

Fachhochschule Gießen-Friedberg

■ Fachbereich Mathematik, Naturwissenschaften und Informatik

Seminararbeit
im Studienschwerpunkt Systemtechnik
zum Thema

Virtual Elastic Services

Ein Ausblick auf Virtualisierungskonzepte

| | |
|------------------|-----------------------------------|
| Eingereicht von: | Kevin Wennemuth |
| Email: | kevin.wennemuth@mni.fh-giessen.de |
| Matrikelnummer: | 712893 |
| Betreuer: | Prof. Dr. P. Kneisel |
| Erstellt am: | 9. März 2007 |

Eingereicht im Wintersemester 2006/2007
im Fachbereich Mathematik, Naturwissenschaften und Informatik (MNI)
der Fachhochschule Gießen-Friedberg

Wiesenstraße 14
D-35390 Gießen

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einleitung | 1 |
| 1.1 | Widmung | 1 |
| 1.2 | Motivation | 1 |
| 2 | Begriffsklärung | 3 |
| 2.1 | Virtualisierungsmethoden | 3 |
| 2.2 | Xen virtual machine monitor | 5 |
| 2.2.1 | Aufbau und Funktionsweise | 5 |
| 2.2.2 | Effiziente Hardwarekonsolidierung | 6 |
| 2.2.3 | Hardware Hotplugging | 6 |
| 2.2.4 | Live-Migration | 6 |
| 2.2.5 | Softwareverfügbarkeit | 7 |
| 2.2.6 | Xen Community | 7 |
| 2.2.7 | Übersicht | 7 |
| 2.3 | ATA over Ethernet | 7 |
| 2.4 | Distributed Replicated Block Device (DRBD) | 9 |
| 2.5 | Oracle Cluster File System (OCFS2) | 9 |
| 2.5.1 | Quorum | 10 |
| 2.5.2 | Fencing | 10 |
| 3 | Anwendungsgebiete | 11 |
| 3.1 | SSI-Hosting | 11 |
| 3.1.1 | Entwicklungssysteme | 12 |
| 3.1.2 | Prototyping und Produktionssysteme | 13 |
| 3.2 | Virtual Clustering | 14 |
| 3.3 | Virtual Elastic Clustering | 14 |
| 4 | Ausflug in den Markt | 17 |
| 4.1 | Appliance Provider | 18 |
| 4.2 | Elastic Hosting | 18 |
| 4.2.1 | Amazon Elastic Compute Cloud (EC2) | 19 |
| 4.2.2 | ElasticLive | 19 |
| 4.3 | Projekt- und Entwicklungsframework ATLAS | 20 |
| 4.3.1 | ATLAS::TSI | 20 |
| 4.3.2 | Entwicklungsumgebung - IST-Stand | 20 |
| 4.3.3 | Serverlandschaft - IST-Stand | 21 |
| 4.3.4 | Ausfallzeiten | 21 |
| 5 | Anforderungen | 23 |
| 5.1 | Allgemeine Anforderungen | 23 |
| 5.1.1 | Aktive Community | 23 |
| 5.1.2 | Aktives BugTracking | 23 |
| 5.1.3 | Dokumentation | 24 |
| 5.1.4 | Einsatzszenarien | 24 |
| 5.1.5 | Lizenzierung | 24 |

| | | |
|----------|---|-----------|
| 5.2 | Anforderungen - Fehlerszenarien | 24 |
| 5.2.1 | Disaster recovery - System crash | 24 |
| 5.2.1.1 | Stromversorgung | 25 |
| 5.2.1.2 | Festplatten | 25 |
| 5.2.2 | Disaster recovery - Communication failure | 25 |
| 5.2.3 | Planned maintenance and shutdown | 25 |
| 5.3 | Anforderungen - Systeme | 25 |
| 5.3.1 | Skalierbarkeit | 26 |
| 5.3.2 | Betriebssystem | 26 |
| 5.3.3 | Dateisystem | 26 |
| 5.3.4 | Backup | 26 |
| 5.4 | Anforderungen - Services | 27 |
| 5.4.1 | Load Balancing und Failover | 27 |
| 5.4.2 | Firewall | 27 |
| 5.5 | Anforderungen - Überwachung | 27 |
| 6 | Projekt ATLAS | 29 |
| 6.1 | Konzeptioneller Aufbau | 29 |
| 6.2 | Ebene 1 - Physikalische Rechner | 29 |
| 6.2.1 | Hardware | 30 |
| 6.2.2 | Festplatten | 30 |
| 6.2.3 | Kommunikation | 30 |
| 6.3 | Ebene 2 - Datensicherung | 31 |
| 6.4 | Ebene 3 - Datenkonsistenz | 32 |
| 6.5 | Ebene 4 - Datenbereitstellung | 33 |
| 6.6 | Ebene 5 - Systemabstraktion | 35 |
| 6.6.1 | Verwaltungsschicht - dom0 | 35 |
| 6.7 | Ebene 6 - Virtuelle Maschinen | 36 |
| 6.7.1 | VM Firewall | 36 |
| 6.7.2 | VM Apache | 37 |
| 6.7.3 | VM Datenbank | 37 |
| 6.7.4 | VM Backup | 37 |
| 6.7.5 | VM Nagios | 37 |
| 6.8 | Ebene 7 - ATLAS Management | 38 |
| 6.8.1 | BixData Server Management Console | 38 |
| 6.8.2 | openQRM | 39 |
| 6.9 | Verbesserungen | 40 |
| 6.9.1 | Lustre statt OCFS2 | 40 |
| 6.9.2 | iSCSI statt ATA over Ethernet | 41 |
| 6.9.3 | Andere Betriebssysteme | 41 |
| 7 | Zusammenfassung | 43 |
| 7.1 | Fazit | 43 |
| 8 | Verwendete Projekte | 45 |

Abbildungsverzeichnis

| | | |
|-----|--|----|
| 2.1 | Native Virtualisierung | 3 |
| 2.2 | Xen™ Paravirtualisierung | 4 |
| 2.3 | Xen™ Hypervisor [Pra06] | 5 |
| 2.4 | Xen™ Live Migration[CC05] | 6 |
| 2.5 | ATA over Ethernet Header Spezifikation | 8 |
| 2.6 | ATA over Ethernet Protokollstack | 8 |
| 2.7 | DRBD im aktiv-passiv Modus | 9 |
| 2.8 | DRBD im aktiv-aktiv Modus | 10 |
| 3.1 | Single System Image Hosting | 11 |
| 3.2 | Exemplarische Entwicklungsumgebungen | 13 |
| 3.3 | Exemplarische Produktivumgebung | 13 |
| 3.4 | Virtual Clustering | 14 |
| 3.5 | Virtual Elastic Clustering | 15 |
| 4.1 | Computing cloud | 19 |
| 4.2 | IST-Stand ATLAS::TSI | 21 |
| 6.1 | ATLAS Systemebenen | 29 |
| 6.2 | BixData Aufbauschema | 38 |
| 6.3 | BixData GUI | 38 |
| 6.4 | BixData Reporting | 38 |
| 6.5 | openQRM | 39 |
| 6.6 | Lustre Dateisystem | 40 |
| 6.7 | iSCSI Schema | 41 |

Listings

| | | |
|------|--|----|
| 6.1 | Installation SMARTMonTools | 30 |
| 6.2 | DRBD: Installation | 31 |
| 6.3 | DRBD: copy | 31 |
| 6.4 | DRBD: Init | 31 |
| 6.5 | DRBD: /etc/conf.d/localstart | 32 |
| 6.6 | OCFS2: Kernelmodul | 32 |
| 6.7 | OCFS2: Kernel Autostart | 32 |
| 6.8 | OCFS2: /etc/fstab | 32 |
| 6.9 | OCFS2: mkfs.ocfs2 | 32 |
| 6.10 | OCFS2: /etc/ocfs2/cluster.conf | 33 |
| 6.11 | OCFS2: /etc/conf.d/ocfs2 | 33 |
| 6.12 | OCFS2: /etc/conf.d/localstart | 33 |
| 6.13 | OCFS2: Autostart | 33 |
| 6.14 | AoE: Kernel Modul | 34 |
| 6.15 | AoE: Autostart | 34 |
| 6.16 | aoetools: Installation | 34 |
| 6.17 | aoetools: aoe-stat | 34 |
| 6.18 | AoE: mount | 34 |
| 6.19 | AoE: /etc/fstab | 34 |
| 6.20 | /etc/make.conf | 35 |
| 6.21 | dom0: Kernel | 35 |
| 6.22 | domU: Kernel | 36 |
| 8.1 | Grundsystem dom0 | 45 |

1 Einleitung

1.1 Widmung





Ich widme diese Ausarbeitung meiner Freundin Nicole. Sie ist das Beste, was mir passieren konnte ...

„Viel Denken, nicht viel Wissen, ist zu pflegen.“
Demokritos von Abdera

1.2 Motivation

Seit Jahren beschäftige ich mich mit dem Thema *Clustering* mit mehr und mehr Begeisterung. Angefangen hat alles mit einem simplen Problem: *„Warum machen meine Rechner im Büro eigentlich nichts Sinnvolles, wenn niemand daran arbeitet?“* Zuerst war die Frage simpel, doch dann kamen andere wie: *„Warum kann ich nicht die Rechenleistung meiner Büronachbarn benutzen?“* Später kamen dann noch mehr Fragen, meist sehr schwierige, und mit schlaflosen Nächten verbundene Fragen.

Mittlerweile bewegen sich professionelle Clusterkonzepte nicht nur in IT Infrastrukturen von Forschungsbereichen oder Universitäten, sondern erhalten auch immer mehr Einzug in die wirtschaftlichen Bereiche wie Hosting oder in DataCentern. Selbst Branchenfremdlinge wie Amazon.com versuchen sich sehr erfolgreich an diesem neuen Markt, wie wir später noch sehen werden.

Die Clusterwelt ist hierbei nicht hochspezialisiert, wie viele denken mögen, sondern hat einen sehr breit gefächertes Repertoire an Projekten, Funktionen und Leistungen. Die Community ist meist noch in der Forschungswelt angesiedelt, doch man bemerkt eine langsame Verlagerung hin zum Allgemeingut. Projekte wie *openMosix*¹  oder *BeoWulf*²  sind wohl die angestammtesten Projekte der Clustering-Szene und spätestens seit der Veröffentlichung von *ClusterKnoppix*³  auch für einfache Anwender zu gebrauchen. Spezialisierte Clustering-Lösungen wie *openSSI*⁴  sind jedoch noch auf dem Weg und werden aktiv von einer großen Community unterstützt.



Die allseits belächelten *„Cluster-Freaks“* haben sich zu einer nicht zu verachteten Marktmacht gewandelt. Die Fortschritte, die im Bereich des klassischen Clustering momentan gemacht werden, sind hervorragend, und ebenso tauchen verstärkt Begriffe wie *„public computing“* und *„grid computing“* aus den Tiefen des Internet an die Oberfläche der normalen Alltagswahrnehmung.

¹openMosix: <http://openmosix.sourceforge.net>

²BeoWulf: <http://www.beowulf.org>

³ClusterKnoppix: <http://clusterknoppix.sw.be>

⁴openSSI: <http://openssi.org>

Hierbei werden nicht einzelne Rechner in einem kleinen Clusterverbund zusammengeschlossen, sondern vielmehr mehrere tausend Rechner über eine öffentlich zugängliche Infrastruktur für Aufgaben zur Verfügung gestellt. Ob es sich dabei um einzelne Rechner, ganze Cluster oder Supercomputer handelt, ist hierbei ohne Belang. Daher wird diese Technik auch gerne als *Meta-Clustering* bezeichnet. Jeder kann die Rechenleistung dieses distributiven Rechnernetzwerkes nutzen. Wenn Sie sich weiter in diese Richtung informieren wollen, kann ich Ihnen zwei in meinen Augen sehr interessante Projekte ans Herz legen: Das momentan von der Cambridge University entwickelte Projekt *XenoServer*⁵  und die Seiten von *The Globus Alliance*⁶  mit Ihrem „*Globus Toolkit*“ für Grid-Computing.

Ich möchte mich mit dieser Ausarbeitung dem Thema „*Virtual Elastic Services*“ widmen, um Mittel und Wege aufzuzeigen, allgemein bekannte und geprüfte Techniken aus dem OpenSource Bereich einzusetzen, um eine hochmoderne Infrastruktur für Hosting- bzw. Clustering-Lösungen abzubilden. Augenmerk bei der hier vorgestellten Lösung ist die Interdisziplinarität, da mehrere Ansätze aus Cluster-, Grid-, Virtualisierungs und Hosting-Lösungen verwendet werden, um ein gemeinsames Ganzes zu schaffen. Es gibt im Bereich des „*Elastic Hosting*“ bereits einige proprietäre und kommerzielle Produkte, die jedoch jenseits des Erreichbaren liegt. Diesen Lösungen werden wir uns in späteren Kapiteln noch ausführlicher widmen.

Den Weg, den ich versuche aufzuzeigen, kann man allein mit Arbeitszeit (ein paar „*alte*“ Rechner vorausgesetzt) begehen. Die benötigte Software ist kostenlos und kann aus frei zugänglichen Quellen bezogen werden. Stellen Sie sich also auf einige schlaflose Nächte ein.

⁵XenoServer: <http://www.xenoservers.net>

⁶The Globus Alliance: <http://www.globus.org>

2 Begriffsklärung

2.1 Virtualisierungsmethoden

Die „*Virtualisierung*“ in der Informatik bezeichnet Methoden, die es erlauben, Ressourcen eines Computers aufzuteilen. Das vorrangige Ziel ist die Abstraktion und Isolation von Ressourcen eines bestehenden Hardwaresystems. Bei der klassischen Virtualisierung bildet eine logische Schicht zwischen Host- und Gast-System eine Koordinationsinstanz, um dem Gast-System exklusiven Zugriff auf eine physikalische Ressource vorzuspielen oder diese gar ganz zu emulieren. Um diese Abstraktionsschicht zu realisieren, kann man zwei primäre Wege beschreiten:

- Native Virtualisierung:
Bei diesem Ansatz wird der komplette Rechner oder genauer die einzelnen Hardwarkomponenten bis hin zum Prozessor vollständig emuliert.

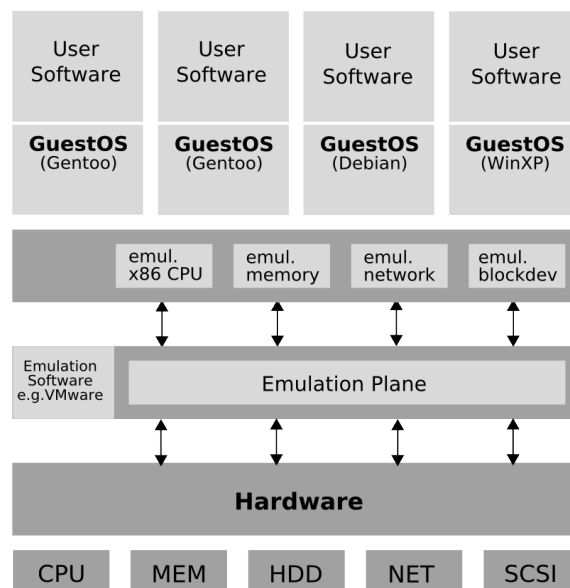


Abbildung 2.1: Native Virtualisierung





Zwar können durch diesen Ansatz flexible Systeme quasi „*on-the-fly*“ zusammengestellt werden, jedoch geht dies auf Kosten der Performance, da alle Befehle abfangen, entsprechend der wirklich darunterliegenden Hardware interpretiert und in beide Richtungen weitergereicht werden müssen.


Einige bedeutende Vertreter dieses Ansatzes sind:


- *QEMU*¹ 
- *Bochs*² 

¹QEMU: <http://fabrice.bellard.free.fr/qemu/>

²Bochs: <http://bochs.sourceforge.net/>

- *Multiple Emulator Super System (M.E.S.S.)* ³ 
- *MS Virtual PC* ⁴ 
- *VMWare* ⁵ 
- *VirtualBox* ⁶ 

Der derzeit bekannteste Vertreter dieser Methodik ist *VMWare* . *VMWare* beherrscht mittlerweile nicht nur die Virtualisierung von normalen Desktoprechner, sondern begibt sich immer mehr in alternative Gebiete, wie Server- bzw. Hosting-Infrastrukturen.

Als OpenSource-Vertreter ist seit Januar 2007 ein vielversprechendes Projekt hinzugekommen, das hier gesondert erwähnt werden sollte: *VirtualBox* ⁷ . *VirtualBox* geht einen sehr ähnlichen Weg wie *VMWare*, indem es ein komplettes System inklusive virtueller Festplatten zur Verfügung stellt. Anders als *VMWare* ist *VirtualBox* jedoch nicht für den Servereinsatz gedacht, sondern spricht eher den Markt der Desktopbesitzer an.

- Paravirtualisierung:
Der Begriff „*Paravirtualisierung*“ leitet sich von Begriff der „*Virtualisierung*“ ab und bezeichnet in unserem Zusammenhang die Ablösung der Betriebssystemschicht von der darunterliegenden Hardware durch Abstraktion in eine neue Vermittlungsschicht.

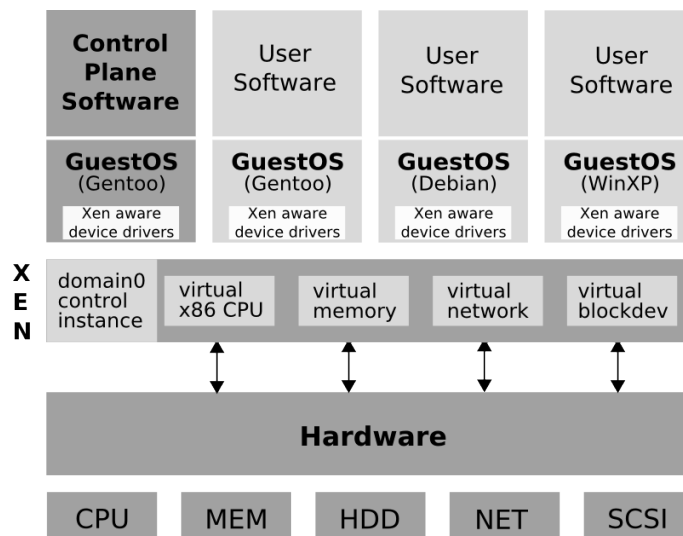



Abbildung 2.2: Xen™ Paravirtualisierung

Bei dieser Methode wird nicht die eigentliche Hardware für das darüberliegende Gast-System emuliert, sondern es wird eine Vermittlungsschicht, der so genannte „*Hypervisor*“ zwischen physikalischer Hardware und Betriebssystem des Gast-Systems etabliert. Bedeutendster Vertreter dieser Abstraktionsart ist derzeit *Xen™ virtual machine monitor* ⁸ .

