University of Applied Sciences
HOCHSCHULE EMDEN·LEER
Fachbereich Technik
Abteilung Elektrotechnik und Informatik

Bachelor-Thesis

Effiziente Speicherung virtueller Festplatten mit bestehender OpenSource-Software (Arbeitstitel)

Bastian de Groot

2. Januar 2011

Prüfer	Prof. Dr. Jörg Thomaschewski
Zweitprüfer	Dr. Arvid Requate

Inhaltsverzeichnis

1 Einleitung

Von “Betriebsystemvirtualisierung” spricht man, wenn sich mehrere virtuelle Betriebsysteminstanzen Hardwareressourcen wie CPU, RAM oder Festplatten teilen. Der Virtualisierungskern (Hypervisor) stellt den virtuellen Betriebsysteminstanzen eine in Software und Hardware realisierte Umgebung zur Verfügung, die für die darin laufenden Instanzen kaum von einer echten Hardwareumgebung unterscheidbar sind. Es gibt unterschiedliche technische Ansätze der Virtualisierung, wie Paravirtualisierung oder Vollvirtualisierung. Diese Kategorisierung bezieht sich darauf, wie der Hypervisor die vorhandene Hardware für die virtuelle Instanz bereitstellt. Auf diesem Gebiet gibt es eine sehr aktive Entwicklung. [Prz] [Bau]

Wenig beachtet bei der Entwicklung von Virtualisierungssoftware ist jedoch die Speicherung von virtuellen Festplatten. In dieser Arbeit wird dieser Punkt aufgegriffen und die Möglichkeit der Optimierung mit der Copy-on-Write Strategie beleuchtet.

Copy-on-Write ist eine Optimierungsstrategie, die dazu dient unnötiges Kopieren zu vermeiden. Diese Strategie wird vom Linux-Kernel genutzt um Arbeitsspeicher einzusparen. Aber auch bei der Desktopvirtualisierung wird Copy-on-Write eingesetzt, um die benötigte Zeit für die Bereitstellung einer geklonten virtuellen Maschine minimieren. Hierbei wird nicht für jeden Benutzer ein eigenes Image kopiert, sondern alle Benutzer verwenden ein Master-Image. Falls ein Benutzer Änderungen an diesem Master-Image vornimmt, werden die Änderungen separat abgespeichert.

1.1 Zieldefinition

Ziel dieser Arbeit ist es, Möglichkeiten zur effizienten Speicherung von virtuellen Festplatten aufzuzeigen. Hierbei wird ausschließlich auf bestehende Open Source Lösungen zurückgegriffen (siehe 1.3). Die freien Open Source Lösungen werden miteinander verglichen und eine effiziente Lösung herausgearbeitet. Außerdem wird betrachtet, wie die für das Copy-on-Write benötigten Master-Images im Netzwerk effizient verteilt werden können.

1.2 Vorgehen und Kurzzusammenfassung

Zunächst werden die vorhandenen Softwarelösungen für Copy-on-Write und für die Verteilung der Master-Images erläutert. Danach werden diese anhand verschiedener anwendungsrelevanter Kriterien miteinander verglichen. Nachdem die besten Lösungen beider Kategorien gefunden wurden, werden Softwaretools erstellt, die die Nutzung der gefundenen Lösung ohne tiefgreifende Vorkenntnisse ermöglicht.

1.3 Anmerkung zur Verwendung von Open-Source

Für die Verwendung von Open-Source gibt es mehrere Gründe. Zum einen setzen führende Hersteller von Virtualisierungssoftware wie zum Beispiel Citrix aktuell auf Open-Source-Lösungen. Somit können in dieser Arbeit die neuesten Entwicklungen aufgezeigt werden. Zum anderen ist der Einsatz von proprietärer Software im finanziellen Rahmen dieser Arbeit nicht möglich. Der letzte wichtige Grund sind

Lizenzprobleme bei proprietärer Software. Sie unterbinden zum Beispiel das Veröffentlichen von Performance-Tests oder die Distribution mit selbst erstellter Software [vmw].

1.4 Anmerkung zu den verwendeten Literaturquellen

Diese Arbeit bezieht sich neben den herkömmlichen Literaturquellen auch auf Mailinglisten- und Forenbeiträge, sowie Blogeinträge.

Bei Quellenangaben im Bereich der Open Source Software gibt es einige Punkte die zu beachten sind. Es gibt keine einheitliche Dokumentation der Software. Häufig sind die Informationen nicht an einer zentralen Stelle vereint, sondern liegen verstreut im Internet in Foren, Blogs, Mailinglisten oder auch in Manpages und den Quelltexten selbst. Die Relevanz und die Richtigkeit einer solcher Quellen ist schwer zu bewerten, da Blogs, Mailinglisten und Foren keinen Beschränkungen unterliegen.

Die oben genannte Verstreuung birgt, neben der schwierigen Bewertbarkeit der Richtigkeit und Relevanz, ein weiteres Problem. Da sehr viele Autoren zum einem Thema etwas schreiben, werden unterschiedliche Begriffe synonym verwendet oder sind mehrdeutig.

Alle Quellen sind mit der zu Grunde liegenden Erfahrung des Autors dieser Arbeit ausgewählt und überprüft, können aber aus den oben genannten Gründen keine absolute Richtigkeit für sich beanspruchen.

2 Analyse Copy-on-Write

Für das Erstellen mehrerer gleichartiger Virtueller Maschinen benötigt man mehrere Virtuelle Festplatten. Das kann man auf herkömmliche Art und Weise lösen, in dem ein vorhandenes Festplattenimage n mal kopiert wird. Durch das häufige Kopieren entstehen allerdings große Mengen an Daten. Außerdem benötigt es viel Zeit Festplattenimages zu kopieren. Um diesen beiden Problemen entgegen zu wirken werden Copy-on-Write-Strategien eingesetzt.

Die Copy-on-Write-Strategie wird von Unix-artigen Betriebsystemen verwendet, um Arbeitsspeicher einzusparen. Es wird eingesetzt um nicht den ganzen Speicherbereich eines "geforkten" Prozesses kopieren zu müssen [CB05]. Die Vorteile der Optimierungsstrategie zeigen sich jedoch auch bei der Speicherung virtueller Festplatten.

Wie in Abbildung 2.1 schematisch dargestellt wird, wird bei Copy-on-Write nicht das gesamte Image kopiert. Es werden in dem Copy-on-Write-Image nur die Veränderungen zu dem so genannten Master- oder Quellimage gespeichert. Für die Platzersparnis werden Sparse-Dateien genutzt, welche im Folgenden erklärt werden. Außerdem werden die unterschiedlichen Verfahren zur Verwendung von Copy-on-Write erläutert und analysiert.

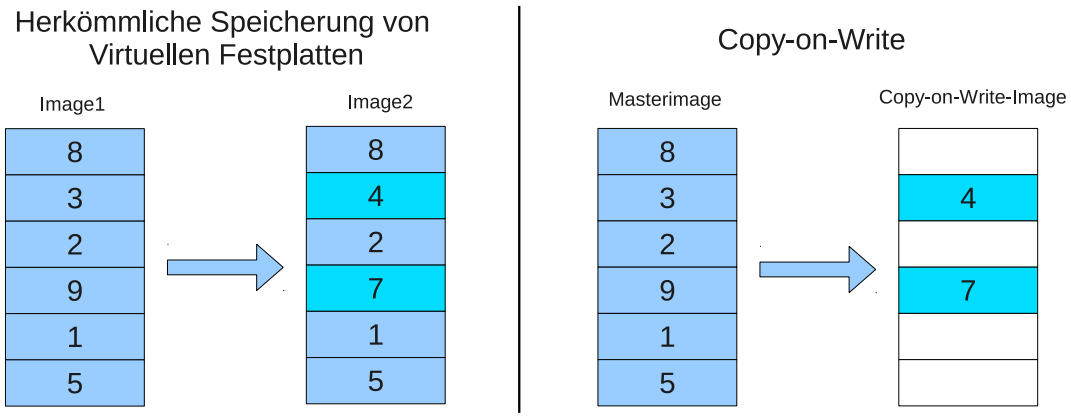

Abbildung 2.1: Copy-on-Write

2.1 Sparse-Dateien

Eine Sparse-Datei ist eine Datei, die nicht vom Anfang bis zum Ende beschrieben ist. Sie enthält also Lücken. Um Speicherplatz zu sparen, werden diese Lücken bei Sparse-Dateien nicht auf den Datenträger geschrieben. Die Abbildung 2.2 zeigt, dass der tatsächlich benutzte Speicherplatz auf der Festplatte weitaus geringer sein kann als die eigentliche Dateigröße [spa].

