Virtualisierung

Betrachtung aktueller Hypervisor wie Xen, KVM und Hyper-V

Guilherme Bufolo
guilherme.bufolo@e-technik.stud.uni-erlangen.de

ABSTRACT

Mit der Einführung von Virtualisierung durch VMware sahen Rechenzentrenbetreiber eine Möglichkeit, ihre brach liegende Rechenleistung zu konsolidieren. Diese mussten bis dahin für jeden Kunden eine Maschine zur Verfügung stellen, was enorme Platzanforderungen mit sich brachte.

Durch die Vielzahl von Lösungen, die für dieses Problem gefunden wurden, minderte sich die Übersichtlichkeit über die Unterschiede einzelner Lösungen. Als geeignete Lösung kristallisierte sich Rechnervirtualisiereng heraus. Diese Virtualisierung wird durch Hypervisors bereitgestellt. Diese sind die treibende Kraft im Cloud-Computing. Es existieren heutzutage verschiedene Hypervisors von verschiedenen Herstellern. Diese werden im Einzelnen vorgestellt und erklärt. Xen wird beispielhaft für eine paravirtualisierende Umgebung verwendet, Hyper-V und KVM für Vollvirtualisierung und VMwares Hypervisor als eine Vollvirtualisierung, die ohne Hardware-Unterstützung (Hardware Assist) auskommt. Als Vertreter von Microhypervisors tritt NOVA auf.

Zusätzlich werden auch moderne Funktionalitäten, wie Memory-Ballooning, Memory-Overcommit, Page-Sharing, Live-Migration und Snapshots, die die neuesten Hypervisors mitbringen, vorgestellt. Diese dienen einer besseren Auslastung der Host-Rechner. Die Möglichkeit der Migration zwischen einzelnen Hypervisors wird diskutiert um zu zeigen, dass sie mit einem gewissen Aufwand durchführbar ist.

1. EINFÜHRUNG

Die Anzahl der Rechenzentren stieg in den letzten Jahren rasant an. Durch das Green-IT-Phänomen und die Motivation Kosten zu senken suchten die Betreiber schon seit Langem nach Möglichkeiten, Rechner, die ihre Rechenleistung nicht komplett ausschöpfen, zu konsolidieren, um so Hardware und Betriebskosten zu senken. Die x86-Architektur gilt aber bis heute als eine schwer zu virtualisierende Plattform. Bis Ende der 90er gab es keine vernünftige Lösung für das Problem. Als es VMware gelang, diese Plattform zu virtualisieren, sahen viele Betreiber sofort das Potential darin und seitdem wächst der Markt konstant [7, 18].

Durch dieses Marktwachstum kamen immer mehr Unternehmen und Software dazu, sodass heute eine große Palette an Lösungen zur Verfügung steht. Die Lösungen, die sich am meisten verbreitet haben, sind Bibliotheksvirtualisierung, Betriebssystemvirtualisierung und Rechnervirtualisierung. Bibliotheksvirtualisierung und Betriebssystemvirtualisierung sind primär als Einzelplatzlösungen gedacht und somit nicht in der Cloud-Computing-Infrastruktur im Einsatz und werden deshalb hier auch nicht weiter erläutert. Rechnervirtualisierung ist für Cloud-Computing eine geeignete Lösung, die durch Hypervisors realisiert wird. Hypervisors erlauben es, eine Vielzahl von parallelen (heterogenen) Betriebsysteminstanzen laufen zu lassen. Manche zielen auf tausende von parallelen Instanzen [1] ab. Hypervisors virtualisieren ganze Betriebssysteme. Dadurch ergibt sich die Möglichkeit die Gast-Betriebssysteme im Betrieb zu migrieren. In der Cloud ist dies ein großer Vorteil, denn es erlaubt die Rechenleistung dort auszunutzen, wo sie zur Verfügung steht.

Cloud-Computing wird mit einem breitem Spektrum an Bedeutungen verwendet. Eine der Bedeutungen ist der Paradigmenwechsel, Rechenleistung und Speicherplatz von einem Dienstanbieter nach Bedarf zu mieten [28], anstatt diese in einem eigenen Rechennetz zur Verfügung zu stellen. Meistens ist dem Mieter nicht bekannt wo die Maschinen stehen, die er gerade benutzt. Um diese Unübersichtlichkeit klar zu formulieren entstand der Begriff der Wolke. Der Leistungsnutzer legt daher keinen besonderen Wert darauf, woher seine Rechenleistung bezogen wird, solange sie in dem von ihm gebuchten Umfang geliefert wird.

Hypervisors, häufig eingesetzt im Bereich des Cloud-Computing, erstellen virtuelle Maschinen (VMs), um Betriebssysteme zu virtualisieren. Eine VM ist ein nachgebildeter Rechner, der mittels eines Hypervisors auf einer realen Maschine läuft. Es ist einem Betriebssystem im Allgemeinen nicht möglich zu unterscheiden, ob seine Hardware virtueller oder reeller Natur ist. Eine VM die auf einem Hypervisor läuft wird im Folgenden auch als Gast, Gast-Instanz oder Gast-Betriebssystem bezeichnet.

Virtualisierung erleichtert auch das Warten von Rechenzentren, da weniger Hardware für dieselbe Anzahl von Betriebssysteminstanzen notwendig ist. Die Sicherheit und Flexibilität ist erhöht, denn es ist möglich eine virtuelle Maschine zu vervielfältigen und somit an verschiedenen Orten zu sichern. In großen Rechenzentren kommen noch andere Vorteile dazu. Das Anschaffen eines neuen Rechners für einen Kunden bedeutet nicht mehr eine ganze Maschine zu kaufen, sondern kann sich darauf beschränken eine neue virtuelle Instanz eines Rechners zu erstellen. Eine Abstraktion der Hardware kann genutzt werden, um heterogene Hardware durch homogene virtuelle Geräte zu ersetzen. Bei Ausfällen der Hardware können die virtuellen Maschinen auf anderen Rechnern problemlos weiterlaufen. Wartungsarbeiten an der Hardware bedeuten nicht mehr, dass ein Dienst eventuell nicht verfügbar ist. Das Wartungspersonal muss sich nicht mehr darum kümmern Daten, zu sichern und Dienste anzuhalten. Sie können die VM auf eine andere Maschine migrieren, auf der sie weiterlaufen kann.

2. VIRTUALISIERUNG

Die Idee der Virtualisierung ist nicht neu, sie wurde schon in den 60er Jahren von IBM für die zSeries-Rechner entworfen [27]. Seitdem hat sich viel getan und es sind mehrere alternative Lösungen und Ansätze entstanden. Im Cloud-Computing-Bereich konnten sich die Hypervisors etablieren, da sie ein ganzes Betriebssystem virtualisieren. Dies hat mehrere Vorteile, unter anderem, ist es möglich, verschiedene Betriebssystemarten und Versionen auf der selben Hardware laufen zu lassen. Zusätzlich ist es möglich jedem VM-Besitzer innerhalb seiner VM Administratorrechte zu geben ohne Sicherheitskompromisse eingehen zu müssen.