³Multiple Emulator Super System (M.E.S.S.): <http://www.mess.org/>

⁴MS Virtual PC: <http://www.microsoft.com/germany/windows/virtualpc/>

⁵VMWare: <http://www.vmware.com/>

⁶VirtualBox: <http://www.virtualbox.org/>

⁷VirtualBox: <http://www.virtualbox.org/>

⁸Xen™ virtual machine monitor: <http://www.cl.cam.ac.uk/research/srg/netos/xen/>

2.2 Xen virtual machine monitor

Wie eingangs schon erwähnt, kommt in dieser Ausarbeitung hauptsächlich *XenTM 3.0* als Paravirtualisierungsebene zur Anwendung. *XenTM* beherrscht gegenüber anderen Virtualisierungsprojekten einige entscheidende Vorteile. Zunächst jedoch möchte ich den grundsätzlichen Aufbau von *XenTM* kurz erläutern.

2.2.1 Aufbau und Funktionsweise

Die Abstraktion bzw. Paravirtualisierung wird bei *XenTM* durch eine Kombination aus Host-Betriebssystem, der so genannte privilegierten Domäne (bei *XenTM*: dom0), und dem so genannte *Hypervisor* erreicht. Dieses Host-System dient als Verwaltungsschicht und stellt allen „virtuellen“ Gast-Systemen (bei *XenTM*: domU) einen gemeinsamen Ressourcenpool (Netzwerk, Festplatten, Benutzerein- und -ausgaben) zur Verfügung. Das Host-System koordiniert hierbei die Ressourcenzugriffe über den *Hypervisor* und so genannte *Hypercalls* [ea03] und gibt sie an die eigentlichen Treiber der Hardware weiter. Jedem Gast-Betriebssystem wird hierbei ein exklusiver Zugriff auf die einzelnen Komponenten vorgespielt. Da die Ressourcen nicht emuliert werden müssen, fällt bei dieser Variante der Flaschenhals der nativen Virtualisierung weg. Der Overhead [Pra06] dieser Form

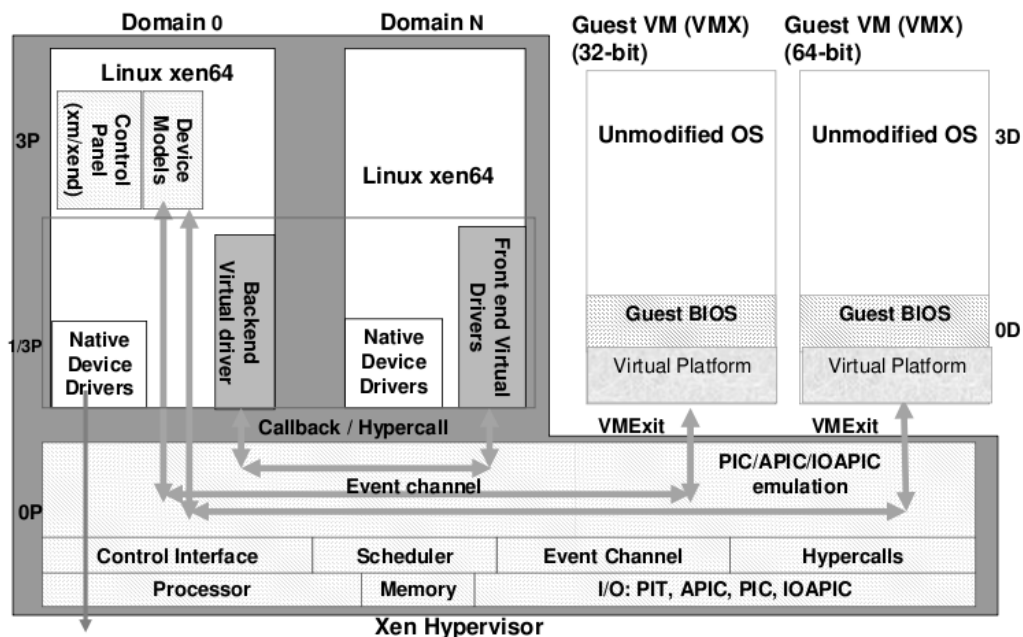




Abbildung 2.3: XenTM Hypervisor [Pra06]

der Paravirtualisierung von *XenTM* liegt nur zwischen 3% und 5% und somit weit unter vergleichbaren Lösungen. Insbesondere IO-intensive Anwendungen wie Datenbanksysteme oder Fileserver, sind innerhalb von *XenTM* weit weniger betroffen als innerhalb einer nativ virtualisierten Umgebung wie *VMWare*.

Ein Nachteil gegenüber der nativen Virtualisierung ergibt sich jedoch gerade aus dieser fehlenden vollständigen Emulation: Das Gast-System oberhalb einer privilegierten Domäne muss Kenntnis von der Paravirtualisierung durch *XenTM* haben. Genauer bedeutet dies eine Anpassung nötiger Treiber an diese Umgebung. Mit *XenTM 3.0.4* bietet sich jedoch mit der Einführung neuer Prozessorgenerationen wie Intels CoreDuo bzw AMDs AMD64

die Möglichkeit, unmodifizierte Gast-Systeme über die Hardwarevirtualisierung dieser Prozessoren (*Intel Virtualization Technology*  bzw. *AMD's Virtualization Solutions* ) zu betreiben.

2.2.2 Effiziente Hardwarekonsolidierung

Viele der heutigen Rechnersysteme sind nicht ausgelastet oder liegen brach [Kot05]. Wie oft kommt es vor, das in einem Unternehmen Rechnerlandschaften nur noch von wenigen benutzt werden, weil die Rechner einfach in Vergessenheit geraten sind, oder die Anwendungen zu spezielle Anforderungen an das Betriebssystem stellen, als das noch andere Applikationen darauf lauffähig wären. Ebenfalls sind die meisten heute im Einsatz befindlichen Webserver oder Firewallserver chronisch unterausgelastet. Hier springt Xen™ in die Bresche. Durch die Möglichkeit, mehrere konkurrente Gast-Betriebssysteme auf einem physikalischen Rechner zu betreiben, können ungenutzte Ressourcen schnell ausfindig gemacht und beliebig an diese verteilt werden. Da die Gast-Umgebungen völlig voneinander isoliert betrieben werden können, machen auch Gast-Systeme mit kuriosen Anforderungen keine Probleme mehr.

2.2.3 Hardware Hotplugging

Xen™ besitzt die Möglichkeit, dynamisch die physikalischen Ressourcen (CPU, Speicher, etc.) durch die Verwaltungsschicht (dom0) und die dazugehörigen Verwaltungsprogramme, zu verwalten und ggf. einem Gast-System zu entziehen oder hinzuzufügen. Hierdurch lassen sich sehr elegant Lastverteilungs- oder Redundanzszenarien erstellen, die jeder Anforderung des Gast-Systems genügen. Gelangt beispielsweise eine Applikation eines Gast-Systems an seine Speichergrenzen, so kann, genügend Speicher vorausgesetzt, die administrative Ebene dieser unprivilegierten Domäne während des laufenden Betriebes neuen Speicher zuweisen. Ausfallzeiten auf Grund von fehlender Hardware lassen sich so minimieren bzw. eliminieren.

2.2.4 Live-Migration

Eine besondere Fähigkeit von Xen™ ist die so genannte „online migration“[CC05]. Dieser Mechanismus erlaubt es, ein Gast-System während des laufenden Betriebes nahezu unterbrechungsfrei von einem physikalischen Host-System auf ein anderes zu migrieren. Alle Daten, Speicherinhalte, Stacks, Sockets, etc. werden hierbei mit umgezogen. Das Gast-System merkt von diesem Vorgang nichts und behält seinen Status genau wie vor der Migration bei. Dieser sehr mächtige Mechanismus benötigt jedoch einige strukturelle Vor-

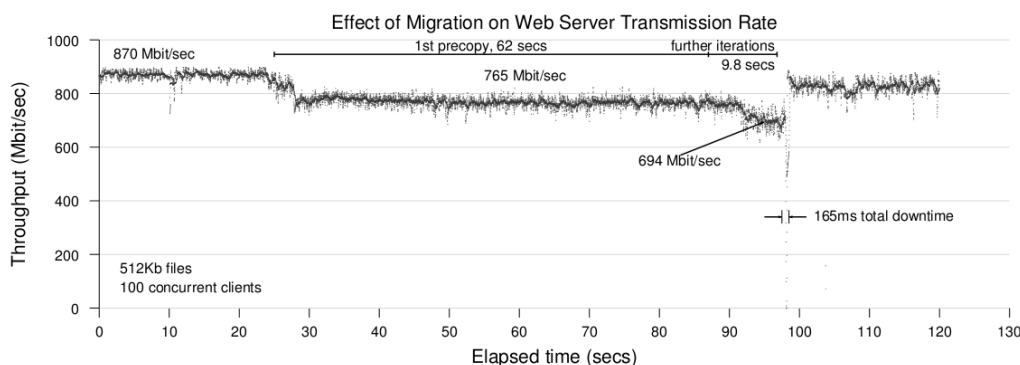


Abbildung 2.4: Xen™ Live Migration[CC05]

kehrungen an der physikalischen Rechnerlandschaft. Der Rechnerpool sollte für eine unterbrechungsfreie Migration zwischen den physikalischen Maschinen mindestens über eine GigaBit-Netzwerkanbindung verfügen, um die zu transferierenden Daten möglichst schnell zu übertragen. Wichtiger jedoch ist ein gemeinsamer Datenpool, auf den alle Rechner konkurrenten Zugriff haben. Eine SAN Lösung (Storage attached Network) oder ein verteiltes Cluster Dateisystem, das die einzelnen Gast-System-Abbilder und Arbeitsdaten enthält, sind hierbei unerlässlich. In späteren Kapiteln werde ich noch genauer auf diese Problematik eingehen, daher soll dies hier erst einmal als Erklärung genügen.


2.2.5 Softwareverfügbarkeit

Durch die o.g. Mechanismen einer XenTM Infrastruktur kann eine für Softwareprojekte ideale Betriebsumgebung geschaffen werden. So können z.B. virtuelle Systeme redundant oder als Failover-System ausgelegt werden, ohne neue Hardware, replektive Rechner, anschaffen zu müssen. Neue Umgebungen können schnell durch einfaches Kopieren, Konfigurieren und Starten bereits bestehender Systeme sehr schnell instanziiert werden. Auch können sich bereits im Produktivbetrieb befindliche Systeme von älteren oder defekten physikalischen Systemen auf neuere Systeme migriert werden. Dies kann z.B. im Falle einer Wartung der Host-Systeme oder im Fehlerfall eine unterbrechungsfreie Laufzeitumgebung begünstigen.

2.2.6 Xen Community

Last but not least - die Community. Die XenTM-Community ist eine der aktivsten und anspruchvollsten Communities im OpenSource Bereich. Meist aus dem Umfeld von Universitäten oder großen Unternehmen wie IBM, Redhat oder der Deutschen Telekom, befinden sich alle Mitglieder und Kontributoren der Community auf einem hohen technischen und wissenschaftlichen Stand. Eine aktive Mitgestaltung durch ein dediziertes Wiki-System sowie eine aktiv gepflegte Mailingliste und der einfache Zugang zu Dokumentationen und Präsentationen runden die Community ab. Der wissenschaftliche Wind in dieser Community sorgt immer wieder für frische Ansätze oder neue Sichtweisen und Einsatzgebiete der XenTM Umgebung. Durch die große Community, die XenTM entweder zu wissenschaftlichen oder zu kommerziellen Zwecken einsetzt, befindet sich das XenTM Framework quasi 24/7 im Regressionstest. Fehler, die hierbei auftauchen, werden meist sehr schnell von anderen Community-Mitgliedern beantwortet oder behoben. Die gute Community ist der Erfolgsgarant dieser Software.

2.2.7 Übersicht

Wenn Sie sich nun, nachdem sie einige der wichtigsten Vertreter kennengelernt haben, selbst einen Überblick verschaffen wollen, so kann ich Ihnen eine vollständige Übersicht aller Virtualisierungsframeworks ans Herz legen. Die Übersicht von *WikiPedia* mit dem Artikelnamen *Comparison of virtual machines*⁹  beinhaltet neben vielen kleinen oder bereits eher historischen Projekten auch eine Übersicht über die derzeit aktuellen, am Markt verfügbaren, Virtualisierungsframeworks.

2.3 ATA over Ethernet

ATA over Ethernet wurde als einfacher RPC-Mechanismus zum Austausch von ATA Befehlen konzipiert. Es dient einem Client zur Kommunikation mit einem beliebigen ATA-Server [BC04] auch über Netzwerkverbindungen hinweg. Die Netzwerkebene von *ATA over*

⁹Comparison of virtual machines: http://en.wikipedia.org/wiki/Comparison_of_virtual_machines

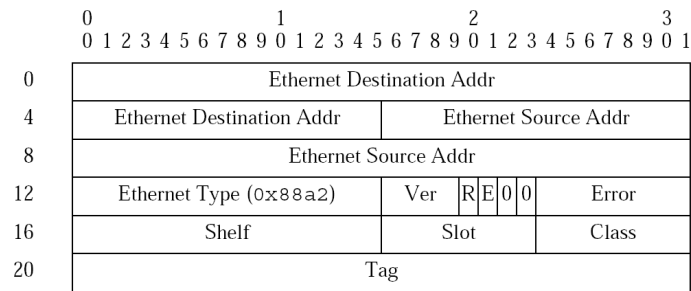



Abbildung 2.5: ATA over Ethernet Header Spezifikation

Ethernet bedient sich einem normalen MAC-Header aus der *IEEE 802.3 Ethernet*¹⁰  Protokollspezifikation, um Pakete auch über eine Netzwerkverbindung zu versenden. Als registrierte Protokollkennzeichnung im Header wird 0x88A2 verwendet. *ATA over Ethernet* kann auf Grund des einfachen Aufbaus als verbindungsloses Protokoll nicht geroutet werden. Daher ist sein Einsatzgebiet auf lokale Netzwerke ohne Router beschränkt. Dies mag

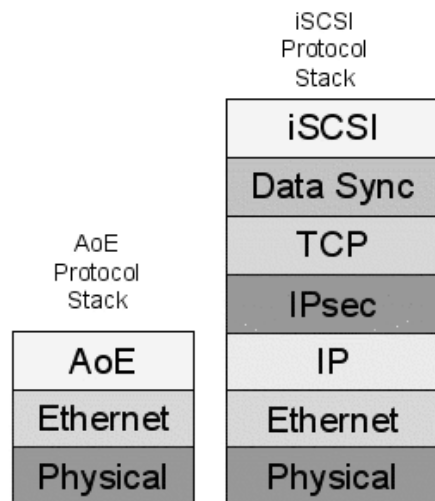




Abbildung 2.6: ATA over Ethernet Protokollstack


auf den ersten Blick als Manko erscheinen, doch ist *ATA over Ethernet* nicht als Werkzeug zum Verwalten von *Remote Storages* gedacht. Im Vergleich zu einer komplexeren Lösung wie *iSCSI*¹¹  erkennt man sehr schnell, wieviel Overhead sowohl bei Verarbeitung als auch bei dem Netzwerktransport mit dem Ansatz von *ATA over Ethernet* eingespart wird. Gerade diese Einfachheit macht es zum perfekten Werkzeug, um Blockgeräte beliebiger Art einfach und schnell zu exportieren.

Auf Serverseite dient das Projekt „*vblade*“ als ATA-Serverschnittstelle und exportiert entsprechende Block-Geräte über eine vorher definierte Netzwerkkarte nach außen. Die Grundimplementierung von „*vblade*“ ist im Userspace vorgenommen, jedoch gibt es mittlerweile auch eine reine Kernelimplementierung von „*vblade*“¹² , mit guten Chancen, in den Mainstream-Kernel aufgenommen zu werden. Die mitgelieferten Werkzeuge *aoetools*



¹⁰IEEE 802.3 Ethernet: <http://grouper.ieee.org/groups/802/3/>

¹¹iSCSI: <http://tools.ietf.org/html/rfc3720>

¹²„vblade“: <http://lpk.com.price.ru/~lelik/AoE/>

¹³ , die für die server- und clientseitige Unterstützung von *ATA over Ethernet* gedacht sind, machen das Implementieren einer einfachen, aber beliebig skalierbaren SAN-Lösung sehr einfach.

2.4 Distributed Replicated Block Device (DRBD)

Das OpenSource Projekt *Distributed Replicated Block Device (DRBD)* ¹⁴  wurde entwickelt, um einfache Block-Geräte unter Linux über ein Netzwerk auf einen anderen Rechner zu spiegeln. Häufig kommt *DRBD* im Linux-High-Availability Bereich, z.B. bei hochverfügbaren Clustern oder Datenhaltungen, zum Einsatz. Grob gesagt, kann man sich *DRBD* als RAID-1 über ein Netzwerk vorstellen. *DRBD* ist clusterfähig, d.h. es kann durch Failover-Mechanismen, wie *heartbeat* ¹⁵ , überwacht und gesteuert werden. *DRBD* spiegelt bei

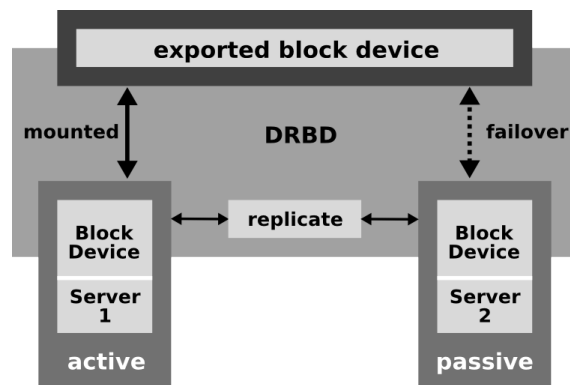



Abbildung 2.7: DRBD im aktiv-passiv Modus

einem synchronen Block-Gerät nicht einzelne Partitionen, sondern ganze Geräte, was es unabhängig vom eingesetzten Dateisystem und seinen Mechanismen macht. *DRBD* eignet sich sehr gut zum Aufbau eines Datenspeichers für Clustereinsätze. Grundsätzlich ist *DRBD* als *aktiv-passiv* Konzept aufgebaut. Hierbei werden die Block-Geräte von der aktiven Seite auf der passiven Seite gespiegelt. Sollte nun der aktive Part ausfallen, übernimmt sofort der passive Part das Ausliefern der Block-Geräte an die Clients. Seit *Version 0.8.0* unterstützt *DRBD* jedoch auch das gleichzeitige Einbinden der gespiegelten Block-Geräte auf beiden Seiten des Konfiguration. Dieses *aktiv-aktiv* Konzept kommt dem Einsatz in Clustersystemen sehr entgegen. Bei einem gleichzeitigen Einbinden sollte jedoch ein dediziertes Cluster-Dateisystem verwendet werden, um die nötige Sicherheit und den Schutz vor gleichzeitigem Zugriff auf eine Ressource zu gewährleisten.

