 are allowed to execute any operation on the CPU, while process in the other rings permissions are handled more restrictively in ascending order. Thus requested operation at ring 3 will be handled and validated in Ring 0 (kernel space). if the request is valid and authorized, the kernel execute the operation and returning the desired data.

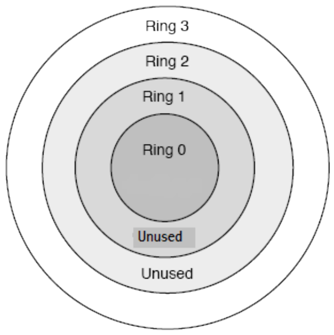

Figure 4.1: x86 rings protection architecture.

In XEN, the hypervisor runs in Ring 0, while XEN virtual machines (VMs) run in Ring 1 and user applications run in Ring 3. At boot time, XEN is loaded into memory in ring 0. It starts a patched Xen kernel in ring 1; this is called domain 0. From this domain, guest operating systems can be created in another domain. The created domains also run their kernels in Ring 1 and use their applications

[1] The XEN kernel is called a hypervisor because it operates at a higher privilege level than the supervisor code of the guest operating systems that it hosts.

in ring 3. Within domain 0, a process called xend runs to manage the system. Xend is responsible for managing and providing access to the virtual machines, and virtual machines gain access to the underlying hardware by making hypercalls. [4] Basic Xens virtualization architecture of 32-bits x86

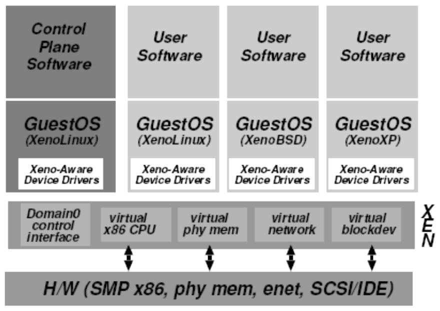

Figure 4.2: Xen architecture

CPU, memory and I/O devices can be found in [4]. The virtualization of these resources supports performance isolation, scheduling priority, memory demand, network traffic and disk access to prevent one instance of a VM to impact the performance of the other.

4.2 Migration Feature of XEN

Live Migration (PUSH/PULL) is one of XENs key features. Live migration simply means to move a virtual machine from one physical computer to another with negligible downtime. Live migration is done by using a per copymethod in which pages of memory are iteratively copied from the source to the destination without stopping the execution of the virtual machine being migrated [5]

A compulsory requirement of live migration is shared storage. Which mean disk images of virtual machines should be accessible from all physical host. Trovostino et al [6] have demonstrated a live migration between Amsterdam, NL and San Diego, USA and the results are astonishing. They measured the application downtime of a live migration as an interval of 0.8-1.6 seconds.

The reason to use XEN can be attributed to it numerous advantages over other virtualization technologies like User Mode Linux, VMWare, QEMU, Solaris Zones, Linux VServer. All of these technologies share some of the advantages of XEN, but the collective advantages makes it suitable for our research.

4.3 Advantages

The main advantages of using XEN are:

- XEN is Open Source. This brings a number of benefits over proprietary solutions, including improved functionality, better performance, and greater extendibility.
- It provide secure isolation by allowing each virtual machine to operate in it own domain. If one virtual machine is compromised, the others are unaffected - hence security.
- User level applications and libraries run unmodified. This makes it possible to use main stream application without modification.
- XENs unique performance benefits accrue from its paravirtualization technology, which allows hosted virtual servers to collaborate with the hypervisor to achieve the best performance for enterprise applications.

- It is scalable. Instances of operating systems can be added or removed on the fly.
- It is flexible. The migration feature can be used to migrate an instance of a virtual machine to another physical host.

Our research aim to utilize the above mentioned advantages by implementing a high availability setup using XEN for virtualization and Heartbeat for high availability.

