KVM on Z vs. ESXi

Dane Wicki

10. Oktober 2017

Zusammenfassung

Virtualisierung hat seinen Ursprung schon in den Frühen 1960er Jahren, als die Computersysteme noch gross und teuer zu warten waren. Der Ursprung findet sich bei den Mainframes, eine Plattform, welche auch heute noch in Gebrauch ist, obschon sie tot geglaubt wurde. IBM entwickelte die Virtualisierung, damit die User die Computer Ressourcen untereinander teilen konnten, und diese Trotzdem noch parallel gebraucht werden konnten. Heutzutage erfreut sich die Virtualisierung eines enormen Hypes, welcher zu vielen verschiedenen Softwarelösungen geführt hat.

Die Virtualisierung ermöglicht das ausführen mehreren Virtuellen Maschinen, welche auf einem einzigen Host laufen. Diese Arbeit soll nun einen Vergleich zweier moderner Virtualisierungslösungen aufzeigen. KVM on Z ist die Erste Lösung, auf welche eingegangen werden wird. Es ist eine Opensource Lösung, welche in den Linux Kernel integriert wurde. ESXi, die 2. Lösung, ist eine Weitere Möglichkeit um zu Virtualisieren. Diese Lösung wird von VMWare vertrieben und ist nicht ganz so Quelloffen wie seine zu gegenüberstellende Lösung.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In der Heutigen welt nimt die Virtualisierung eine wohl immer Stärkere Position ein. Die stetig steigende Leistung der Hardware ist dabei ein Wesentlicher bestandteil dabei. Zudem bietet die Virtualsiserung eine vereinfachung der Administration, da mehrere Betriebssysteme auf einer Hardware laufen kann und dadurch nur diese eine Hardware gewartet werden muss. In der Welt der Virtualisierung gibt es immer weitere Lösungen und man kann zwischen vielen verschiedenen Anbietern aussuchen. Hier wird auf zwei Lösungen eingegangen und diese in einem Kleinen vergleich gegenübergestellt.

Kapitel 2

Grundlagen

2.1 Grundlagen der Virtualisierung

Virtuell ist laut Duden etwas, was nicht echt, nicht in der Wirklichkeit vorhanden, also nur scheinbar vorhanden ist. Wenn wir nun von einer Virtualisierung sprechen im Bereich der IT hat dies viel mit dieser Definition zu tun. Wenn wir nun eine Betriebssystem Instanz auf einem nicht vorhanden, also virtuellen Rechner installieren, nennen wir das Virtualisierung. Aber was meine ich nun mit dem virtuellen Rechner und wie kann ich darauf etwas laufen lassen wenn er doch nicht existiert?

Nehmen wir doch mal an ich habe einen nicht Virtuellen Rechner, also einen wirklich vorhanden rechner vor mir. Nun Installiere ich das Betriebssystem auf eben jenem Rechner, so habe ich einen uns gewohnten Rechner. Nun könnte ich jedoch auf diesem Rechner eine Software laufen lassen, die einen weiteren Rechner herstellt, also einer, welcher nicht wirklich vorhanden ist, nur scheinbar vorhanden ist, so ist dieser Rechner virtuell. Und wie auf einem vorhandenen Rechner, kann ich auf diesem virtuellen Rechner eine Betriebssystem Instanz installieren,

so habe ich einen Virtuellen Rechner.

Also versteht man unter der Informatik unter dem Begriff Virtualisierung unter anderem die Technologie, bei welcher das Betriebssystem nicht unmittelbar auf einer physischen Instanz, einem physisch vorhanden Rechner installiert ist, sondern auf der Hardware über einer Zwischenschicht. Diese Zwischenschicht abstrahiert die reale physische Hardware, ist also nicht wirklich vorhanden. Die auf dieser Zwischenschicht laufende Betriebssystem Instanz wird als Gastsystem bezeichnet.

Auf der realen physikalischen Hardware muss jedoch auch eine Software laufen, welche diese virtuellen Systeme erstellt, diese Software wird als Hypervisor bezeichnet. Der Hypervisor dient in erster Linie der Verwaltung und Zuteilung der Ressourcen für die einzelnen Gastsysteme. Dabei soll der Hypervisor die Ressourcen so verteilen, dass das Gastsystem die Ressourcen zur Verfügung gestellt bekommt, wenn es diese auch benötigt.