2.5 Oracle Cluster File System (OCFS2)

Oracle Cluster File System (OCFS2) [Fas06] ist ein Dateisystem, das speziell für Cluster entwickelt wurde. Es ist *POSIX* ¹⁶  kompatibel und sorgt unter anderem dafür, dass in einem Szenario mit einer geteilten Festplatte (*shared-disk*), die von mehreren Systemen gleichzeitig benutzt wird, kein Datenverlust durch die konkurrierenden Zugriffe auf diese Ressource entstehen. *OCFS2* bedient sich hierbei den Prinzipien des *IO-Fencing* und *Quorum-Managements*.

¹³aoetools: <http://aoetools.sourceforge.net/>

¹⁴Distributed Replicated Block Device (DRBD): <http://www.drbd.org>

¹⁵heartbeat: <http://linux-ha.org>

¹⁶POSIX: http://www.opengroup.org/austin/papers/posix_faq.html

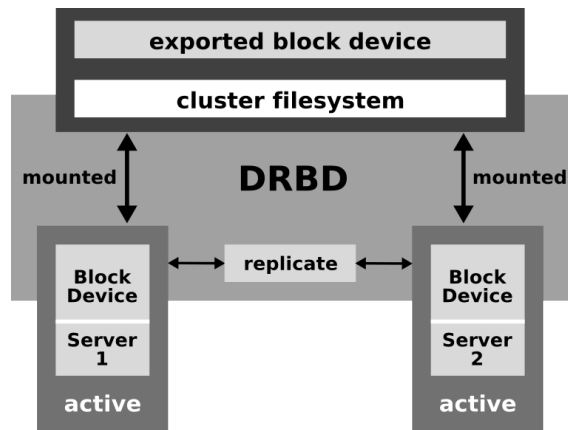


Abbildung 2.8: DRBD im aktiv-aktiv Modus

2.5.1 Quorum

Ein *Quorum* ist eine Aussage über eine Gruppe von Systemen in einem Cluster, die gemeinsamen Zugriff auf eine Ressource (hier: Festspeicher) haben. Ein *Quorum* wird benötigt, wenn die Gruppe, z.B. durch Fehler in der Kommunikation, in kleinere Gruppen aufgeteilt wird. Hierbei können die einzelnen Systeme innerhalb der kleinen Gruppen noch miteinander kommunizieren, jedoch nicht mehr die Gruppen untereinander. Ein System darf die gemeinsame Ressource nur benutzen, wenn sie dafür *Quorum* hat. Ein *Quorum* kann auf verschiedene Arten ermittelt werden. Hier ein Beispiel, wann ein System den Status *Quorum* erhält:

- Es sieht eine ungerade Anzahl an anderen Systemen und hat mindestens zur Hälfte der Systeme eine Verbindung, **oder**
- es sieht eine gerade Anzahl an anderen Systemen und hat mindestens zur Hälfte eine Verbindung **und** hat eine Verbindung zum System mit der kleinsten Quorum-Nummer.

2.5.2 Fencing

Während des ständigen Zugriffs von verschiedenen Systemen aus auf eine Ressource, kann es vorkommen, dass ein System sich nicht mehr mit dem Rest des Verbundes synchronisieren kann. In diesem Fall hat das System kein *Quorum* und nimmt sich selbständig aus dem Verbund und schreibt auf keinen Fall mehr auf die Ressource (*self-fencing*).

3 Anwendungsgebiete

Dieses Kapitel beinhaltet eine Auswahl an möglichen Gebieten zum Einsatz der Paravirtualisierung sowie ein konkretes Anwendungsbeispiel im Rahmen dieser Ausarbeitung für ein großes deutsches Unternehmen.

3.1 SSI-Hosting

SSI oder „*Single System Images*¹“ ist ein Begriff aus dem Clusterbereich, für den es momentan zwei Bedeutungen gibt:

- *Innere Sicht:*
Ein System (meistens ein Betriebssystem), das mit allen notwendigen Applikationen in einer einzigen Datei untergebracht ist.
- *Äußere Sicht:*
Ein System, das einem Benutzer eine einheitliche Umgebung zur Nutzung zur Verfügung stellt, egal auf wievielen Rechnern (oder Clustern) diese Umgebung verteilt ist.

Ich beziehe mich bei diesem Anwendungsgebiet auf die erste Variante, da diese Anwendungsform Hand in Hand mit den Möglichkeiten der Xen™-Paravirtualisierung geht. Hierbei werden frei verfügbare Grundsysteme an die speziellen Anforderungen des jeweiligen Nutzers angepasst und dann entsprechend auf die Systeme des SSI-Hosting Providers aufgespielt. Dort können diese Systemabbilder entsprechend in virtuellen Domains (*domU*)

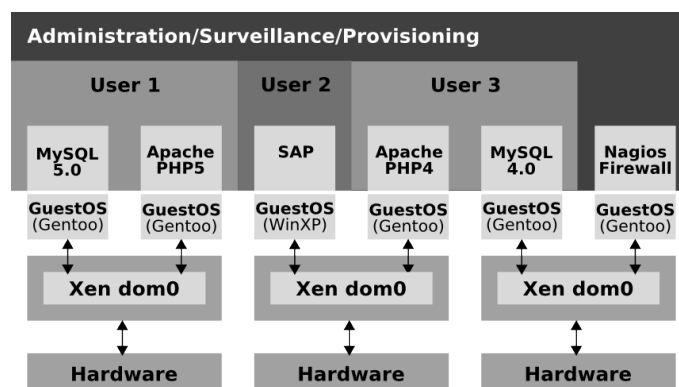


Abbildung 3.1: Single System Image Hosting

gestartet werden. Ein Vorteil dieser Vorgehensweise ist neben dem Isolationslevel (der Benutzer hat keinen physischen Zugriff auf den Server) auch der Vorteil der Konsolidierung der Serverkapazitäten. So können beispielsweise mehrere Systeme von verschiedenen Nutzern auf einem physikalischen und nicht ausgelasteten Server zusammengefasst werden, ohne dass der jeweilige Nutzer dies bemerkt. Intelligente Load-Balancing Systeme können diesen Vorgang auch automatisiert vornehmen, was eine manuelle Administration nur

¹Definition für Single System Images: http://en.wikipedia.org/wiki/Single-system_image

im Fehlerfall oder auf Wunsch nötig macht. Diese Technik eröffnet mehrere Aspekte für Einsparpotenzial:

- *Server Konsolidierung:*
Freie oder brachliegende Rechenleistung kann genutzt bzw. verbucht werden.
- *Wenig manuelle Administration:*
Intelligentes Load-Balancing gleicht Engpässe an Rechenleistung oder Speicherplatz gezielt nach festgesetzten Regeln aus. Ein manueller Eingriff ist nur im Fehlerfall nötig.
- *Hoher Grad an Sicherheit und Wartungserleichterung der physikalischen Systeme durch Isolation:*
Die paravirtualisierten Systeme haben keinen direkten Zugriff auf das darunterliegende physikalische System. Wartungsarbeiten, sowie ausgehebelte Sicherheitssysteme, die bei physikalischem Zugriff kompromittiert werden können, sind ausgeschlossen.
- *Isolation der paravirtualisierten Systeme untereinander:*
Mehrere parallel, auf einer physikalischen Maschine, laufende Systeme haben jeweils nach innen exklusiven Zugriff auf sämtliche ihnen zur Verfügung stehenden Ressourcen. Die Systeme können untereinander nur über fest definierte Schnittstellen (so genannte virtuelle Devices wie Netzwerk oder shared storage) kommunizieren.
- *Weniger anfällig durch Hardwareausfälle:*
Sollte ein Hardwaredefekt auftreten, bei dem normalerweise das System heruntergefahren werden muss, besteht die Möglichkeit, die laufenden SSI durch Online-Migration auf andere physikalische Systeme auszulagern. Nachdem die Wartungsarbeiten abgeschlossen sind, werden die SSI wieder zurückmigriert.

Meist werden Systeme, die im normalen Hosting auf einzelnen physikalischen Maschinen laufen, durch dieses Konzept zusammengefasst. Einige unerlässliche Systeme, wie Apache-Server, MySQL-Server, Mailsysteme oder Firewalls, lassen sich so sehr effizient zusammenlegen und somit der physikalische sowie administrative Aufwand erheblich reduzieren.

3.1.1 Entwicklungssysteme

Ein recht einfaches Anwendungsszenario von *SSI-Hosting* ist das Etablieren von dedizierten Entwicklungsumgebungen. Hierbei steht die Mehrfachnutzung der vorhandenen Entwicklungsinfrastruktur für mehrere Projekte im Vordergrund. Oft haben Projekte, die konkurrent entwickelt werden, Anforderungen die sich nicht gänzlich auf einem Entwicklungssystem vereinbaren lassen oder sich gar völlig ausschließen. Unter normalen Umständen müssten nun mehrere isolierte Entwicklungsumgebungen auf verschiedenen Rechnersystemen etabliert werden, um eine Interaktion zu verhindern.

Die Methode des *SSI-Hostings* bietet die Möglichkeit, die Projektumgebungen isoliert und konkurrent auf den gleichen Rechnersystemen laufen zu lassen, ohne dass die o.g. Interaktion stattfindet. So können mehrere Projektumgebungen voneinander isoliert auf der physikalischen Infrastruktur untergebracht werden und es lassen sich erheblicher administrativer Aufwand und auch physikalische Ressourcen einsparen. Auch der zeitliche Aufwand für das Etablieren neuer Projektumgebungen wird so reduziert, da lediglich eine vorhandene Projektinfrastruktur, sofern möglich, kopiert und angepasst werden muss. Als Beispiel lässt sich gut eine Projektumgebung zum Entwickeln aus dem Webbereich vorzeigen. Oft werden z.B. bei CMS-Implementationen verschiedenste PHP Versionen oder Einstellungen an Apache Servern verlangt, die nicht konkurrent mit anderen Projekten

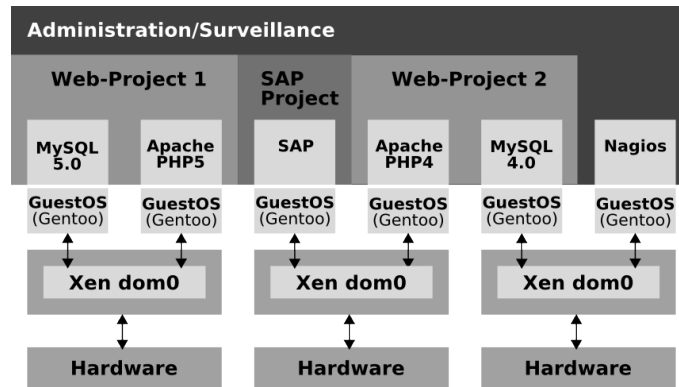


Abbildung 3.2: Exemplarische Entwicklungsumgebungen

lauffähig sind. Die einfachste Möglichkeit ist nun, verschiedene Grund-SSI's aufzubauen, die im ersten Schritt nur LAMP Systeme enthalten. Für ein konkretes Projekt wird nun diese Umgebung angepasst und entsprechend als unprivilegierte Domäne (domU) gestartet. Die Entwicklung kann nun völlig isoliert, aber konkurrent zu anderen Projektumgebungen stattfinden. Wie wir im nächsten Kapitel sehen werden, lassen sich sogar Produktivsysteme so sehr effizient testen.

3.1.2 Prototyping und Produktionssysteme

Oft kommt es vor, dass ein Produktivsystem in einer Entwicklungsphase oder auf Grund eines aufgetretenen Fehlers isoliert getestet werden muss. Oft hat dies auch die Anforderung, das es auf den dafür vorgesehenen Servern geschehen muss. Ohne ein System zur Paravirtualisierung müsste das Produktivsystem physikalisch redundant ausgelegt sein, um dieses System testen zu können, während es noch produktiv weiterarbeitet. Dies stellt einen nicht zu unterschätzenden Kostenfaktor dar, da dieses Testsystem noch nicht einmal als Failover-System genutzt werden kann, obwohl es dafür vom Konzept her prädestiniert dafür wäre. In einer isolierten paravirtualisierten Umgebung können jedoch Entwicklungs-

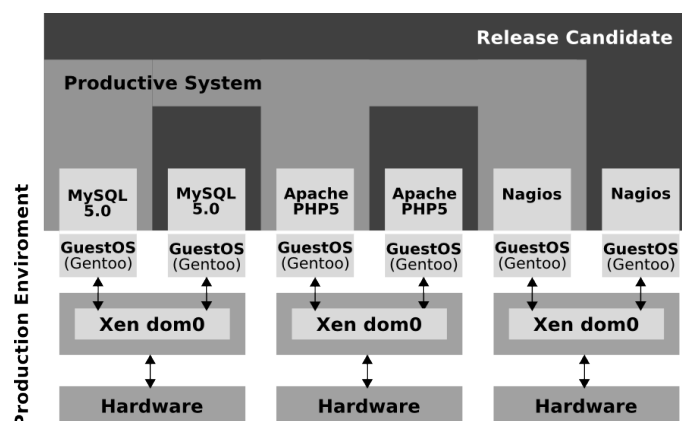


Abbildung 3.3: Exemplarische Produktivumgebung

systeme innerhalb der physikalischen Maschinen des Produktivsystems erstellt werden. Auf Grund der Isolation bleibt das eigentliche Produktivsystem unbeeinträchtigt. Auch ganze Abbilder von Produktivsystemen lassen sich so generieren und auf Testsysteme zur Fehlersuche oder zum Testen übertragen.

Auch Mechanismen aus der Softwaretechnik, wie Releasezyklen oder Testsysteme lassen sich so isoliert in den einzelnen Projekt- oder Produktivumgebungen darstellen. Hierbei werden die Releasesysteme konkurrent zu dem derzeitigen Produktivsystem aufgebaut und gestartet. Ein Update bzw. Austausch von Release- und Produktivsystem ist nun nur noch minimaler administrativer bzw. zeitlicher Aufwand. Ausfall- oder Wartungszeiten lassen sich so effizient vermeiden. Des weiteren fördert natürlich die Isolation der einzelnen Systeme untereinander die Einhaltung der Mechanismen der Softwaretechnik selbst.

3.2 Virtual Clustering

Wie im vorigen Kapitel schon angesprochen, eröffnet ein Anwendungsgebiet des SSI-Hosting in Verbindung mit Paravirtualisierung durch Xen™ die Möglichkeit, mehrere angepasste Systeme auf physikalischen Rechnern zu isolieren. Jedes dieser Single-System-Images hat seine eigene, in sich abgeschlossene Umgebung mit allen notwendigen Ressourcen, wie Netzwerk, HDD, etc. zur Verfügung.

Eine sehr einfache Methode dieses Konzept darzustellen, bietet die Implementierung der privilegierten Domäne in Verbindung mit dem *openMosix* Clustersystem. Als größter Vorteil lässt sich die Lastverteilung und Ressourcenverwaltung durch *openMosix* ansehen. In diesem kompletten Rechnerpool wird eine effiziente Last- und Prozessverteilung durch die Mechanismen von *openMosix* sehr gut dargestellt. Unprivilegierte Domänen lassen sich in dieser Gesamtumgebung durch einzelne verteilte Prozesse darstellen. Einige Nachteile

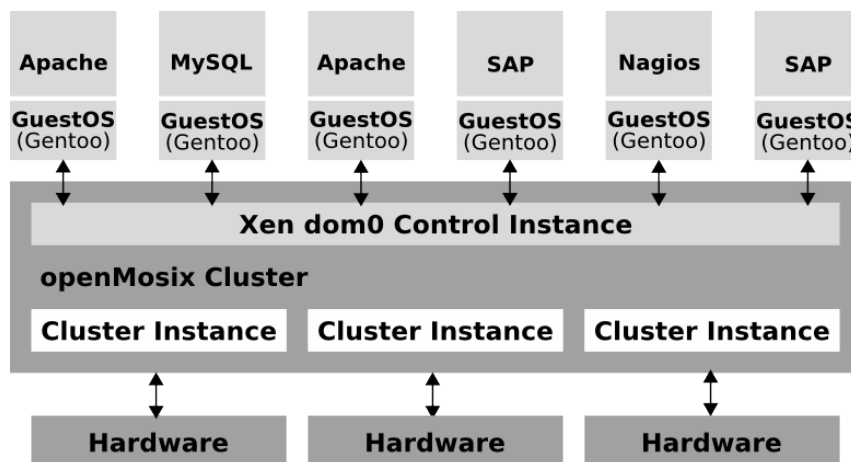


Abbildung 3.4: Virtual Clustering

dieses Anwendungsgebietes machen es jedoch strukturell unsicher, dieses Modell einzusetzen: Als erstes ist die quasi doppelte Abstraktion der Hardware, sowohl durch *openMosix*, als auch durch Xen™ zu nennen. Ein zweiter, aber viel gravierender Nachteil, den dieses System im produktiven Einsatz offenbart, ist die fehlende Möglichkeit, mehrere Clustersysteme auf der selben physikalischen Hardware einzusetzen. Wie wir jedoch im nächsten Kapitel sehen werden, kann sich dieser Nachteil gezielt beheben lassen.

3.3 Virtual Elastic Clustering

Ein Schritt weiter als der Ansatz des *Virtual Clustering* geht der Ansatz des *Virtual Elastic Clustering*. Hierbei wird ein solches System nicht auf Basis eines *openMosix* in Verbindung mit einer privilegierten Domäne (*dom0*) aufgebaut, sondern es werden auf den

physikalischen Maschinen lediglich privilegierte Domänen eingesetzt. Die Clusterumgebung (z.B: *openMosix*) wird als SSI auf diesen Systemen angelegt und entsprechend einmal oder mehrmals auf einer oder mehreren privilegierten Domains (*dom0*) gestartet. Die Ebene der Clusterumgebung wird hierbei vollständig in unprivilegierte Domänen (*domUs*) verschoben, wodurch sich einige Vorteile gegenüber dem *Virtual Clustering* ergeben.

