High availability clustering of virtual machines - possibilities and pitfalls

Paper for the talk at the 12th Linuxtag, May 3rd-6th, 2006, Wiesbaden/Germany
Version 1.01

**Werner Fischer (wfischer(at)thomas-krenn.com) and
Christoph Mitasch (cmitasch(at)thomas-krenn.com)**

Virtualization techniques are getting more and more popular. They allow to run multiple virtual servers on a single physical machine. But in case of a hardware outage, tens of virtual servers are down. This paper gives some insight how to prevent this situation. “Problems of virtualization” discusses why it is necessary to think about clustering in virtualized environments. Then, “Types of virtualization” explains characteristics of different virtualization techniques and shows some examples of virtualization. “HA cluster configurations in virtual environments” looks into different kinds of possible cluster configurations, their respective advantages and drawbacks, and possible pitfalls. Finally “Examples” shows some concrete examples of HA (high availability) clusters in virtual environments.

Problems of virtualization

Without virtualization, an outage of a single physical machine leads to the outage of a single server. But if a physical machine uses virtualization technologies, and tens of virtual servers are running on that single physical machine, an outage of this machine causes much more trouble. With a single outage tens of virtual servers are down. So taking precautions to prevent such situations is critical in a virtualized environment. By combining virtualization and HA clustering, it is possible to benefit from increased manageability and savings from server consolidation through virtualization without decreasing uptime of critical services [1].

Types of virtualization

Virtualization techniques are getting more and more popular. They allow to run multiple virtual servers on a single physical machine. We refer to this virtual servers as virtual machines. For the purpose of this paper, we distinguish between three virtualization techniques:

- Hardware-Virtualization (e.g. QEMU [2], VMware [3])

- Para-Virtualization (e.g. Xen [4], Denali [5], User Mode Linux [6])
- OS-Virtualization (e.g. OpenVZ [7], Linux-Vserver [8], Sun Solaris Containers [9])

This definition is not suitable for all discussions about virtualization. Also some overlapping between the three virtualization techniques is possible. Another possible definition of virtualization techniques in Linux is shown in [10]. A detailed survey of virtualization techniques can be found in [11].

The three techniques shown above differ not only in the kind of virtualization, but also in the provided features. Virtualization is used for different reasons, and necessary features vary depending on the purpose of an virtualized environment. All techniques have different pros and cons, also in respect of different kinds of clustering. We look at these three techniques more detailed to be able to evaluate the three virtualization techniques for the use in different kinds of cluster configurations.

Hardware-Virtualization

Hardware-virtualization provides virtual hardware to the virtual machines. It allows to run guest operating systems in the virtual machines without any modifications. The guest OSs are not aware that they are actually not running on a physical computer – the virtual hardware seems like real hardware to them.

Virtual hardware is provided through emulating this hardware with the help of a virtual machine monitor (VMM). This VMM needs real resources from the physical server, and can be implemented in two ways. It can be either host-based, or not host-based. In the first case a host OS, host specific drivers and some user-space code are necessary for virtual machines. Examples are QEMU or VMware Workstation and VMware Server. In the other case the VMM is not host-based. In [10] such a VMM is called a pure virtual machine monitor. VMware ESX Server is an example for this. With this method, some of the overhead that is necessary when running on top of a host OS can be removed, and therefore performance within the virtual machine can be increased. On the other side, such pure VMMs themselves must include more hardware support than host-based VMMs have to (see [10] p. 587).

Para-Virtualization

Emulating hardware like in hardware-virtualization requires some additional amount of resources. To avoid this drawback, some virtualization solutions provide a virtual machine abstraction layer which is only similar to the underlying hardware – but not identical. This approach is known as para-virtualization [12], and promises improved performance compared to full hardware-virtualization. As the abstraction is only similar to the underlying

hardware, modifications to the guest operating systems are necessary. In this way, the guest operating systems work co-operatively with the VMM (which is often called hypervisor in this context). This has some advantages. For example, the hypervisor can tell the guest that real time has passed between its last run, and its present run, permitting it to make smarter re-scheduling decisions to appropriately respond to a rapidly changing environment [13].