Dank der Virtualisierung lassen sich also mehrere Gastsysteme auf einem einzelnen Hypervisor zum Laufen zu bringen. Diese Gastsystem sind zudem nicht direkt abhängig von der bestehenden Hardware, so dass es auch möglich ist, dasselbe Gastsystem auf mehreren verschiedenen physikalischen Rechner zum Laufen zu bringen sofern der Hypervisor auf dieser neuen physikalischen Instanz zum Laufen gebracht werden kann. Dies bringt den Enormen Vorteil, dass ein virtualisiertes Betriebssystem auf einfache Art und Weise auf einen neuen Rechner gezogen werden kann. Weiter bietet die Möglichkeit, dass mehrere Gastsysteme auf einer physikalischen Instanz laufen können den Vorteil, dass die Ressourcen besser ausgenutzt werden können. Dies wird vor allem bei Servern und Mainframes geschätzt, da die Hardware in diesen Bereichen teuer ist und man diese, wenn man schon viel Geld

bezahlt auch ausnutzen möchte.

2.1.1 Hypervisor

Hypervisors werden auch Virtual-Maschine-Monitor (kurz. VMM) genannt. Diese Hypervisoren sind eine Klasse von Systemen, die als eine abstrahierende Schicht zwischen tatsächlicher vorhandener Hardware und weiteren zu installierenden Betriebssystems dienen.
Sie dienen der Verwaltung der Ressourcen für die einzelnen Gastsysteme. Im Bereich der Hypervisoren gibt es ein Klassifizierungssystem, bei welchem diese in 2. Typen unterteilt werden.

Type 1

Ein Typ-1-Hypervisor (Bare-Metal oder auch native genannt) setzt direkt auf der Hardware auf. Es benötigt kein Betriebssystem, auf welchem dieses Läuft. Es wird jedoch vorausgesetzt, dass die Hardware des Host Systems vom Typ-1-Hypervisor durch entsprechende Treiber unterstützt wird.

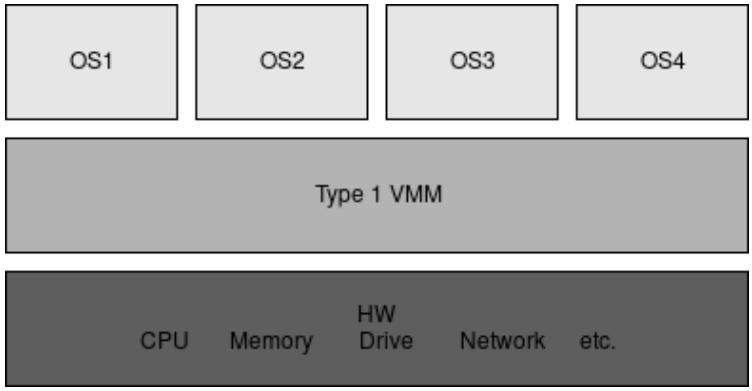

Abbildung 2.1: Aufruf des FilterVIs

Type 2

Ein Typ-2-Hypervisor setzt auf einem vollwertigen Betriebssystem, welches auf dem Hostsystem installiert ist, auf und nutzt die Treiber, des Betriebssystems, um auf die Hardwareressourcen des Hostsystems zugreifen zu können. Dementsprechend sind alle Typ-2-Hypervisoren auf einem Hostsystem lauffähig, sofern sich auf diesem das unterstützte Host Betriebssystem installieren lässt.