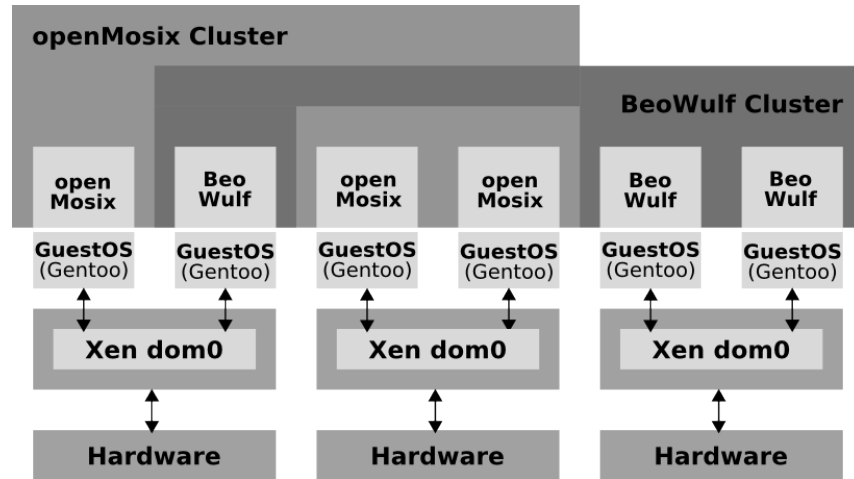


Abbildung 3.5: Virtual Elastic Clustering




SSIs auf Basis eines Clustersystemes, wie z.B. *openMosix* oder *BeoWulf*, können beliebig oft auf einzelnen bzw. mehreren, untereinander verbundenen, physikalischen Maschinen gestartet und verbunden werden. So kann z.B. ein *openMosix* Clustersystem völlig isoliert, aber konkurrenzfähig mit einem *BeoWulf* Clustersystem auf dem selben Rechnerpool betrieben werden. Durch die einfache Handhabung der unprivilegierten Domänen lässt sich eine paravirtualisierte Clusterumgebung leicht nach unten bzw. oben skalieren. Ein einfaches Kopieren und Starten der SSI macht es möglich. Es können so sehr leicht virtuelle elastische Clusterumgebungen in dieser paravirtualisierten Umgebung aufgebaut werden.


Auch *Ausfall-* und *Crash-Recovery-Szenarien* lassen sich leicht durchspielen, indem man lediglich eine oder mehrere der zum System gehörenden unprivilegierten Domänen herunterfährt und beendet oder gar ohne Herunterfahren einfach im laufenden Betrieb stoppt.

4 Ausflug in den Markt

Marktgeschehen ist für einen Bereich, in dem man selbst tätig ist, immer sehr interessant. Man kann Trends beobachten, neue Strukturen kennenlernen und immer wieder neue Bereiche für seine tägliche Arbeit entdecken. Vor allem dann, wenn sich neue Trends gerade im Verborgenen entwickeln und kurz vor dem Durchbruch stehen, wird ein Thema sehr interessant. Dies geschieht gerade im Bereich der IT Infrastrukturen und Virtualisierung. Diverse Virtualisierungsansätze entsteigen langsam, aber sehr gezielt, den Kinderschuhen. So bietet z.B. *VMWare* seit einiger Zeit Produkte nicht nur für den Heimanwendermarkt an, sondern verlegt seinen Hauptgeschäftszweig immer mehr in die Virtualisierung von Servern und Serverinfrastrukturen, wie die von Hosting- oder DataCenter-Providern. Auch Xen™ hält immer mehr Einzug in die professionelle Welt der Hostingprovider.

Abseits dieser Virtualisierungsprovider werden auch immer mehr kleine wie große Firmen auf Xen™ aufmerksam. Es bietet eine einfache Möglichkeit, seine IT Infrastruktur im laufenden Betrieb und - betriebswirtschaftlich wichtig - ohne große Zusatzkosten zu konsolidieren oder in beliebige Richtungen zu skalieren. Früher noch ein nicht zu unterschätzender Aufwand, können heute, dank der Abstraktion durch Xen™ oder ähnlichen Techniken, prototypische Testumgebungen oder Entwicklungssysteme „on-the-fly“ erstellt, archiviert und auch wieder deaktiviert werden.

Ebenfalls ist mittlerweile ein neuer Geschäftszweig aus dieser Branche erwachsen. Unternehmen, wie *VirtualIron*¹  sind entstanden, die sich darauf spezialisieren, ganze IT Infrastrukturen für andere auf Basis von Xen™ zu konsolidieren bzw. virtualisieren. Provider mit virtuellen Servern auf Basis von Xen™ tauchen ebenfalls verstärkt auf dem Markt auf. Firmen, die Administrationswerkzeuge oder gar Überwachungswerkzeuge für ganze Unternehmensnetze mit virtuellen Maschinen in ihrer Produktpalette haben, tummeln sich ebenso am Markt wie OpenSource Projekte wie *JailTime*²  oder das kommerzielle Pendant *rPath*³ , das vollständige SSI's als so genannte „Appliances“, also vollständige Systeme mit einzelnen Anwendungen (z.B. Apache + MySQL + Mediawiki) und die dazugehörige Beratung zur Verfügung stellen.

Ein neuer Trend wird geboren und auch die Entwickler des Linux Kernels können sich nicht mehr der Entwicklung verschließen und arbeiten derzeit an einer direkten Implementierung einer Virtualisierungstechnik in den Kernel. *Kernel based Virtual Machine (KVM)*⁴  wird das Kind getauft und ist bereits dazu fähig, unmodifizierte Gast-Betriebssysteme oder andere Linux-Derivate zu virtualisieren. KVM baut hierbei hauptsächlich auf einer modifizierten Variante von *Qemu* auf, um die Virtualisierung der Gast-Systeme zu vervollständigen. Im aktuellen Kernel 2.6.20 ist bereits ein großer Teil der KVM fertiggestellt und die Linuxgemeinde prüft das System bereits auf Herz und Nieren.

¹VirtualIron: <http://www.virtualiron.com>




²JailTime: <http://www.jailtime.org/>



³rPath: <http://www.rpath.com>

⁴Kernel based Virtual Machine (KVM): <http://kvm.qumranet.com>

Um Ihnen einen kurzen Überblick über das Thema und die dazugehörigen Trends zu geben, machen wir in diesem Kapitel gemeinsam einen kleinen Ausflug in einige sehr interessante Branchen und Techniken. Der Ausflug ist jedoch nicht als Wandertour mit Streckenrekord geplant, also stellen Sie sich auf einen gemütlichen Ritt durch die holprige Virtualisierungswelt ein. Vielleicht bekommen Sie ja Appetit auf mehr.

4.1 Appliance Provider

Unter *Appliances* versteht man im Virtualisierungsumfeld Systemabbilder, die einem gewissen Zweck dienen. So kann eine *Appliance* eine Kombination aus einem Debian Linux Grundsystem und einer dedizierten Firewall bestehen, aber auch einfache *Appliances* wie LAMP-Systeme (Linux, Apache, MySQL, PHP) gehören dazu. Diese Systeme werden von verschiedenen Anbietern entweder kommerziell oder nicht kommerziell vertrieben und können meist in verschiedenen Formaten für verschiedene Virtualisierungsframeworks bezogen werden. Diese *Appliances* müssen nicht installiert, sondern lediglich konfiguriert werden und stellen somit eine Möglichkeit dar, sehr schnell eine Evaluation der einzusetzenden Software durchzuführen, ohne selbst ein komplettes System aufsetzen zu müssen. Die beiden bekanntesten Vertreter sind zum einen *rPath*⁵  und *enomalism*⁶ . Mittlerweile existieren auch Suchmaschinen, wie *VMfind*⁷ , die sich auf das Auffinden von bestimmten *Appliances* spezialisiert haben.

Auch die OpenSource Gemeinde ist schon seit längerem in diesem Feld tätig, wie die Seite von *FreeOsZoo*⁸  eindrucksvoll beweist. Hier ist es möglich, verschiedene Systeme für XenTM oder *Qemu* herunter zu laden und auszuprobieren. Neben verschiedenen Linux Derivaten bietet ein Ableger von *FreeOsZoo*, *Live FreeOsZoo*⁹ , sogar die Möglichkeit, bestimmte Systeme online zu testen, ohne sie vorher herunterzuladen. Hierbei werden die Instanzen des gewünschten Systems - zeitlich begrenzt - auf den Servern von *FreeOsZoo* als virtuelle Maschine instanziiert. Dort kann ein Interessent das System evaluieren und ggf. danach als „*Appliance*“ herunterladen.

4.2 Elastic Hosting

Hosting ist in heutigen Zeiten von WebApplications, Webseiten und Webservices ein unabdingbarer Bestandteil eines erfolgreichen Produktes. Aber immer wieder werden während der Entwicklung und des Betriebes komplexer Systeme einige grundlegende Fragen aufgeworfen. Diese Fragen sind meist nicht belanglos, sondern können ganze Entwicklungen oder bereits produktive Systeme gefährden. „*Wieviele Rechner braucht unsere Applikation?*“ - „*Welche Art von Rechnern (z.B. Datenbanken, Webserver, etc.) brauchen wir?*“ - „*Was passiert im Falle eines Absturzes?*“ - „*Was passiert, wenn wir plötzlich erfolgreicher werden, als wir erwartet haben?*“ - „*Wird unser System dem Ansturm Stand halten?*“ - „*Wie hoch ist der Aufwand der Erweiterung unserer Infrastruktur?*“ „*Elastic Hosting*“ ist das Zauberwort, das im Zusammenhang mit all diesen Fragen immer wieder auftaucht.

⁵rPath: <http://www.rpath.com>

⁶enomalism: <http://www.enomalism.com>

⁷VMfind: <http://www.vmfind.com>

⁸FreeOsZoo: <http://www.oszoo.org>

⁹Live FreeOsZoo: <http://connessi.webminds.cs.unibo.it:8880/>

Im Gegensatz zum traditionellen Hosting (z.B. Webhosting) wo ein Kunde nur eine statische Anzahl von Rechnern bzw. Rechenleistung buchen kann, ist der Ansatz des „*Elastic Hosting*“ losgelöst vom statischen Modell eines Rechners. Rechner bzw. Rechenleistung wird (para-)virtualisiert und somit von der Hardwareebene des traditionellen Hostings gelöst. Es werden nicht mehr statische Rechner und damit Rechenleistung über einen be-

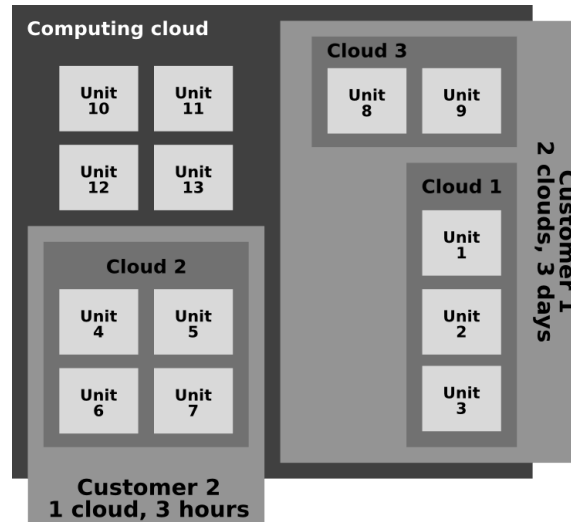


Abbildung 4.1: Computing cloud

stimmten Zeitraum angeboten, sondern eine so genannte „*computing cloud*“, eine flexible virtuelle Umgebung, in der beliebig viele oder wenige virtuelle Rechner instanziiert werden können.

4.2.1 Amazon Elastic Compute Cloud (EC2)

Amazon.com Inc. stellt seit kurzem einen neuen Webservice für flexible Hosting- und Computing-Lösungen als Beta-Version vor. „*Amazon Elastic Compute Cloud (Amazon EC2)*“¹⁰ stellt Benutzern die Möglichkeit zur Verfügung, beliebig viele virtuelle Rechner oder sogar Rechnerverbünde aus einem größeren Gesamtsystem zu buchen. Die Systeme können hierbei entweder von Amazon.com vorinstallierte Standardsysteme mit Datenbank- oder Applikationsservern sein oder es werden gänzlich vom Benutzer per „*Amazon Machine Image Manager*“ erstellte Images (AMI) eingespielt und gestartet. Hierdurch ist es möglich, Rechenleistung und Funktionalität „*on-demand*“ zu buchen, wobei Amazon.com für die Wartung und Instandhaltung der Hardware aufkommt. Leider ist nicht genau bestätigt, welche Systeme Amazon für diesen Dienst einsetzt, Gerüchten aus mehreren unabhängigen Quellen weisen jedoch auf ein System mit Einsatz des *XenTM virtual machine monitor* hin. Seit Januar 2007 stellt Amazon.com den Teilnehmern seines Beta-Programmes öffentlich *Appliances*¹¹ mit verschiedenen Betriebssystemen bzw. Applikationen zur Verfügung und fördert den Austausch dieser AMI's der Nutzer untereinander.

4.2.2 ElasticLive

*ElasticLive*¹² ist eine Application, die ähnlich Amazon.com's *EC2* ein Framework für *Elastic Hosting* bereitstellt. Dieses Framework wirbt mit der Einfachheit der Installation und

¹⁰ Amazon EC2: <http://aws.amazon.com/ec2>

¹¹ Appliances: <http://developer.amazonwebservices.com/connect/index.jspa>


¹² ElasticLive: <http://www.elasticlive.com>

der Benutzung, selbst durch nicht versierte Nutzer. Es stellt eine Palette bereits einsatzfähiger Systeme (*Appliances*), Frameworks und Applikationen bereit. Der Funktionsumfang der *Appliances* ist bei diesem Produkt laut *ElasticLive* größer als bei Amazon.com's *EC2*.

4.3 Projekt- und Entwicklungsframework ATLAS

Diese Ausarbeitung bzw. die Resultate dieser Betrachtung von Elastic Hosting und Clustering im Bereich der WebApplications fließen unmittelbar in die Konzeption einer IT Infrastruktur zur Abwicklung von Logistik-Workflows im Mobilfunkbereich ein. Dieses Konzept bietet die Grundlage für eine ausfallsichere und fast wartungsfreie Infrastruktur mit besonderem Augenmerk auf redundante Failover-Systeme und Skalierbarkeit.

4.3.1 ATLAS::TSI

Das momentan eingesetzte Produktivsystem *TSI*¹³  wird zur Abwicklung von Garantie- und Reparaturdaten mit mehr als 30 Herstellern aus dem Mobilfunk- und Consumer Electronic-Bereich benutzt. Es bietet momentan auf Basis von einzelnen dedizierten physikalischen Systemen mit PHP und MySQL eine hochspezialisierte Plattform mit speziell an jeden Hersteller angepassten Workflow- und Reportingsystemen zur Erfassung, Abwicklung und Abrechnung von Geräten aller Art.

Die Plattform bietet sowohl Online-, als auch Offlineschnittstellen zu diversen Systemen von Herstellern (meist SAP), sowie großen Partnern und Distributoren. Die Plattform steht, auf Grund der Masse an Benutzern, die Tag und Nacht an dem System arbeiten, unter einer SLA mit 99,999% Verfügbarkeit und sichert über verschiedene Failover- und Redundanzsysteme die Ausfallsicherheit und Plattformverfügbarkeit. Dieses Projekt und andere Projekte sollen sukzessive eine Änderung der zugrundliegenden Infrastruktur erfahren. Zum Einsatz kommt hierbei das in dieser Ausarbeitung entwickelte Projekt- und Entwicklungsframework ATLAS.

4.3.2 Entwicklungsumgebung - IST-Stand

Die bei der Entwicklung dieser Plattform zum Einsatz kommenden Test- und Entwicklungssysteme sind strikt getrennt. Teilweise werden in der Entwicklungsphase Softwarekomponenten, z.B. für die Evaluation oder das Reporting, benötigt, die eine parallele Entwicklung anderer Projekte auf diesen noch physikalischen Maschinen behindert oder gar unmöglich macht. Die Folgerung, jeweils ein physikalisches System pro Projekt zur Verfügung zu stellen, birgt ebenfalls eine Kostenfalle, da auch diese Systeme erstellt und administriert werden müssen.

Der momentane Stand erzwingt jeweils ein physikalisches System pro Projekt, um die nötige Isolation untereinander zu gewährleisten. Erschwerend kommt hinzu, dass externe Mitarbeiter jeweils auf eigenen Laptops bzw. Rechnersystemen entwickeln und somit Aufwand bei der Integration des entwickelten Projektes in die bereits bestehende Serverlandschaft verursachen. Auch bereits bestehende Datenbankserver werden mehrfach genutzt, was ein Risiko im Falle eines Ausfalls darstellt. Im diesem Falle wären nicht nur einzelne Projekte betroffen, sondern gleich mehrere. Das ist ein unhaltbarer Zustand.

¹³TSI: <http://tsi.handyreparatur.de>

4.3.3 Serverlandschaft - IST-Stand

Die Serverlandschaft der Plattform TSI ist zur Zeit noch in rein physikalischer Hinsicht in getrennte Systeme unterteilt. Rechnerpools mit redundanter Hardware (wie Netzwerk, Festplatte, Netzteil) sorgen für die nötige Hardwaresicherheit im Backend des TSI. Ein zweiter Rechnerpool, geographisch in einem zweiten Rechenzentrum untergebracht, sorgt für den nötigen Failover im Falle eines Ausfalles einer ganzen Leitung oder des Rechenzentrums. Backup und dedizierte Fileserver, jeweils auf eigenen physikalischen Maschinen untergebracht, sind für die Datenversorgung zuständig. Da die physikalischen Maschinen

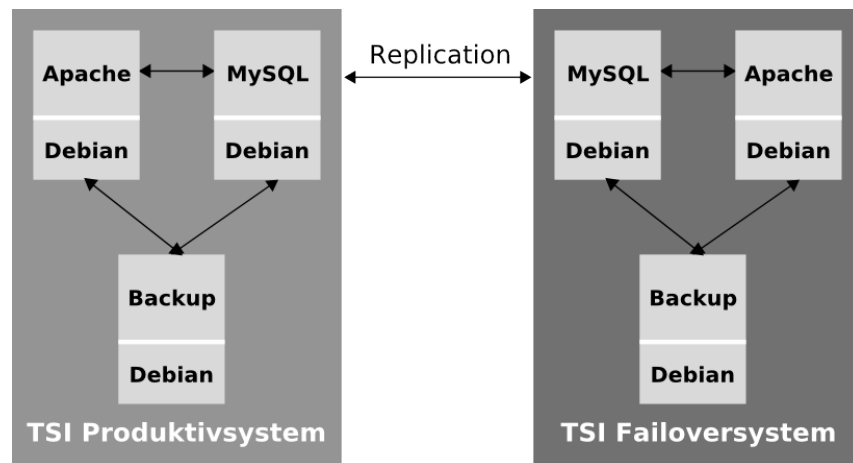


Abbildung 4.2: IST-Stand ATLAS::TSI

zur Zeit nur jeweils zwischen 15% und 20% ausgelastet sind, kommt der Ansatz der Server-Konsolidierung voll zum Tragen. Zum einen werden die verfügbaren physikalischen Systeme besser ausgelastet, zum anderen hat die Unabhängigkeit von dem physikalischen System, in Kombination mit der Möglichkeit, im laufenden Betrieb die virtualisierten Rechner auf andere physikalische Maschinen zu transferieren, eine erhebliche Kostenersparnis zur Folge. Im Kapitel 6 *Projekt ATLAS (S.29)* werde ich genauer auf diese Vorzüge eingehen.