Examples of para-virtualization are Xen and Denali. Also User Mode Linux can be seen as some kind of para-virtualization. For Xen, the paper [14] outlines that although the OS must be modified for the guest, no changes to the application binary interface (ABI) are necessary and hence no modifications are required to guest applications. The paper shows some performance tests shown, but it has to be kept in mind that they reflect program versions as of 2003.

Like with hardware-virtualization, also in a para-virtualized environment each virtual machine runs its own kernel, has it's own dedicated memory, and it's own disk space.

OS-Virtualization

The third technique is OS-virtualization. Examples for OS-virtualization are OpenVZ, Linux-Vserver, and Sun Solaris Containers. The main difference compared to hardware-virtualization and para-virtualization is that with OS-virtualization only one single kernel runs on a physical server. This single kernel is used by both the host operating system itself and the guest operating systems. The idea behind that design is to avoid processing data twice as it is often necessary when a guest is running its own kernel.

In [15], Herbert Pötzl, maintainer of Linux-Vserver gives an example what happens when using multiple kernels: “a syscall to read a file is first processed by the guest kernel, to be then handed upwards and result in an actual I/O by the host kernel, which in turn has to hand back the data to the guest kernel before it reaches the process.” There are also other benefits with this technique. First, virtual machines do not require a dedicated amount of RAM. Although a minimal amount of resources, like the amount of guaranteed available RAM can be configured, other virtual machines can use these resources if a virtual machine does not currently need them itself. In this way, it does not happen that a virtual machine runs out of RAM, while another virtual machine only uses 10 MB of 256 MB dedicated RAM. Second, RAM consumption can be further decreased by loading shared libraries only once for multiple virtual machines into the memory. And third, virtual machines can be started in seconds, as they use the same kernel as the host, which is already running.

When using OS-virtualization only a single kernel runs for both the host and the guest systems. This technique lacks the possibility to run different operating system types in the virtual machines. But it is still possible to run different Linux distributions in the guests, as long as they work with the virtualized

kernel and the host itself runs on Linux.

HA cluster configurations in virtual environments

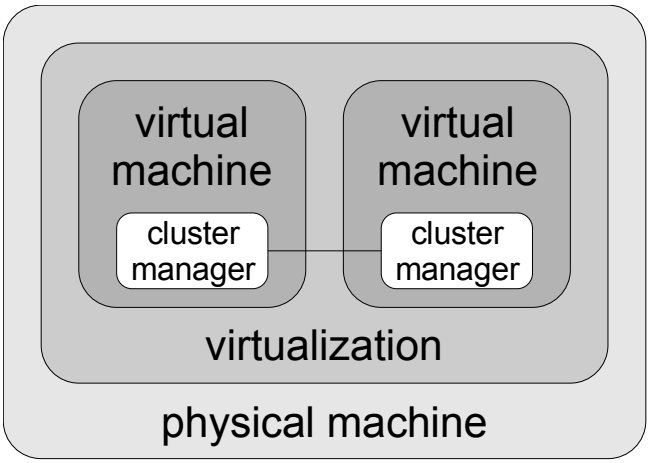
The majority of today's high availability clusters is based on real physical hardware. Because of the fact that virtualization is coming more and more popular nowadays, one has to think about possible combinations of virtualization and high availability. Until now only very few implementations and literature is available about that topic but this is expected to rapidly emerge in the near future.

The following scenarios (see also [16]) are shown below:

- virtual/virtual, single physical machine
- virtual/virtual, multiple physical machines
- physical/virtual
- virtual/physical
- physical/physical with fail-/switchover of virtual machines

Virtual/virtual, single physical machine

This setup builds an HA cluster between two or more virtual machines on a single physical machine. The cluster manager itself is running directly in the virtual machines. The physical server only hosts the virtual machines, like shown in the figure below.