4.4 Disadvantages

Notwithstanding the advantages, XEN has the following disadvantages:

- In the nearby future OS can be delivered with standard built-in virtualization technology.[2]
- The virtual machine (guest OS) has to be modified. Xen can only support unmodified guest VMs like Windows if you are using either a VT capable Intel processor or an SVM capable AMD processo. [7]

4.5 Usability

XEN can be used in many virtualization scenarios. Possible usage scenarios are:

- Internet hosting service company to provide virtual dedicated servers.
- to provide server consolidation thereby it benefits from increased hardware utilization, an ability to rapidly start and stop a virtual machine
- for load balancing in a cluster environment with minimum downtime
- development purposes; running the new system as a guest avoids the need to reboot the computer whenever a bug is encountered.
- can be used in computer security research by proving a sandboxed environment. This makes it possible to study the effects of some virus or worm without the possibility of compromising the host system.

4.6 Other Products

4.6.1 User Mode Linux

User Mode Linux (UML) is a virtual Linux machine that runs on Linux. It is basically a modified Linux kernel application on a normal Linux system(host). It process runs in a user space, this approach provides the user with a way of running multiple virtual Linux machines on a single piece of hardware without affecting the host environments configuration or stability. Even though Linux has been ported to many different processors, including x86, Suns SPARC, IBM and Motorolas PowerPC, DECs Alpha, and a variety of others, UML has only been ported to x86 and powerPC (work is on the way to port it to SPARC). This is because porting UML to other architecture involves porting UML to the software interface defined by Linux rather than the hardware interface defined by the processor and the rest of the physical computer. [10] . It is mostly used in educational and development enviroment.

The performance of UML is very slow because only one program can run in privilegde mode: the host kernel that support the guest kernle

[2]kernel virtual machine (KVM).

4.6.2 VMWare

VMware ESX Server uses a modified kernel that replaces the Linux kernel after hardware initialization. It adds a virtualization layer between the hardware and the operation system, and it partitions a physical server into multiple secure and portable virtual machines that can run side by side on the same physical server sharing a single hardware.This virtualization approach provides lower overhead and better control and granularity for allocating resources (cpu time, disk bandwidth, network bandwidth, memory utilization) to virtual machines. It also increases security, thus positioning VMware ESX as an enterprise-grade product. [3]

4.6.3 Linux VServer

The Linux-VServer technology is a soft partitioning concept based on Security Contexts which permits the creation of multiple virtual machines. The implementation is done adding operating system-level virtualization capabilities to the Linux kernel. Security is enhanced by a chroot mechanism which securely partition resources on a system. It is very fast and lightweight: virtual machines share the same system call interface and do not have any emulation overhead.[4]

4.6.4 Qemu

From the QEMU web site, [5] QEMU is a generic and Open Source processor emulator which achieves a good emulation speed by using dynamic translation. QEMU has two operating modes:

- Full system emulation. In this mode, QEMU emulates a full system (for example a PC), including a processor and various peripherials. It can be used to launch different Operating Systems without rebooting the PC or to debug system code.
- User mode emulation (Linux host only). In this mode, QEMU can launch Linux processes compiled for one CPU on another CPU.
- It supports wide range of architecture: IA-32(x86), AMD64, MIPS,SPARC, ARM and PowerPC.

4.6.5 Solaris Zones

Zones is a light weight operating system-level virtualization technology which completely isolate virtual machine within a server. Like Linux VServer, security is enhanced by a jail mechanism which prevent one process from interfering with the other A comparisons of the above software is shown below

[3]http://www.vmware.com/products/vi/esx/.
[4] http://linux-vserver.org
[5]http://fabrice.bellard.free.fr/qemu/about.html

Chapter 5

Research phase and designs

This phase has the focus on the development of components and other features of the system been developed. We chose Heartbeat because it is a widely used and important standard used in many High Availability solutions. However, Heartbeat has no functionality to manage virtual machines, doing live migration on demand and do resource monitoring (memory and CPU monitoring on the physical hosts). This chapter covers the features necessary to make a fully functional cluster manager that uses virtualization technology in a production environment.

5.1 System design

5.1.1 Common requirements

- Each virtual machine has one dedicated service so they cant interfere with each other.
- All physical nodes and virtual machines have access to the same HA data pool. The physical nodes mount this data pool for the images of the VM. Data like MySQL databases or website content are hold separately in the common data pool.
- Heartbeat V2 on the virtual machines will monitor services and will try to restart them if they fail. If that hasnt have any effect; Heartbeat will disable itself and other nodes will detect this.