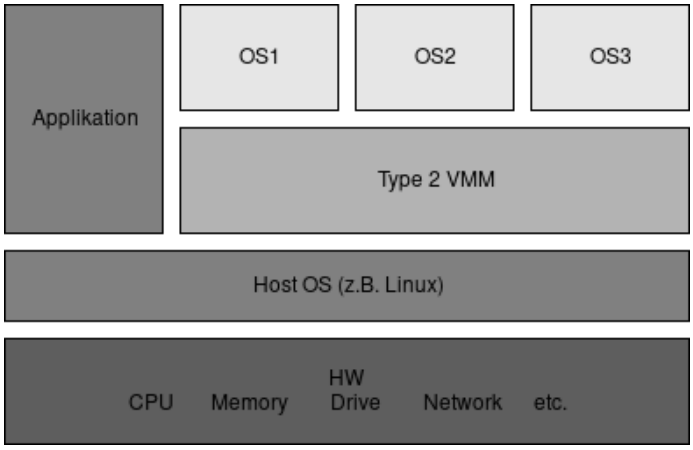

Abbildung 2.2: Aufruf des asd

Kapitel 3

KVM

KVM steht für Kernel Based Virtual Machine. bei KVM handelt es sich um eine Infrastruktur des Linux-Kernels zur Virtualisierung. Die Virtualisierung auf verschiedenen Plattformen unterstützt, unter anderem die folgenden Intel, AMD, ARM, PowerPC und System Z. KVM selbst nimmt keine Emulation vor, es ist also nicht möglich eine Bestimmte Hardware zu emulieren mit KVM selbst. so würde es sich als enorm Schwer erweisen ein vollwertiges Betriebssystem auf dem Hostsystem nur mit Hilfe von KVM zu installieren. Um die Emulation zu ermöglichen hat sich QEMU als de facto Standard etabliert. QEMU steht für Quick Emulator uns stellt für virtualisierte Gastsysteme die notwendigen Geräte wie Festplatten, Netzwerk-, Sound- und Grafikkarten zur Verfügung.
KVM ist desweiteren auch eine Quelloffene Lösung, und ist neben XEN der Weitverbreitete Quelloffene Hypervisor. Trotz einigen aussagen ist KVM jedoch ein Typ 2 Hypervisor und nicht wie XEN ein Typ 1. [1]

3.1 Funktionsumfang

KVM ist bereits bei der Installation eine sehr schlanke und einfache Umgebung. Es sind im Wesentlichen nur die KVM Kernel Module dem bestehenden System zu installieren sowie Qemu und Management-Tools einzurichten. Der Vorteil liegt hier auch, dass viele schon bestehende Linux-Server nachträglich zu einem Virtualisierungssystem aufgerüstet werden kann ohne damit eine Komplette Neuinstallation tätigen zu müssen. Weiter verhält sich jede Virtuelle CPU (kurz VCPU) im Gastsystem wie ein gewöhnlicher Linux-Process und kann so beispielsweise auch über normale Kommandos wie kill oder top kontrolliert und gesteuert werden. Dasselbe gilt auch für die Gerätelandschaft. Da mit KVM und Qemu die normalen Linux-Treiber genutzt werden können, ist eine Umgewöhnung für einen Linux-Administrator nicht nötig.

KVM in Zusammenarbeit mit Qemu unterstützt eine enorme Anzahl an Gast-Betriebssysteme, darunter finden sich fast Sämtliche wichtigen Windows-Varianten sowie Linux, Solaris, BSD, FreeBSD und einige exotischere wie ReactOS.

Ein Nachteil der KVM ist jedoch, dass sie nicht auf Graphische Systeme ausgerichtet ist, es ist also die Idee, dass ein Gastsystem, welches auf einer KVM Instanz läuft, nur über ein Terminal bedient wird.
Beim Management zeigt sich, wie stark sich die Marktposition dieses Quelloffenen Open-Source-Hypervisors verbessert hat. Es ist inzwischen eine Fülle von Administrationswerkzeugen verfügbar, zudem wird KVM in vielen Cloud-Plattformen eingebaut. Durch die simplen Tools virt-manager sowie virsh ist schon eine Remote-Management möglich. Für fortgeschrittene Funktionen wie Orchestrierung ganzer Pools von Virtuellen Maschinen mit weitergehenden Funktionen wie Failover, High Availability und dergleichen gibt es weitere Lösungen.

Hier springen Dritt-Objekte sowie Softwarehersteller in die Bresche, normalerweise auch Open Source.