This setup does not protect against hardware failures. So the physical machine itself is a SPOF (single point of failure). Other advantages normally offered by

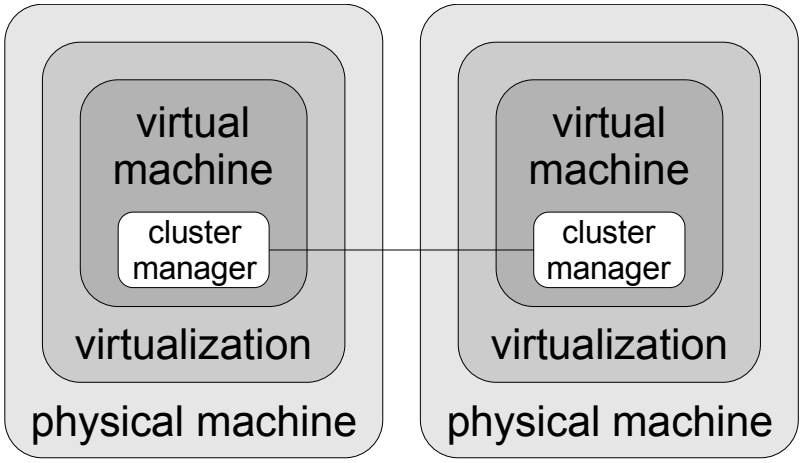
HA clusters like planned outage for maintenance purposes also get lost.

The remaining advantages mainly depend on the used virtualization type. If hardware- or para-virtualization is used this setup brings additional protection since separate instances of the operating system are running in the virtual machines. It protects against a kernel panic in a virtual machine for example. When using OS-virtualization the virtual machines on one physical machine are effectively all running the same kernel and therefore are not protected against a kernel panic. The ability of a cluster manager to monitor applications and do local recoveries is not negatively affected by this setup.

As a conclusion this scenario is mainly useful for test environments. It is a very comfortable way to test different HA cluster setups without requiring much hardware. The paper [17] discusses the use of User Mode Linux for testing purposes.

Virtual/virtual, multiple physical machines

In this scenario an HA cluster is built between two or more virtual machines, each of them running on different physical machines. The SPOF of a single physical machine is eliminated. Again, the cluster manager is running directly in the virtual machine. The figure below shows this setup.

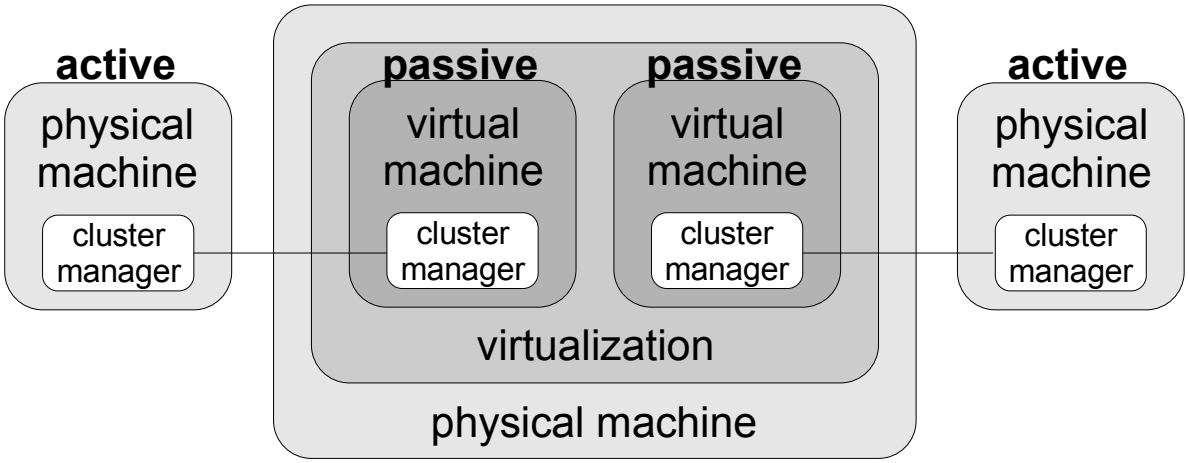


The main advantage of this setup lies in the area of easier management of the cluster nodes. Since the cluster is running in a virtual machine it can be easily moved to other physical machines. Moreover backup and recovery procedures are traditionally simplified through the use of virtualization. However it is also necessary that the physical server and its operating system have to be included in backup, recovery and security update procedures as well.