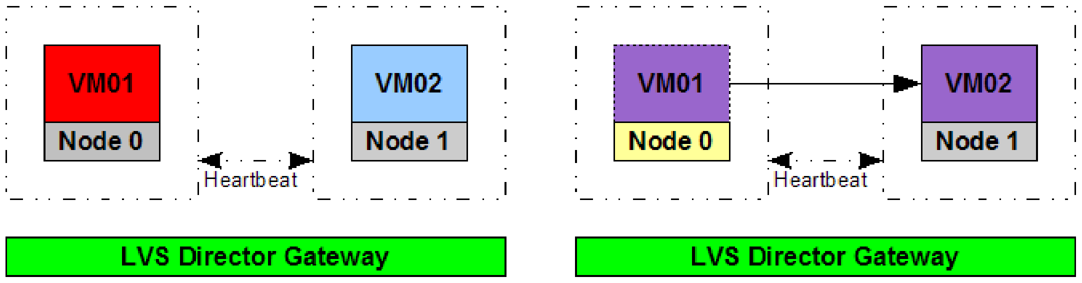

Figure 5.1: Shows cold failover due to a failed service, the second picture shows migration.

5.1.2 HA Environment using LVS, XEN and Heartbeat

System design 1 describes a system pool of N services and N nodes. Heartbeat is installed on both the physical and virtual machines. System design 2 describes a system pool of N nodes with N services and a redundant LVS director. Redundancy is applied to prevent that the load balancer might become a SPOF for the whole system. Two heartbeat daemons run on the primary and the backup respectively on the load balancer. When the heartbeat daemon of the backup cannot hear the heartbeat message from the primary in the specified time, it will take over the virtual IP address to provide the load-balancing service. When the failed load balancer comes back to work, there are two solutions, one is that it becomes the backup load balancer automatically, the other is the active load balancer releases the virtual IP address address, and the recover one takes over the VIP address and becomes the primary load balancer again.

The primary load balancer has state of connections, i.e. which server the connection is forwarded to. If the backup load balancer takes over without those connections information, the clients have to send their requests again to access service. In order to make load balancer failover transparent to client applications, we need connection synchronization in IPVS [1], the primary IPVS load balancer synchronizes connection information to the backup load balancers through UDP multicast. When the backup load balancer takes over after the primary one fails, the backup load balancer will have the state of most connections, so that almost all connections can continue to access the service through the backup load balancer.

Figure 5.1 shows a minimum of two physical machines for simplicity but may be extended. The number of physical machines and shared storage represent the total amount of system resources. This design is very efficient because it maximizes resource usage. More physical machines are waked from standby state when the need for resources are higher.

5.2 Production environment

Heartbeat needs extra functionality built in to support XEN failover scenarios. This section provides the requirements which needed to be implemented in such an environment. Procedures for a controlled outage:

- heartbeat on the current virtual machine is notified for a controlled shutdown and ask the domain controller for a suitable physical machine which can takeover the VM with dedicated service.
- the domain controller will make a selection on resources and will choose the one with the most system resources available for virtual machine failover.
- the script issued by heartbeat will perform a live migration (PUSH) to the chosen physical machine.

Procedures for a uncontrolled outage:

- heartbeat on multiple nodes in the network detect that one of the virtual machines or services is not responding. Once again; the domain controller will decide which host should takeover the virtual machine if its not already running somewhere in the cluster.
- The virtual machine will be cold booted from the chosen node.

The tasks:

- Build a resource management system into Heartbeat which monitors the resources from physical hosts and develop a system which controls XEN from Heartbeat.
- Implement a hardware STONITH device which ensures that a physical node is really dead and control this device from the heartbeat system.
- Use a Global File System to make sure that the data pool is HA.
- Build in safety measures into the heartbeat scripts to make sure that a live migration is reliable. If live migration fails; the uncontrolled outage procedure should be followed.

[1]IPVS (IP Virtual Server) implements transport-layer load balancing inside the Linux kernel

Chapter 6

Proof of concept

The main objective of this experiment is to test the interoperability of XEN and Heartbeat in providing High Availability services.

