KVM on Z vs. ESXi

Dane Wicki

10. Oktober 2017

Zusammenfassung

Virtualisierung hat seinen Ursprung schon in den Frühen 1960er Jahren, als die Computer systeme noch gross und teuer zu warten waren. Der Ursprung findet sich bei den Maneframes, eine platform, welche auch heute noch in gebrauch ist, obschon sie totgeglaubt wurde. IBM entwickelte die Virtualissierung, damit die users die Computer resourcen untereinander teilen konnten, und diese Trozdem 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 machinen, 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

TODO

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 Betriebssysteminstanz auf einem nicht vorhanden, also virtuellen rechner installieren, nennen wir das Virtualisirung. 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 gewonnten 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 vorhanden rechner, kann ich auf diesem virtuellen Rechner eine Betriebssysteminstanz installieren, so

habe ich einen Virtuellen rechner.

Also versteht man unter der Informatik unter dem begriff Virtualisierung unter anderem die Technologie, bei welcher das Betriebsystem nicht unmittelbar auf einer physischen Instanz, einem physisch vorhanden rechner installiert ist, sondern auf der HArdware über einer Zwischenschicht. Diese Zwischenschicht abstrahiert die reale phyische hardware, ist also nicht wirklich vorhanden. Die auf dieser Zwischenschicht laufende Betriebssysteminstanz wird als Gastsystem bezeichnet.

Auf der realen physikalischen Hardware muss jedoch auch eine software laufen, welche diese virtuellen systeme erstellt, diese Software werden als Hypervisor bezeichnet. Der Hypervisor dient in erster linie der verwaltung und Zuteilung der Resourcen für die einzelnen Gastsysteme. Dabei soll der Hypervisor die Resourcen so verteillen, dass das Gastsystem die Resourcen 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, das Selbe gastsystem auf mehreren verschiedenen physikalsichen renchner 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 physikalsichen instanz laufen können den vorteill, dass die Resourcen 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

Hypervisoren werden auch Virtual-Machine-Monitor (kurz. VMM) genannt. Diese Hypervisoren sind eine Klasse von Systemen, die als eine abstrahierende Schicht zwischen tatsächlicher vorhandener Hardware und weiteren zu installierenden Betriebssysteme dienen.
Sie dienen der Verwaltung der Resourcen 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-matel 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 Hostsystems vom Typ-1-Hypervisor durch entsprechende Treiber unterstützt wird.

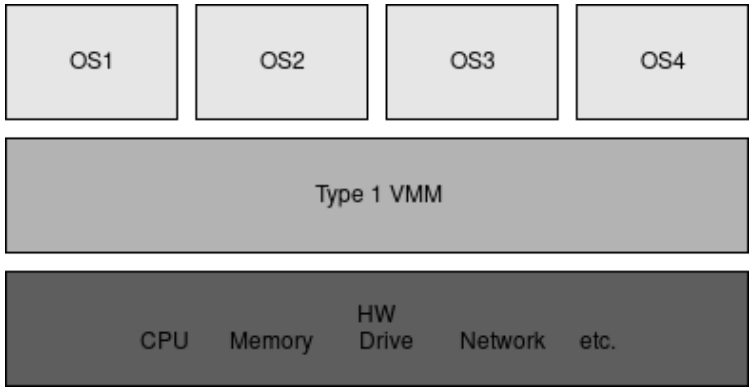

Abbildung 2.1: Aufruf des FilterVIs

Type 2

Ein Typ-2-Hypervisor setzt auf einem vollwertigen Betriebssystem, welches auf dem Hostsystem installiert ist, auf und nutzt die Treiber, des Betriebssystemes, um auf die Hardwareresourcen des Hostsystemes zuzugreifen zu können. Dementsprechend sind alle Typ-2-Hypervisoren auf einem Hostsystem lauffähig, sofern sich auf diesem das unterstützte Hostbetriebssystem installieren lässt.

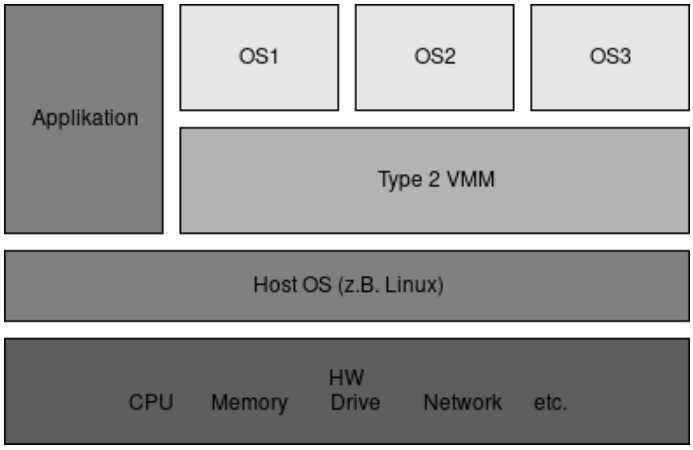

Abbildung 2.2: Aufruf des asd

2.2 KVM

KVM steht für Kernel Based Virtualmachine. bei KVM handelt es sich um eine Infrastruktur des Linux-Kernels zur Virtualisirung. 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 defacto standart etabliert. QEMU steht für Quick Emulator uns stellt für virtualisierte Gastsysteme die notwendigen Geräte wie Festplatten, Netwerk-, Sound- und Grafikkarten zur Verfügung.
KVM ist desweiteren auch eine Quellofene lösung, und ist neben XEN der Weitverbreiteste Quellofene Hypervisor. Trotz einigen aussagen ist KVM jedoch ein Type 2 Hypervisor und nicht wie XEN ein Type 1.