Eine Sparse-Datei ist kein eigenes Imageformat sondern eine Optimierungsstrategie. Sie verhilft Copy-on-Write-Images zu einer großen Platzersparnis. In Imageformaten wie qcow2 oder VHD ist diese Optimierungsstrategie ein fester Bestandteil.

2.2 qcow2

Das Imageformat qcow2 ist im Rahmen des qemu Projekts entwickelt wurde [qem]. Es ist der Nachfolger des ebenfalls aus dem qemu Projekt stammenden Formats qcow [McL].

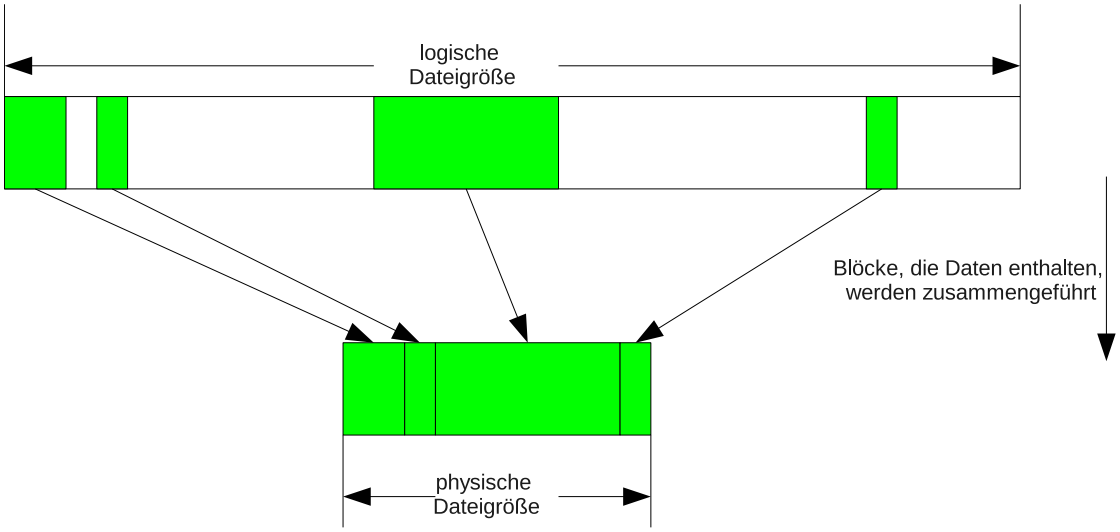

Abbildung 2.2: Sparse-Datei

Vorteile

- einfache Einrichtung
- aktive Entwicklung im Rahmen der Projekte KVM und qemu

Nachteile

- aktuell fehlende Unterstützung durch mit Xen und andere offene Virtualisierungstechniken (z.B. VirtualBox)

2.3 VHD

Das Format VHD ist von Conectix und Microsoft entwickelt worden. Die Spezifikation des Imageformats wurde von Microsoft im Zuge des “Microsoft Open Specification Promise” freigegeben [mso] [vhd]. Seit der Freigabe der Spezifikation bieten einige Open Source Virtualisierungslösungen wie qemu, Xen oder VirtualBox die Möglich-

keit dieses Format zu verwenden.

Vorteile

- einfache Einrichtung
- Unterstützung durch Softwarehersteller mit hoher Marktakzeptanz

Nachteile

- Weiterentwicklung scheint derzeit fragwürdig
- Verwendung der Copy-on-Write-Funktion von VHD mit KVM nicht möglich

2.4 dm-snapshots

Die dm-snapshots sind eine Funktion des Device Mappers. Device Mapper ist ein Treiber im Linux-Kernel. Er erstellt virtuelle Gerätedateien, die mit bestimmten erweiterten Features wie zum Beispiel Verschlüsselung ausgestattet sind [Bro]. Bei dm-snapshots wird eine solche virtuelle Gerätedatei erstellt, die aus zwei anderen Gerätedateien zusammengesetzt wird. Die erste Gerätedatei ist der Ausgangspunkt, wenn daran Änderungen vorgenommen werden, werden sie als Differenz in der zweiten Gerätedatei gespeichert [dmk].

Die von Device Mapper erstellten Gerätedateien benötigen keine Unterstützung der Virtualisierungstechnik, da sie für diese nicht von physikalischen Festplattenpartitionen unterscheidbar sind. Dieses ist nicht nur ein Vorteil, sondern zugleich auch ein Nachteil. Es muss immer vor dem Starten einer virtuellen Maschine das Copy-on-Write-Image und das Masterimage zu einer Gerätedatei verbunden werden.

Vorteile

- hohes Entwicklungsstadium
- gesicherte Weiterentwicklung
- unabhängig von Virtualisierungstechnik

Nachteile

- Aufwendige Einrichtung
- erfordert zusätzlichen Programmstart vor dem VM-Start

2.5 LVM-Snapshots

LVM-Snapshots sind ein Teil des Logical Volume Managers. LVM ist eine Software-Schicht die über den eigentlichen Hardware-Festplatten einzuordnen ist [lvma] [lvmc]. Sie basiert auf Device Mapper [lvmb]. LVM ermöglicht das Anlegen von virtuelle Partitionen (logical volumes). Diese können sich über mehrere Festplatten-Partitionen erstrecken und Funktionen wie Copy-on-Write bereitstellen.

Vorteile

- hohes Entwicklungsstadium
- gesicherte Weiterentwicklung
- unabhängig von Virtualisierungstechnik

Nachteile

- Aufwendige Einrichtung
- Live-Migration nicht möglich

- Nutzung von Sparse-Dateien schwer umsetzbar

2.6 Benchmarks

Ein wichtiger Punkt für die Entscheidung welche Copy-on-Write Implementierung optimal ist, ist die Lese- und Schreibgeschwindigkeit. Hierbei gibt es zwei Zugriffsarten, einmal den sequentiellen Zugriff und den wahlfreien oder auch zufälligen Zugriff.

2.6.1 Testbedingungen

Das Hostsystem für die Performance-Tests hat einen AMD Athlon II X2 250 Prozessor und 4 GiB RAM. Als Betriebsystem kommt sowohl auf Host- als auch Gastsystem ein 64 bit Debian squeeze zum Einsatz. Bei den KVM-Tests ist 2.6.32-5-amd64 der eingesetzte Kernel, für Xen wird der gleiche Kernel mit Xen-Unterstützung verwendet.

Während der Performance-Tests laufen neben der Virtuellen Maschine auf dem Hostsystem keine anderen aktiven Programme, die das Ergebnis verfälschen könnten. Als Referenz zu den Copy-on-Write-Techniken dient eine echte Festplattenpartition. Zum Testen der Performance werden IOzone und Bonnie++ eingesetzt.

IOzone
IOzone ist ein Tool mit dem in einer Reihe von unterschiedlichen Tests die Lese- und Schreib-Geschwindigkeit überprüft werden kann. Es wird hier zur Überprüfung der sequentiellen Lese- und Schreibgeschwindigkeit verwendet.

Bonnie++
Bonnie++ dient wie IOzone als Tool zum Testen von Festplatten. Es wird hier zur Überprüfung der sequentiellen Lese- und Schreibgeschwindigkeit sowie zum Testen

des wahlfreien Zugriffs verwendet.

2.6.2 Testergebnisse

Die Testergebnisse werden in diesem Kapitel zusammenfassend aufgeführt und analysiert. Die kompletten Testergebnisse befinden sich im Anhang.