6.1 Test Environment

6.1.1 Hardware

An Intel(R)Pentium D CPU 3.00.GHz machine with two network cards. The computers are already in a network topology, therefore, we only abstract out our experimental network topology. For shared storage we planned on using NFS. Due to time constraint, and we did not want to get bugged down by to much network intricacies, we used an identical disk image on both physical hosts.

6.1.2 Software

The software on the two physical machines are identical. We use Kubuntu 6.10 GNU/Linux on all the machines, both physical and virtual, as OS. The XEN kernel, Xen-hypervisor-3.0-i386 is used on the physical hosts. The disk images for the virtual machines are created using a loopback partitioning. XEN configurations are the same.

The debian packages Heartbeat-2 and Heartbeat-2-dev are installed in all the machines, both physical and virtual. The configuration files of the tested machine are identical. The details is explained in the next section. Apache2 is used on both virtual machines as web service, and they are configured the same.

The uniformity in hardware, software and sometimes in configuration files are necessary to prevent problems in heterogeneous communication, and software versioning.

We tested network connectivity for a moving IP address during IP failover. This is done by pinging a virtual machine for connectivity, and then shutting it down to demonstrate a failover. The ping would hang, print destination unreachable and then access the IP on the other machine. A browser is use to estimate the time taken for a web switch. We also use the tail -f command to constantly monitor Hartbeats logfile. This is use to track configuration and communication errors.

We implemented three scenarios to demonstrate a failover mechanism using XEN and Heartbeat. We use XEN to create a virtual machine which runs a web service, and Heartbeat monitors nodes or services based on the scenario.

6.1.3 Scenario 1

The objective of scenario 1 is to test a failover between two physical host. In figure 1, node 0 host the virtual machine VM01, and the heartbeat communication monitor the existence of node 0. If node 0 fails 3.1, node 1 will create a new virtual machine with the same service and IP address of VM01.

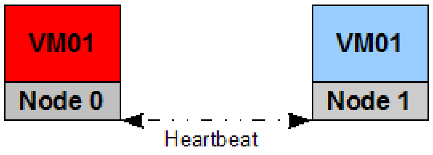

Figure 6.1: Heartbeat communication between physical Node0 and Node1.

And when node 0 is in the air, the process is reversed. In this scenario no IP address is moved. The network configuration of VM01 is static configured.

This scenario only monitor the existence of a node and not the service. It does not monitor web service on the virtual machine which is a disadvantage. If the service on VM01 or VM01 itself crash but node 0 is still in the air, it will not trigger any failover. This means HA cannot be guaranteed.

6.1.4 Scenario 2

The objective of scenario 2 is to test the failure of a service 3.1.

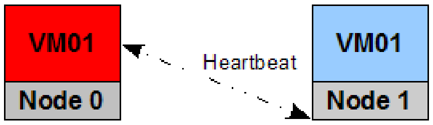

Figure 6.2: Heartbeat communication between VM01 and physical host node1.

In figure 2, the heartbeat communication monitor the existence of the web service on VM01. If the web server crash, Heartbeat tries to restart the service. Unsuccessful restart can initiate a failover which will lead to the creation of VM01 on node 1. The version 2 of Heartbeat can monitor service [11]. In this scenario, there are two other events that can trigger the creation of VM01 on node 1: the failure 3.1 of VM01 or node 0.

6.1.5 Scenario 3

The objective of this scenarion is to test active/active HA of a web service. In Figure 3, the Heartbeat

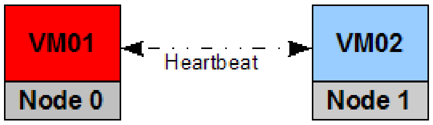

Figure 6.3: Heartbeat communication between VM01 and VM02.

communication is between the two virtual machines, and it is configured to monitor web service on VM01. VM02 is on standby but not running web service. If the web service on VM01, VM01 or node 0 fails 3.1, the IP address of VM01 moves to VM02 and the web server is started.

6.1.6 Results

The test is basically to test the availability of a service after an uncontrol failure 3.1 which is the key to HA. During failover in scenarios 1 and 2, we access the same web service with a downtime of 6-8 seconds. This is due to the fact that a virtual machine was created and the web server started. In scenario 3, the downtime is 1-2 seconds. In scenario 3 only the web server is started which explains the reduction in time. On a busy server which takes some delay to load a page, the downtime would not be noticed. Also we do not use dedicated path for Heartbeat, which should be used in production environment. If a dedicated path was used, it would have reduced the downtime.