Es gibt zudem für auf KVM aufbauende Komplettlösungen für die Servervirtualisierung. allen voran RHEV (Red Hat Virtualization). Für die Verwaltung der Gastsysteme geben im Wesentlichen viele verschiedene Tools, von einfachen verwaltungssoftware für den Desktop PC wie zum Beispiel Gnome-Boxe [1]

3.2 KVM on IBM z System

KVM for IBM z Systems ist eine kostengünstige Alternative zu der von der IBM selbst entwickelten Virtualisierungslösung z/VM. KVM for IBM z System besitzt ein einfaches und gewohntes User-Interfaces, welche eine einfache Integration der z System Plattform in die IT Infrastruktur ermöglicht. KVM for IBM z System ermöglicht zudem das den einzelnen Gastsystemen mehr Ressourcen zugeteilt wird, als das z System tatsächlich zur Verfügung stellt, dieser Umstand ermöglicht die optimale Auslastung der virtuellen Umgebung. Zusätzlich macht es die Plattform Mobilität einfacher und ist sogar in der Lage VMs und workloads zwischen verschiedenen Instanzen von KVM for IBM z System zu verschieben ohne dabei eine Ausfallzeit zu haben. [4]

Feature	**Benefit**
KVM Hypervisor	Unterstützt das ausführen mehrere nicht gleicher Linux VMs auf einem einzelnen System.
CPU sharing	Erlaubt es die Ressourcen der CPU über mehrere VMs hinweg zu teilen.
I/O sharing	Ermöglicht das teilen von I/O Ressourcen zwischen den VMs.
Memory and CPU over-commitment	Erlaubt es einer VM mehr CPU, Memory und swapping space zu zuteilen als effektiv vorhanden ist.
Live VM relocation	Ermöglicht die Arbeitsbelastung zu migrieren mit minimalem Einfluss.
Dynamic addition and deletion of virtual I/O devices	Reduziert Downtime um die I/O Geräte für die VMs zu konfigurieren.
Thin-provisioned VMs	Die VMDisk benötigt nur den belegten Speicherplatz, es wird so speicher gespart.

Hypervisor Performance Management	Unterstützung von policy-based, goal-oriented Management und Monitoring der virtuellen CPU Ressourcen.
Installation and Konfiguration Tools	Mitliefern von Tools um KVM for IBM z System zu installieren und Konfigurieren.
Transactional execution use	Bietet verbesserte Performance für das ausführen von multi-threaded Applikationen.

Tabelle 3.1: Auszug der features für IBM z System [2]

http://www.redbooks.ibm.com/redbooks/pdfs/sg248332.pdf

3.2.1 Management Tools für KVM on IBM z System

Dynamic Partition Manager kurz DPM. DPM ist ein geführtes Management Interface im HMC (IBM Hardware Management Konsolen), mit welchem die z System Hardware und die virtuelle Infrastruktur mitsamt dem integriertem dynamischen I/O Management definiert werden kann.
Unattended Installation (Unbeaufsichtigte Installation) Unattended Installation des KVM Hypervisors vereinfacht die Administrations. Dies wird erreicht, indem ein Kickstart File erstellt wird, sowie dem hinzufügen des inst.auto=path_to_kickstart Parameters im generic.prm File.

Single Hypervisor Management GUI Kimchi ist ein OpenSource Management Tool, welches über ein intuitives Grafisches User Interface (GUI) für die folgende Management aufgaben verfügt.

- Netzwerk Konfiguration für die OSA-based NICs

- Storage Konfiguration für ECKD und SCSI Geräte
- Standard System Informationen und Statistiken
- Debug Reports
- Hypervisor Neustart

Enhanced Problem Determination Support Viele verschiedene Tools wurden noch hinzugefügt um Probleme zu finden:

- Ziorep -¿ eine Suite bestehend aus Tools, welche die SCSI Performance überwachen können.
- Das DASD dump Tool um einzelne DASD Geräte entleeren zu können.
- Das KDUMP Feature, welches Crash Dumps erstellt, falls ein Kernel Crash eintritt.
- dbginfo Skript, welches verschiedene system-related Files für Debug zwecke sammelt.
- Das sosreport Skript sammelt System fehlerbehebungs Informationen

Kapitel 4

ESXi

VMware ESXi (früher bekannt als ESX) ist ein Enterprise-Class, Typ-1 Hypervisor. Entwickelt wird dieser Typ-1 Hypervisor von VMware. Als ein Typ-1 Hypervisor, ist ESXI nicht eine Software Applikation, welche auf einem bestehenden Betriebssystem installiert wird, sondern, es beherbergt in sich schon wesentliche Komponente eines Betriebssystems, wie etwa den Kernel.

Nach der Version 4.1 benannte VMware ESX zu ESXi, ESXi ersetzte damals die Service Konsolen welches lediglich ein rudimentäres Betriebssystem war, mit einem besser integrierten Betriebssystem. ESX/ESXi ist die wichtigste Komponente der VMware Software Infrastruktur.