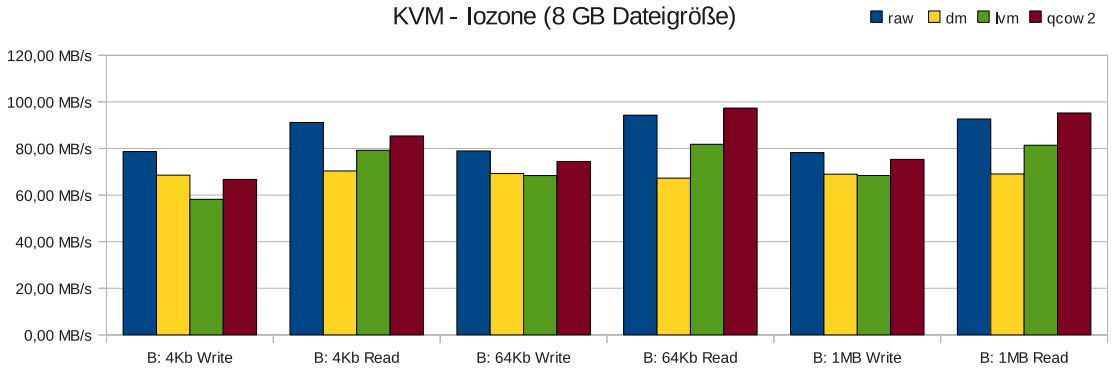

Abbildung 2.3: Iozone-kvm-8gb

Die Abbildung 2.3 zeigt, dass mit KVM qcow2 gegenüber den anderen Copy-on-Write-Techniken einen Geschwindigkeitsvorteil beim sequentiellen Lesen und Schreiben hat. LVM-Snapshots und dm-snapshots liegen hingegen ungefähr gleich auf.

Abbildung 2.4 ist zu entnehmen, dass qcow2 wie auch bei den sequentiellen Tests vor LVM-Snapshots und dm-snapshots liegt. Der Unterschied zu der echten Festplattenpartition ist in beiden Tests sehr gering. Die guten Werte von qcow2 sowohl beim sequentiellen als auch beim zufälligem Zugriff auf die Festplatte, hängen mit der guten Integration in KVM zusammen.

In Xen schneiden die dm-snapshots besser ab als LVM-Snapshots und vhd beim sequentiellen Lesen und Schreiben, wie in Abbildung 2.5 zu sehen ist.

Beim zufälligen Zugriff auf die Festplatte ist unter Xen vhd abgeschlagen hin-

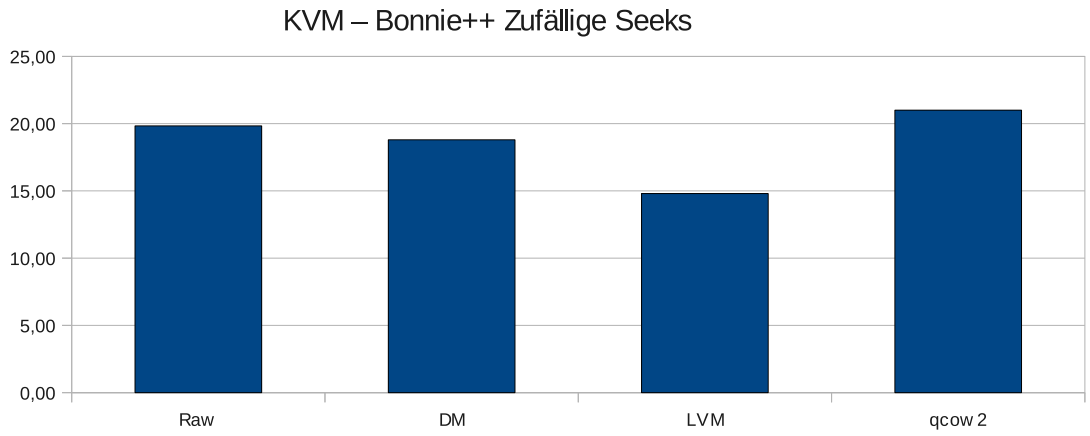

Abbildung 2.4: bonnie-kvm-random-seek

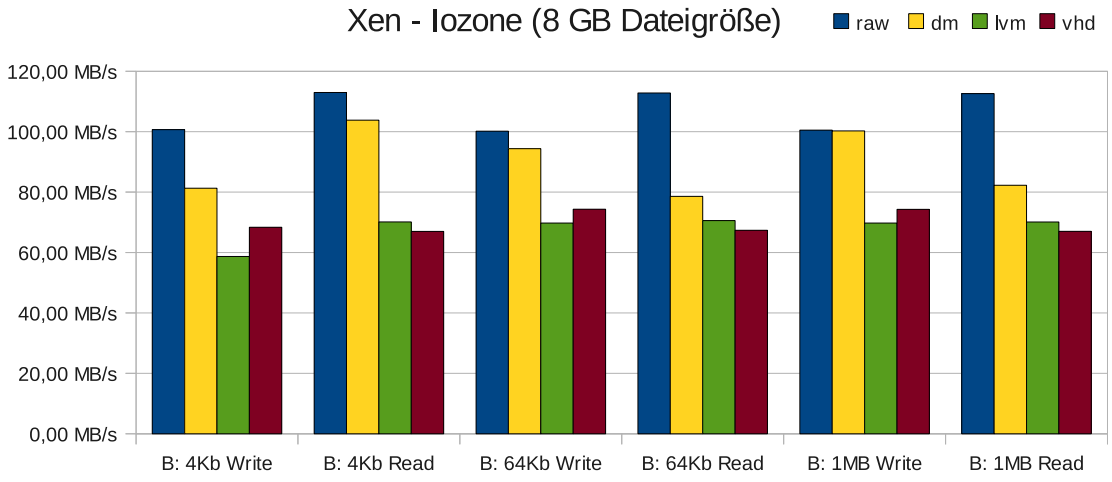

Abbildung 2.5: Iozone-xen-8gb

ter LVM-Snapshots und dm-snapshots. Diese sind ungefähr gleichauf und liegen nicht weit hinter der Festplattenpartition (Abbildung 2.6). Das mittelmäßige Abschneiden des Imageformats vhd verwundert, da Citrix, die treibende Kraft der Xen Weiterentwicklung, eine optimierte vhd-Unterstützung entwickelt hat [Cro].

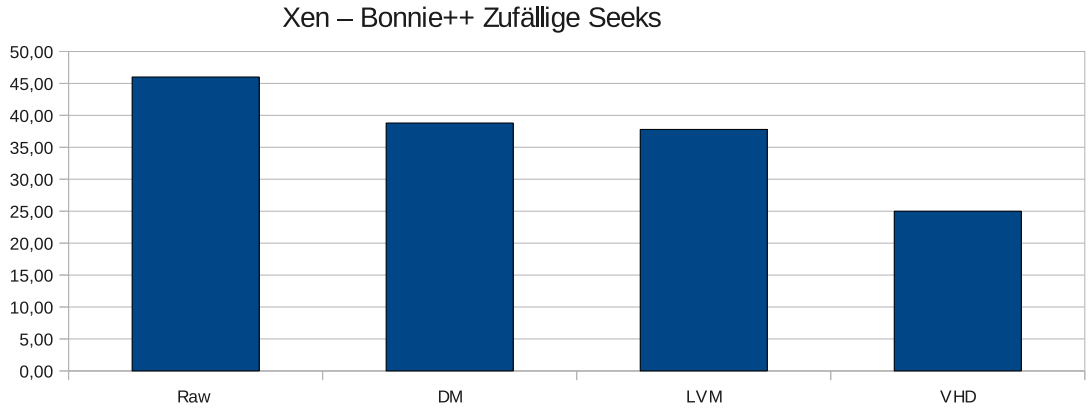

Abbildung 2.6: bonnie-xen-random-seek

Die Testergebnisse zeigen, dass es Geschwindigkeitsunterschiede zwischen den Copy-on-Write-Technniken gibt. Diese Unterschiede in der Geschwindigkeit sind aber nicht so gravierend, dass man einzelne Copy-on-Write-Lösungen aufgrund der Performance-Tests kategorisch ausschließen müsste. Dennoch sind besonders die Vorteile von qcow2 in Verbindung mit KVM zu erwähnen. Für Xen gibt es kein Image-Format, dass ähnliche Testergebnisse wie qcow2 in Verbindung mit KVM vorweisen kann.