Chapter 7

Future work

As stated in the design requirements; future work can be based on adding more functionality to the system design. Research can be done on the following concepts:

- Monitoring software that is aware of the cluster level of organization and can show resource (hardware) use and show the location of virtual machines
- Balancing resource allocations by migrating running virtual XEN nodes
- An improved scheduling system to manage booting virtual XEN cluster nodes.
- Performance measurements

Chapter 8

Conclusion

Our assignment was to research the possibilities for High Availability (HA) failover mechanisms using the XEN virtualization technology and the requirements necessary for implementation on technical level. Our tests showed that it is possible to use XEN in combination with Hearbeat to provide a HA environment. Even though combining XEN virtualization technology and High Availability software is still in the beginning stages at this moment, our research showed that XEN can be used with Heartbeat to realize a flexible, reliable and efficient HA environment. This is done by adding extra functionality in the cluster manager (Heartbeat) 5.1.

Bibliography

[1] Mark F. Mergen, Volmar Uulig, Olilig Orran Kriegen, Jim Xenidis. *Virtualization for High-Performance Computing.* IBM T.J Watson Research Center Yorktown Heights, NY 10603.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris,Alex Ho, Rolf Neugebauer,Ian Pratt, Andrew Warfiel. *XEN and the Art of Virtualization.* ACM Symposium on Operating Systems Principles. In Proceedings of the 19th ACM SOSP, pages 164177, October 2003.

[3] IA-32 Intel Architecture Software Developers Manual, Volume 1: Basic Architecture, section 4.5 (privilege levels)

[4] Developer Manual (Xen interfaces) The XEN Team. Xen Interface Manual, Xen v3.0 for x86. University of Cambridge, UK

[5] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen,Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield. *Live Migration of Virtual Machines.* Published at NSDI 2005.

[6] Franco Travostinoa, Paul Daspitb, Leon Gommansc, Chetan Joga, Cees de Laatc, Joe Mambrettid, Inder Mongaa, Bas van Oudenaardec, Satish Raghunatha, Phil Yonghui Wange. *Seamless live migration of virtual machines over the MAN/WAN.* Future Generation Computer Systems Volume 22, Issue 8 , October 2006, Pages 901-907.

[7] Stephen Alan Herrod, The future of virtualization technology, 2006

[8] Jeff Dijk. User Mode Linux.

[9] Alan Robertson. Linux-HA release 2. Linux.conf.au 2007 conference in Sydney, Australia 17 January 2007.

[10] Heartbeat The Linux High Availability project.

[11] Bryan Clark, Todd Deshane, Eli Dow, Stephan Evanchik, Mathew Finlayson, Jason Herne, Jeanna Neete Matthews. *XEN and the Art of Repeated Research* In Proceedings of the FREENIX Tracks: USENIX Annual Technical Conference. Bosten, MA, USA. June 27-July 2, 2004.

[12] Keir Fraser, Steven Hand, Rolf Neugebauer, Ian Pratt, Andrew Warfield, Mark Williamson. *Safe Hardware Access with the Xen Virtual Machine Monitor.* 1st Workshop on Operating System and Architectural Support and Implementation, 2004. Published at the OASIS ASPLOS 2004 workshop.

[13] A Robertson - Proceedings of the 4th Annual Linux Showcase and Conference. *Linux-HA Heartbeat System Design.* SuSE Labs.

[14] Linux World 2005 Virtualization BOF Presentation. gives an overview of Xen 2.0, live migration, and the roadmap for Xen 3.0 and beyond.

[15] Xen Summit 3.0 Status Report, Cambridge April 2005. Description and status report on the Xen 3.0 development.

[16] Espen Braastad, *Management of High Availability Services Using Virtualization.* Masters Thesis, May 22, 2006. University of Oslo.

[17] Kyrre Begnum, Karst Koymans, Arjen Krap, John Sechres. *Using Virtual Machines in System and Network Administration Education.* UNENIX/SANE Conference 2004, 2004.