Another advantage of this setup is the possibility to run multiple instances of an application that normally is not able to run in one single OS instance. This is possible when using multiple virtual machines with one application instance each running on one physical machine.

Physical/virtual

Here an HA cluster is established between a physical machine (normally active node) and a virtual machine (normally passive node). The cluster manager is on the one side running directly on the physical machine and on the other side running in the virtual machine. Let's call the physical machine running the virtual machines for standby nodes “passive node appliance”. The figure below shows the setup.



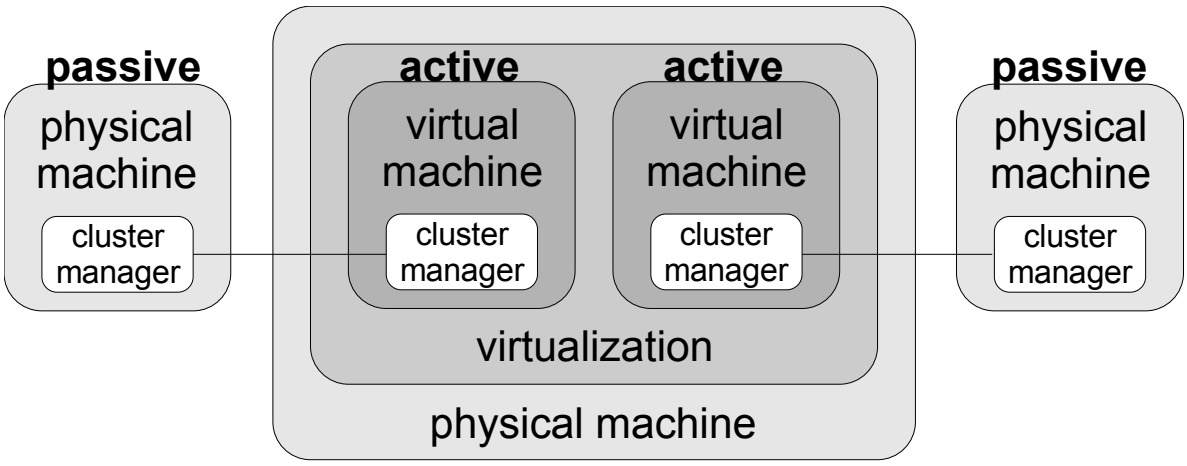
The advantage of this setup is the possibility to combine multiple standby nodes on one physical machine whereas normally one would need a second physical machine for each standby node. This offers great potential for cost savings.

Imagine a company uses ten different active/passive HA cluster systems. When the standby node is virtualized some form of overbooking for the standby nodes can be done. Indeed it seems to be quite unlikely that all ten HA clusters would ever need their standby nodes at the same time. However in case of fail-/switchover of multiple clusters to the “passive node appliance” this physical server may run out of resources. So appropriate sizing is very important.

Another advantage is the possibility to run different Linux distributions or even operating systems on the “passive node appliance”. Disadvantageous is the necessity to use different mechanisms for backup, recovery and security updates for the physical machine and the “passive node appliance”.

Virtual/physical

This scenario is the counterpart to physical/virtual. In theory it is quite similar except that the active and passive roles are interchanged. So the cluster manager for the active node is normally running in the virtual machine, where the cluster manager for the passive node is running on a physical machine. Let's call the physical machine running the virtual machines “active node appliance”.