2.1 Hypervisor

Hypervisors[1], auch als Virtual Machine Monitors (VMMs) bekannt, sind Anwendungen die dafür sorgen, dass Instanzen von VMs richtig gestartet und gestoppt werden können. Sie verwalten die Hardware und sorgen dafür, dass die VMs reibungslos nebeneinander laufen können.

Hypervisors werden nach ihrem Aufbau kategorisiert und nach der Art der Virtualisierung, die sie zur Verfügung stellen, unterschieden. Typ 1 läuft direkt auf der „nackten" Hardware und muss daher eigene Treiber mitbringen, um auf Geräte zugreifen zu können. Typ 2 dagegen setzt auf ein schon vorhandenes Betriebssystem auf und lässt dieses die Zugriffe auf die Hardware koordinieren. Typ-1-Hypervisors bringen öfters ein Betriebssystem mit und leiten die Hardware-Zugriffe auf dieses Betriebssystem um. Dieses Betriebssystem wird dann auch meistens zur Verwaltung der Gastrechner verwendet.

Die Art der Virtualisierung kann variieren. Sogenannte vollvirtualisierende Hypervisors können den Gast direkt auf die Hardware zugreifen lassen. Hypervisors die durch Kernelmodifikationen Aufrufe des Gast-Betriebssystem an den Hypervisor weiterleiten, werden als paravirtualisierend bezeichnet.

2.2 Vollvirtualisierung

Diese Art der Hardware-Weitergabe hat den Nachteil, dass die meisten Betriebssysteme geschrieben werden mit der Annahme, dass sie auf Ring 0 einer x86-Architektur betrieben werden [34]. Da auf diesem Ring jetzt der Hypervisor existiert, muss das Betriebssystem auf einem der anderen Ringe betrieben werden mit der Folge, dass alle seine Instruktionen, die privilegierten Status brauchen, nicht mehr verwendbar sind.

Die x86-Architektur ist in Ringen aufgebaut. Der Ring 0 ist mit der höchsten Priorität versehen und kann alle Befehle ausführen sowie auf alle Register und auf beliebige Daten zugreifen. Er ist der innerste Ring. Danach gibt es noch drei weitere Ringe 1, 2 und 3 (äußerster Ring) mit jeweils absteigender Priorität. Ringe mit höherer Priorität dürfen auf Daten äußerer Ringe zugreifen, Ausführung von niederprivilegiertem Code ist nicht zugelassen. Code anderer Ringe, darf nur über Gates, spezielle Funktionen für den Ringwechsel, aufgerufen werden. Ring 0 ist für Betriebssysteme vorgesehen, Ring 3 für Anwenderprogramme [26].

[1] „Hyper" stammt aus dem Griechischen und bedeutet „über". „Visor" lässt sich aus dem Lateinischen „videre" ableiten, was „sehen" bedeutet. Sinngemäß übersetzt handelt es sich also um ein System, welches als „Aufseher" etwas bzw. weitere Systeme „überblickt" bzw. „etwas überwacht".

Die Probleme, die durch die Verschiebung des Betriebssystems auf einen anderen Ring entstehen, können durch zwei verschiedene Ansätze gelöst werden. Es gibt einmal die Möglichkeit Teile des Kernels des Gast-Betriebssystems zu patchen (VMware ESX [34]) oder Hardware-Unterstützung zu verwenden. Die Hardware-Unterstützung heißt bei Intel VT-x (x steht dabei für die Architektur, z. B. x86) (Codename Vanderpool), bei AMD heißt sie AMD-V (Codename Pacifica). Beide Technologien, sind im Gegensatz zu den Instruktionssatzerweiterungen (SSE, MMX, etc.), nicht miteinander kompatibel, aber sie bieten im Endeffekt dieselbe Funktionalität an.

Die Hardware-Unterstützung erlaubt es, einen nicht modifizierten Kernel auf Ring 0 laufen zu lassen. In einem „Root Mode Privilege Level" [34], auch Ring -1 genannt, wird dann der Hypervisor installiert. Wenn das Gast-Betriebssystem versucht einen privilegierten Befehl auszuführen, greift die Hardware-Unterstützung ein. Dieser wird dann automatisch an den Hypervisor weitergeleitet. Somit entfällt die Notwendigkeit den Kernel binär zu patchen oder der Paravirtualisierung. VMware ist der Meinung, die Technologie zeige großes Potential für die Zukunft, sei aber noch nicht ausgereift [34]. Die Hardware-Unterstützung ist in allen aktuellen Intel-CPUs vorhanden (ausgenommen Celeron) und bei AMD in allen {Athlon, Thurion}-64-Prozessoren und der Opteron- und Phenom-Serie.

Das Patchen des Codes on-the-fly hat den großen Nachteil, dass dies bei jedem neuen Release/Patch eines Betriebssystems überprüft werden muss. Zusätzlich ist es notwendig, alle Instruktionen zu modifizieren, die auf Ressourcen zugreifen, um z. B. Seitentabellenkonsistenz zu wahren, Seitentabellenänderungen geschehen bei Prozesswechsel, deshalb kann dies hohe Performance-Einbußen zur Folge haben [15].

2.3 Paravirtualisierung

Bei der Paravirtualisierung wird der Kernel des Gast-Betriebssystems modifiziert. Alle Bereiche, die privilegierte Befehle nutzen, werden umgeschrieben, um Funktionen des Hypervisors zu nutzen; diese werden mit Hypercalls aufgerufen. Hypercalls sind Funktionsaufrufe, die an den Hypervisor weitergeleitet werden. Wie in Betriebssystemen bei Systemaufrufen üblich sind Hypercalls Interrupts [22], die dem Hypervisor zugestellt werden. Kontextwechsel eines Betriebssystems in den Hypervisor sind komplex und teuer, deshalb existiert die Möglichkeit, diese Aufrufe zu gruppieren um den Kontextwechsel nur einmalig durchzuführen.

Der große Nachteil dieser Lösung ist, dass es erforderlich ist, den Kernel zu modifizieren. Dies ist von besonderem Nachteil bei Closed-Source-Betriebssystemen, da dort die Unterstützung nur von Seite des Herstellers möglich ist. Hypervisors dieser Art, bieten deshalb auch Hardware-Unterstützung an, um z. B. Windows virtualisieren zu können.

Diese Lösung bietet eine gute Leistung. Mittlerweile sind sogenannte Paravirtualisierungsoperationen (paravirt-ops) Teil des Linux Kernels (seit Version 2.6.21).