2.2.1 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 dazuzuinstallieren 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 üüber normale Kommandos wie kill oder top kontrolliert und gesteuert werden. Das selbe gilt auch für die Gerätelandschaft. Da mit KVM und Qemu die normalen Linux-Treiber genutzt werden können, ist eine Umgewühnung 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 instaz 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 ein 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 vortgeschrittene Funktionen wie Orchestrierung ganzer Pools von Virtuellen Maschinen mit weitergehenden funktionen wie Failover, High Availability und dergleichen gibt es weitere Lösungen. Hier springen Drittobjekte sowie Softwarehersteller in dir 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 gibt im Wesentlichen viele verschiedene Tools, von einfachen verwaltungssoftware für den Desktop PC wie zum Beispiel Gnome-Boxe

2.2.2 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 platform in die IT infrastruktur ermöglicht. KVM for IBM z System ermöglicht zudem das den einzelnen Gastsystemen mehr resourcen zugeteilt wird, als das z System tatsächlich zur verfügung stellt, dieser umstand ermöglicht die optimale auslastung der virtuellen umgebung. Zusätzlich macht es die platformmobilität 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. [(]ibm) http://www.redbooks.ibm.com/redbooks/pdfs/sg248332.pdf

Feature	**Benefit**
KVM hypervisor	Unterstzt das ausführen mehrere nicht gleicher Linux VMs auf einem einzelnen System.
CPU sharing	Erlaubt es die resourcen der CPU über mehrere VMs hinweg zu teilen.
I/O sharing	Ermöglicht das teilen von I/O resourcen 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 minimalen 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 resourcen.
Installation and configuration tools	Mitlieferung von tools um KVM for IBM z System zu installieren und Konfigurieren.
Transactional execution use	Bietet verbeserte performance für das ausführen von multi-threaded Applikationen.

Managementtools für KVM on IBM z System

Dynami Partition Manager kurz DPM. DPM ist ein geführtes management Interface im HMC (IBM Hardware Management Console), 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 Graphisches User Interface (GUI) für die folgende Management aufgaben verfügt.

- Netzwerk configuration für die OSA-based NICs
- Storage Konfiguration für ECKD und SCSI geräte
- Standart 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 performence überwachen können.
- Das DASD dump tool um einzelne DASD geräte entleeren zu können.

- Das KDUMP feature, welches Crash Dumps erstellt, fals ein Kernel crash eintritt.
- dbginfo script, welches verschiedene system-related files für Debug zwecke sammelt.
- Das sosreport skript sammelt system felerbehebungs informationen

2.3 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 Application, welche auf einem bestehenden Betriebssystem installiert wird, sondern, es beherbergt in sich schon wesentliche komponente eines Betriebssystemes, wie etwa den kernel.

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

2.3.1 Funktionsumfang

TODO

Kapitel 3

Vergleich

Um dem Vergleich einen fairen touch 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 einsatztgebiete gebraucht werden und dementsprechend kein guter vergleich zustande kommen könnte.

Es werden nur die Folgenden Punkte miteinander verglichen:

- Performance
- Integration
- Cost
- Complexity
- Maturity
- Scalability
- Functionality support

Performance

Hypervisors may be classified into two types, which can impact their performance. Type 1 hypervisors, also known as "bare metal" hypervisors, run directly on the physical hardware, and the OS of each guest runs on top of the hypervisor. These hypervisors typically allow some guests to control the hypervisor. Most businesses use Type 1 hypervisors.

A Type 2 hypervisor, also known as a hosted hypervisor, runs within an OS that runs on the physical hardware. The OS of each guest then runs on top of the hypervisor. Desktop hypervisors are usually Type 2 hypervisors.

Xen is probably the best example of a pure Type 1 hypervisor, although ESXi is clearly a Type 1 hypervisor as well because it isn't an application that's installed onto an OS. ESXi includes a kernel and other OS components that it integrates into the native OS.

The classification of KVM is more challenging because it shares characteristics of both types of hypervisor. It's distributed as a Linux component, meaning that a Linux user can start KVM from a command line or graphical user interface (GUI). These methods of starting KVM make it appear as if the hypervisor is running on the host OS, even though KVM is actually running on the bare metal.