2.7 Fazit

Es gibt bei den Testergebnissen keinen klaren Gewinner oder Verlierer. Im Großen und Ganzen fallen bei den Ergebnissen unter den einzelnen Copy-on-Write Verfahren keine bemerkenswerten Unterschiede auf. Aufgrund des unterschiedlichen Implemen-

tierungen der Copy-on-Write-Techniken in KVM und Xen, wird auch für die beiden Virtualisierungslösungen ein jeweiliges Fazit gezogen.

2.7.1 KVM

Unter KVM gibt es die Alternativen dm-snapshots LVM-Snapshots oder qcow2. Das von Microsoft entwickelte vhd kommt nicht in Frage. KVM unterstützt zwar das vhd-Format, jedoch nicht die Copy-on-Write-Funktion des Formats.

Die effizienteste Lösung für Copy-on-Write mit KVM ist qcow2. Dafür gibt es mehrere Gründe. Das qcow2-Format ist Teil des qemu-Projekts und damit sehr gut in dem darauf basierendem KVM integriert. Durch die gute Integration werden sehr gute Performance-Werte erreicht. Außerdem lässt es sich im Gegensatz zu dm-snapshots und LVM-Snapshots leichter administrieren.

2.7.2 Xen

Die für Xen zur Verfügung stehenden Copy-on-Write-Formate sind dm-snapshots, LVM-snapshots und VHD. Xen ünterstützte in einigen vergangenen Versionen qcow2, diese Unterstützung ist jedoch nicht in der aktuellen Version enthalten (Version 4.0.1) [qco].

Für Xen ist VHD aktuell die attraktivste Copy-on-Write-Lösung. Es ist zwar laut der Performance-Tests nicht die schnellste Lösung, hat aber wesentliche Vorteile gegenüber dm-snapshots und LVM-Snapshots. Es werden keine Änderungen am Xen-Quelltext benötigt, wie es bei dm-snapshots der Fall ist [rac]. Die Funktion der Live-Migration ist mit vhd leichter zu realisieren als mit LVM-Snapshots und dm-snapshots. Die im weiteren Verlauf dieser Arbeit verwendete Lösung ist VHD. Falls Xen in den nächsten Versionen wieder qcow2 unterstützt, sollte jedoch die Verwendung von qcow2 auch unter Xen geprüft werden.

3 Analyse Verteilung von Images

Der Copy-on-Write-Mechanismus benötigt immer eine Vorlage - das Masterimage. Um es auf mehreren Virtualisierungsservern nutzen zu können, muss es über das Netzwerk verteilt werden oder über ein gemeinsam genutztes Storage-Backend zur Verfügung gestellt werden. Dieses Kapitel soll Wege aufzeigen diese Verteilung oder Bereitstellung möglichst effizient vorzunehmen.

Die Verteilungslösungen werden darauf überprüft, wie störanfällig sie sind. Ein anderer Punkt für die Entscheidungsfindung ist die benötigte Dauer der Verteilung. Außerdem wird einbezogen, wie skalierbar die Lösungen sind.

3.1 Multicast

Ein Multicast ist eine Mehrpunktverbindung. Der Sender schickt die Daten gleichzeitig an mehrere Empfänger. Durch das einmalige Senden an mehrere Empfänger wird Bandbreite eingespart. Die Daten werden nur an Rechner im Netz versendet diese auch angefordert haben, wie in Abbildung 3.1 schematisch dargestellt. Die Ausnahme bilden Switches die Multicasting nicht unterstützen, sie versenden die gesendeten Daten an alle damit verbundenen Netzwerkknoten [mul].

Da es bei den Masterimages darauf ankommt, dass sie komplett und fehlerfrei dupliziert werden, kann der Sender maximal so schnell senden, wie es der langsamste Empfänger entgegen nehmen kann. Dadurch ist die Verwendung von Multicast, in

einer heterogenen Umgebung mit einem langsamen oder weit entfernten Empfänger, sehr ineffizient. Anwendung findet Multicast heute vor allem bei der Verteilung von Multimediadaten [Lei].

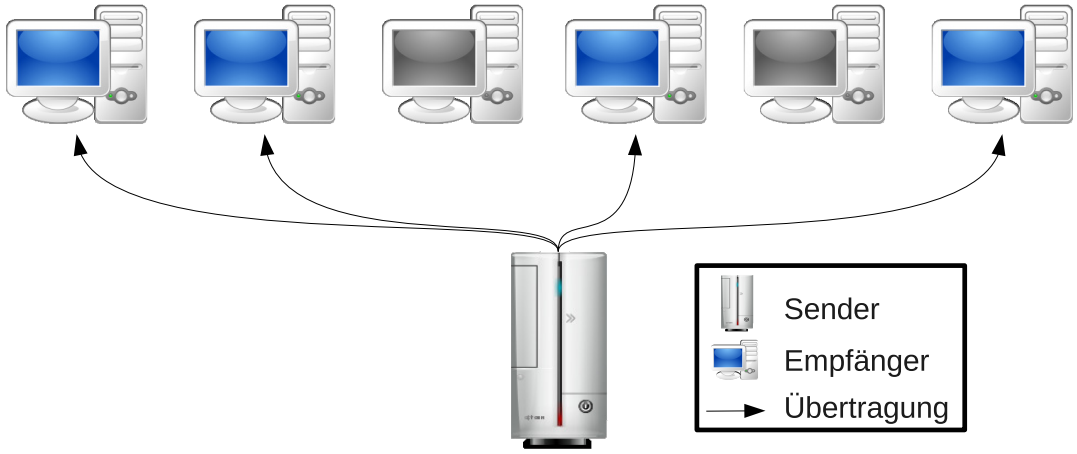

Abbildung 3.1: Multicast Beispiel

Vorteile

- sehr hohe Geschwindigkeit durch Parallelität

Nachteile

- hohe Netzwerklast
- Geschwindigkeitseinbruch bei heterogener Umgebung oder schlechten Netzanbindungen

3.2 BitTorrent

BitTorrent ist ein Netzwerkprotokoll zum effizienten Verteilen großer Datenmengen. Die Empfänger der Daten sind hierbei gleichzeitig auch Sender, sie werden Peers genannt [Coh08]. Damit wird nicht ein einziger zentraler Sender ausgelastet,

sondern die Last wird auch auf alle Empfänger verteilt (zu sehen in Abbildung 3.2). Für die Kontaktaufnahme der Peers untereinander wird ein sogenannter Tracker benötigt. Aktuellere BitTorrent-Clients können aber auch trackerlos über eine verteilte Hashtabelle (engl. “Distributed Hash Table”; DHT) andere Peers finden [Loe08].

Die zu übertragenden Daten werden nicht komplett in einem Stück übermittelt, sondern in Blöcke aufgeteilt. Bei zwischenzeitlichen Netzausfällen müssen somit auch nicht alle Daten noch einmal übertragen werden. Der BitTorrent-Client setzt nach dem Netzwerkausfall die Datenübertragung problemlos fort und muss nur gegebenenfalls die bereits übertragenen Daten einen Blockes verwerfen.