4.1 Funktionsumfang

Bei ESXi gibt es noch eine Sonderheit. ESXi ist zwar an sich kostenlos. So kann auf jedem Server / Rechner ESXi installiert werden ohne eine Lizenz bezahlen zu müssen. Möchte man jedoch ESXi auch verwalten und Konfigurieren so muss man sich den VMware vSphere hinzukau-

fen. Dies ist das Management Tool um den ESXi Hypervisor über die Distanz, also über das Lokale LAN zu konfigurieren. Dieses vSphere wird von VMware in 3 Lizenzversionen verkauft, welche über eine unterschiedliche Funktionspalette verfügt. Diese wurde nun in der Folgenden Tabelle dargelegt.

Funktion und Merkmale	**Standard**	**Enterprise Plus**	**Operations Management Enterprise Plus**
VMotion (Live-Migration)	✓	✓	✓
Storage VMotion (Live-Migration des Storages, z.B. in ein Cluster)	✓	✓	✓
Data Backupsolution	✓	✓	✓
Fault Tolerance (Unterbrechungsfrei bei bis zu x VCPUs)	2 VCPUs	4 VCPUs	4 VCPUs
Anti Virus System für alle VM (läuft nicht in den VMs)	✓	✓	✓
VSphere Replication (ermöglicht Replikation von Daten aus VM heraus)	✓	✓	✓
VCenter HA	✓	✓	✓
RecoverySolution	✓	✓	✓
Verschlüsselung der VMs		✓	✓

Virtuelle Laufwerke	✓	✓	✓
Storage Policy-Based Management	✓	✓	✓
Distributed Resource & Power Sheduler (Schaltet nicht gebrauchte VMS ab)		✓	✓
Storage DRS (Speichert Daten der VMs so, dass Sie schneller gefunden werden)		✓	✓
NVIDIA GRID vGPU		✓	✓
Flexible Operations Policies and Operations Groups			✓
Role-based Access Control			✓
Scale-out Platform			✓

Tabelle 4.1: Darstellung eines kleinen Funktionsauszuges von VMware[2]

Kapitel 5

Vergleich

Um dem Vergleich einen fairen Tuch zu verleiten wurde hier Ausschliesslich KVM und ESXi miteinander verglichen und nicht KVM on IBM z System, da dieses Komplettpacket wohl für zwei total unterschiedliche einsatzgebiete gebraucht werden und dementsprechend kein guter vergleich zustande kommen könnte.

Es werden nur die Folgenden Punkte miteinander verglichen:

- Performance
- Integration
- Kosten
- Komplexität
- Reife (Ausgereift?)
- Skalirbarkeit
- Funktionssupport

Performance

Hypervisoren sind in zwei unterschiedlichen Typen unterteilt, welche einen Einfluss auf die Performance haben. Type 1 Hypervisoren, welche auch bekannt als "Bare-Metalßind, laufen direkt auf der physikalischen Hardware, und das Betriebssystem jedes Gastsystemes läuft auf dem Hypervisor. Diese Hypervisoren erlauben normalerweise einigen Gästen den Hypervisor zu Kontrollieren. Viele Unternehmen setzen auf den Typ 1 Hypervisor.

Ein Typ 2 Hypervisor, auch bekannt als hosted Hypervisor, läuft in einem Betriebssystem, welches wiederum auf einer Physikalischen Hardware läuft. Das Gastsystem läuft anschliessend auf dem Hostsystem. Desktop Hypervisoren sind meistens Typ 2 Hypervisors.

Das beste Beispiel für einen puren Typ-1 Hypervisor ist XEN, jedoch auch ESXi, welches definitiv auch ein Type-1 Hypervisor ist, da es keine Applikation ist, welche auf einem Betriebssystem installiert wird. ESXi besteht aus einem Kernel und weiteren OS Komponenten, welche es ins eigene native OS integriert hat.