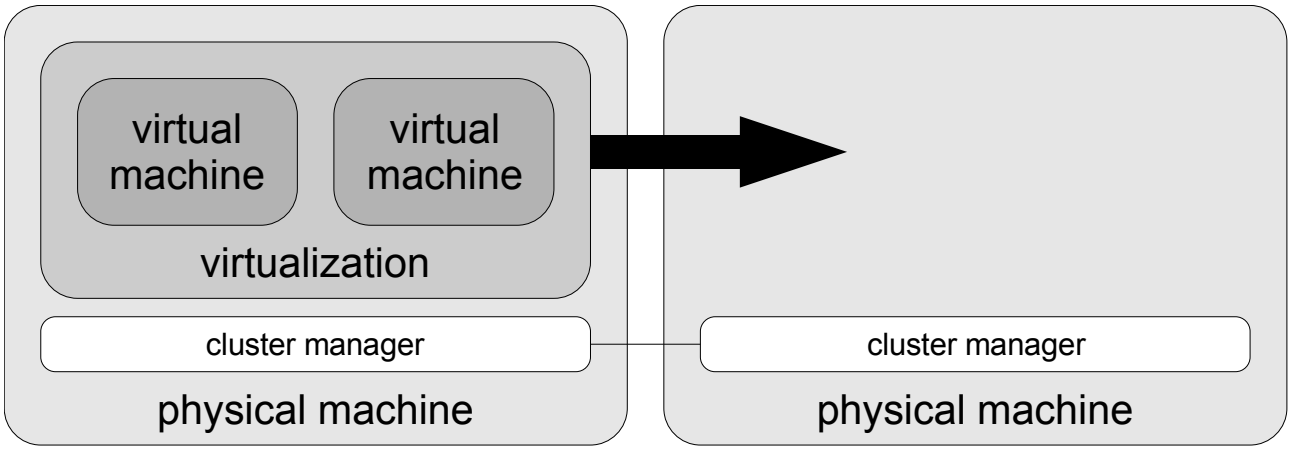


In that case the “active node appliance” has to be sized as large to be able to run all virtual machines at the same time. A reason for the deployment of that scenario would be if someone likes to use a very powerful and redundant hardware during normal operations for the “active node appliance”. For the standby nodes each running on a dedicated physical machine, cheaper and less powerful hardware could be used.

However it is questionable if cheap and less redundant hardware should be used for HA clustering at all.

Physical/physical with fail-/switchover of virtual machines

In contrast to the previously discussed scenarios, in this one the cluster manager is always directly running on the physical machine. So the HA cluster is built between two or more physical machines just like in traditional setups. The difference is that applications are not running directly on the physical machine but instead running in one or more virtual machines. Virtual machines themselves are configured as a resource in the cluster and are managed by the cluster manager. In case of a fail-/switchover the whole virtual machine is failed over to the standby node, as shown in the figure below.



This setup offers some reasonable advantages. One advantage is the simplification of the cluster configuration. Since the cluster manager is only responsible for controlling the virtualization software, the configuration is kept simple and straightforward. No complex resource hierarchies and no complicated application integration procedures like sharing or synchronization of configuration files are needed. One big advantage is the fact that applications are running in a single virtual machine. Therefore managing clustered applications is as easy as managing applications on a single virtual or physical machine. The virtual machine itself does not even know anything about the fact that it is actually clustered.

Another advantage is the possibility to cluster multiple virtual machines with only one cluster manager. Instead of running multiple clusters, multiple virtual machines can be consolidated on a single cluster.

A disadvantage is that a fail-/switchover can take longer than in a traditional cluster. This is because it is not sufficient to relocate the application only in the cluster but moreover the whole virtual machine has to be started on the other cluster node. Therefore a whole operating system has to be started which always takes longer than just starting an application. The time the virtual machine takes to startup is significantly affected by the virtualization technology used. When using hardware- or para-virtualization, the startup procedure normally takes longer since usually the OS kernel has to be started and in addition some form of virtual hardware always has to be initialized. If OS virtualization is used instead, the overhead for starting the kernel and for initializing the hardware is eliminated or significantly reduced. In that case the overhead compared to starting a single application is often negligible.

Possible Pitfalls

Today's HA cluster solutions have different requirements for hardware, software, network and storage. HA cluster managers have not been built to run in virtual environments. This has to be considered when using HA clustering in combination with virtualization.