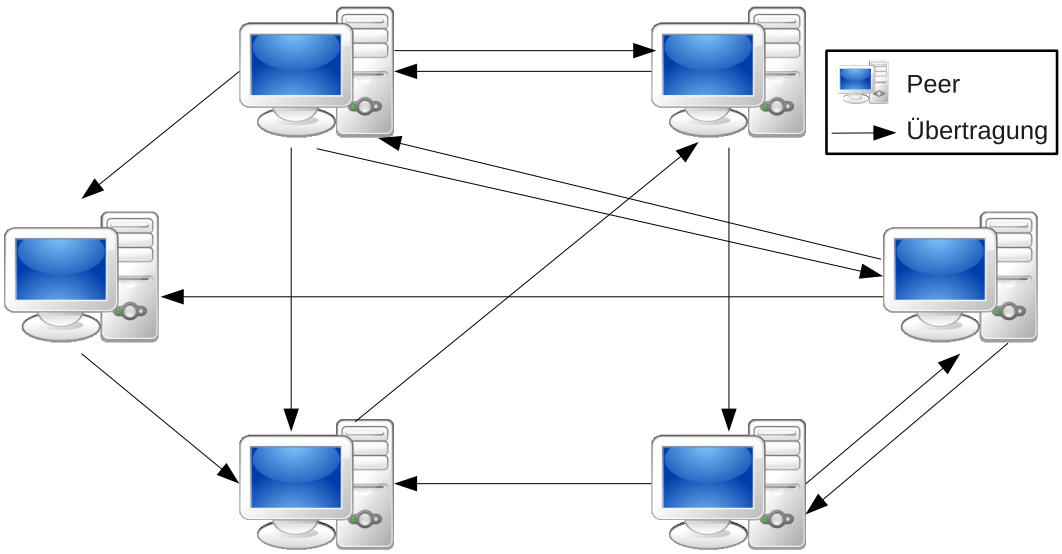

Abbildung 3.2: Bittorrent Beispiel

Vorteile

- hohe Skalierbarkeit
- Netzwerklast auf teilnehmende Netzwerksegmente beschränkt
- sehr effizient auch in heterogenen Umgebungen

Nachteile

- höherer Administrationsaufwand

3.3 NFS

NFS (Network File System) ist ein Protokoll für das Bereitstellen von Daten über das Netzwerk. Das ist ein großer Unterschied zu den beiden vorher genannten Technologien. Die Daten werden nicht von einem Rechner auf den anderen kopiert, sondern über das Netzwerk wie eine lokale Festplatte zur Verfügung gestellt [nfs03]. Der Server macht hierbei eine Freigabe die von dem Clientrechner "gemountet" wird [nfs]. Die vom Clientrechner gemountete Freigabe wird in der Verzeichnisbaum eingebunden und kann wie lokales Verzeichnis angesteuert werden.

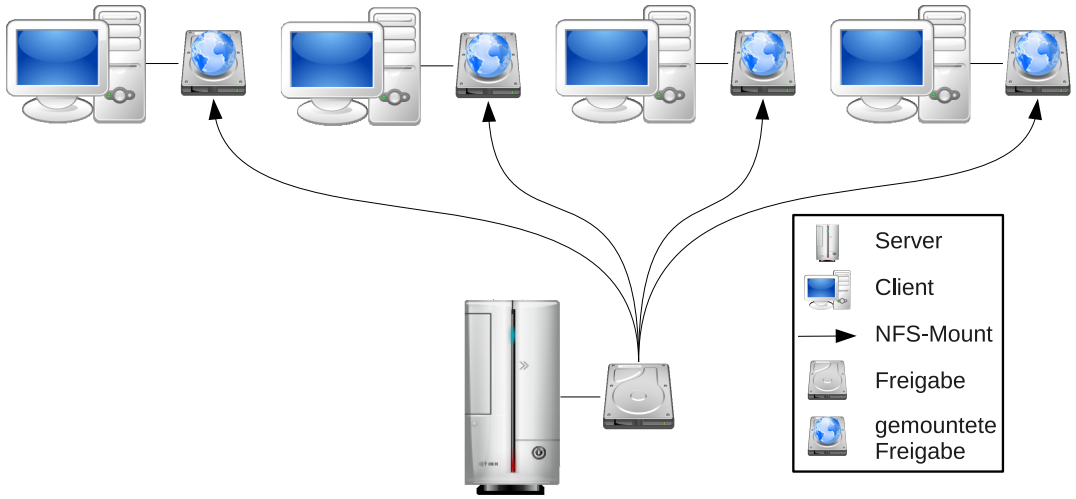

Abbildung 3.3: NFS Beispiel

Vorteile

- geringer Administrationsaufwand

Nachteile

- schlechte Skalierbarkeit
- viele von einer NFS-Freigabe gestartete virtuelle Maschinen, können zu einer permanent hohen Netzwerklast führen
- schlechte Lastenverteilung

3.4 Vergleich

Im Folgenden sollen die Verteilungsalternativen in Hinsicht auf die Kriterien Skalierbarkeit, Netzwerkausfall und Geschwindigkeit untersucht werden.

3.4.1 Skalierbarkeit

Eine gute Skalierbarkeit zeichnet sich dadurch aus, dass der Aufwand nicht signifikant ansteigt oder sich verlangsamt, wenn das Masterimage an einen weiteren Virtualisierungsserver verteilt wird. NFS zeigt dabei eine Schwäche, die Last steigt des NFS-Servers stetig mit jedem neuen NFS-Client an [Ker00].

Der Aufwand der Verteilung per Multicast steigt bei einem zusätzlichem Empfänger nicht an. Jedoch wird die Übertragung erheblich langsamer, wenn der zusätzliche Empfänger eine langsame Verbindung zu dem Server hat.

Der dezentrale Aufbau des BitTorrent-Netzes macht es sehr skalierbar. Jeder zusätzliche Empfänger des Masterimages, wird auch gleichzeitig zu einem Sender. Wenn der Upload und der Download gleich hoch sind, wird das Netz theoretisch also nicht langsamer. Das BitTorrent-Netz profitiert sogar von zusätzlichen Peers, da sie die Störanfälligkeit des Netzes verringern [EK].

3.4.2 Störanfälligkeit

Hier wird verglichen wie sich der Ausfall eines Netzwerkknotens auf die Verteilung auswirken. BitTorrent ist besonders unanfällig auf Ausfälle im Netz. Dieses wird durch die dezentrale Struktur ermöglicht. Wenn ein einzelner Netzwerkknoten ausfällt, besteht trotzdem unter den noch verfügbaren Knoten ein Netz.

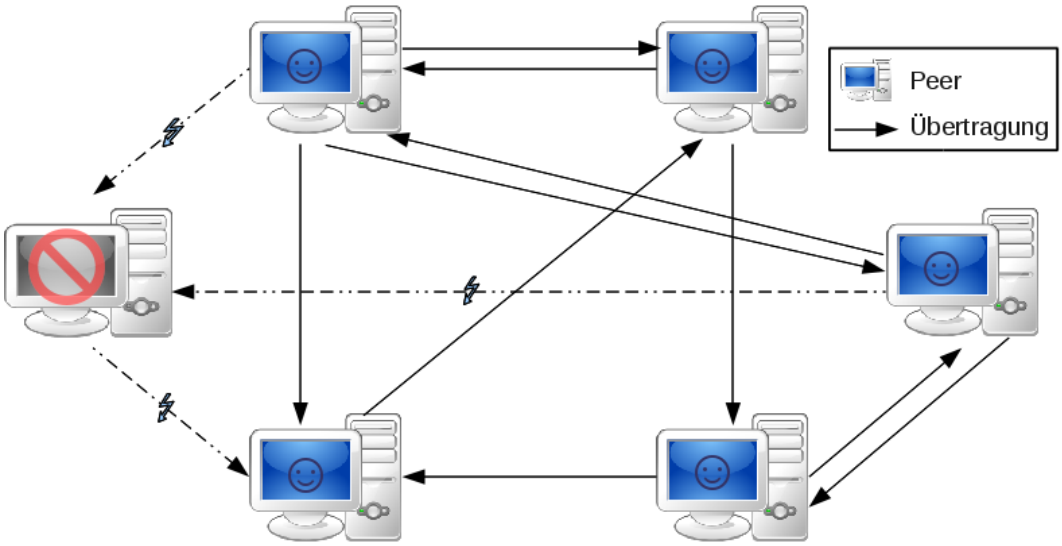

Abbildung 3.4: Bittorrent Netzwerkausfall

NFS und Multicast haben im Unterschied zu BitTorrent einen wunden Punkt, da die Verteilung über einen einzigen Knoten stattfindet. Der Ausfall eines bestimmten Knotens führt also zum kompletten Abbruch der Verteilung. Man nennt diesen Punkt *Single Point of Failure*.