4.3.4 Ausfallzeiten

Um eine kurze, aber effektive Überleitung in das nächste Kapitel zu bewerkstelligen, lassen sie uns anhand von ATLAS::TSI ein kleines Rechenexempel statuieren. Die meisten Lösungen in kleinen oder mittelständischen Unternehmen haben Netzwerke mit ca. 150 bis 300 Benutzern, die auf Dienste wie Drucken, Dateiserver oder Email zugreifen. Im Falle von ATLAS::TSI arbeiten tagsüber (08.00 Uhr - 22.00 Uhr) ca. 300 deutsche Werkstätten produktiv mit ATLAS::TSI. Hinzu kommen periphere Benutzer wie 4.000 Händler und ca. 160.000 Endkunden. Sollte das Kernsystem von ATLAS::TSI ausfallen, können die Werkstätten und Händler ihre Arbeitsleistung nur zu 50% erbringen. Die nicht abgeschlossenen Aufträge sowie verlorengegangene Werbeeinnahmen bei einem solchen Ausfall sind noch nicht mitgerechnet.

Bei einer durchschnittlichen Verfügbarkeit von 98% bedeutet dies einen 2% Ausfall aller Services, die ATLAS::TSI zur Verfügung stellt. Werkstätten könne keine Ersatzteile buchen, ihr Werkstattmanagement und Controlling ist quasi lahmgelegt. Händler können in dieser Zeit keine Geräte erfassen oder an die Werkstätten weiterleiten. Ein erheblicher Verdienstausfall geht damit einher. Doch zurück zur Rechnung:

- 6 Server mit verschiedenen Services

4 Ausflug in den Markt

- 300 Benutzer aus dem Bereich der Werkstätten
- 4.000 Benutzer aus dem Bereich der Händler
- Jeder Techniker in der Werkstatt arbeitet ca. 1.900 Stunden im Jahr
- Jede Werkstatt arbeitet im Schnitt mit 2 Technikern ($2 * 1.900 \text{ h} = 3.800 \text{ h}$)
- Jeder Händler verbringt ca. 1 h pro Tag mit ATLAS::TSI = 240 Stunden pro Jahr
- 2% Ausfall des ATLAS::TSI im Jahr

Bei einem Ausfall der Server von 2%, einem effektiven Arbeitsausfall der Benutzer von 50%, einen Lohn von ca. 1.200 € für einen normalen Techniker und einen fiktiven Lohn von 1.500 € für einen Händler, ergibt sich folgende Kalkulation für die Ausfallzeiten:

| | | | |
|------------------------|---------------------------------------|---|-------------|
| Arbeitszeit (gesamt): | $300 * 3.800 + 4.000 * 240$ | = | 2.100.000 h |
| Verlorene Zeit: | $2.100.000 \text{ h} * 0,02$ | = | 42.000 h |
| Verlorene Arbeitszeit: | $42.000 \text{ h} * 0,5$ | = | 21.000 h |
| Verlust pro Jahr: | $21.000 \text{ h} * 8,55 \text{ €/h}$ | = | 179.550 € |

Dies ergibt einen Gesamtverlust von ca. 15.000 € pro Monat über alle Kunden an reiner Arbeitszeit. Hierbei sind die Rechenleistung und die entgangenen Aufträge nicht mitberechnet. Wie man leicht erkennen kann, darf eine 2%-ige Aufallsquote nicht sein. Ein Konzept zur Streuung bzw. Vermeidung von Ausfallzeiten ist ein unbedingtes Muss. Die Konzepte der Paravirtualisierung liefern hier einen guten Ansatz, wie wir in den späteren Kapiteln sehen werden, um die Ausfallzeiten wirkungsvoll gegen Null streben zu lassen.

5 Anforderungen

Die im Kapitel 3 *Anwendungsgebiete (S.11)* erarbeiteten Anwendungsgebiete und Umgebungen bilden die Grundlage für die in diesem Kapitel erarbeiteten Anforderungen. Die Planung der zu realisierenden IT Infrastruktur ist hauptsächlich bestimmt durch den Einsatz von OpenSource Software bzw. Projekten. Dies hat vornehmlich den Hintergrund der Kosteneinsparung, die auch kleinen Unternehmen erlaubt, im sehr kostenintensiven Bereich der SAN- und Clustering-Lösungen zu navigieren.

Die meisten OpenSource Projekte, die hierbei zum Einsatz kommen, sind entweder Projekte die über eine große und aktive Community verfügen oder bereits von großen namhaften Unternehmen (z.B. RedHat) oder Universitäten (z.B. University of Cambridge, MIT) eingesetzt und entwickelt werden. Die Projekte sind allesamt erprobt und werden aktiv weiterentwickelt, was die allgemeinen Kosten zur Einführung bzw. Anpassung erheblich verringert.

5.1 Allgemeine Anforderungen

Um Projekte ab einer gewissen Größenordnung mit Mitteln der OpenSource Gemeinde zu restrukturieren, sind einige Vorbedingungen notwendig, die die eingesetzte Software erfüllen muss. In diesem Kapitel werde ich einige dieser Kriterien vorstellen.

5.1.1 Aktive Community

Eine der wichtigsten Kriterien im Bereich OpenSource ist eine aktive Community. Hierbei geht es nicht um die Zahl der Nutzer insgesamt, obwohl dies auch bei Unterschreiten einer gewissen Größe Auswirkungen auf die Projektqualität hat, sondern eher um die aktive Teilnahme der Nutzer einer Projektes oder Software am Entwicklungsprozess [uDHH03]. Aktive Teilnahme durch Dokumentationen, Howtos oder Patches ist für jedes erfolgreich angewendete Projekt von essentieller Wichtigkeit.

Im Falle von OpenSource werden Tests der Software zuerst nur im kleinen Kreis der Kerntwickler durchgeführt. Sobald eine Version entweder per Versionierungssystem oder als Release Candidate verfügbar ist, übernimmt die Community die Funktion der Regressionstester. Gerade diese Regressionstests unter hunderten oder tausenden von verschiedenen Bedingungen geben einem aktiven Projekt die nötige Stabilität und Skalierbarkeit zum produktiven Einsatz.

5.1.2 Aktives BugTracking

Ein aktiv gepflegtes BugTracking System ist für die Zusammenarbeit mit einer aktiven Community unerlässlich. Fehlerdokumentation oder die Verwaltung und spätere Eingliederung von Fehlerbehebungen in den Entwicklungszyklus machen dieses System zum Kommunikationskanal für Entwickler aus der Community. Auch um eventuell bei einer eigenen Implementation aufgetretene Fehler zu finden und eine eventuelle Lösung zu suchen, ist ein solches System unentbehrlich.

5.1.3 Dokumentation

Die größte Schwäche der meisten OpenSource Projekte, sind sie auch noch so innovativ bzw. interessant oder stabil, ist meist die fehlende Dokumentation. Es gibt bei einer progressiven Planung in einem eigenen Projekt nichts schlechteres als fehlende Dokumentation bestimmter Bereiche oder offene Fragen bezüglich wichtiger Bestandteile. Um Abschätzungen zur Skalierbar- oder Fehleranfälligkeit treffen zu können, oder sei es nur um bestimmte Aspekte der Interoperabilität mit anderen Projekten zu ermitteln, ist eine detaillierte und umfassende Dokumentation nötig. Oft kommt es vor, dass Projekte nicht eingesetzt werden, weil sie die Anforderungen nicht erfüllen, sondern weil eine Dokumentation fehlt. Dies bezieht sowohl Dokumentationen der Entwickler, als auch Anwendungsbeispiele der Nutzer, z.B. über eine Wiki System, mit ein. Zum Austausch von Erfahrungswerten sind diese meist unerlässlich.

5.1.4 Einsatzszenarien

Produktive oder wissenschaftliche Einsatzszenarien in großen Unternehmen bzw. Universitäten sind ein gutes Kriterium für neue oder interessierte Benutzer, das Projekt im Kontext der eigenen Anwendung zu evaluieren. Vor- und Nachteile des Projektes unter speziellen Anwendungskriterien lassen sich meist über diese so genannte ShowCases identifizieren. Eine Projektbeschreibung, das Einsatzgebiet, die aufgetretenen Schwierigkeiten und der Projektzeitraum sind meist die Fakten, die ein potenzieller Interessent klären möchte. In Kombination mit einer aktiven Community, in der es möglich ist, mit den Initiatoren dieser Einsatzszenarien direkt in Kontakt zu treten, ersetzt die Möglichkeit eines aktiven Supports.

5.1.5 Lizenzierung

Die freie Verfügbarkeit unter *GPL*, *GPL2* oder *LGPL* ist unerlässlich für den Einsatz von OpenSource Projekten [uDHH03]. Dies hat auf der Seite des Einsetzenden den Vorteil der Kostenersparnis für die Anschaffung und den aktiven Support der Applikation. Ein weiterer Vorteil ist die Sicherheit vor Patentansprüchen Dritter, was in letzter Zeit einen nicht unerheblichen Faktor in der Einsatzentscheidung dargestellt hat.

5.2 Anforderungen - Fehlerszenarien

Um den Kontext zum aktuell geplanten Projekt wieder herzustellen, werden in diesem Kapitel ausführlich die konkreten Anforderungen an das Projekt- und Entwicklungsframework *ATLAS*¹ erläutert. Beginnen wird das Kapitel mit den Anforderungen an die Überbrückung von verschiedenen Fehlerquellen, seien diese von aussen, durch Kommunikation oder Standort, oder innen, durch Hardwareausfälle oder Kommunikationsprobleme, induziert. Als nächstes werden wir die nötigen administrativen Vorgänge unter die Lupe nehmen. Im Abschluss dieses Kapitels werde ich ebenfalls die Failover-Mechanismen sowohl auf Systemebene, als auch auf Serviceebene, innerhalb der geplanten Umgebung analysieren.

5.2.1 Disaster recovery - System crash

Der wohl kritischste Abschnitt der Ausfallsicherheit ist die dahinterliegende Hardware. Ohne funktionierenden Speicher oder Festplatten ist das beste Abstraktionsmodell völlig

¹[A]rvato [T]eleservice [L]ogistic [A]pplication [S]ervice

nutzlos. Daher werden die wichtigsten, oft beanspruchten Hardwarekomponenten redundant oder als Failover ausgelegt.

5.2.1.1 Stromversorgung

Das wichtigste Element zum Betrieb eines Rechnerpools ist Strom. Ohne Strom, kein Rechner. Daher sind alle Netzteile der eingesetzten physikalischen Systeme redundant ausgelegt und separat über Sensoren überwachbar.

5.2.1.2 Festplatten

Auch der Datenbestand des kompletten Frameworks, obwohl redundant per Cluster-Filesystem auf mehreren Rechnern verteilt, wird auf Hardwareebene durch ein RAID-1 System unterstützt. Auch dieser RAID-1-Verbund wird ständig durch Sensoren überwacht und kann ggf. Fehler schnell, durch Austausch und Reorganisation der defekten Platten, beheben.

5.2.2 Disaster recovery - Communication failure

Ein zweiter, häufiger vorkommender Vorfall ist der Ausfall einer Netzwerkkarte oder der Abbruch der Verbindung des Serverpools untereinander. Hierbei ist es essentiell, eine redundante Route zum physikalischen Serverpool über eine zweite Netzwerkkarte herstellen zu können. Diese redundante Route sollte nach Möglichkeit transparent vom System selbst hergestellt werden, so dass eine möglichst kurze Ausfallzeit erreicht werden kann. Im Falle eines Hardwarefehlers sollte das System notwendigerweise sofort über einen entsprechenden Netzwerk-Bonding eine alternative Route über eine zweite Netzwerkkarte aufbauen. Im Falle eines reinen Software- oder Treiberfehlers sollte dies ebenfalls geschehen. Sollten beide Mechanismen nicht funktionieren, so muss sich die betreffende Maschine selbständig aus dem Rechnerpool entfernen und eine andere Maschine übernimmt, durch redundante Auslegung der Services, deren Arbeit.

5.2.3 Planned maintenance and shutdown

Nicht ganz dem Bereich der Fehler zuzuordnen, aber meist mit den gleichen Auswirkungen, sind administrative Notwendigkeiten, das gesamte physikalische System herunterzufahren. Dies kann z.B. durch ein Hardwareaustausch oder eine Hardwareerneuerung notwendig sein. In diesem Fall bildet das System der Online-Migration von Xen™ die perfekte Möglichkeit zur Überbrückung dieser Ausfallzeiten.

Vor einem geplanten „Ausfall“ werden alle virtuellen Systeme von einem zu wartenden physikalischen System auf ein anderes umgezogen. Dann wird die physikalische Maschine heruntergefahren und eventuelle Arbeiten daran werden getätigt. Wenn alle Arbeiten abgeschlossen sind, wird der physikalische Rechner wieder hochgefahren und die virtuellen Systeme werden zurückmigriert. Die in den virtuellen Umgebungen laufenden Applikationen bekommen auf Grund ihrer Isolierung von der physikalischen Hardware von diesem Vorgang nichts mit. Im Falle eines Webservers, der als *Appliance* in einer solchen Umgebung betrieben wird, kommt es in Bezug auf die Latenz nach außen, auf Grund der sehr geringen Migrationskosten an Zeit und Ressourcen, zu keinerlei Störungen.

5.3 Anforderungen - Systeme


Die Betrachtung der möglichen Hardware bildet im System der Anforderungen die eine Seite der Medaille. Die andere Seite bildet die Betrachtung der Host- bzw. Gast-Systeme.

Diese Systeme müssen einen Teil der Fehlerquellen möglichst selbst beheben können und jederzeit synchronisiert und überwachbar sein.

5.3.1 Skalierbarkeit

Eine der Hauptanforderung konzeptioneller Art ist die Beherrschung der Skalierung im Zuge des Wachstums der Projekte oder des Unternehmens. Neue Projekte sollten isoliert aufgesetzt und beliebig vergrößert werden können. Die physikalischen Maschinen sollten idealerweise von der Skalierung der Projekte unabhängig sein. Ebenso sollte eine Skalierung nach unten, also eine Verkleinerung der Projekte und Rechnerpools, ohne großen administrativen Aufwand möglich sein. Insbesondere sollte eine Erweiterung oder Verkleinerung ohne Ausfall oder geplantes Herunterfahren einhergehen können. Das Hauptaugenmerk der gesamten Tätigkeit in einem Projekt sollte in der Entwicklung des Projektes und nicht seiner Umgebung als solches liegen. Daher sollten die administrativen Tätigkeiten sowohl auf ein Minimum beschränkt, als auch leicht auf andere Personen übertragbar sein. Dies spart sowohl Kosten für die Administration als auch für die Projektübergabe der Administration ein.

5.3.2 Betriebssystem

Das zum Einsatz kommende Betriebssystem sollte frei verfügbar und von Lizenzkosten Dritter befreit sein. Es soll Mehrbenutzerbetrieb ebenso zur Verfügung stellen können, wie Einzelnutzerbetrieb. Das Betriebssystem an sich sollte leicht wartbar und ein eventuell eingesetztes Paketsystem muss konsistent und überschaubar sein. Alle Änderungen an einem System müssen dokumentier- und nachvollziehbar sein. Es sollte regelmäßige Sicherheitsupdates und Fehlerbeseitigungen für die im Betriebssystem eingesetzte Software geben und außerdem sollte es möglich sein, mehrere Softwareversionen einer Software zu installieren (*Multislottting*² ). Außerdem gelten für die Betriebssysteme die Anforderungen aus dem Kapitel „5.1 Allgemeine Anforderungen (S.23)“.

5.3.3 Dateisystem

Das zu Einsatz kommende Dateisystem sollte ebenfalls allen Anforderungen an Stabilität und Skalierbarkeit gerecht werden. Augenmerk ist hierbei auf Grund des Konzeptes der Onlinemigration durch Xen™ auf die Ausfallsicherheit zu legen. Das Dateisystem muss, ebenso wie die physikalischen Komponenten, nach oben bzw. unten skalierbar sein. Um von dem Ansatz der SAN Lösung abzukommen, sollte das Dateisystem asynchrone, aber redundante Schreibweisen in einer verteilten Umgebung beherrschen. Hierbei muss das Dateisystem dem POSIX Standard entsprechen und auch das simultane Einbinden auf mehreren Systemen gleichzeitig sollte für das Dateisystem keine Schwierigkeiten darstellen. Es muss über die Mechanismen des *Quorum* und *Fencing* bzw. äquivalente Mechanismen zur Einhaltung von Locks in verteilten Systemen beherrschen. Alle nötigen Werkzeuge zur Datensicherung und Wiederherstellung nach einem Fehlerfall sind obligatorisch.

5.3.4 Backup

Im Zusammenhang mit der Ausfallsicherheit und Datenkonsistenz sind Datenbackups in verschiedenen Variationen anzulegen. Dies können zum einen ganze Abbilder einzelner Systeme, zum anderen nur einzelne Logfiles oder auch ganze Datenbank- oder Subversion-Repositories sein. Die Daten sind je nach Dringlichkeit einer Wiederherstellung bzw. nach Wichtigkeit für das betreffende System, jeweils in einem variierenden Rhythmus, skalierend

²Multislottting: <http://devmanual.gentoo.org/general-concepts/slotting/index.html>

von stündlich bis jährlich, zu sichern. Ebenfalls sollten Backups, die länger zurückliegen oder besonders wichtig sind, regelmäßig auf externe Festspeicher zur Archivierung verlagert werden.