It becomes clear when thinking about shared resource protection. According to [18] the following mechanisms exist:

- Lock Manager
    - SLM (Single Lock Manager)
    - RLM (Redundant Lock Manager)
    - DLM (Distributed Lock Manager)
- Quorum

        - Quorum Disk
        - Real Quorum of Servers
        - Tie-Breaker Mechanisms
    - Fencing Mechanisms
        - Resource Fencing
        - System Reset/Node Fencing

Depending on the shared resource protection mechanism used, different problems can be expected in virtual environments with that mechanisms.

Resource fencing guarantees an exclusive use of a shared resource. Many cluster managers use SCSI-2 reservations or SCSI-3 persistent reservations to assure this exclusive access for SCSI disks. These mechanisms can be already critical and error-prone in traditional environments as mentioned in [19]. Especially when using virtual/physical or vice versa setups, SCSI reservations can cause troubles. If used at all in a virtualized environment, intensive testing is necessary.

Node fencing excludes a complete physical machine from the cluster, usually by power cycling the cluster node. This is often called STONITH (Shoot The Other Node In The Head) as mentioned in [20]. When using node fencing in a virtualezed environment one has to avoid situations where multiple “virtual” cluster nodes are excluded by power cycling a single physical machine.

In general one should always remember the key HA principle: simplicity (see [21] p. 101). The more complex an HA cluster gets, the more potential failures can occur. Indeed using virtualization itself adds complexity and therefore potential for additional failures. A short example: device drivers. When using hardware virtualization often two device drivers are needed, called double device driver model. One in the host system accessing the real hardware, and one in the virtual machine accessing the host's hardware through a virtual device driver. Therefore errors can occur in two different device drivers. At this point, it is noticeable that [22] has shown that device drivers have error rates up to three to seven times higher than the rest of the kernel. When using para-virtualization the split device driver model is often used. In that case again multiple drivers have to cooperate in a smooth and reliable way.

Future

Combining virtualization and HA clusters is just in its beginning stages at the moment. Until now only very few ready-to-run solutions exist.

Real-time migration is currently becoming increasingly popular for virtualization

technologies. It allows to freeze a virtual machine at a particular time and restore it on another physical machine while preserving memory, processes and network connections. In best case the client will not notice any outages during migration.

This feature can be useful for switchovers in HA environment since the outage normally caused by starting and stopping virtual machines can be minimized. Moreover the use of real-time migration in case of failover in HA environment could be an interesting approach too.

Examples

Below a cluster with OpenVZ, Heartbeat [23], and DRBD [24] is shown, as well as some references to other cluster examples.

Cluster with OpenVZ, Heartbeat, and DRBD

It possible to build a physical/physical scenario with fail-/switchover of virtual machines with the OS-virtualization OpenVZ, the cluster manager Heartbeat and the data replication DRBD (Distributed Replicated Block Device).

Let's assume a two node cluster with Heartbeat installed directly on both nodes' OS on the physical machines. OpenVZ is able to start and stop all currently configured virtual machines, called VPSs (Virtual Private Servers), with a single init script called /etc/init.d/vz. This init script is configured as a cluster resource in Heartbeat. As a result in case of a switchover Heartbeat simply stops all VPSs on one node and starts them again on the other node.

DRBD is used to share data and configuration files for the VPSs and is also configured as a resource in Heartbeat. At least the following directories have to be shared when using OpenVZ:

- /vz – contains the VPSs
- /etc/sysconfig/vz-scripts/, /etc/sysconfig/vz – configuration files for VPSs
- /var/vzquota – quota information for VPSs

It is not necessary to configure the IP addresses of the VPSs as a resource in the cluster manager. When a VPS is started the command /usr/sbin/arpsend is called by OpenVZ and sends out a gratuitous arp. By default only one gratuitous arp is sent out, but this value can be increased in the script /usr/lib/vzctl/scripts/vps-functions.