Bei NFS gibt es beim Bereitstellen der Masterimages zusätzlich die Problematik, dass der Festplattenzugriff der virtuellen Maschinen von der Verfügbarkeit des NFS-Servers abhängt. Ein Ausfall führt damit zu dem Abstürzen der virtuellen Maschinen.

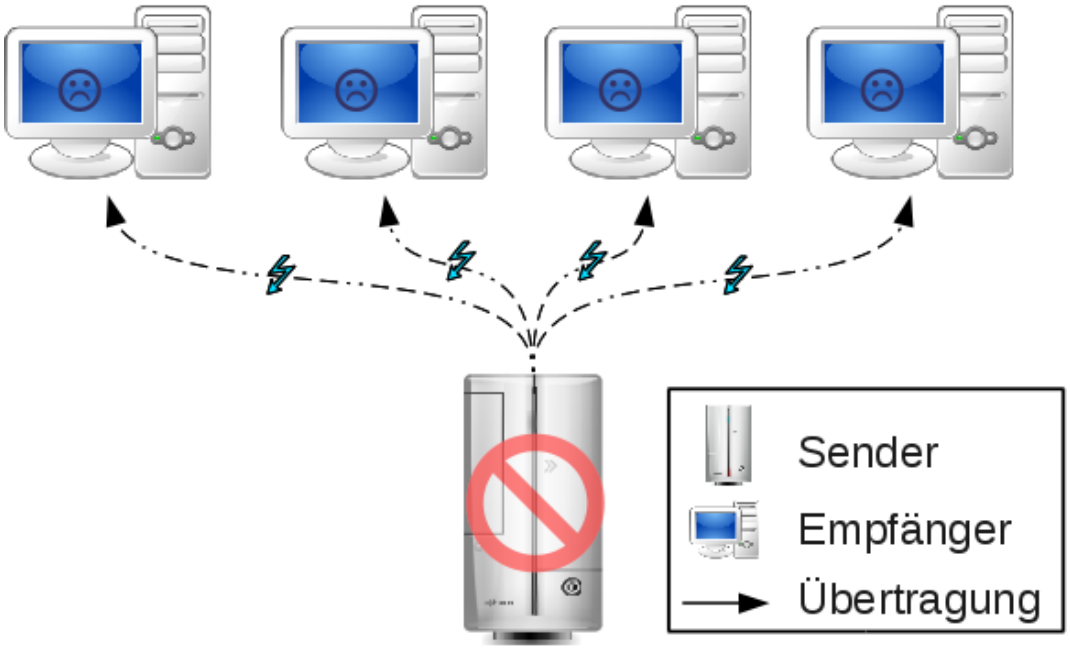

Abbildung 3.5: Multicast Netzwerkausfall

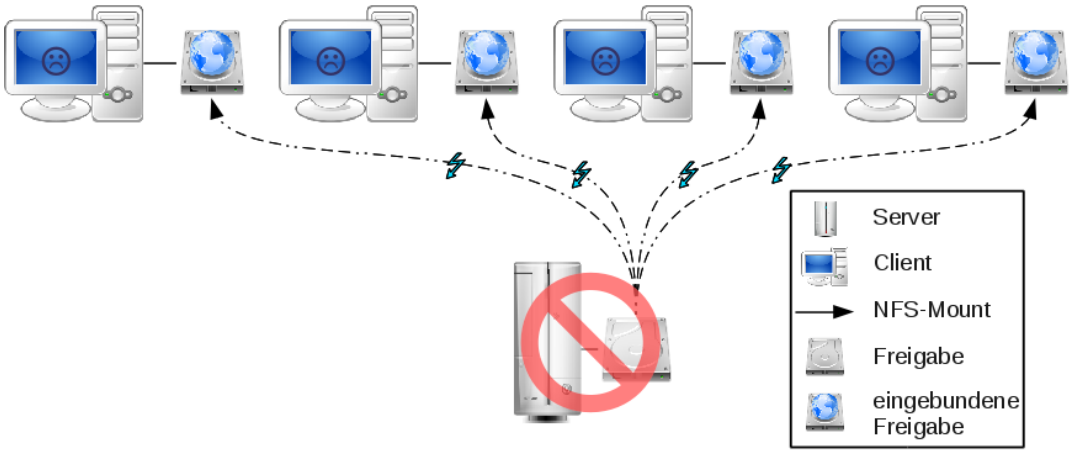

Abbildung 3.6: NFS Netzwerkausfall

3.4.3 Verteilungsdauer

Besonders hervorzuheben ist NFS, da es nicht wie BitTorrent und Multicast die Masterimages verteilt sondern bereitstellt. Dadurch benötigt es keine Zeit die Masterimages zu verteilen und kann sie direkt zur Verfügung stellen.

Die Dauer der Übertragung ist bei Multicast vom langsamsten beteiligten Netzwerkknoten abhängig. Ideal ist es, wenn alle Empfänger und der Sender über die gleiche Download- und Upload-Bandbreite verfügen (homogene Umgebung). So kann die gleichzeitige Übertragung an alle Empfänger optimal ausgenutzt werden.

BitTorrent zeichnet sich vor allem durch aus, dass es auch gute Ergebnisse erzielt, wenn die Peers über unterschiedliche Download- und Upload-Geschwindigkeiten verfügen. In einer homogenen Umgebung benötigt es mehr Zeit für die Verteilung als Multicast.

3.5 Fazit

Alle aufgezeigten Lösungen für das Verteilen von Masterimages haben ihre Vor- und Nachteile. Jedoch zeigt sich, dass BitTorrent wesentliche Vorteile gegenüber den anderen beiden Lösungen hat. Eine geringe Störanfälligkeit ist im produktiven Einsatz sehr wichtig. Auf diesem Gebiet liegt BitTorrent weit vor NFS und Multicast. Auch die Erweiterbarkeit um zusätzliche Virtualisierungsserver unterstützt die Schlussfolgerung, dass BitTorrent für den hier diskutierten Einsatz die effizienteste Lösung ist.

4 Synthese

4.1 Konzept

4.1.1 Steuerung und Kommunikation

Um in einem Netz mit mehreren Virtualisierungsservern Masterimages zu teilen und zu klonen, bedarf es einer Kommunikation zwischen den Rechnern. Diese Kommunikation sollte von einem zentralen Server gesteuert werden. Dieser Verwaltungsserver kann selbst ein Virtualisierungsserver sein oder nur mit der Verwaltung beschäftigt sein.

Die Virtualisierungstechniken sollen über eine einheitliche Schnittstelle verwaltet werden. Durch die einheitliche Schnittstelle soll die Verwaltung vereinfacht und zusätzlicher Aufwand vermieden werden. Auch die Möglichkeit bei einer Weiterentwicklung des Programms neue Virtualisierungstechniken zu integrieren soll gegeben sein.

4.1.2 Verteilung

Die Verteilung der Masterimages findet über das BitTorrent-Protokoll statt (siehe Kapitel 3). Um die Verteilung über ssh zu verwalten, wird ein BitTorrent-Client benötigt, der sich über Kommandozeile bedienen lässt. Eine weitere Vorraussetzung

ist die Unterstützung des Protokolls DHT.

Zum Starten der Verteilung der Masterimages wird zunächst eine .torrent-Datei erstellt und an alle Virtualisierungsserver gesendet, die es erhalten sollen. Danach wird der BitTorrent-Client gestartet und der Download initiiert.

Nicht jeder Virtualisierungsserver kann das Verteilen initiieren, sondern nur das Verwaltungsprogramm des Verwaltungsservers. Dies gewährleistet, dass nicht jeder Virtualisierungsserver auf jeden anderen zugreifen kann.

4.1.3 Klonen

Das Klonen wird, wie auch die Verteilung, von dem zentralen Verwaltungsserver verwaltet. Für das eigentliche Klonen der virtuellen Festplatten werden die in den Virtualisierungstechniken integrierten Programme eingesetzt.