Die Klassifikation von KVM stellt ein etwas grösseres Problem dar. Der Grund für die Schwierigkeiten ist, dass KVM eigenschafte beider Hypervisor Typen beinhaltet. Es wird als eine Linux Komponente vertrieben. Dies bedeutet, dass ein Linux User KVM von der Command Line oder per GUI starten kann. Das KVM so gestartet werden kann lässt es so aussehen, als sei es eine Applikation, welche auf einem Betriebssystem läuft, obschon KVM im Kernel integriert ist und dadurch auch direkt auf der Maschine Läuft. Das Host Betriebssystem stellt KVM ein Start Mechanismus, nach welchem es eine Co-Processing Beziehung mit ihm aufbaut. Diese Co-Processing Beziehung erlaubt es KVM die physische Hardware direkt zu Kontrollieren. KVM macht von den Prozessor Virtualisierung Instruktionen auf der x86 Hardware gebrauch. Dies erlaubt es dem Hypervisor und all seinen Gästen direkt auf der Hardware zu laufen, also Bare-Metal. Die physikalische

Hardware für die meisten Resource Translation aus, so das KVM die traditionellen Kriterien für einen Typ 1 Hypervisor erfüllen würde und dennoch wird es als ein Typ 2 Hypervisor eingestuft.

Auch wenn bekanntermassen ein Typ 1 Hypervisor schneller laufen sollte als ein Typ2 so ist dies bei KVM und ESXi nicht der Fall. KVM kann hier mit einer besseren Startzeit punkten und auch einer etwas besseren Performance, was sich jedoch bei einem normalen Workload kaum unterscheiden lässt. [3]

Integration

Die verschiedenen Hypervisoren nutzten auch verschiedene Methoden um mit der physikalischen Hardware des Hosts zu Kommunizieren. KVM benutzt dazu einen Agent, welcher auf dem Host installiert ist um mit der Hardware zu Kommunizieren, ESXi hingegen nutzt VMware Management plane um mit der Hardware zu kommunizieren. VMware's Lösung ermöglicht es ESXi auf anderen VMware Produkte, welche auch die Management plane nutzten, zuzugreifen. Dies erfordert jedoch auch, dass ESXi VMware control stack verwenden muss, was zu einer erhöhten Hardware Requirement führt.

Starke Integration mit dem Host OS ist normalerweise der primäre Grund, weshalb Linux Entwickler KVM bevorzugen. Da KVM kurz nach dessen Release in den Linux Kernel integriert wurde. Zum Vergleich, Xen wurde nicht offiziell in den Kernel integriert bis 2011, also 8 Jahre nach dessen Einführung. Zudem wird es bevorzugt, da viele Linux distributionen KVM als präferierten Hypervisor adaptiert haben. [3]

Kosten

KVM gewinnt klar gegen VMware wenn es zu den Kosten kommt. KVM ist OpenSource, dies bedeutet, dass keine weiteren Kosten für

den Endbenutzer anfallen. Es wird oft auf verschiedenen Wegen vertrieben, meist als ein Teil einer open Source Lösung.

VMware erhebt licenz gebühren für den gebrauch ihrer Produkte. Sie sind dazu auch im stande, da sie das erste Unternehmen waren, welches Enterprise Softwarelösungen für die Virtualisation veröffentlicht haben. Dies führte zu dem Umstand, dass sie auch heute noch der Marktführer in diesem segment sind. Die Marke VMware ist deshalb immernoch relevant für einen Business end user, auch wenn die Entwickler anders darüber denken. Um den ESXi betreiben zu können muss die Licence für den gebrauch von vSphere gekauft werden. vSphere ist VMware's Funktionspallete um ESXi zu konfigurieren. Es könnte sein, dass zu diesen Licenz kosten noch weitere Licenzen dazugekauft werden müssen, was die Kosten für die Betreibung eines ESXi steigen lässt.

IBM hat die Total Cost of Ownership für KVM und VMware im Jahre 2012 ausgerechnet. Die Berechnungen haben aufgezeit, dass KVM durchschnittlich 39% günstiger waren als VMware. Die Kosten hängen jedoch stark von den Firmen-Specifischen faktoren wie etwa den Operational setting und des Workloads. [3]

Komplexität

Ein vergleich auf ebene des Quellcodes von KVM und VMware führte zu einer klaren differenz der länge, welche einen Effekt auf die Wartungskosten eines Hypervisors ausübt. KVM wurde ursprüngliche entwickelt um die Eingebauten Processor funktionen, welche die Virtualisierung ohne übersetzten des Binary Codes erlaubten, zu gebrauchen. Dieser umstand verdankt es KVM, dass der Hypervisor zum Release zeitpunkt eine sehr leichte virtualisierunglösung war mit etwas mehr als 10,000 zeilen code. Bei VMware kann die Angaben nur vermutet werden, da der Quellcode im gegensatzt zu KVM nicht öffentlich zugänglich ist. Es wird jedoch vermutet, dass der Codes mehr als 6