Other cluster examples

There are also other resources available that describe concrete combinations of HA clustering and virtualization:

- [25] describes how to setup a cluster with Linux-Vserver, Heartbeat and DRBD.
- In [26] it is mentioned how the RedHat Cluster Suite can be used to run inside a Xen Guest. The cluster filesystem GFS is used this example.
- In a posting [27] on the Linux-HA mailing list a Heartbeat 2.X resource agent for Xen guests has been published.

Bibliography

[1] James Bottomley. The Risks of Over-Virtualization - why virtualization is not high availability. *LinuxWorld Magazine*, February, 27th 2006. http://linux.sys-con.com/read/183014.htm

[2] http://fabrice.bellard.free.fr/qemu/

[3] http://www.vmware.com/

[4] http://www.cl.cam.ac.uk/Research/SRG/netos/xen/

[5] http://denali.cs.washington.edu/

[6] http://user-mode-linux.sourceforge.net/

[7] http://openvz.org/

[8] http://linux-vserver.org/

[9] http://www.sun.com/software/solaris/ds/utilization.jsp

[10] Chris Wright. Virtually Linux - Virtualization Techniques in Linux. In *Proceedings of the Linux Symposium*, Ottawa, Ontario, Canada, July 21th-24th 2004. http://www.linuxsymposium.org/proceedings/reprints/Reprint-Wright-OLS2004.pdf

[11] Nadir Kiyanclar. A Survey of Virtualization Techniques - Focusing on Secure On-Demand Cluster Computing. University of Illinois at Urbana-Champaign, National Center for Supercomputing Applications, November 2nd 2005. http://citeseer.ist.psu.edu/733373.html

[12] A. Whitaker and M. Shaw and S. Gribble. Denali: Lightweight virtual machines for distributed and networked applications. In *Proceedings of the*

USENIX Annual Technical Conference, Monterey, CA, June 2002. http://citeseer.ifi.unizh.ch/whitaker02denali.html

[13] Moshe Bar. Xen, the virtual machine monitor. *Free Software Magazine*, Issue 5, June 2005. http://www.freesoftwaremagazine.com/articles/focus-xen

[14] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebar, Ian Pratt and Andrew Warfield. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, New York, NY, USA, October 2003. http://www.cl.cam.ac.uk/Research/SRG/netos/papers/2003-xensosp.pdf

[15] Herbert Pötzl. Linux-VServer - Resource efficient context isolation. *Free Software Magazine*, Issue 5, June 2005. http://www.freesoftwaremagazine.com/articles/focus-linux_vserver

[16] LifeKeeper for Linux in Virtual Machine Environments. White Paper, SteelEye Technology, Inc., Palo Alto, CA, USA, 2005. http://licensing.steeleye.com/support/download_get.php/LKinVirtualEnvironments.pdf?dir=0&file=LKinVirtualEnvironments.pdf

[17] Armin M. Warda. Virtuelle Testumgebungen mit User Mode Linux. Postbank Systems AG, Oktober 2004. http://www.user-mode-linux.org/~jdike/armin/Virtuelles-UML-Labor.pdf

[18] Werner Fischer. Implementation of a Disaster Resilient Linux Cluster with Storage Subsystem Based Data Replication. Master's thesis, Fachhochschule Hagenberg, Computer- und Mediensicherheit, Hagenberg, Austria, September 2004. http://www.wefi.net/thesis/diploma_thesis_werner_fischer.pdf

[19] Alan Robertson. Resource fencing using STONITH. 2001.

[20] http://www.linux-ha.org/STONITH

[21] Evan Marcus and Hal Stern. *Blueprints for High Availability*. Wiley Publishing, Indianapolis, IN, USA, second edition, 2003.

[22] Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem, and Dawson Engler. An Empirical Study of Operating Systems Errors. Stanford University, 2001. http://www.stanford.edu/~engler/metrics-sosp-01.ps

[23] http://www.linux-ha.org/

[24] http://www.drbd.org/

[25] http://linux-vserver.org/Vserver+DRBD

[26] http://people.redhat.com/pcaulfie/docs/xencluster.html

[27] http://lists.linux-ha.org/pipermail/linux-ha/2006-March/018798.html