Aktuell werden auch Treiber speziell für virtuelle Hardware entwickelt, sodass sie sich dieser Tatsache bewusst ist und spezielle Befehle des Hypervisors verwendet. Somit lassen sich auch ohne Paravirtualisierung des Kernels annehmbare Leistungen erzielen.

3. HYPERVISORS

In den vergangen Jahren haben viele Hersteller ihre Lösungen für Virtualisierung herausgebracht. In diesem Zeitraum, gewannen am meisten Hyper-V, Xen und KVM an Bedeutung. Hyper-V, Microsofts Hypervisor, wurde für den Windows Server 2008 freigegeben und Kunden auch kostenlos zur Verfügung gestellt. Xen, ursprünglich in der Universität von Cambridge entstanden wurde, wurde von Citrix aufgekauft und in den ersten Jahren von Suse und Red Hat als die Standardlösung angeboten. KVM beginnt jetzt bekannt zu werden, nachdem es als erster Einzug in den Linux-Kernel erhielt. KVM wird immer noch nur durch eine Open-Source-Community entwickelt. VMware hat mit dem Hype der Virtualisierung begonnen und bietet auch im VM-Management immer noch die umfangreichsten Lösungen an und auch die breiteste Produktpalette. Im Hypervisor-Bereich aber haben alle anderen gut aufgeholt. VMware bleibt einer der wenigen, wenn nicht der einzige, der Vollvirtualisierung auch ohne Hardware-Unterstützung anbietet.

3.1 Besondere Eigenschaften

Im praktischen Betrieb hat sich gezeigt, dass durch verschiedene Technologien mehr Gäste auf einem Wirt zum Laufen gebracht werden können als z. B. der RAM dies zulassen würde.

Ein Betriebssystem benötigt selten den ganzen RAM, der ihm zugesichert wird. Um das auszunutzen, haben manche Hypervisor-Hersteller Technologien entwickelt, um mehr Gäste auf einem Wirt zu platzieren als es der physikalische Speicher zulassen würde. Drei dieser Mechanismen sind Memory-Overcommit, Page-Sharing und Memory-Ballooning. Somit sind Rechenzentren in der Lage ihr Rechenpotential zu konsolidieren, um Kosten und Platz zu sparen.

Im Sinne der Cloud-Computing-Utility-Grids, wo Leistung nach Bedarf gemietet werden kann, ist der Gedanke interessant laufende Betriebssysteme bei steigenden Anforderungen (z. B. hoher Rechenlast, hohem Speicherverbrauch) auf leistungsstärkere Maschinen zu migrieren und bei Abklingen dieser Lastspitze wieder zurückzuholen. Dies wird mit Live-Migration erreicht. Zusätzlich sind Snapshots und VM-Vorlagen von Bedeutung, die es erlauben Wiederherstellungspunkte zu erstellen und eine Vorlage auf mehreren Rechnern zu kopieren.

3.1.1 Memory-Overcommit

Memory-Overcommit bedeutet, dass in der Summe aller zugesicherten Arbeitsspeichergrößen mehr RAM verteilt wurde als das System installiert hat. Der Sinn dahinter ist, dass öfters Betriebssysteme unter regulärer Last nicht ihren ganzen RAM ausschöpfen. Das Memory-Overcommit sinnvoll sein kann zeigt [21].

3.1.2 Page-Sharing

Page-Sharing bedeutet, dass identische Speicherseiten gesucht werden; meistens sind diese leere Seiten [32]. Diese werden dann mehreren VMs zur Verfügung gestellt und mit Copy-On-Write verwendet. Mit großen Seitentabellen (Speicherseiten, die größer als 4 KB sind), sinkt die Wahrscheinlichkeit zwei gleiche Seiten zu finden enorm und somit die Einsatzmöglichkeit dieser Technologie.

3.1.3 Memory-Ballooning

Memory-Ballooning heißt, dass einem System je nach Last mehr oder weniger Speicher (RAM) zur Verfügung gestellt wird. Dieser Anpassungsvorgang passiert, während das Gastsystem noch läuft. Für eine VM sieht es so aus als hätte sie diese Menge an flüchtigen Speicher (RAM) physisch vorhanden [35].

3.1.4 Live-Migration

Live-Migration ist der wichtigste Aspekt für die Cloud. Live-Migration ist das Migrieren des Gastes von einer Hardware zu einer anderen, im laufenden Betrieb. Dazu wird zuerst der nicht-flüchtige Speicherinhalt (Local Persistent State, Festplatteninhalt) kopiert, falls das VM-Image nicht auf dem Ziel-Rechner vorhanden ist; anschließend der RAM- und Register-Inhalt. Das komplizierte an dieser Prozedur ist zu erkennen, welche Seiten sich geändert haben seit dem letzten Kopieren, denn diese müssen gegebenfalls erneut kopiert werden. Es sollte besonders acht darauf gegeben werden nicht unnötig viele Daten zu kopieren. Seiten, die sich oft ändern, sollten einmalig am Ende kopiert werden. Xen bietet dazu eine Bitmaske an, die kennzeichnet, welche Seiten sich geändert haben. Andere Hypervisors haben ähnliche Mechanismen, auch Dirty Page Marking genannt.

Mit Live-Migration ist es möglich Unterbrechungszeiten von wenigen Millisekunden im LAN zu erreichen [17] und ein paar Sekunden im WAN [16]. Migrieren über ein WAN bedeutet meistens, dass die VM eine neue IP-Adresse auf dem neuen Wirt haben wird. Dies lässt sich aber für bestehende Verbindungen mittels eines IP-Tunnels verbergen und mit dynamischen DNS-Aktualisierungen können neue Anfragen an die neue IP-Adresse weitergeleitet werden [16].

3.1.5 Snapshots

Um Wiederherstellungspunkte zu erstellen, wurden Snapshots eingeführt. Diese speichern nur Veränderungen, die an der VM stattgefunden haben. Da sie alle Änderungen immer protokollieren, sollten sie nicht mit inkrementellen Backups verwechselt werden. Snapshots laufen paralell zu dem aktiven System und haben somit, einen Leistungsverlust als Folge. Zu beachten ist, dass Snapshots mit der Anzahl der Veränderungen an der Original-VM größer werden und deshalb nur für kurze Zeiten gedacht sind.

3.1.6 VM-Vorlagen

Es werden vorgefertigte Gäste mit verschiedenen Konfigurationen angeboten. Diese können verwendet werden um mehrere virtuelle Maschinen zu erstellen. Dies erspart es für die selbe Konfiguration wiederholt die selben Einstellungsschritte durchgehen zu müssen.

3.1.7 Linked-VM

Mit Vorlagen und Snapshots lassen sich Linked-VMs erstellen [10]. Bei Erstellung einer Linked-VMs wird der aktuelle Betriebszustand der Root-VM verwendet, um einen gemeinsamen Zustand zu stellen. Alle anderen Linked-VMs speichern Änderungen zu diesem Zustand. Indem diese nur Änderungen speichern, lässt sich Platz sparen, um so die Festplattenplatznutzung zu konsolidieren.