5.4 Anforderungen - Services

Dieses Kapitel wird sich hauptsächlich mit der inneren Redundanz, also der redundanten Softwarekomponenten *in* einem virtuellen Rechnerpool befassen. Auch hier kommen wieder sämtliche vorherigen Anforderungen zum Tragen.

5.4.1 Load Balancing und Failover

Alle Services (z.B. Apache, MySQL, SOAP) in einem entsprechenden System müssen eine Skalierung nach unten oder oben gefahrlos zulassen. Gerade im Bereich der Serverdienste wie Apache sollte sich durch den Einsatz gleicher Systeme gezielt eine Load Balancing Struktur aufbauen lassen. Alle Services werden durch ein virtuelles Pendant auf einem physikalisch anderen Rechner redundant gehalten. So kann im Falle eines physikalischen Fehlers auf einer Maschine nicht gleich der komplette Service ausgeschaltet werden. Die Redundanzen übernehmen, bis der Fehler behoben ist, die Arbeit des originalen Services.

5.4.2 Firewall

Wie auch schon im vorherigen Kapitel erwähnt, ist die Firewall natürlich ebenfalls redundant zu halten. Eine Besonderheit sollte hier jedoch erwähnt sein: Es sind nicht für jeden Host einzelne Firewalls von Nöten. Die virtuellen Maschinen können entweder direkt miteinander bzw. über das Host-System kommunizieren oder befinden sich als Gruppe in einem eigenen VLAN. Hierbei kann diese Aufgabe eine einzelne virtuelle Umgebung mit entsprechender Firewall pro Projekt- oder Rechnerpool erledigen. Hochgesicherte Systeme können aber selbst bei diesem Konzept nochmals eine eigene Firewall beherbergen.

5.5 Anforderungen - Überwachung

Die eingesetzten physikalischen und virtuellen Systeme müssen jederzeit überwachbar sein und eine sichere Umgebung darstellen. Dies können simple Mechanismen wie Suchmaschinen für Viren und Rootkits sein, aber auch komplexe IDS-Systeme. Alle Mess- und Auswertungsdaten sollten zentral an einer Stelle abruf- und überwachbar sein. Das Überwachungssystem sollte ebenfalls, bei bestimmten Fehlersituationen, autonom reagieren können. Eine dedizierte Hierarchie bei der Benachrichtigung der Verantwortlichen bzw. Administratoren muss gewährleistet sein.

Alle Auswertungs-, System- und Servicemeldungen sind sowohl direkt auf den jeweils physikalischen bzw. virtuellen Rechnern, als auch auf einer gesicherten Umgebung zu speichern. Zu Reporting- und Nachweiszwecken sind alle Daten nach täglichen, monatlichen und quartalsmäßigen Daten zu sortieren. Alle Daten sind wöchentlich oder monatlich, je nach Bedarf, extern auf einen Festspeicher zu sichern.

6 Projekt ATLAS

Im diesem Kapitel werde ich den konzeptionellen Aufbau des Frameworks „*ATLAS*“ erörtern, um letztendlich im Abschluss dieser Ausarbeitung zur konkreten Umsetzung zu gelangen.

6.1 Konzeptioneller Aufbau

Der Aufbau von ATLAS gliedert sich in mehrere konzeptionelle Ebenen, die unabhängig voneinander austauschbar sind. Die unterste Ebene beginnt mit der Hardwareausstattung. Dann kommen die Ebenen der Datensicherung und der Datenkonsistenz. Gefolgt werden

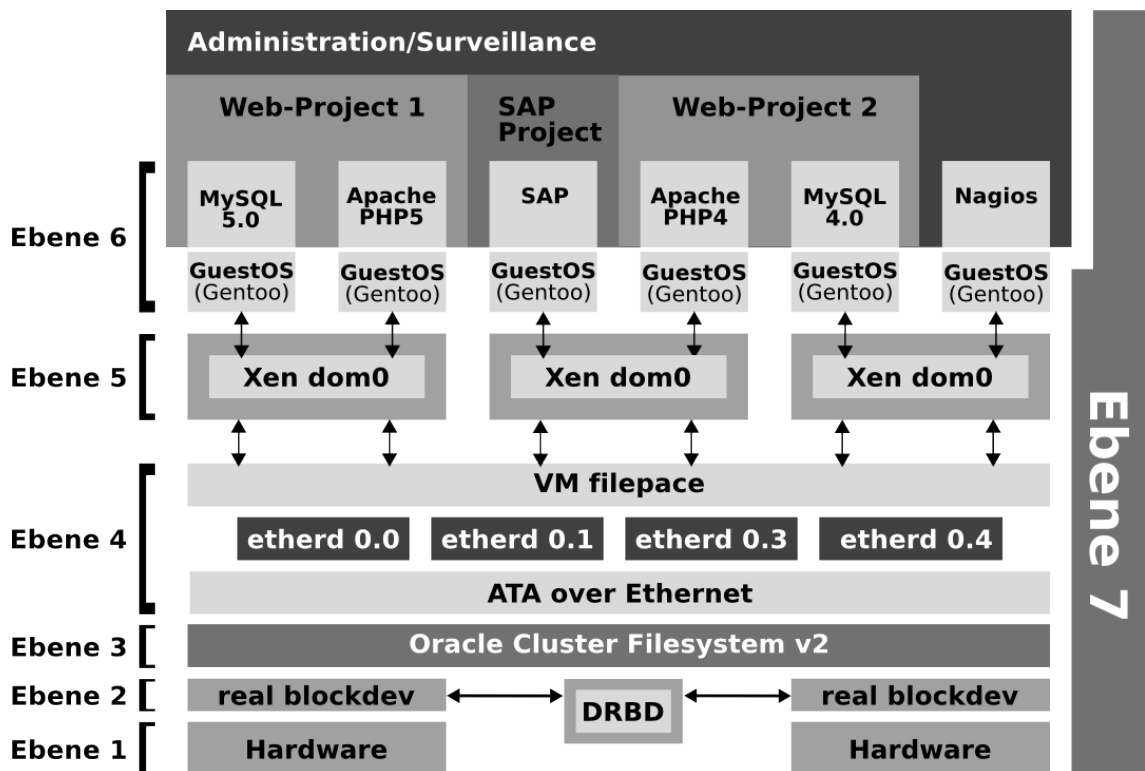


Abbildung 6.1: ATLAS Systemebenen

diese Ebenen von der Datenbereitstellungsschicht und der Systemabstraktion, sowie der virtuellen Maschinen. Als oberste Ebene folgt die Managementschicht.


6.2 Ebene 1 - Physikalische Rechner

Die physikalischen Rechner des ATLAS-Frameworks bestehen zumeist, außer bei speziellen Anforderungen, aus normalen Home- bzw. Root-Servern diverser Anbieter. Die folgenden Erläuterungen beziehen sich jedoch nur auf die bereits in diversen Rechenzentren integrierten Root-Server. Die Ausstattung dieser Server variiert sowohl in der Speicherausstattung als auch in der Ausstattung der Festplatten, Prozessoren und Netzwerkkarten.

6.2.1 Hardware

Die kleinste Ausstattung ist hierbei ein AMD 64 mit 3,2 GHz, 2 GB Speicher und 160 GB Festplatte im RAID-1 Verbund. Dies ist das Minimum an Rechenleistung und Speicher, das in ATLAS verbaut werden sollte. Zwar ist der Prozessor schnell genug für ein einfaches System, doch bei zwei oder mehr Systemen gleichzeitig, wird die Rechenreserve immer kleiner. Jeder der eingesetzten Rechner muss mindestens über 2 GB Speicher verfügen, um die virtuellen Systeme versorgen zu können. Alle Rechnersysteme sind jeweils mit zwei Netzteilen ausgestattet, um eine redundante Stromversorgung zu gewährleisten. Root-Server mit dieser Ausstattung lassen sich mittlerweile mit unter 100 € pro Monat hosten.

6.2.2 Festplatten

Alle Datenspeicher entsprechen in ihrer Größe mindestens zweimal 160 GB und laufen in einer RAID-1 Formation, um die nötigen Redundanzen auf Hardwareebene zu verschaffen. Die Platten werden permanent durch *SMARTMonTools*¹  auf ihren S.M.A.R.T.-Status hin überprüft. Die physikalischen Festplatten sind somit ausreichend abgesichert. Das RAID-1 wird auch hardwareseitig durch den jeweiligen Provider überwacht und bei einem Fehler ggf. ausgetauscht.

Listing 6.1: Installation SMARTMonTools

```

1 MasterNode1 ~ $ emerge sys-apps/smartmontools
2 MasterNode1 ~ $ nano /etc/smartd.conf
3     ...
4     DEVICSCAN
5     /dev/sdx -a -d scsi -o on -S on -s (S/../../02 \
6                                     |L/../../03)
7     ...
8 MasterNode1 ~ $ rc-update add smartd default
9 MasterNode1 ~ $ /etc/init.d/smartd start

```

6.2.3 Kommunikation

Die Netzwerkkarten der einzelnen Rechner sind ebenfalls redundant ausgelegt, um einem Hardwarefehler zuvorzukommen. Alle Karten haben mindestens eine GigaBit-Übertragungsleistung, um die nötige Bandbreite für Datentransfers, wie bei der Onlinemigration zur Verfügung zu stellen. Außerdem ist darauf zu achten, dass keine der Netzwerkkarten „on-board“ ist, da hierfür die Rechenleistung des Rechners für die Verarbeitung der Netzwerklast mißbraucht wird. Dies ist nicht wünschenswert. Die Karten laufen im so genannte Bonding-Modus [Soc00], d.h. eine IP-Adresse wird von mehreren physikalischen Netzwerkkarten bedient. Sollte eine dieser Karten ausfallen, übernimmt die andere Netzwerkkarte des virtuellen Bonding-Gerätes den Datentransfer. Diese Bonding-Geräte werden durch das Überwachungswerkzeug Nagios überwacht.

Ein Element dieses Rechnerverbundes lässt sich wie folgt verdeutlichen:

- CPU: Intel DualCore 2,4 GHz
- Speicher: min. 2 GB
- Festplatte: min. 300 GB
- Controller: RAID Controller

¹SMARTMonTools: <http://smartmontools.sourceforge.net/>

- Netzwerk: min. 2x GigaBit LAN
- Netzteil: redundant (2x) 450 Watt
- Sonstiges: Serielle Konsole, Reboot Service, Rescue System

6.3 Ebene 2 - Datensicherung

Im Zuge der Datensicherung der einzelnen Systeme kommen neben mehreren Backupservern auch Systeme mit *DRBD 0.8.0* zum Einsatz. Gerade Systeme, die die Abbilder der virtuellen Umgebungen beherbergen, sind mit *DRBD* gesichert. Jeweils zwei physikalische Maschinen bilden hierbei einen Datenknoten.

Listing 6.2: DRBD: Installation

```

1 MasterNode1 ~ $ emerge =sys-cluster/drbd-0.8.0
2 MasterNode1 ~ $ nano /etc/drbd.conf
3     resource mirror {
4         on test1 {
5             device      /dev/drbd0;
6             disk        /dev/hda5;
7             address     192.168.1.1:7788;
8             meta-disk   /dev/hda6[0];
9
10        on test2 {
11            device      /dev/drbd0;
12            disk        /dev/hda5;
13            address     192.168.1.2:7788;
14            meta-disk   /dev/hda6[0];
15        }
16    }

```

Diese Datei muss auf der Gegenstelle ebenfalls gleich lauten:

Listing 6.3: DRBD: copy

```

1 MasterNode1 ~ $ scp /etc/drbd.conf user@192.168.1.2:/home/user/
   drbd.conf

```

Und an beiden Endpunkten muss natürlich der *DRBD*-Dienst gestartet werden:

Listing 6.4: DRBD: Init

```

1 MasterNode1 ~ $ rc-update add drbd default
2 MasterNode1 ~ $ /etc/init.d/drbd start
3 MasterNode1 ~ $ drbdadm -- --do-what-I-say primary all
4 MasterNode1 ~ $ cat /proc/drbd
5     version: 0.8.0 (api:77/proto:74)
6     SVN Revision: 2093 build by tester@test1, 2007-01-27
7     12:41:17
8     0: cs:SyncSource st:Primary/Secondary ld:Consistent
9     ns:122084 nr:0 dw:662016 dr:782689 al:165 bm:174 lo
10    :19 pe:67 ua:82 ap:0
11    [==>.....] sync'ed: 19.2%
12    (540188/662008)K
13    finish: 0:01:45 speed: 5,080 (4,872) K/sec
14    ...

```

Wie man hier erkennen kann, synchronisiert *DRBD* bereits das Block-Gerät. Nun muss noch dafür gesorgt werden, dass *DRBD* bei jedem Start die Platten synchronisiert:

Listing 6.5: DRBD: /etc/conf.d/localstart


```

1 MasterNode1 ~ $ nano /etc/conf.d/local_start
2     ...
3     drbdadm primary all
4     ...

```

Die Block-Geräte der beiden Maschinen werden über DRDB synchron gehalten, um im Falle eines Ausfalls über *heartbeat* die andere Maschine dazu zu bringen, die Belieferung mit Daten zu übernehmen.

6.4 Ebene 3 - Datenkonsistenz

Für die Auswahl der Cluster-Dateisysteme stehen zwei Kandidaten zur Verfügung: Das schon im vorherigen Kapitel erwähnte *OCFS2* und das von RedHat entwickelte *Global File System*² . Da jedoch GFS keine dateibasierte Einbindung (so genannte loop-devices) beherrscht [Cah04], scheidet es aus, bleibt aber unter Beobachtung. Der Bereich der Datenkonsistenz wird daher auf Dateisystemebene von *2.5 Oracle Cluster File System (OCFS2) (S.9)* übernommen. Es sorgt durch *Quorum* und *IO-Fencing* dafür, dass die gemeinsam genutzten Ressourcen, die per ATA over Ethernet exportiert werden, nicht bei gleichzeitiger Nutzung aus dem Tritt geraten. Um *OCFS2* verwenden zu können, sind einige Einstellungen im Kernel notwendig:

Listing 6.6: OCFS2: Kernelmodul

```

1 --> File systems
2     --> <M> OCFS2 file system support (EXPERIMENTAL)

```

Listing 6.7: OCFS2: Kernel Autostart

```

1 MasterNode1 ~ $ nano /etc/modules.autoload.d/kernel-2.6
2     ...
3     ocfs2
4     ...

```

Dieser Schritt ist auf allen Nodes, die das Dateisystem einbinden wollen, gleich. Um den *OCFS2*-Dienst erfolgreich starten zu können, ist noch eine Anpassung der */etc/fstab* notwendig:

Listing 6.8: OCFS2: /etc/fstab

```

1 MasterNode1 ~ $ nano /etc/fstab
2     ...
3     none /config configfs
4     defaults,noauto 0 0
5     none /dmlm ocfs2_dlmfs
6     defaults,noauto 0 0
7     ...

```

Ein Block-Gerät, welches durch *OCFS2* verwaltet werden soll, muss natürlich erst initialisiert werden:

Listing 6.9: OCFS2: mkfs.ocfs2

```

1 MasterNode1 ~ $ mkfs.ocfs2 -L ocfs2Disk01 /dev/sdx

```

²Global File System: <http://sources.redhat.com/cluster/gfs/>

Nachdem das Block-Gerät initialisiert wurde, bleibt noch die Anpassung von *OCFS2* selbst. In diesem Schritt wird der Verwaltungsschicht von *OCFS2*, dem so genannte Domain-Locking-Manager (DLM) mitgeteilt, welche Systeme Zugriff auf das gemeinsam genutzte Medium haben. In unserem Falle sind es zwei Systeme:

Listing 6.10: OCFS2: /etc/ocfs2/cluster.conf

```

1 MasterNode1 ~ $ nano /etc/ocfs2/cluster.conf
2     cluster:
3         node_count = 2
4         name = atlas
5
6     node:
7         ip_port = 7777
8         ip_address = 10.1.1.2
9         number = 0
10        name = node01
11        cluster = atlas
12
13    node:
14        ip_port = 7777
15        ip_address = 10.1.1.3
16        number = 1
17        name = node02
18        cluster = atlas

```

Listing 6.11: OCFS2: /etc/conf.d/ocfs2

```

1 MasterNode1 ~ $ nano /etc/conf.d/ocfs2
2     ...
3     OCFS2_CLUSTER="'atlas'"
4     ...


```

Listing 6.12: OCFS2: /etc/conf.d/localstart

```

1 MasterNode1 ~ $ nano /etc/conf.d/local_start
2     ...
3     echo "'61'" > /config/cluster/atlas/heartbeat/
4     dead_threshold
5     ...

```

Die letzte Einstellung verhindert ein vorzeitiges *Fencing* der einzelnen Systeme durch die Latenz von *ATA over Ethernet*. Diese Dateien **müssen** auf allen, am Dateicluster beteiligten, Systemen absolut gleich sein. Die Syntax kann dem *OCFS2 UserGuide* ³  entnommen werden. Nun noch den *OCFS2*-Dienst starten:

Listing 6.13: OCFS2: Autostart

```

1 MasterNode1 ~ $ rc-update add ocfs2 default
2 MasterNode1 ~ $ /etc/init.d/ocfs2 start

```

Das mit *OCFS2* verwaltete Block-Gerät steht nun bereit zum Export durch *ATA over Ethernet*.