4.2 Realisierung einer Komplettlösung

4.2.1 Rahmenbedingungen

Die Komplettlösung wird auf einem Debian squeeze System implementiert. In der Komplettlösung werden ein paar wenige debianspezifische Befehle wie zum Beispiel *apt-get* verwendet. Diese können aber leicht für andere Linux-Distributionen portiert werden. Neben den oben genannten Lösungen kommen ssh und rsync zum Einsatz.

Für die Programmierung wird die Skriptsprache Python eingesetzt. Da die hier entwickelte Verwaltungslösung nicht zeitkritisch ist, hat die Performanz keine hohe Priorität. Viel wichtiger ist es den Wartungsaufwand niedrig zu halten. Mit diesen Bedingungen ist die Skriptsprache Python eine sehr gute Wahl.

4.2.2 Programmierstil?

4.2.3 Steuerung und Kommunikation

Um die Steuerung der Virtualisierungsserver zu vereinfachen und zu vereinheitlichen wird in dieser Arbeit die Virtualisierungs-API libvirt verwendet. Die Virtualisierungstechniken Xen und KVM können beide mit libvirt verwaltet werden. Die Fähigkeiten von libvirt umfassen zum Beispiel das Erstellen, Starten, Stoppen, Pausieren sowie die Migration von virtuellen Maschinen.

Alle virtuellen Maschinen liegen libvirt als XML-Beschreibungen vor. Sie enthalten Informationen zu der virtuellen Hardware und eine eindeutige Identifikationsnummer. Eine solche XML-Beschreibung ist beispielhaft im Folgenden zu dargestellt.

```
<domain type='kvm'>
  <name>debian</name>
  <memory>512000</memory>
  <currentMemory>512000</currentMemory>
  <vcpu>1</vcpu>
  <os>
    <type>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/bin/kvm</emulator>
    <disk type='file' device='disk'>
     <driver name='qemu' type='qcow2'/>
      <source file='/var/lib/libvirt/images/debian.qcow2'
         />
```

```
21        <target dev='hda'/>
22      </disk>
23      <interface type='network'>
24        <source network='default'/>
25      </interface>
26      <input type='mouse' bus='ps2'/>
27      <graphics type='vnc' port='-1' listen='0.0.0.0'/>
28    </devices>
29  </domain>
```

Listing 4.1: libvirt-XML Beispiel

Libvirt bietet die Möglichkeit über das Netzwerk angesprochen zu werden. Außerdem unterstützt libvirt neben Xen und KVM noch andere Virtualisierungstechniken, die bei einer weiteren Entwicklung in die Komplettlösung integriert werden können.

Die einzelnen Aufgaben wie das Verteilen und das Klonen werden auf den Virtualisierungsservern von lokal installierten Skripten erledigt. So kann vermieden werden, dass unnötig viele Befehle über das Netzwerk gesendet werden müssen. Die Skripte werden über das Netzwerkprotokoll ssh gestartet.

4.2.4 Einrichtung eines Virtualisierungshosts

Die Einrichtung wird durchgeführt um alle verwalteten Virtualisierungshosts auf einen einheitlichen Stand zu bringen. Während der Einrichtung wird die nötige Software installiert und es werden Einstellungen vorgenommen. Sie ermöglichen das ???reibungslose??? Kopieren und Klonen von virtuellen Maschinen. Der Ablauf der Einrichtung wird im Folgenden dargelegt.

Ablauf

Der Benutzer gibt zunächst die Hostadresse des Virtualisierungshosts an. Ebenfalls wird die Virtualisierungstechnik des neuen Hosts abgefragt. Nach der Eingabe wird

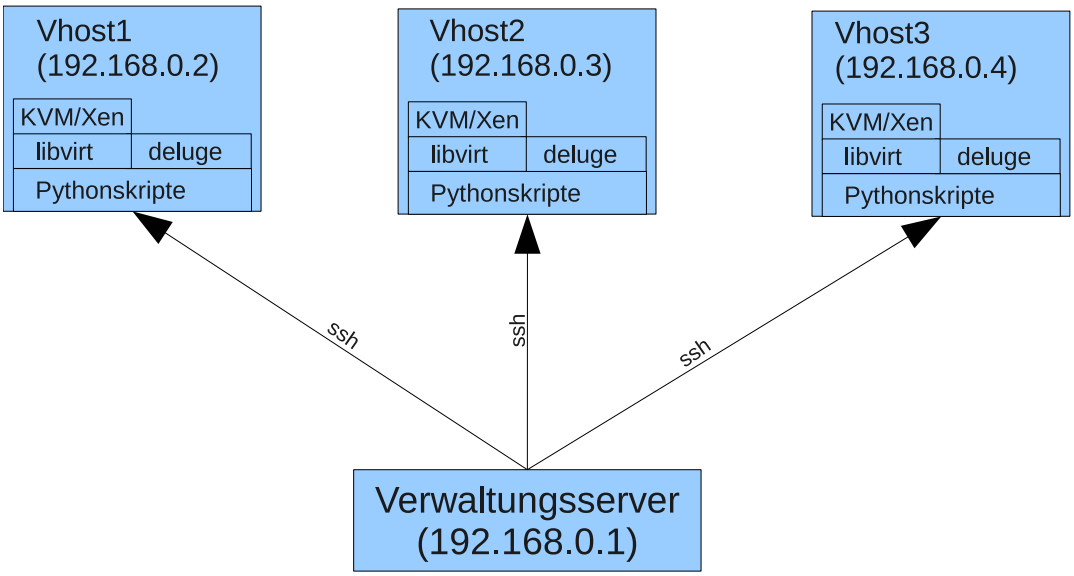

Abbildung 4.1: Kommunikation

der Rechnername abgefragt ???. Für die einfache Kommunikation wird auf Host der ssh-key des Verwaltungsservers hinzugefügt.

Um nicht alle Aktionen remote über das Netzwerk ausführen zu müssen, werden die Funktion des Klonens und der Verteilung in Skripte ausgelagert. Diese Skripte werden von dem Verwaltungsserver auf den neuen Host übertragen.

Im Anschluss folgt die Installation der benötigten Software-Pakete. Es werden die Pakete für libvirt, deluge, sowie für administrative Tools installiert.

Für die Abfrage über das Netzwerk verwendet libvirt Zertifikate. Es gibt drei unterschiedliche Zertifikate. Das Server-Zertifikat dient dazu die Echtheit des Virtualisierungshosts zu validieren. Das Client-Zertifikat wird von dem Server dazu verwendet den Client zu validieren und ihm somit Zugrif zu gewähren. Das CA-Zertifikat wird benötigt um die anderen beiden Zertifikate zu erstellen und zu zertifizieren. Das Server-Zertifikat für den Virtualisierungshost erstellt der Verwaltungsserver und legt ihn danach auf dem neuen Virtualisierungshost ab. (**Hinweis:** Dies ist nur eine vereinfachte Darstellung des Zertifikate-Systems. Eine ausführliche Beschreibung ist

unter http://wiki.libvirt.org/page/TLSSetup zu finden.)

4.2.5 Verteilung

Für den Zweck der Verteilung, kommt in dieser Arbeit *deluge* als BitTorrent-Client zum Einsatz. Er kann komplett über die Kommandozeile gesteuert werden und hat die Möglichkeit per DHT andere Peers zu finden.