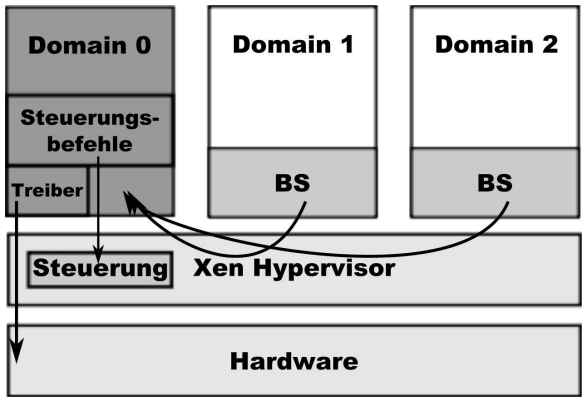

Abbildung 1: Xen-Architektur

3.2 Xen

Xen ist eine Virtualisierungslösung, die direkt auf der Hardware installiert werden kann (Typ 1). Der Hypervisor teilt die Gast-Betriebssysteme in zwei Arten. Eine virtuelle Maschine läuft in der Domain 0 (Dom0) Umgebung. Diese darf direkt auf die Geräte zugreifen und wird zum Verwalten der anderen VMs verwendet. Die anderen Gast-Betriebssysteme laufen in sogenannten Benutzer-Domänen [15] (DomU, Abb. 1). Diese greifen auf virtualisierte Hardware sowie auch direkt auf reale Hardware zu. Zum Beispiel sind die Netzwerkschnittstellen rein virtuell vorhanden, während um die Speicherzugriffsgeschwingdigkeit zu beschleunigen, direkt mit dem RAM kommuniziert werden kann. Xen bietet für die Unterstützung von Betriebssystemen ohne Kernelmodifikationen, Vollvirtualisierung an, setzt dafür aber geeignete Hardware voraus.

Xen wurde entwickelt, um bis zu 100 parallel laufende Gast-Betriebssysteme zu unterstützen. Es kann die Gast-Betriebssysteme mit bis zu 128 virtuellen CPUs, 1 TB RAM per Host, 128 physischen CPUs per Host [31] versorgen, Unterstützung für virtuelles LAN (VLAN) ist auch vorhanden, sowie das Verwenden von mehreren Netzwerkadaptern in einem Gast-Betriebssystem [13].

Um die virtuellen Maschinen auszuführen, verwendet Xen das Open-Source-Programm QEMU. QEMU existierte schon lange vor dem Xen-Projekt und konnte ein ganzes Betriebssystem verwalten; die Simulation der Hardware geschah mittels Software, was große Einbußen in der Performance mit sich brachte. Für das Xen-Projekt wurde es erweitert, um die Möglichkeiten der Paravirtualisierung zu nutzen.

Wie in Abschnitt 2.3 beschrieben werden im Kernel Datenstrukturen und Funktionen so modifiziert, dass sie keine privilegierten Instruktionen benötigen, somit auf Ring 1 laufen können. Die Instruktionen werden durch Hypercalls ersetzt, die diese dann auf Gültigkeit und Sicherheit prüfen können. Dadurch, dass der Gast auf Ring 1 läuft, ist immer noch sichergestellt, dass die Trennung von Kernel-Modus und User-Modus eingehalten wird.

Die Modifikationen, die Xen an dem Kernel vornimmt, und die Abstraktion, die Xen von der Hardware bietet, werden im Folgenden vorgestellt.

3.2.1 CPU

Um die Abschottung der einzelnen Betriebssysteme von dem Hypervisor zu garantieren müssen diese auf einer niedrigeren Prioritätsstufe laufen. Gast-Betriebssysteme werden bei Xen auf Ring 1 betrieben. Um Exceptions korrekt an den Gast weiterleiten zu können, müssen VMs an dem Hypervisor einen Exception-Handler registrieren. Dieser wird auf Korrektheit überprüft, bevor er zugelassen wird. Systemaufrufe, die an das Gast-Betriebssystem gerichtet sind, müssen den ganzen Hypervisor durchlaufen; dies würde einen Kontextwechsel bedeuten. Um dies zu verhindern bietet Xen die Möglichkeit an „schnelle" Bearbeitungsfunktionen zu installieren. Systemaufrufe, die an diese weitergeleitet werden, müssen nicht den Hypervisor passieren. Das ist nicht möglich für Seitenfehler; in Abschnitt 3.2.3 wird das Problem mit den Seitenfehlern genauer beschrieben. Hardware-Interrrupts werden durch ein Ereignissystem ersetzt. Dies erlaubt es die Ereignisse zeitversetzt zu liefern, sowie auch künstliche Hardware-Interrrupts in Software zu erzeugen.

3.2.2 Speicherverwaltung

Virtuelle Maschinen dürfen keine Segmentdeskriptoren erstellen, die vollprivilegiert sind, das sind z. B. Seitentabellendeskriptoren. Dies ist eine natürliche Einschränkung durch den Ring, in dem sie laufen. Auf Seitentabellen darf lesend ohne Beschränkung zugegriffen werden, Aktualisierungen dürfen aber nur über Hypercalls geschehen. Speicher, der einer virtuellen Maschine zugewiesen wird, kann auch segmentiert sein. Das dient einer hohen Speicherausnutzung auf dem Wirt. Um einen geringeren Overhead zu erzeugen, können Seitentabellenaktualisierungen auch zu Stapeln zusammengefasst in einem Aufruf an den Hypervisor übergeben werden.

3.2.3 Interrupt-Bearbeitung

Da es dem Gast-Betriebssystem nicht mehr möglich ist direkt auf manche Ausnahmeregister zuzugreifen, erstellt Xen eine Kopie dieser im Stapel des Gastes und übergibt sie der VM.

Nur zwei Interrupts treten häufig genug auf um einen Leistungseinfluss auf die Virtualisierung zu haben. Zum einen sind dies Systemaufrufe, zum anderen Seitenfehler.

Seitenfehler sind von größerer Bedeutung. Die relevante Information ist in einem Register enthalten, das nur aus Ring 0 lesbar ist. Der Inhalt dieses Register muss zuerst in den Stack der VM kopiert werden bevor die Exception weitergeleitet werden kann.

Systemaufrufe, wie in Abschnitt 3.2.1 beschrieben, können dem Gast direkt weitergegeben werden, wenn dieser einen geeigneten Handler installiert hat.

3.2.4 I/O

Xen gibt fast keine Geräte direkt an das Gast-Betriebssystem weiter, sondern abstrahiert diese mittels einfacher virtueller Geräte. Diese werden durch Treiber angesteuert, die Hypercalls nutzen. I/O wird mit Shared-Memory und asynchronen Pufferringen realisiert. So kann Xen mit hoher Leistung diese Bereiche auf Gültigkeit zu prüfen.