The host OS provides KVM with a launch mechanism and establishes a co-processing relationship with it, allowing KVM to share control over physical hardware with the Linux kernel. KVM uses the processor's virtualization instructions when it runs on x86 hardware, allowing the hypervisor and all of its guests to run directly on the bare metal. The physical hardware performs most of the resource translations, so KVM meets the traditional criteria for a Type 1 hypervisor.

A Type 1 hypervisor should outperform a Type 2 hypervisor, all other factors being equal. Type 1 hypervisors avoid the overhead that a Type 2 hypervisor incurs when it requests access to physical resources

from the host OS. However, other factors also play an important role in a hypervisor's performance. For example, ESXi generally requires more time to create and start a server than KVM. ESXi also has slower performance when running servers, although this difference may be insignificant for typical loads.

Integration

Hypervisors use different methods to communicate with the host's physical hardware. KVM uses an agent installed on the host to communicate with hardware, while ESXi uses VMware's management plane to communicate with hardware. The process does provide the advantage of allowing ESXi to access other VMware products that use this management plane. However, it also requires ESXi to use VMware's control stack, which can increase hardware requirements.

Close integration with the host OS is the primary reason that Linux developers typically prefer KVM, which was incorporated into the Linux kernel shortly after its release in 2007. In comparison, Xen didn't become officially part of the Linux kernel until 2011, eight years after its initial release. Linux developers are also more likely to use KVM because Red Hat and other Linux distributors have adopted it in preference to other hypervisors. Illumos is an open-source OS based on OpenSolaris that also chose KVM over other hypervisors when it added support for hardware virtualization.

Cost

KVM clearly wins over VMware on the basis of cost. KVM is open source, so it doesn't incur any additional cost to the user. It's also distributed in a variety of ways, often as part of an open-source OS.

VMware charges a license fee to use its products, including ESXi. It's able to do this because VMware was the first company to release

enterprise-class virtualization software and is still the market leader in this segment. Its brand is therefore still relevant to an business's end users, regardless of what developers may think about it. An ESXi user must also purchase a license to use vSphere, VMware's suite of tools for cloud computing that uses ESXi. Additional software licenses may be needed, which will further increase the cost of implementing ESXi.

IBM performed some calculations regarding the total cost of ownership (TCO) for KVM and VMware in 2012. These calculations showed the KVM's TCO was typically 39 percent less than VMware, although the actual TCO will depend on site-specific factors such as the operational setting and workload. This difference in TCO indicates that cloud service providers will probably want to implement KVM on at least one cluster, regardless of the other factors to consider.

Complexity

A comparison of KVM and VMware also show a clear difference in the size of the code base, which affects a hypervisor's maintenance costs. KVM was initially released to take advantage of processor extensions that allowed them to virtualize guests without translating binary code. This origin meant that the first stable release of KVM was essentially a lightweight virtualization driver, with little more than 10,000 lines of code (LOC).

VMware is believed to have over 6 million LOC, although this fact can't be verified since its source code isn't publicly available. This total doesn't directly affect performance since VMware uses hardware extensions to virtualize guest. Nevertheless, its original code has never been completely rewritten, resulting in a more complex code base than KVM.

Maturity

KVM and ESXi are both highly mature and stable. KVM has been part of the Linux kernel for over a decade, and ESXi has been publicly available since 2006. However, KVM is more widely deployed since it's open source and is included in many packages such as Redhat Enterprise Virtualization (RHEV). KVM also supports more features than any other hypervisor.

Scalability

KVM is generally more scalable than VMware, primarily because vSphere has some limitations on the servers it can manage. Furthermore, VMware has added a large number of Storage Area Networks (SANs) to support various vendors. This feature means that VMware has more storage options than KVM, but it also complicates VMware's storage support when scaling up.

Functionality Support

Hypervisors vary greatly in their support of functionality. Network and storage support are especially important and are probably more important than any other factor besides OS integration. It shouldn't come as a surprise to learn that ESXi's support for other VMware products is unmatched by any other hypervisor. On the other hand, KVM offers more options for network support than VMware.

Kapitel 4

Fazit

KVM is typically the most popular choice for users who are concerned about the cost of operating each VM and less interested in enterprise-level features. This rule primarily applies to providers of cloud and host services, who are particularly sensitive to the cost and density of their servers. These users are highly likely to choose open-source hypervisors, especially KVM.

The tight integration with the host OS is one of the most common reasons for developers to choose KVM, especially those who use Linux. The inclusion of KVM in many Linux distributions also makes it a convenient choice for developers. KVM is also more popular among users who are unconcerned about brand names.

Kapitel 5

Anhang

https://www.cs.hs-rm.de/ linn/fachsem0910/hirt/KVM.pdf https://www.tecchannel.
virtualisierungssoftware-im-vergleich,2051909,4 https://www.tecchannel.de/a/kostenl
virtualisierungssoftware-im-vergleich,2051909,5 https://www.rippleweb.com/vmware-
vs-kvm/