Millionen Zeilen aufweist. Diese Imense menge an Code hat vermutlich keinen grösseren effekt auf die Performence, da auch VMware auf die HArdware extensions der Processoren aufbaut, jedoch sorgt es dafür, dass mehr Fehler auftreten können und diese auch länger brauchen um gefunden zu werden. [3]

Reife

KVM sowie ESXi sind beide sehr Ausgereift und stabil. KVM ist nun bereits seit über einem Jahrzehnt in den Linux Kernel integriert und ESXi ist auch schon seit 2006 in gebrauch. Jedoch ist KVM wohl weiter verbreitet über alle felder hinweg (auch Desktop PC), da es eben open Source ist und bereits in Sehr vielen Distributionen integriert ist. Zudem unterstützt KVM mehr features als jeder andere Hypervisor. [3]

Scalability

KVM ist im Allgemeinen etwas besser Skalierbar als ESXi. Vorallem weil vSpehre, die verwaltunssoftware von VMware, einige limitationen mitbringt, mit welchem es den Server managenkann. Es gibt jedoch bereiche in denen Klar ESXi die Nase forn hat, einer der punkte liegt in der Storage option. VMware hat eine grosse anzahl an Storage Area Network hinzugefügt um verschiedene Hersteller zu unterschtützen, dies kann jedoch bei einem Scaling up auch ein kleiner Nachteil sein. [3]

Functionality Support

Hypervisoren unterscheiden sich stark in ihrer Funktionalität. Netzwerk- und Speicherunterstützung sind besonders wichtig und sind wahrscheinlich wichtiger als viele weitere Funktionen ausser vielleicht die Integration in das Betriebssystem. Es sollte keine allzu grosse Überraschung

sein, dass die ESXi-Unterstützung für andere VMware-Produkte von keinem anderen Hypervisor erreicht wird. Auf der anderen Seite bietet KVM mehr Möglichkeiten für die Netzwerkunterstützung als VMware. [3]

Kapitel 6

Fazit

KVM ist in der Regel die beliebteste Wahl für Benutzer, die sich Sorgen über die Kosten für den Betrieb jeder einzelnen VM machen und weniger an Funktionen auf Unternehmensebene interessiert sind. Diese gilt in erster Linie für Anbieter von Cloud- und Host-Services, die besonders sensibel auf die Kosten und die Dichte ihrer Server reagieren. Weshalb die anbieter von Cloud- und Host-Services wohl eher eine Open-Source Hypervisor wählen, höchstwahrscheinlich KVM.

Die enge Integration mit dem Host-Betriebssystem ist einer der häufigsten Gründe für Entwickler, sich für KVM zu entscheiden, insbesondere für diejenigen, die Linux verwenden. Die Einbindung von KVM in viele Linux-Distributionen führt dazu, dass es die praktischere Wahl für Entwickler ist.

Es gilt jedoch immer zu beachten, dass für die Entsprechende Umgebung und den gwünschten Funktionsumfang eine eigene Kurze Analyse gemacht werden muss um so siene den Umständen entsprächende Optimale wahl zu treffen.

Literatur

[1] Von Andrej Radonic. *Kostenlose Virtualisierungssoftware im Vergleich*. 22. Sep. 2017. URL: https://www.tecchannel.de/a/kostenlose-virtualisierungssoftware-im-vergleich,2051909 (besucht am 06. 10. 2017).

[2] VMWare. *vSphere und vSphere with Operations Management*. URL: https://www.vmware.com/ch/products/vsphere.html (besucht am 05. 10. 2017).

[3] *VMware vs KVM*. Okt. 2017. URL: https://www.rippleweb.com/vmware-vs-kvm/.

[4] Bill White; Marian Gasparovic; Franco Pinto; Richard Young. *Red books Front cover Getting Started with KVM for IBM z Systems*. URL: http://www.redbooks.ibm.com/redbooks/pdfs/sg248332.pdf (besucht am 05. 10. 2017).