I/O-Interrupts werden durch einen Event-Mechanismus ersetzt. Diese Ereignisse werden in einer Bitmaske notiert. Sie kennzeichnet, ob ein Interrupt aufgetreten ist. Diese veranlasst das Aufrufen des Event-Handlers. Mit diesem System ist es möglich, auf Wunsch des Gast-Betriebssystems die Hardware-Interrrupts künstlich zu unterbinden ohne andere Gäste zu beeinflussen.

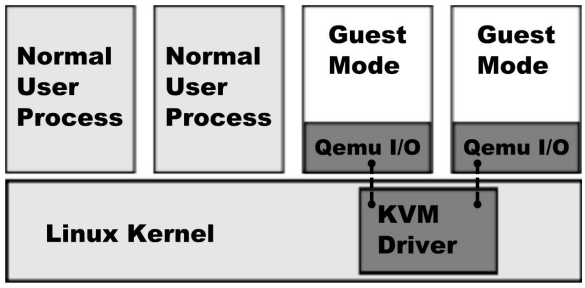

Abbildung 2: KVM-Architektur [24]

3.3 KVM

KVM ist ein relativ neues Projekt, das aber für viel Aufsehen sorgte, indem es als erstes Teil des Linux-Kernels wurde. Xen probierte dies schon seit Jahren ohne Erfolg. KVM ist eine Vollvirtualisierungslösung, die geeignete Hardware voraussetzt (AMD-V oder Intel VT-x). KVM ist nicht leicht einem Typ zuzuordnen, denn einerseits ist es möglich es direkt auf Hardware zu installieren, andererseits setzt es ein Betriebssystem voraus, das es aber selbst mitbringt (somit auch Scheduler, Speicherverwaltung, Treiber, etc. [36]).

Auch KVM verwendet für das Ausführen seiner Gast-Betriebssysteme das Open-Source-Programm QEMU. Für das KVM-Projekt wurde es erweitert, sodass es jetzt auch die Vorteile einer Vollvirtualisierung ausnutzen kann.

Indem KVM Bestandteil des Linux-Kernels wurde, ist es in der Lage immens von Entwicklungen und Erweiterungen, die am Kernel vorgenommen werden, zu profitieren. Neue Treiber, die in den Kernel ein gepflegt werden, können sofort verwendet werden. Erweiterungen und neue Funktionalitäten des Kernels können auch von KVM genutzt werden.

3.3.1 Einbettung in Linux

KVM stellt sich als ein Gerät dar, das unter */dev/kvm* eingebunden wird. So kann es wie ein Gerät mittels ioctl() gesteuert werden. Bereitgestellt werden folgende Operationen:

- Starten einer neuen VM
- Bereitstellung von Speicher für eine VM
- Lesen und Schreiben der Register einer VM
- Erzeugung von Interrupts in einer VM
- Starten einer virtuellen CPU

Für die Anforderungen, die das Bereitstellen eines Hypervisors mit sich bringt, wurde speziell ein dritter Laufzeitmodus erschaffen, zusätzlich zu den bekannten, User-Modus und Kernel-Modus wurde der Gast-Modus (Guest mode) (Abb. 2) hinzugefügt.

VMs, die im Gast-Modus gestartet werden, bekommen einen eigenen Speicher, der unabhängig von dem des startenden Prozesses ist. Eine VM läuft in ihrem eigenen Kontext, bis sie einen Interrupt verursacht, eine Anweisung eintritt, die vom Hypervisor abgearbeitet werden muss oder ein Fehler passiert. Der Kernel überprüft dann, ob I/O-Operationen oder Interrupts anstehen. Ist dies der Fall, wird die Kontrolle an die User-Modus-Software weitergegeben, die für dieses Ereignis verantwortlich ist [25]. Abbild 3 zeigt diese Interaktion.

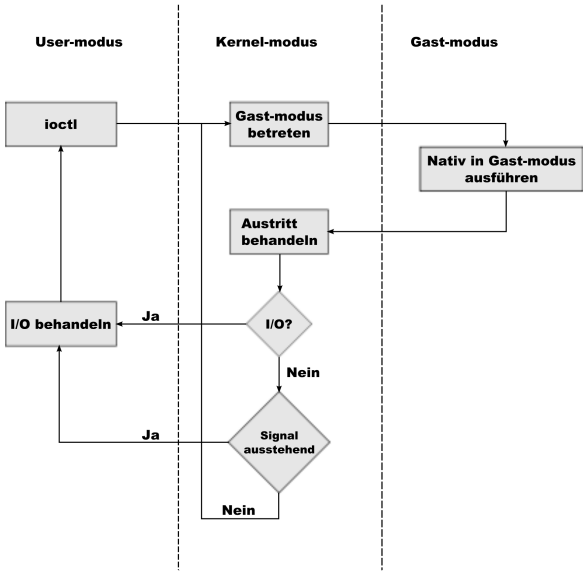

Abbildung 3: Gast Ausführung flowchart. [25]

Das Wirt-Linux sieht die VM wie ein regulärer Prozess. Das bedeutet, mit bekannten Prozessverwaltungs-Software eine VM bedienen zu können. Mit *kill(1)* ist es z. B. möglich eine Gast-Instanz zu terminieren.

3.3.2 Speicher

Ein Problem, mit dem Hardware-unterstützte Virtualisierung zu kämpfen hat, ist, dass die Memory Management Unit (MMU) keinen Weg bietet, um die Abstraktion des virtuellen Speichers des Gastes auf den des physischen Speichers des Wirtes abzubilden. Beim Abbilden von segmentierten Bereichen als ein kontinuierliches Segment im Gast ist dies notwendig.

Da hier im Gegensatz zu Xen der Gast-Kernel keine Modifikationen enthält, können die Speicherseitentabellenzugriffe nicht direkt an den Hypervisor weitergeleitet werden, um sie umzuschreiben, sondern erst, wenn ein Interrupt ausgelöst wird ist der Hypervisor in der Lage den Zugriff zu überprüfen. Um dieses Defizit auszugleichen, arbeitet KVM mit einem Trick, jede Änderung des Transfer Lookaside Buffers (TLB) wird abgefangen. Dieser Buffer enthält einen Cache aus Seiten, die zuletzt verwendet wurden. Bei diesem Interrupt wird festgestellt, welche Seite eine VM bekommen soll, um so parallel eine Shadow Page Table zu verwalten, die den virtuellen Speicher des Gastes auf den physischen Speicher des Hosts abbilden kann.