6.5 Ebene 4 - Datenbereitstellung

Alle aus 6.3 Ebene 2 - Datensicherung (S.31) bekannten Block-Geräte werden mittels *ATA over Ethernet* im Netzwerk bekannt gemacht und können von den einzelnen Host-Systemen

³OCFS2 UserGuide: http://oss.oracle.com/projects/ocfs2/dist/documentation/ocfs2_users_guide.pdf

eingebunden und gleichzeitig verwendet werden. Um *ATA over Ethernet* verwenden zu können, müssen einige Anpassungen am Kernel vorgenommen werden:

Listing 6.14: AoE: Kernel Modul

```
1 Device Drivers --->
2   Block devices --->
3     <M> ATA over Ethernet support
```

Das Modul muss natürlich beim Systemstart entsprechend geladen werden:

Listing 6.15: AoE: Autostart

```
1 ...
2 MasterNode1 ~ $ nano /etc/modules.autoload.d/kernel-2.6
3   ...
4   aoe
5   ...
```

Nach einem Neustart des Systems steht nun die Funktionalität von *ATA over Ethernet* zur Verfügung. Um *ATA over Ethernet* jedoch außerhalb der Kernelebene benutzen zu können, werden die *aoetools* benötigt. Auf der Serverseite wird nun der Dienst *vblade* initialisiert, um per *ATA over Ethernet* ein Block-Gerät zu exportieren:

Listing 6.16: aoetools: Installation

```
1 MasterNode1 ~ $ emerge =sys-block/aoetools
2 MasterNode1 ~ $ nano /etc/conf.d/vblade
3   ...
4   config_vblade0="'0 0 eth0 /dev/sdx'"
5   ...
6 MasterNode1 ~ $ rc-update add vblade.vblade0 default
7 MasterNode1 ~ $ /etc/init.d/vblade.vblade0 start
```

Nach dem erfolgreichen Start des Dienstes, lassen sich die exportierten Block-Geräte auf einem anderen beliebigen Rechner im Netzwerk mittels *aoe-stat* überprüfen.

Listing 6.17: aoetools: aoe-stat

```
1 MasterNode1 ~ $ aoe-stat
2   e0.0          8.00GB  eth4  up
```

Auf den jeweiligen Rechnern lässt sich dieses Block-Gerät nun einfach per *mount* einbinden. Die Besonderheit hierbei ist das Einbinden per *LABEL* und nicht per Block-Knoten. Das Einbinden per *LABEL* verhindert hierbei, dass *OCFS2* sich über eine falsche UID des Block-Gerätes beschwert:

Listing 6.18: AoE: mount

```
1 MasterNode1 ~ $ mount -t ocfs2 -L ocfs2Disk01 /mnt/ocfs2
2 MasterNode1 ~ $ mount
3   /dev/etherd/e0.0 on /mnt/ocfs2 type ocfs2
```

Um die Block-Geräte beim Start einer Node automatisch einzubinden, sind nur noch folgende Änderungen nötig:

Listing 6.19: AoE: /etc/fstab


```
1 MasterNode1 ~ $ nano /etc/fstab
2   ...
3   LABEL=ocfs2Disk01          /mnt/ocfs2          ocfs2
4   rw,defaults                0 0
5   ...
```


Bei jedem Systemstart wird so das Gerät mit der Bezeichnung *ocfs2Disk01* eingehängt.

6.6 Ebene 5 - Systemabstraktion

Als Abstraktionskonzept kommt Xen™ zum Einsatz. Zum einen hat Xen™ anderen Virtualisierungslösungen bei weitem einen Performancevorsprung [ea04] voraus, zum anderen bietet es die Möglichkeit der Onlinemigration.

6.6.1 Verwaltungsschicht - dom0

Als Betriebssystem kommt auf allen Host-Systemen *Gentoo 2006.1*⁴  zum Einsatz. Diese Linux Distribution erfüllt alle o.g. Anforderungen an Lizenz, Community und Stabilität. Gerade die sehr aktive Community der Gentoo Gemeinde ist ein reicher Pool an Wissen und Erfahrung über alle möglichen Aspekte und Projekte des Betriebssystems und Linux/Unix im Allgemeinen.

Das einzelne Host-System sich basiert auf einer Standardinstallation von Gentoo. Der Prozessor und die Architektur sollten bei Systeminstallation so festgelegt werden, dass es auch später möglich ist, das Betriebssystem, egal ob 32 oder 64 Bit Prozessor bzw. Intel oder AMD, auf einem anderen Rechner einzusetzen. Dies kann durch geeignete *Einstellungen des GCC*⁵  unter Gentoo in der `/etc/make.conf` erreicht werden.


Listing 6.20: `/etc/make.conf`

```

1 ...
2 CHOST="i686-pc-linux-gnu"
3 CFLAGS="-march=i686 -pipe -O2 -fomit-frame-pointer"
4 CXXFLAGS="${CFLAGS}"
5 ...

```

Um diesen Abschnitt nicht unnötig in die Länge zu ziehen, zeige ich hier nur die wichtigsten, nicht in der Grundinstallation enthaltenen Programme, Services und Einstellungen auf. Am Ende dieser Ausarbeitung sowie auf der beiliegenden DVD sind alle Paketauflistungen enthalten.

Der Kernel des Systems basiert auf dem *Vanilla-Kernel 2.6.18*⁶  Abbild und enthält außerdem alle Patchsets für Xen™ 3.0.4. Hier die wichtigsten Einstellungen des Kernels auf einen Blick:

Listing 6.21: dom0: Kernel

```

1 File systems -->
2     <M> OCFS2 file system support (EXPERIMENTAL)
3
4 Device Drivers -->
5     Block devices --->
6         <M> ATA over Ethernet support
7
8 Processor type and features --->
9     [*] Enable Xen compatible kernel
10
11 XEN -->
12     [*] Privileged Guest (domain 0)
13     <*> Block-device backend driver
14     <*> Network-device backend driver

```

⁴Gentoo 2006.1: <http://www.gentoo.org>

⁵Einstellungen des GCC: http://gcc.gnu.org/onlinedocs/gcc-3.4.3/gcc/i386-and-x86_002d64-Options.html



⁶Vanilla-Kernel 2.6.18: <http://kernel.org/>

```

15      <*>   Network-device loopback driver
16      [*]   Scrub memory before freeing it to Xen
17      <*>   Export Xen attributes in sysfs
18
19 Networking --->
20     Networking options --->
21         [*] IP: tunneling
22         [*] 802.1d Ethernet Bridging
23
24 Device Drivers --->
25     Block devices --->
26         <*> Loopback device support

```

6.7 Ebene 6 - Virtuelle Maschinen

Die virtuellen Systeme der einzelnen *Appliances* bestehen ebenfalls aus einem *Gentoo 2006.1* ⁷  Grundsystem mit *Kernel 2.6.16* ⁸ . Der Kernel ist entsprechend den Bedürfnissen der virtuellen Maschinen angepasst. Hier die wichtigsten Kernelparameter des Grundsystems:

Listing 6.22: domU: Kernel


```

1 Processor type and features --->
2     [*] Enable Xen compatible kernel
3
4 XEN -->
5     [ ] Privileged Guest (domain 0)
6     [ ] Block-device backend driver
7     [ ] Network-device backend driver
8     [*] Block-device frontend driver
9     [*] Network-device frontend driver
10    [ ] Piplined transmitter (DANGEROUS)
11    [*] Scrub memory before freeing it to Xen
12    Processor Type (X86) --->

```

Die restlichen Kernelparameter sind relativ großzügig ausgelegt, um nicht für jedes eingesetzte System einen neuen Kernel generieren zu müssen. Das Grundsystem wird je nach späterem Einsatzgebiet den Bedürfnissen in Systemkonfiguration und Programmauswahl angepasst. In den folgenden Kapiteln werde ich nicht detailliert auf die Installation der einzelnen Programme eingehen, sondern vielmehr den Sinn und ihren Zweck im Gesamtkonzept erläutern.

6.7.1 VM Firewall

Diese virtuelle Maschine liefert eine in sich abgeschlossene Umgebung zur Netzwerksicherung. Unter anderem wird mit Hilfe von *shorewall* ⁹  eine dedizierte Verwaltungsumgebung für das angeschlossene Netzwerk bereitgestellt. Diese VM dient als Sicherungsschicht für dahinterliegende VM's. Auch die Verwaltung der VLAN's einzelner Umgebungen wird hier festgelegt. Ebenfalls bringt diese Umgebung alle notwendigen Netzwerkverwaltungswerkzeuge mit.

⁷Gentoo 2006.1: <http://www.gentoo.org>

⁸Kernel 2.6.16: <http://kernel.org/>

⁹shorewall: <http://www.shorewall.net/>



6.7.2 VM Apache

Die VM-Apache stellt eine isolierte Umgebung für einen Webserver dar. Sie kann nach Belieben angepasst werden, um z.B. einen Tomcat-Server oder beliebige Module wie PHP, PERL oder WebDav zu enthalten.


6.7.3 VM Datenbank

Datenbanken wie MySQL, PostGreSQL oder SAP-DB sind in dieser VM untergebracht. Vorteilhaft ist hierbei die eigene Clusterfähigkeit der einzelnen Datenbanksysteme. Dies ermöglicht es, einzelne VM's zu starten und in einen bereits bestehenden Datenbankcluster zu integrieren.

6.7.4 VM Backup

Für Backups beliebiger Art ist diese VM zuständig. Unter anderem werden hier die täglichen und wöchentlichen Backups verwaltet. Das System ist mit einigen Backup-Programmen wie *csync2*¹⁰  und *flexBackup*¹¹  für dedizierte Backupstrategien ausgestattet und ermöglicht so sowohl volle, als auch inkrementelle Backups beliebiger Art.

6.7.5 VM Nagios

Eine der wichtigsten VM-Lösungen ist momentan die der Überwachung durch *Nagios*¹² . Nagios bietet eine breit gefächerte Zahl von Sensoren beliebiger Art. Die meisten anderen VM's enthalten mindestens einen, normalerweise aber mehrere dieser Sensoren zur Überwachung einzelner Prozesse oder des Systems selbst. Auch die Überwachung der Netzwerkinfrastruktur, wie Router, Switches und Netzwerkverbindungen wird per Nagios veranlasst. Nagios bildet hierbei ein sehr gutes Reporting- und Meldungssystem ab.

¹⁰csync2: <http://oss.linbit.com/csync2/>

¹¹flexBackup: <http://www.edwinh.org/flexbackup/>

¹²Nagios: <http://www.nagios.org>

6.8 Ebene 7 - ATLAS Management

Das ATLAS Management wird momentan noch manuell gemacht, es stehen jedoch zwei Frameworks zur Verwaltung in der Evaluationsphase.

6.8.1 BixData Server Management Console

Neben den eigentlichen *xen-tools*, die in der Systeminstallation von Xen™ enthalten sind, kann für das Management der physikalischen und virtuellen Systeme *BixData*¹³ zum Einsatz kommen. BixData kann Informationen über die eingesetzten Systeme auf drei ver-

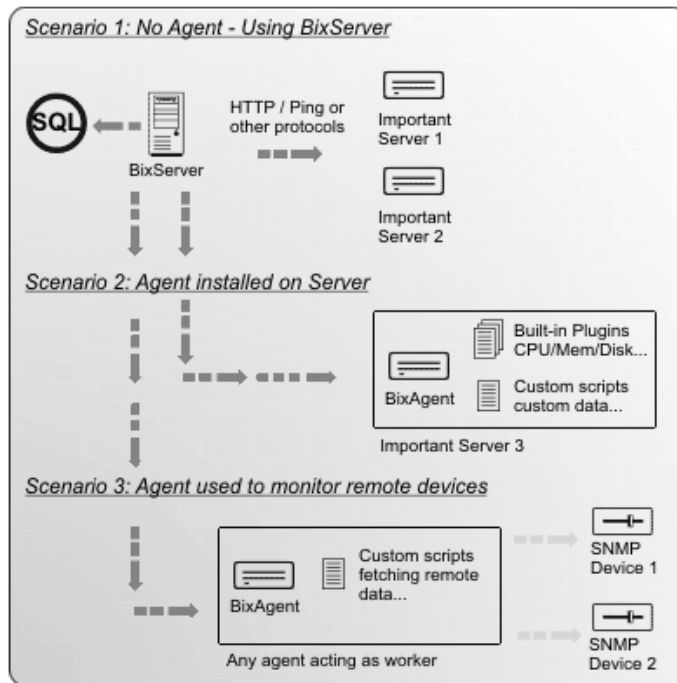


Abbildung 6.2: BixData Aufbauschema

schiedenen Wegen bekommen. Der erste Weg ist der direkte: Von einem BixServer aus werden Informationen, die über die Systeme von außen (z.B. Pingzeiten, SNMP) erreichbar sind, abgefragt. Der zweite Informationsweg benutzt so genannte *Agents*, Dienste, die direkt auf den Systemen laufen, um Informationen von Innerhalb des Systems zu erreichen. Der dritte Weg ist die indirekte Kommunikation zwischen zwei *Agents*, deren Informationen dann im BixServer gesammelt werden. BixData bietet eine ausgereifte und

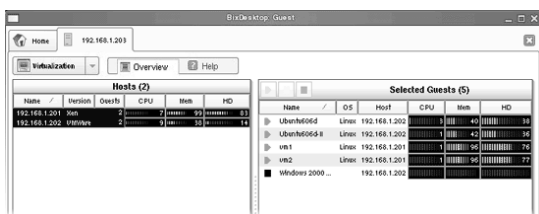


Abbildung 6.3: BixData GUI

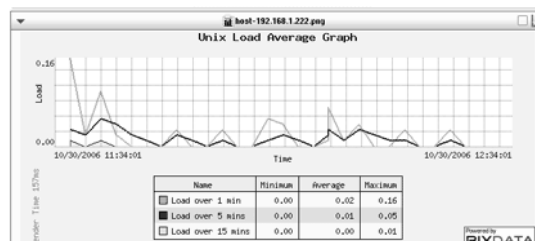


Abbildung 6.4: BixData Reporting

vollständige Oberfläche zum Verwalten von physikalischen Servern, virtuellen Umgebun-

¹³BixData: TODO

gen, Netzwerken und Applikationen. Im Gegensatz zu *openQRM* (siehe 6.8.2 *openQRM* (S.39)), ist BixData sehr schnell und einfach installiert, bietet jedoch nicht den gesamten Funktionsumfang von *openQRM*. Ebenfalls ist zu beachten, dass BixData lediglich eine so genannte *Community Edition* seines kommerziellen Produktes zur Verfügung stellt. *openQRM* hingegen ist unter der MPL, einer an die GPL angelehnten Lizenz, zu erhalten.

6.8.2 openQRM

Der große Bruder von BixData ist *openQRM*¹⁴. *openQRM* ist als DataCenterlösung gedacht und kann ganze Infrastrukturen samt Netzwerken, Switches und Routern verwalten. Es baut hierbei auf einigen zentralen Servern auf, die per NFS Gast-Systeme zur Verfügung stellen. Die Gast-Systeme booten und laden sich hierbei über DHCP (so genannte PXE [Sys99]) von diesem zentralen Servern. Der Aufbau von komplexen Strukturen

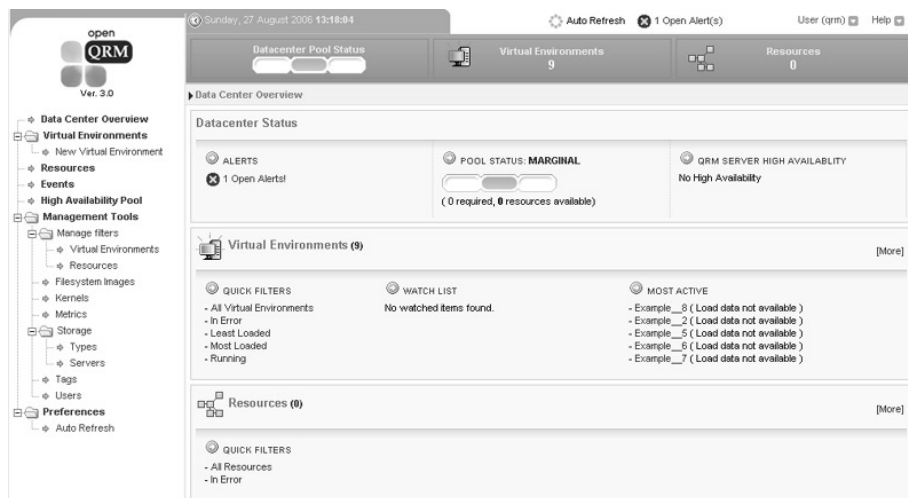


Abbildung 6.5: openQRM

kann ebenfalls mit einer Nagios ähnlichen Überwachungsinfrastruktur beobachtet werden. Genau wie sein kleiner Bruder *BixData*, stellt auch *openQRM* umfangreiche Reporting- und Managementmechanismen zur Verfügung, die sogar bei Fehlern selbständig reagieren können.

¹⁴openQRM: <http://www.openqrm.org/>

6.9 Verbesserungen

6.9.1 Lustre statt OCFS2

Wie schon oben erwähnt, kann sich das Konzept eines Cluster-Dateisystems bedienen, um die hohen Kosten einer SAN-Lösung zu umgehen. Zum Einsatz kann hier das Cluster-Dateisystem *Lustre*¹⁵ in der Version 1.6 beta8 kommen. Lustre ist nach eingehender Evaluation ein sehr vielversprechendes Projekt. Seine *Roadmap*¹⁶ enthält einige Funktionalitäten, die andere Cluster-Dateisysteme bei weitem missen lassen [Wik07]. Lustre basiert grundlegend auf dem Client-Server Prinzip, um Daten effektiv und redundant auf mehreren Rechnern zu verteilen. Lustre verfügt über eine ausgereifte Verwaltung der Metadaten des jeweiligen Dateisystems in eigenen *Metadata Servers (MDS)*. Diese MDS koordinieren die kompletten Zugriffe auf das verteilte Dateisystem. Die eigentlichen Daten

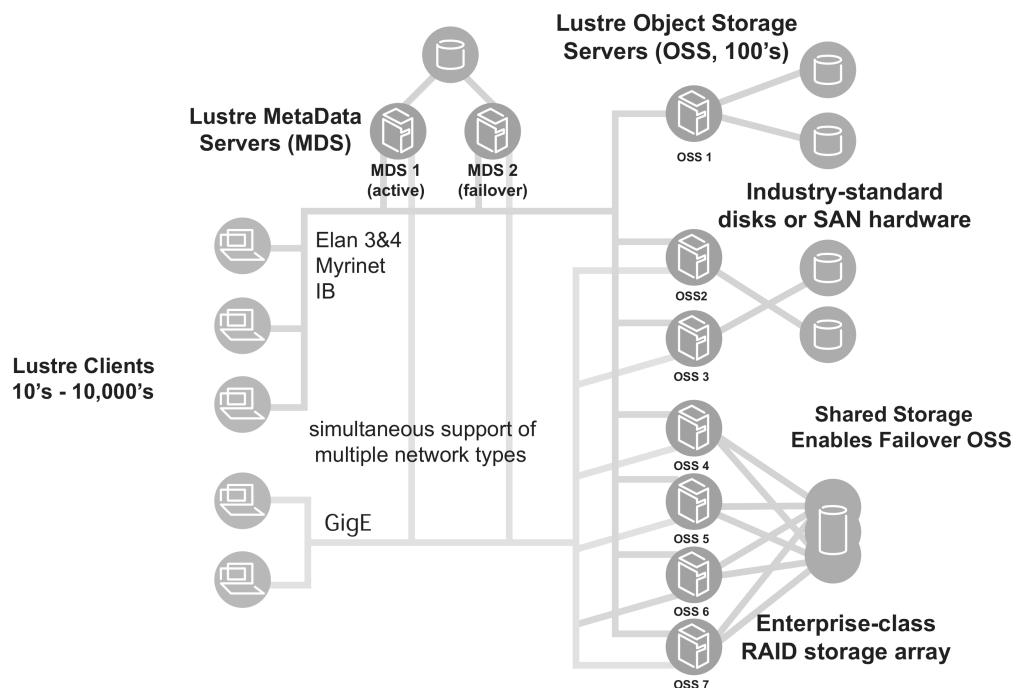


Abbildung 6.6: Lustre Dateisystem

liegen als Objekte auf den so genannten *Object Storage Servers (OSS)*. Um Datenverlust vorzubeugen, verfügt Lustre über ein intelligentes Verteilungssystem der Daten. Daten werden redundant auf mehreren OSS hinterlegt und bei eventuellen Ausfällen kann ein anderer OSS die Daten weiterhin liefern. Wenn der ausgefallene OSS wieder aufgeschaltet wird, synchronisiert er sich über die Angaben der Metadaten des MDS selbst. Als Datenbackend können sowohl SAN-Lösungen, als auch normale Festplatten in normalen physikalischen Rechnern verwendet werden. Einen großen Vorteil bietet die Option, dass ein benutzender Rechner sowohl Client, als auch OSS und MDS sein kann. Dies macht Lustre vor allem für kleine Clusterlösungen interessant. Lustre ist POSIX kompatibel.