Ablauf

Zunächst wählt der Benutzer einen Virtualisierungs-Host aus, der die zu verteilende virtuelle Maschine beherbergt.

```
1  def chooseVHost():
2    hList = hostList()
3    if hList:
4      print 'Wählen␣Sie␣den␣Virtualisierungshost␣aus:'
5      print 'ID\tHost\tTyp'
6      for host in hList:
7        print str(host[0]) + '\t' + host[1] + '\t' + host
             [2]
8      hostId = intInput('ID:␣')
9      return [hList[hostId][1], hList[hostId][2]]
10   else:
11     print 'kein␣Virtualisierungshost␣vorhanden'
12     return [None,None]
```

Listing 4.2: VHost-Auswahl

Die Methode für die Auswahl lässt den Benutzer zwischen allen ausgeschalteten virtuellen Maschinen auswählen.

```
1  def chooseVm(hostName,vType):
2    vOffList = vmOffList(hostName,vType)
```

```
3   if vOffList:
4     print 'Wählen Sie eine VM aus:'
5     print 'ID\tName\tState'
6     for i in range(0,len(vOffList)):
7       print str(i) + '\t' + vOffList[i].name() + '\t' +
          str(vOffList[i].info()[0])
8     vmId = intInput('ID: ')
9     return vOffList[vmId]
```

Listing 4.3: VM-Auswahl

Außerdem gibt der Benutzer an welche Virtualisierungsserver die virtuelle Maschine verteilt werden soll. Nach der Auswahl der Server und der VM, erstellt das Skript `maketorrent.py` (siehe 4.2.4) eine torrent-Datei aus der XML-Beschreibung von libvirt und den virtuellen Festplatten. Sie wird an alle ausgewählten V-Hosts mit rsync weitergegeben. Nun werden alle BitTorrent-Clients gestartet und die erstellte torrent-Datei hinzugefügt.

4.2.6 Klonen

Für das Klonen der virtuellen Maschinen werden die von den Virtualisierungstechniken mitgebrachten Tools verwendet. Auf einem Xen-Server ist es das Tool *vhd-util*, bei KVM *kvm-img*.

Ablauf

Beim Klonen einer virtuellen Maschine wählt der Benutzer, wie bei der Verteilung, einen Virtualisierungs-Host und eine virtuelle Maschine aus (siehe 4.3). Zusätzlich dazu wird die Anzahl der Klone und die Option alle Klone sofort zu starten abgefragt. Nach den erfolgten Benutzereingaben ruft das Programm das auf dem Virtualisierungsserver befindliche Skript `clone.py` zum Klonen auf.

Im ersten Schritt des Klonvorgangs generiert das Skript einen neuen Namen. Der

Name setzt sich aus dem alten Namen und 6 zufälligen und Buchstaben zusammen. Der nächste Schritt ist es die Beschreibung der Vorlage aus libvirt zu laden. Aus ihr werden die Festplatten der Vorlage ausgelesen und geklont. Die Identifikationsnummer und die MAC-Adresse aus der Beschreibung werden gelöscht und der neue Name eingetragen. Die MAC-Adresse und die Identifikationsnummer generiert libvirt neu beim Anlegen der geklonten VM.

```
1  def prepareXml(xmlDescription):
2    hList = []
3    description = parseString(xmlDescription)
4    hardDisks = description.getElementsByTagName("disk")
5    for i in range(0,len(hardDisks)):
6      if hardDisks[i].getAttribute("device") == "disk":
7        hardDisk = hardDisks[i].getElementsByTagName("
           source")[0].getAttribute("file")
8        newHddPath = config.imageDir + "/" + newVmName + "-
           " + os.path.basename(hardDisk)
9        cloneHdd(hardDisk, newHddPath)
10       hardDisks[i].getElementsByTagName("source")[0].
           setAttribute("file", newHddPath)
11   description.getElementsByTagName("name")[0].childNodes
       [0].data = newVmName
12
13   for removeTag in description.getElementsByTagName("mac"
       ):
14     removeTag.parentNode.removeChild(removeTag)
15
16   for removeTag in description.getElementsByTagName("uuid
       "):
17     removeTag.parentNode.removeChild(removeTag)
18   return description
```

Listing 4.4: XML-Bearbeitung

4.2.7 Fehlerbehandlung

Benutzereingabe Hostausfall Debugging

5 Zusammenfassung und Ausblick

5.1 Fazit

Der wichtigste Punkt der in dieser Arbeit deutlich werden soll, ist das man bei der Softwareentwicklung nicht immer das Rad neu erfinden muss. Dieses gilt vor allem im Bereich von Open Source. Hier gibt es bereits sehr viele freie Biblitheken und Programme die man problemlos in eine Eigententwicklung verwenden und einbinden kann.

6 Anhang

Literaturverzeichnis

[Bau] Baun, Christian: *Vorlesung Systemsoftware*. http://jonathan.sv.hs-mannheim.de/~c.baun/SYS0708/Skript/folien_sys_vorlesung_13_WS0708.pdf, Abruf: 31.10.2010

[Bro] Brož, Milan: *Device mapper*. http://mbroz.fedorapeople.org/talks/DeviceMapperBasics/dm.pdf, Abruf: 17.10.2010

[CB05] Cesati, Marco ; Bovet, Daniel P.: *Understanding the Linux Kernel*. dritte Ausgabe. Linux-Server-Praxis, 2005

[Coh08] Cohen, Bram: *The BitTorrent Protocol Specification*. http://www.bittorrent.org/beps/bep_0003.html. Version: 2008, Abruf: 01.01.2011

[Cro] Crosby, Simon: *We've Open Sourced Our Optimized VHD Support*. http://community.citrix.com/x/OYKiAw, Abruf: 11.11.2010

[dmk] *Device-mapper snapshot support*. http://www.kernel.org/doc/Documentation/device-mapper/snapshot.txt, Abruf: 17.10.2010

[EK] Eger, Kolja ; Killat, Ulrich: *Scalability of the BitTorrent P2P Application*. http://www3.informatik.uni-wuerzburg.de/ITG/2005/presentations/kolja.eger.pdf, Abruf: 01.01.2011

[Ker00] Kerr, Shane: *Use of NFS Considered Harmful*. http://www.

time-travellers.org/shane/papers/NFS_considered_harmful.html. Version: 2000, Abruf: 01.01.2011

[Lei] Leitner, Felix von: *Wir erfinden IP Multicasting*. http://www.fefe.de/multicast/multicast.pdf, Abruf: 01.01.2011

[Loe08] Loewenstern, Andrew: *DHT Protocol*. http://www.bittorrent.org/beps/bep_0005.html. Version: 2008, Abruf: 01.01.2011

[lvma] *Linux LVM-HOWTO*. http://www.selflinux.org/selflinux/html/lvm01.html, Abruf: 18.10.2010

[lvmb] *LVM2 Resource Page*. http://sourceware.org/lvm2/, Abruf: 18.10.2010

[lvmc] *What is Logical Volume Management?* http://tldp.org/HOWTO/LVM-HOWTO/whatisvolman.html, Abruf: 18.10.2010

[McL] McLoughlin, Mark: *The QCOW2 Image Format*. http://people.gnome.org/~markmc/qcow-image-format.html, Abruf: 18.10.2010

[mso] *Microsoft Open Specification Promise*. https://www.microsoft.com/interop/osp/default.mspx, Abruf: 18.10.2010

[mul] *Multicast FAQ File*. http://www.multicasttech.com/faq/, Abruf: 01.01.2011

[nfs] *4. Setting up an NFS Client*. http://nfs.sourceforge.net/nfs-howto/ar01s04.html#mounting_remote_dirs, Abruf: 01.01.2011

[nfs03] The Internet Society: *Network File System (NFS) version 4 Protocol*. http://tools.ietf.org/html/rfc3530. Version: 2003, Abruf: 01.01.2011

[Prz] Przywara, André: *Virtualization Primer*. http://www.andrep.de/virtual/, Abruf: 01.11.2010

[qco] *Qcow2 Support*. http://lists.xensource.com/archives/html/xen-devel/2010-11/msg00256.html, Abruf: 14.11.2010

[qem] *QEMU Emulator User Documentation*. http://wiki.qemu.org/download/qemu-doc.html#disk_005fimages, Abruf: 18.10.2010

[rac] *Race condition in /etc/xen/scripts/block*. http://lists.xensource.com/archives/html/xen-devel/2010-07/msg00827.html, Abruf: 14.11.2010

[spa] *Sparse files*. http://www.lrdev.com/lr/unix/sparsefile.html, Abruf: 18.10.2010

[vhd] *Virtual Hard Disk Image Format Specification*. http://technet.microsoft.com/en-us/virtualserver/bb676673.aspx, Abruf: 18.10.2010

[vmw] VMware: *VMware Benchmarking Approval Process*. http://www.vmware.com/pdf/benchmarking_approval_process.pdf, Abruf: 05.11.2010

Abbildungsverzeichnis

Listings