Diese Adressumsetzung muss bei jedem Schreiben auf diesen Seiten stattfinden, damit die Adressumsetzung korrekt funktioniert. Um dies zu realisieren, werden alle Seiten in einen Nur-Lese-Modus versetzt, was zur Folge hat, dass jeder Versuch in diese Seiten zu schreiben einen Seitenfehler auslöst. In diesem Moment wird die Adresse mithilfe der Shadow Page Table neu berechnet und zeigt auf die korrekte Seite [25].

3.3.3 I/O

Software und Hardware bedienen sich bei Programmed I/O (PIO) und Memory-Mapped I/O (MMIO), um zu kommunizieren. Die Hardware-Voraussetzung für KVM hat den

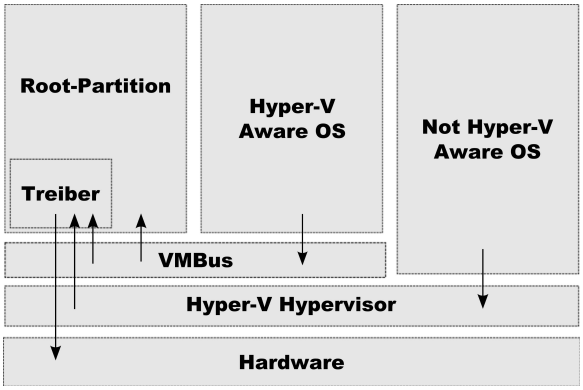

Abbildung 4: Hyper-V-Architektur

Vorteil, dass diese automatisch bei PIO einen Interrupt erzeugt, der an den Hypervisor weitergeleitet wird. MMIO aber ist komplexer, da die selben Mechanismen auch verwendet werden um auf Speicher zuzugreifen. Bei MMIO-Zugriffen wird ein x86-Emulator verwendet anstatt einer Shadow Page Table wie in Abschnitt 3.3.2. Alle MMIO- und PIO-Aufträge werden an User-Modus-Software weitergegeben, die auch richtige Hardware-Interrrupts auslösen kann.

3.4 Hyper-V

Microsofts Hyper-V integriert sich nahtlos in das Betriebssystem. Somit lässt es als ein Typ-1-Hypervisor klassifizieren. Hyper-V, am Anfang spezialisiert auf die Virtualisierung von Windows Betriebssystemen, unterstützt seit Release 2 (Hyper-V R2) Linux in allen Variationen [32]. Offiziell wird nur Suse und Red Hat unterstützt, für diese bietet Microsoft Support an. Treiber für Linux, die die Leistungsfähigkeit an das übliche Niveau einer Vollvirtualisierungslösung bringen, sind seit neuestem verfügbar und Open-Source. Erwähnenswert ist noch, dass Microsoft Paravirtualisierung „Enlightement" nennt und einen Satz von Treibern und virtuellen Geräten „Integration Components".

In einer virtuellen Maschine kann das Betriebssystem nicht direkt auf die Hardware zugreifen, alle Geräte sind nur virtuell präsent. Bei Treibern ohne Hypervisorunterstützung gehen alle Operationen mit Geräten an den Hypervisor, bei Hypervisor-optimierten Treibern an ein internes Busssystem (VMBus), das sie an die Root-Partition weitergibt [3]. Die Verwendung des VMBuses verhindert einen Kontextwechsel und ist somit effizienter als die nicht optimierte Version. Die Root-Partition entspricht der Dom0 unter Xen (Abbild 4).

3.5 VMware

VMware ist Pionier der Virtualisierungstechnologie. Die x86-Plattform gilt seit jeher als eine Plattform die sich nur schwer virtualisieren lässt. Ihr Befehlssatz ist groß und nicht alle privilegierten Befehle melden einen Fehler (Trap) bei Ausführung aus einem niederprivilegierten Ring. VMware gelang es als erstes diese Plattform zu virtualisieren [20]. VMWares Produktpalette ist groß, aber alle ihre Produkte, abgesehen von der Hosted-Virtualisierungssoftware, basieren auf dem selben Hypervisor; der Funktionsumfang betrifft nur die Administrationsoberfläche und mitgelieferte Tools [19].

Die ersten VMware-Produkte entstanden bevor es Hardware-Unterstützung für Virtualisierung gab. Sie arbeiteten von Anfang an mit binären Kernel-Patches [34]. Diese Hypervisors setzen kein Betriebssystem voraus und sind deshalb als Typ-1-Hypervisors einzuordnen. Der Nachteil ist, dass VMware-Hypervisors nur mit ausgewählter Hardware funktionieren, da sie die Treiber mitliefern müssen. Extra VMware-zertifizierte Hardware muss deshalb mit ihren Lösungen verwendet werden. Mit der Evolution der Hardware-unterstützten Virtualisierung entwickelten sich auch alle VMware-Hypervisors, sodass sie, falls vorhanden, Hardware-unterstützte Virtualisierung nutzen können [14].

3.6 NOVA

NOVA OS Virtualization Architecture (NOVA) ist ein junges Projekt der TU Dresden. Es ist ein Microhypervisor, der ähnlich eines Microkernels nur mit den absolut notwendigen Prozessen ausgestattet ist und alles andere auf User-Modus auslagert. So bietet der NOVA-Hypervisor nur Scheduling, IPC-Dienste und Ressourcen-Weitergabe an. Alle anderen notwendigen Dienste wie Treiber und Systemdienste bieten Applikationen an. Diese werden in einem getrennten User-Mode-Kontext betrieben. Zusätzlich wird ein Capabilities-Model unterstützt; dieses soll die Sicherheit gegen Einbrüche erhöhen [33]. NOVA wird im weiteren nicht mehr berücksichtigt, da es noch keine Bedeutung im Cloud-Computing-Bereich gewonnen hat.

3.7 Gemeinsamkeiten und Unterschiede

Die vorgestellten Hypervisors weisen deutliche Unterschiede auf bei der näheren Betrachtung der tatsächlich bereitgestellten Ressourcen und Anforderungen an den Wirt. Hyper-V unterstützt 64 virtuelle CPUs (vCPUs) pro Host während vSphere von VMware mit 512 vCPUs eine deutlich höhere Anzahl unterstützt. Beide wiederum unterstützen aber, mit 64 CPUs, die gleiche Anzahl von physikalischen CPUs pro Host. Red Hat Enterprise Virtualization for Servers (RHEVfS) (basiert auf KVM) kann mit 256 physikalischen CPUs pro Host beide deutlich übertreffen.

Im Bereich Netzwerkschnittstellen bietet sich eine ähnliche Sicht. RHEVfS unterstützt 8 virtuelle Netzwerkkarten (vNICs) pro Gast, vSphere 10 vNICs. Hyper-V unterstützt 8 vNICs für Gast-Gast Kommunikation und 4 für Kommunikation mit anderen Rechnern.