Weitere vielversprechende, aber noch in der Forschung oder Entwicklung befindliche Cluster-Dateisysteme sind:

- Self-certifying File System [DAR07]

¹⁵Lustre: <http://www.lustre.org/>

¹⁶Roadmap: <http://www.clusterfs.com/lustre-roadmap-v1.15.pdf>

- Ceph Petabyte Storage [Cen07]
- Parallel Virtual File System, Version 2 [Tea03]
- AFS und GFS [Gri06]

6.9.2 iSCSI statt ATA over Ethernet

Anstatt *ATA over Ethernet* kann wie oben erwähnt auch *iSCSI* [JS04] zum Einsatz kommen. Sollen die Daten via Internet oder VPN in einem anderen Netzwerk oder Standort

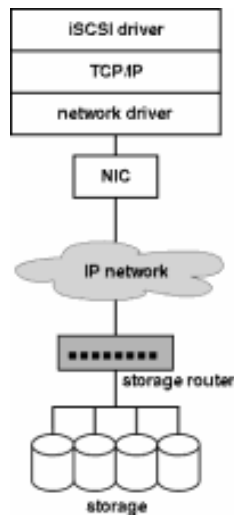


Abbildung 6.7: iSCSI Schema

verfügbar sein, empfiehlt sich auf Grund der Routing- und Authentifizierungseigenschaften die Verwendung von *iSCSI*. Einen sehr guten Überblick bieten die Folien von fm-berger [Ber06].

6.9.3 Andere Betriebssysteme

Natürlich kann XenTM auch mit anderen Betriebssystemen verwendet werden. Gentoo ist hierbei kein Zwang, erleichtert jedoch erheblich den administrativen Aufwand. Wer sich mit XenTM unter Debian [Kem06] auseinandersetzen möchte, der soll dies gerne tun. Es gibt sehr gute Anleitungen [Pfe06] zu diesem Thema. Auch Clustering mit Hilfe von XenTM [Cau05] erfreut sich in der OpenSource Gemeinde immer mehr Beliebtheit.

7 Zusammenfassung


7.1 Fazit

Eine der größten Schwierigkeiten im Umgang mit virtuellen Systemen ist das Benutzen von virtuellen Systemen. Dies mag Paradox erscheinen, jedoch ergeben sich die meisten Probleme aus Thematiken, die augenscheinlich nichts mit der Virtualisierung zu tun haben. Die korrekte Auswahl der einzelnen hier verwendeten Projekte, hat den Großteil der Recherchen in Anspruch genommen. Die Zusammenführung einzelner Projekte sowie die korrekte Auswahl des Linux-Kernels ist das Schwierigste dabei gewesen. Nicht jede Kernelversion unterstützt alle Projekte. Gibt es einen Kernel, der z.B. XenTM unterstützt, heißt dies noch lange nicht, das auch *OCFS2* damit funktioniert. Leider schlägt in diesem Bereich immer wieder das *try-and-error*-Prinzip voll ein. Die Konzipierung des Datenbackends war die zweite wichtige Hürde, die es zu nehmen galt. Leider gibt es noch kein Dateisystem, das alle geforderten Eigenheiten besitzt, die für einen optimalen Aufbau notwendig wären. Lustre gilt hierbei als Favorit, aber da es momentan sehr schlecht an aktuelle Kernelversionen angepasst ist, blieb es bei einem kurzen Test. Da die Entwicklung jedoch erfahrungsgemäß sehr schnell in diesem Bereich voranschreitet, bleibt Hoffnung auf baldige Lösung aller Probleme.

Die Entwicklung des Marktes für virtuelle Umgebungen wächst derzeit ständig und stetig an. Immer neue Firmen mit neuen Geschäftsbereichen sind am Markt zu sehen. Doch auch eingessessene Hosting-Firmen und Provider stellen ihre IT Infrastruktur sukzessive auf virtuelle Umgebungen um. Selbst kleine Firmen können davon profitieren, wie man anhand des Projekts ATLAS sehr leicht sehen kann.

Im Allgemeinen kann man sagen, das der Kostendruck und die Konsolidierungsmöglichkeiten den ausschlaggebenden Aspekt bilden, eine solche Infrastruktur einzuführen. Auch die Kosteneinsparung an relativ teurem Personal für die Administration der Infrastruktur kann so reduziert werden.

Der Markt an sich ist momentan sehr schnell in Bewegung, was auch dazu führt, dass immer neue Technologien und Konzepte auf den Markt kommen, die teilweise sehr gut sind, teilweise aber auch sehr schnell wieder verschwinden. Auch branchenfremde Riesen, wie Amazon.com, entdecken ihr brachliegendes Potential an Rechenkraft und versuchen es entsprechend zu vermarkten.

Ein Ausblick in Richtung Forschung sollte jedoch nicht fehlen: Der eigentliche Zweck, warum XenTM überhaupt konzipiert wurde, zielt in Richtung „public grid computing“. *XenoServer*¹  stellt ein Projekt dar, das versucht, ungenutzte Ressourcen in einem globalen Netzwerk zu bündeln. XenoServer benutzt hierbei die durch XenTM geschaffenen Umgebungen, um virtuelle Netze an beliebigen Standorten aufzubauen, die ein beliebiger Benutzer nutzen kann [Kot05]. Die Granularität geht hierbei von einzelnen Rechenminuten bis hin zu ganzen Clustern, die bestimmte Aufgaben erledigen. Der Clou ist die integrierte

¹XenoServer: <http://www.xenoservers.net/>

Management-Plattform von XenoServer, die es ermöglicht, einzelne Bestandteile dieses „public computing grids“ zu buchen und entsprechend zu provisionieren.

Der Ansatz der Paravirtualisierung und die neuen Technologien von Intel und AMD sind der Schlüssel für eine völlig neue Ära der Informatik. Momentan vollziehen sich in dem Bereich des Clustering und Hosting zwei Paradigmenwechsel: Nicht die einzelne Hardware entscheidet über Rechenleistung, sondern die gesamte Infrastruktur. Das zweite Paradigma ist die Abstraktion der eigentlich für die Kommunikation mit der Hardware gedachten Betriebssysteme. Sie kommunizieren nicht mehr, wie Ursprünglich konzipiert, direkt mit der Hardware, sondern werden selbst zum Programm für bestimmte Aufgaben degradiert.

Es bleibt spannend...

8 Verwendete Projekte

Dieses Kapitel enthält lediglich eine Auflistung der hier verwendeten Projekte sowie ein Listing des kompletten Host-Systems auf Gentoo-Basis:

Listing 8.1: Grundsystem dom0

```
1 ...
2 app-emulation/xen [3.0.4_rc1]
3 sys-kernel/xen-sources [2.6.18.6]
4 sys-kernel/xen-sources [2.6.16.28-r1]
5 app-emulation/xen-tools [3.0.4_rc1]
6 sys-kernel/linux-headers [2.6.17-r2]
7 net-misc/bridge-utils [1.0.6-r3]
8 app-crypt/gnupg [1.9.21]
9 app-misc/screen [4.0.3]
10 sys-process/vixie-cron [4.1-r9]
11 sys-kernel/module-rebuild [0.5]
12 sys-apps/iproute2 [2.6.19.20061214]
13 net-misc/iputils [20060512]
14 sys-cluster/heartbeat [2.0.7-r2]
15 net-analyzer/netio [1.26]
16 net-analyzer/netcat [110-r8]
17 net-analyzer/traceroute [1.4_p12-r5]
18 sys-fs/sysfsutils [1.3.0-r1]
19 app-admin/syslog-ng [1.6.11-r1]
20 app-admin/logrotate [3.7.2]
21 app-portage/eix [0.7.9]
22 net-misc/dhcpd [2.0.5-r1]
23 sys-apps/smartmontools [5.36-r1]
24 sys-cluster/gnbd-kernel [1.03.00]
25 sys-cluster/gfs-kernel [1.03.00]
26 sys-cluster/gnbd-headers [1.03.00]
27 sys-cluster/gfs-headers [1.03.00]
28 sys-cluster/cman-headers [1.03.00]
29 sys-cluster/dlm-headers [1.03.00]
30 sys-fs/gfs [1.03.00]
31 sys-cluster/dlm [1.03.00]
32 sys-cluster/fence [1.03.00]
33 sys-cluster/cman [1.03.00]
34 sys-cluster/gnbd [1.03.00]
35 sys-fs/fuse [2.6.3]
36 sys-block/iscsitarget [0.4.14]
37 sys-block/iscsi-initiator-core-tools [2.3]
38 sys-block/vblade [10-r2]
39 sys-block/aoetools [10]
40 sys-fs/ocfs2-tools [1.2.1]
```

Index

- 'vblade', 8
- AFS, 41
- AMD's Virtualization Solutions, 6
- aoetools, 9
- Appliances, 19
- ATA over Ethernet, 7–9, 33, 34, 41
- BeoWulf, 1, 15
- BixData, 38
- Bochs, 3
- Ceph, 41
- ClusterKnoppix, 1
- Comparison of virtual machines, 7
- csync2, 37
- Distributed Replicated Block Device (DRBD), 9
- DRBD, 9, 31
- Einstellungen des GCC, 35
- enomalism, 18
- Fencing, 9, 10, 26, 32, 33
- flexBackup, 37
- FreeOsZoo, 18
- Gentoo 2006.1, 35, 36
- Global File System, 32, 41
- grid computing, 1
- heartbeat, 9
- IEEE 802.3 Ethernet, 8
- Intel Virtualization Technology, 6
- iSCSI, 8, 41
- JailTime, 17
- Kernel 2.6.16, 36
- Kernel based Virtual Machine (KVM), 17
- Live FreeOsZoo, 18
- Lustre, 40
- MDS, 40
- MS Virtual PC, 4
- Multiple Emulator Super System (M.E.S.S.), 4
- Multislottting, 26
- Nagios, 37
- OCFS2, 9, 32–34, 43
- OCFS2 UserGuide, 33
- openMosix, 1, 14, 15
- openQRM, 39
- openSSI, 1
- POSIX, 9, 26, 40
- public computing, 1
- PVFS2, 41
- QEMU, 3
- Quorum, 9, 10, 26, 32
- Roadmap, 40
- rPath, 17, 18
- Self-certifying File System (SFS), 40
- shorewall, 36
- Single System Images, 11
- SMARTMonTools, 30
- SSI-Hosting, 11, 12, 14
- The Globus Alliance, 2
- TSI, 20
- Vanilla-Kernel 2.6.18, 35
- VirtualBox, 4
- VirtualIron, 17
- VMfind, 18
- VMWare, 4
- XenTM, 4–7, 11, 14, 17–19, 25, 26, 35, 38, 41, 43
- XenTMvirtual machine monitor, 4
- XenoServer, 2, 43

Literaturverzeichnis

- [BC04] BRANTLEY COILE, SAM HOPKINS: *The ATA over Ethernet Protocol*, Oktober 2004. URL: <http://www.coraid.com/pdfs/documentation/AoEDescription.pdf>.
- [Ber06] BERGER, FRANK: *iSCSI unter Linux - das SAN für Zuhause*, März 2006. URL: <http://www.fm-berger.de/iscsi/>.
- [Cah04] CAHILL, BEN M.: *What is OpenGFS*, April 2004. URL: [http://opengfs.sourceforge.net/showdoc.php?docpath=cvsmirror/opengfs/docs/WHATIS-ogfs&doctitle=WHATIS_OpenGFS&docauthor=ben.m.cahill\(at\)intel.com](http://opengfs.sourceforge.net/showdoc.php?docpath=cvsmirror/opengfs/docs/WHATIS-ogfs&doctitle=WHATIS_OpenGFS&docauthor=ben.m.cahill(at)intel.com).
- [Cau05] CAULFIELD, PATRICK: *Running a Xen Cluster*, Juni 2005. URL: <http://people.redhat.com/pcaulfie/docs/xencluster.html>.
- [CC05] CHRISTOPHER CLARK, ANDREW WARFIELD ET AL.: *Live Migration of Virtual Machines*, Mai 2005. URL: <http://www.cl.cam.ac.uk/research/srg/netos/papers/2005-nsdi-migration.pdf>.
- [Cen07] CENTER, STORAGE SYSTEMS RESEARCH: *Ceph Petabyte Storage*, Januar 2007. URL: <http://ceph.sourceforge.net/overview.html>.
- [DAR07] DARPA: *Self-certifying File System*, Januar 2007. URL: <http://www.fs.net/sfswww/sfs.html#Overview>.
- [ea03] AL., IAN PRATT ET: *Xen an the Art of Virtualization*. University of Cambridge, UK, Oktober 2003. URL: <http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf>.
- [ea04] AL., BRYAN CLARK ET: *Xen and the Art of Repeated Research*, April 2004. URL: <http://www.clarkson.edu/class/cs644/xen/files/repeatedxen-usenix04.pdf>.
- [Fas06] FASHEH, MARK: *OCFS2: The Oracle Clustered File System, Version 2*, Mai 2006. URL: <http://oss.oracle.com/projects/ocfs2/dist/documentation/fasheh.pdf>.
- [Gri06] GRIMME, DIPL. INF. MARK: *Workshop: Vergleich AFS und GFS + Konfiguration eines GFS-Clusters im Rahmen des GUG FFG*, März 2006. URL: <http://www.atix.de/event-archiv/vergleich-afs-und-gfs-konfiguration-eines-gfs-clusters-german-deutsch/>.
- [JS04] J. SATRAN, K. METH, C. SAPUNTZAKIS M. CHADALAPAKA E. ZEIDNER: *Internet Small Computer Systems Interface (iSCSI)*, April 2004. URL: <http://www.ietf.org/rfc/rfc3720.txt>.
- [Kem06] KEMP, STEVE: *Debian Sid gets Xen 3.0*, Mai 2006. URL: <http://www.debian-administration.org/articles/396>.

- [Kot05] KOTSOVINOS, EVANGELOS: *Global public computing.* , Cambridge University, Computer Laboratory, Januar 2005. URL: <http://www.cl.cam.ac.uk/TechReports/>.
- [Pfe06] PFENNIG, THILO: *Xen*, Juli 2006. URL: <http://www.linuxwiki.org/Xen?highlight=%28xen%29>.
- [Pra06] PRATT, IAN: *Xen and the Art of Virtualization - Masterclass Presentation*, April 2006. URL: www.cl.cam.ac.uk/research/srg/netos/papers/2006-xen-lw-masterclass.pdf.
- [Soc00] SOCIETY, IEEE COMPUTER: *Amendment to carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications-aggregation of multiple link segments*, Januar 2000. URL: <http://ieeexplore.ieee.org/ISOL/standardstoc.jsp?punumber=6867>.
- [Sys99] SYSTEMSOFT, INTEL CORPORATION : *Preboot Execution Environment (PXE) Specification*, September 1999. URL: <http://www.pix.net/software/pxeboot/archive/pxespec.pdf>.
- [Tea03] TEAM, PVFS2 DEVELOPMENT: *Parallel Virtual File System, Version 2*, September 2003. URL: <http://www.pvfs.org/pvfs2-guide.html>.
- [uDHH03] DR. HEIDI HOHENSOHN, JIAYIN HANG UND: *Global File System Installation on a GenToo Distribution*, Juli 2003. URL: http://now.c-lab.de/veroeffentlichungen/einfuehrung/Anforderungskatalog_Teil_final.pdf.
- [Wik07] WIKIPEDIA, THE FREE ENCYCLOPEDIA: *Distributed file system*, Januar 2007. URL: http://en.wikipedia.org/wiki/Distributed_File_System.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Kevin Wennemuth)

Copyright 2007 Kevin Wennemuth.

All Rights Reserved.

Aside from my professor's sole, personal review as part of his/her private, single-human, software-free grading process (checking for plagiarism with Google is acceptable), neither my professor nor my academic institution may otherwise copy, transfer, distribute, reproduce, publicly/privately perform, publicly/privately claim, publicly/privately display, or create derivative works (including „digital fingerprints“) of my copyrighted document (*intellectual property*).

The same restrictions apply to **Turnitin.com and all similar services** if my document should somehow come into their possession. Neither my professor nor my academic institution may submit my copyrighted document, in whole or in part, to be copied, transformed, manipulated, altered, or otherwise used by or stored at **Turnitin.com (iParadigms, LLC) or any other physical or electronic database or retrieval system** without my personal, explicit, voluntary, uncoerced, written permission.

Regardless of supposed intent (e.g., „to create a digital fingerprint,“), **no part of my copyrighted document** may be temporarily or permanently transferred, by any party, to Turnitin.com or any other service, program, database, or system for analysis, comparison, storage, or any other purpose whatsoever. Violators will be monetarily punished to the fullest extent allowed by the DMCA (Digital Millennium Copyright Act), the German Urheberrecht and/or international law.