[18] Vuk Marojevic, Marisa Gil. *Evaluating Hyperthreading and VMware.* University of Catalonia, Spain.

Chapter 9

Appendixes

9.1 Appendix A XEN Installation Procedures

This section describes how to install XEN under Ubuntu Edgy. Several packages need to be installed using the package management system:

```
# sudo apt-get install bridge-utils
```

Contains utilities for configuring the Linux ethernet bridge which can be used for connecting multiple ethernet devices together. You also need the following:

```
# apt-get install iproute bridge-utils python-twisted gcc-3.3 binutils make libcurl3-dev
zlib1g-dev python-dev transfig bzip2 screen ssh debootstrap libcurl3-dev libncurses5-d
python-xen3.0 xen-doc-2.6.17 xen-docs-3.0 xen-hypervisor-3.0-i386
xen-image-xen0-2.6.17-6-generic-xen0 xen-ioemu-3.0
xen-restricted-modules-2.6.17-6-generic-xen0 xen-tools xen-utils-3.0
```

The XEN kernel (xen-image-xen0-2.6.17-6-generic-xen0) is used for both the host OS as the virtual machines. XEN tools and utils provide commandline utilities to create, destroy, migrate, suspend and resume virtual machines. Next you have to update your grub configuration. Add to /boot/grub/menu.lst:

```
title           XEN/2.6.17
root            (hd0,1)
kernel          /boot/xen-3.0-i386.gz
module          /boot/xen0-linux-2.6.17-6-generic-xen0 root=/dev/sda2 ro quiet
module          /boot/xen0-linux-2.6.17-6-generic-xen0.initrd.img
```

After this you must create a 'initrd' image to mount the root filesystem:

```
mkinitramfs -o /boot/xen0-linux-2.6.17-6-generic-xen0.initrd.img 2.6.17-6-generic-xen0
```

Boot your XEN kernel!!

After rebooting; check if the system works (you should see dom0) with the command:

```
xm list
```

And check if your system supports VT (hardware based Virtual Technology). This can be done by checking if the 'vmx' flag is present in 'cat /proc/cpuinfo'. It must also be enabled: 'xm dmesg — grep VMX' You need images of virtual machines to boot; we create our own images with deboorstrap: Images for virtual machines (5GB and 500MB):

```
dd if=/dev/zero of=/vm/vm_base.img bs=1024k count=5000
dd if=/dev/zero of=/vm/vm_base-swap.img bs=1024k count=500
mkfs.ext3 /vm/vm_base.img
mkswap /vm/vm_base-swap.img
```

9.1.1 Install a basic edgy system in our loopback image

```
mount -o loop /vm/vm_base.img /mnt/vm
debootstrap edgy /mnt/vm
cp -a /lib/modules/2.6.17-6-generic-xen0/ /mnt/vm/lib/modules
```

Configure your basic install

```
mount -t sysfs none /mnt/vm/sys
mount -t proc /proc /mnt/vm/proc
chroot /mnt/vm
```

Setup /etc/apt/sources.lst, setup /etc/fstab

```
/dev/hda1               /               ext3    defaults        1       2
/dev/hda2               none            swap    sw             0       0
/dev/pts                devpts          gid=5,mode=620    0       0
none                    /dev/shm        tmpfs   defaults     0       0

apt-get install localeconf
```

Setup /etc/network/interfaces

```
auto lo
iface lo inet loopback
address 127.0.0.1
netmask 255.0.0.0

auto eth0
iface eth0 inet static
address 145.92.25.202
netmask 255.255.255.0
broadcast 145.92.25.255
gateway 145.92.25.1
```

Setup /etc/hosts

```
127.0.0.1       localhost.localdomain   localhost

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Exit, umount /mnt/vm

Make a basic configuration file for you first virtual machine:

```
/etc/xen/vm01.cfg
kernel = "/boot/xen0-linux-2.6.17-6-generic-xen0"
ramdisk = "/boot/xen0-linux-2.6.17-6-generic-xen0.initrd.img"
builder='linux'
memory = 128
name = "vm01"
vif = [ 'bridge=xenbr0' ]
disk = [ 'file:/vm/vm_base.img,ioemu:hda1,w','file:/vm/vm_base-swap.img,hda2,w' ]
root = "/dev/hda1 ro"
```

Start you virtual machine at boot time:

```
ln -s /etc/xen/vm01.cfg /etc/xen/auto/vm01
```

Start your virtual machine with:

```
xm create -c /etc/xen/vm01.cfg
```

Shutdown a domain (shutdown immediately) with the following command:

```
xm shutdown vm01
```

Dispatch virtual machine with: control+] (telnet)

9.1.2 Setup XEN Networking

Enable IPV4 forwarding

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

You can use your dedicated ip range for your virtual machines. You can give all your virtual machines internet access by bridging the connections to the gateway on dom0. Enable bridging in /etc/xen/xend-config.sxp: Change (network-script network-dummy) in (network-script network-bridge)

```
##
# To bridge network traffic, like this:
#
# dom0: fake eth0 -> vif0.0 -+
#                            |
#                          bridge -> real eth0 -> the network
#                            |
# domU: fake eth0 -> vifN.0 -+
#
# use
#
# (network-script network-bridge)
#
# Your default ethernet device is used as the outgoing interface, by default.
# To use a different one (e.g. eth1) use
#
# (network-script 'network-bridge netdev=eth1')
#
# The bridge is named xenbr0, by default.  To rename the bridge, use
#
# (network-script 'network-bridge bridge=<name>')
#
# It is possible to use the network-bridge script in more complicated
# scenarios, such as having two outgoing interfaces, with two bridges, and
# two fake interfaces per guest domain.  To do things like this, write
# yourself a wrapper script, and call network-bridge from it, as appropriate.
#
(network-script network-bridge)
```

The network range on dom0 does not cover the dedicated ip range: Fix this temporarily with: (Note don't use my ip range; this is just a example)

```
ip addr add 145.92.25.200/29 dev eth0
```

9.2 Appendix B Heartbeat Configuration

9.2.1 /etc/ha.d/ha.cf

```
#ha.cf
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
# keepalive: how long between heartbeats? in seconds
keepalive 2
# deadtime: how long-to-declare-host-dead? in seconds
deadtime 30
# Very first dead time (initdead)
initdead 120
#       What UDP port to use for bcast/ucast communication?
udpport 694
bcast   eth0            # Linux
ping 145.92.25.203
auto_failback on
node    vm01
node    vm02
crm no
respawn hacluster /usr/lib/heartbeat/ipfail
debug 1
```

9.2.2 /etc/ha.d/haresources

```
#haresources
node-name resource1 resource2 ... resourceN
vm01 145.92.25.204 apache2
```

9.2.3 Script to generate SHA-1 encryption

```
!/bin/bash
#
# Idea and code due to Sean Reifschneider <jafo@tummy.com>
# Both somewhat mangled by alanr@unix.sh
#
#       RANDMAX is the upper bound (in 512-byte blocks) on how much
#       randomness to consume...
#
#       It may be the case that a single block is enough, since it's
#       supposedly quite random data, and all we need is a 32 byte
#       checksum string out of it...
#
RANDMAX=3 # upper bound (in blocks) on how much randomness to consume...
RDEV=/dev/urandom
AUTH=/etc/ha.d/authkeys
RAND=${RANDOM}

#       Make sure ${RANDOM} is supported
case $RAND in
  ?*)   ;;
  *)    RAND=50;;
esac
```

```
#
#       Figure out how many blocks to checksum
#
CNT=‘expr \( $RAND % $RANDMAX \) + 1‘

if
  [ ! -c $RDEV ]
then
    echo "Random device $RDEV not supported on this OS"
    exit 1
fi


umask 077
touch $AUTH 2>/dev/null
chmod 600 $AUTH

cat <<-! >$AUTH
        # Key automatically generated by $0
        auth 1
        1 sha1 ‘dd if=$RDEV count=$CNT 2>/dev/null | md5sum | cut -c1-32‘
!
```

Output: /etc/ha.d/authkeys

```
Key automatically generated by ./generatekeys
auth 1
1 sha1 adf432eb9ea42f3584fe48ba46563787
```

9.2.4 Generating CIB XML from haresources:

```
python /usr/lib/heartbeat/haresources2cib.py /etc/ha.d/haresources >
/var/lib/heartbeat/crm/cib.xml
```