3.7.1 Speicherplatznutzung

VMware bietet aktuell den kleinsten Hypervisor, der kommerziell im Einsatz ist. Er belegt 60 MB auf der Festplatte [23], im Vergleich dazu setzt Hyper-V eine Windows Server 2008 Installation voraus mit 3GB, KVM/Xen eine ganze Linux-Installation. Dies ist nennenswert, um zu verdeutlichen, wie minimal ein Hypervisor ausgelegt sein kann, spielt aber in der Cloud keine große Rolle. Die Verwaltungs-Software, die noch benötigt wird, um diesen Hypervisor in eine nutzbare Cloud-Lösung umzuwandeln, ist um einiges größer als der Hypervisor selbst.

Um es an einem Projekt zu verdeutlichen, wie kompakt ein Hypervisor sein kann, bietet sich NOVA an. Der NOVA-Microhypervisor besteht aus 9.000 Lines of Code (LoC) und zusätzlich 27.000 LoC für den Zustandsmonitor, Applikationen und Treiber. Das ist sehr wenig verglichen mit den 100.000 LoC die Xen allein für den Hypervisor benötigt [33].

3.7.2 Speicherung der Gast-Betriebssysteme

Jede Hypervisor-Lösung verwendet ein anderes Format für die Speicherung ihrer Gastsysteme. Hyper-V setzt auf Virtual Hard Drives (VHD). Diese sind ein offenes und dokumentiertes Format. Citrix und Microsoft schlossen ein Abkommen ab, das es ermöglichen soll Xen und Hyper-V näher zu bringen. Daraus resultierte, dass sie sich auf VHD als ein Standardformat für die Speicherung von VMs einigten [8]. VMware speichert die virtuellen Maschinen im Virtual Machine Disk Format (VMDK), das bei Hypervisors in einem Virtual Machine Filesystem (VMFS) gelagert ist. VMDKs können ohne Probleme verschoben und kopiert werden. Wie VHD ist VMDK ein offenes und dokumentiertes Dateiformat, wird aber nicht direkt unterstützt von anderen Lösungen. KVM speichert VMs in einem von QEMU vorgegebenen Dateiformat qcow/qcow2. Alle Formate können mit geeigneten Tools untereinander konvertiert werden.

3.7.3 Speicherverwaltung

Page-Sharing und Memory-Ballooning werden von allen hier vorgestellten Hypervisors unterstützt. Dies ist von großer Bedeutung in der Cloud, denn nur so lässt sich RAM-Nutzung konsolidieren. Hyper-V unterstützt kein Memory-Overcommit. Microsoft vertritt den Standpunkt, es ergäbe sich kein Vorteil bei Produktions-Servern. Microsoft zufolge kann nach diesem Prinzip keine Stabilitätsgarantie gewährleistet werden, denn selbst Migrationen brauchen eine gewisse Zeit die unter Randbedingungen nicht ausreicht um gestiegene Speicheranforderungen zu kompensieren. Neuere Betriebssysteme, insbesondere Windows 7, nutzen in optimalen Fällen ihren ganzen RAM-Speicher aus, um Programmladezeiten zu verringern [32] indem sie viel genutzte Programme im Hintergrund in den Speicher laden (Superfetch).

3.7.4 VM-Vorlagen

Vorlagen für Gast-Betriebssysteme werden von allen Hypervisors in verschiedenem Maße unterstützt. Während VMware und Hyper-V eigene Verwaltungsprogramme für diese Aufgabe bereitstellen (VMware Virtual Center [5] und System-Center Virtual Machine Manager 2008 [4]), ist unter Xen und KVM noch vieles manuell zu erledigen [6, 30]. Mit Hilfe dieser Programme ist es auch möglich auf Remote-Rechnern Instanzen eines Betriebssystems zu starten.

3.7.5 Linked-VM

Diese Art VMs zu erstellen wird derzeit nur von VMware angeboten. Es ließen sich auch nach intensiver Recherche keine glaubhaften Quellen finden, die über die Aufnahme dieser Technologie in andere Hypervisors berichteten.

4. WECHSEL DER VIRTUALISIERUNGSUMGEBUNG

Wie in Abschnitt 3.1.4 beschrieben ist es möglich Betriebssysteme im laufendem Betrieb von einem Rechner auf einen anderen zu migrieren. Dies lässt vermuten, dass in einem Utility-Grid die Möglichkeit besteht eine VM ohne weiteres von einem Anbieter zu einem anderen zu migrieren. Die Realität sieht aber anders aus. Wie in Abschnitt 3.7 kurz beschrieben unterscheiden sich alle Hypervisors in ihren Gastsystemspeicherabbildungen und in den Funktionalitäten, die sie aufweisen. Somit ist es nicht möglich von einem Anbieter zu einem anderen zu wechseln ohne Vorkehrungen zu treffen.

Der Begriff Vendor Lock-in wird verwendet, um die Abhängigkeit an einem Anbieter zu verdeutlichen. Dies besteht bei aktuellen Lösungen in Bezug auf die Virtualisierungsumgebung nicht. VHD, VMDK, qcow lassen sich mit mehr oder weniger Aufwand auf andere Formate konvertieren [29, 37]. Die Möglichkeit an diese Images zu gelangen ist durchaus auch bei kommerziellen Anbieter wie Amazon EC2 gegeben [12].

Die Betreiber solcher Cloud-Computing-Utility-Grids bieten auch Programmierschnittstellen (APIs) an, um den VM-Benutzern eine bessere Kontrolle ihrer Gastsysteme zu ermöglichen. Diese können dem Wirt gestiegene Anforderungen mitteilen, um eine Migration einzuleiten. Diese APIs sind herstellerspezifisch und nicht miteinander kompatibel. Es existieren Initiativen diese API zu standardisieren. VMware stellte ihre vCloud API allen zur Verfügung, um eine Vereinheitlichung der Plattformen zu ermöglichen [11] sie schlägt diese auch dem Distributed Management Task Force, Inc. (DMTF) als Standard vor. Google App Engine stellt den eigenen Source-Code zur Verfügung [2]. Diese API ist nun auch in der Amazon EC2 S3 Cloud anwendbar [9]. Dies alles mindert das Vendor-Lock-in-Problem, aber eine von allen adoptierte Schnittstelle und Speichermöglichkeit existiert noch nicht.

5. FAZIT

In den letzten Jahren ist das Thema Virtualisierung sehr in die Aufmerksamkeit der Öffentlichkeit gerückt, wie die verschiedenen Angebote an Virtualisierungs-Software zeigen. Es wurden sowohl Lösungen der kommerziellen Software-Hersteller als auch von der Open-Source-Gemeinde präsentiert.

Für den produktiven Einsatz in Cloud-Computing-Rechenzentren werden ebenfalls die wichtigsten Punkte erfüllt wie Live-Migration und Speicherausnutzungsmechanismen, um Rechenleistung und Speicherkapazität effizient einzusetzen.

Im ganzen unterscheiden sich die Hypervisors nur in ihren Feinheiten, wie unterstützte Hardware und Größe des Hypervisors. Dennoch sind sie nicht äquivalent.

Eine standardisierte API und Speichermöglichkeit für die Gast-Betriebssysteme ist noch nicht vorhanden, es existieren dennoch diverse Initiativen, um diese Lücke in den nächsten Jahren zu schließen. Dies sollte die Hochverfügbarkeit der Cloud weiter erhöhen und eine einfachere Migration ermöglichen.

6. LITERATUR

[1] Denali: a brief overview. *http://denali.cs.washington.edu/overview.html*.

[2] Google App Engine. *http://code.google.com/appengine/downloads.html*.

[3] Hyper-V Architecture. *http://msdn.microsoft.com/en-us/library/cc768520\%28BTS.10\%29.aspx*.

[4] R2 Features. *http://www.microsoft.com/systemcenter/en/us/virtual-machine-manager/vmm-features.aspx*.

[5] VMware VirtualCenter 1.3.x Support Documentation. *http://www.vmware.com/support/vc13/doc/c13templateintro.html*.

[6] Xen Templates. *http://conshell.net/wiki/index.php/Xen_Templates*.

[7] Erdbeben im Virtualisierungsmarkt. *http://www.computerwoche.de/heftarchiv/2007/34/1220457/index.html*, 2007.

[8] Microsoft and Citrix Extend Virtualization Alliance. *http://www.microsoft.com/presspass/press/2007/sep07/09-11VirtualizedDesktopPR.mspx*, 2007.

[9] Exclusive: Google App Engine ported to Amazon's EC2. *http://waxy.org/2008/04/exclusive_google_app_engine_ported_to_amazons_ec2/*, 2008.

[10] Linked Virtual Machines. *http://www.vmware.com/support/developer/vc-sdk/linked_vms_note.pdf*, 2009.

[11] VMware Submits VMware vCloud API Specification to the Distributed Management Task Force (DMTF) – First Ever Submission of Key Cloud Interface. *http://www.vmware.com/company/news/releases/vcloud-api-vmworld09.html*, 2009.

[12] Converting Amazon EC2 Windows Instances to VMware Workstation or VMware vSphere Virtual Machines. *http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1018015*, 2010.

[13] XenNetworking. *http://wiki.xensource.com/xenwiki/XenNetworking#head-d7bb01c62c0cc1b20602dbeca666ec5be873e099*, 2010.

[14] O. Agesen. Software and Hardware Techniques for x86 Virtualization. *http://www.vmware.com/resources/techresources/10036*, 2009.

[15] P. Barham, B. Dragovicand, K. Fraserand, S. Handand, T. Harrisand, A. Hoand, R. Neugebauery, I. Prattand, and A. Warfield. Xen and the Art of Virtualization. *http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf*, 2003.

[16] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live Wide-Area Migration of Virtual Machines Including Local Persistent State. *VEE '07*, 2007.

[17] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. *Proceedings of the 2nd Symposium on Networked Systems Design & Implementation*, pages 273–286, 2005.

[18] T. Cloer. Sehr gesundes Wachstum im Markt für Virtualisierungs-Software. *http://www.computerwoche.de/knowledge_center/virtualisierung/1887063/*, 2009.

[19] D. Davis. How does VMware ESXi Server compare to ESX Server? *http://www.virtualizationadmin.com/articles-tutorials/vmware-esx-articles/general/vmware-esxi-server-compare-esx-server.html*, 2009.

[20] S. W. Devine, E. Bugnion, and M. Rosenblum. Virtualization system including a virtual machine monitor for a computer with a segmented architecture. *United States Patent*, 1998.

[21] M. DiPetrillo. Memory Overcommitment in the Real World. *http://blogs.vmware.com/virtualreality/2008/03/memory-overcomm.html*, 2008.

[22] D. M. Engel and M. Haupt. Case Study: Xen. *http://www.uni-marburg.de/fb12/verteilte_systeme/lehre/vl/virtual_tech*, 2010.

[23] E. Gray. If VMware ESXi 4 is so small, why is it so big? *http://www.vcritical.com/2009/08/if-vmware-esxi-4-is-so-small-why-is-it-so-big/*, 2009.

[24] Q. Inc. KVM: Kernel-based Virtualization Driver. *http://www.linuxinsight.com/files/kvm_whitepaper.pdf*, 2006.

[25] A. Kivit, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. KVM: the Linux Virtual Machine Monitor. *www.kernel.org/doc/ols/2007/ols2007v1-pages-225-230.pdf*, 2007.

[26] D. Lohmann. Betriebsysteme, Vorlesung 6, IA-32 Das Programmiermodell der Intel-Architektur. *http://www4.informatik.uni-erlangen.de//Lehre/WS09/V_BS//Vorlesung/folien/06-IA32-2x2.pdf*, 2009.

[27] N. Paulussen. Virtualisierung von Betriebssystemen: Das VM Betriebssystem von IBM. *http://www.fh-wedel.de/~si/seminare/ws06/Ausarbeitung/02.VMware/vmware1.htm*, 2006.

[28] D. O. Singer. Aktueller Begriff Cloud Computing. *http://www.bundestag.de/dokumente/analysen/2010/cloud_computing.pdf*, 2010.

[29] E. Sloof. Convert Virtual Disks from VHD to VMDK. *http://www.ntpro.nl/blog/archives/194-Convert-Virtual-Disks-from-VHD-to-VMDK.html*, 2007.

[30] H. Solomon. How you can use QEMU/KVM base images to be more productive (Part 1). *http://www.linux-kvm.com/content/how-you-can-use-qemukvm-base-images-be-more-productive-part-1*, 2008.

[31] S. Spector. Xen 4.0 Feature List. *http://blog.xen.org/index.php/2010/04/13/xen-4-0-feature-list/*, 2010.

[32] C. Steffen. Guest post: Setting the Record Straight - 9 Reasons Why Hyper-V is a Great Choice for Enterprises. *http://blogs.technet.com/virtualization/default.aspx?p=2*, 2010.

[33] U. Steinberg and B. Kauer. NOVA: A Microhypervisor-Based Secure Virtualization Architecture. *Eurosys 2010, Session 6, Talk 16*, 2010.

[34] VMware. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. *http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf)*, 2007.

[35] C. Waldspurger. Ballooning. *http://www.usenix.org/event/osdi02/tech/waldspurger/waldspurger_html/node6.html*, 2002.

[36] K. Ward. KVM: Bare-Metal Hypervisor? *http://virtualizationreview.com/blogs/mental-ward/2009/02/kvm-baremetal-hypervisor.aspx*, 2009.

[37] B. Zazen. Convert VMware .vmdk to KVM .qcow2 or Virtualbox .vdi. *http://blog.bodhizazen.net/linux/convert-vmware-vmdk-to-kvm-qcow2-or-virtualbox-vdi/*, 2009.