

Evaluation und Implementierung einer Accounting-Lösung für die Private Cloud mit Eucalyptus

BACHELORARBEIT

für die Prüfung zum

Bachelor of Science

des Studienganges Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Karsten Knese

23.08.2010

Bearbeitungszeitraum	12 Wochen
Matrikelnummer	21187703
Kurs	TAI07B2
Ausbildungsfirma	Karlsruher Institut für Technologie
Betreuer der Ausbildungsfirma	M. Sc. Christian Baun Alexander Helget
Gutachter der Studienakademie	Dr. Markus Reischl

Erklärung

gemäß § 5 (2) der „Studien- und Prüfungsordnung DHBW Technik“ vom 18. Mai 2009.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Eine unzulässige Verwendung von Quellen (Plagiarismus) führt nach §9 (3) zur Note "nicht ausreichend"!

Danksagung

Ein herzlicher Dank geht an Christian Baun, den fachlichen Betreuer für die Unterstützung mit ausführlichem Informationsmaterial und zielstrebigem Leitung dieser Bachelorarbeit. Vielen Dank an Dr. Markus Reischl, der mir als Betreuer seitens der Dualen Hochschule Karlsruhe mit Ideen und Hinweisen zur Ausarbeitung zur Seite stand. Einen weiteren Dank an Alexander Helget, der als Ausbilder bei organisatorischen Problemen zur Hilfe kam. Allen drei sei gedankt für die ausführliche und konstruktive Korrektur der Ausarbeitung.

Für den erfolgreichen Verlauf meines Studiums danke ich meinen Eltern Heinrich und Erika Knese, die mich drei Jahre lang stolz und motivierend begleiteten. Vielen Dank auch meiner Freundin Katharina Bernhardt für das unermüdliche Korrekturlesen.

Karsten Knese

Karlsruhe, den 11. August 2010

Inhaltsverzeichnis

I	Einleitung	1
1	Zusammenfassung	2
2	Motivation	3
3	Zieldefinition	5
II	Grundlagen	6
4	Virtualisierungsgrundlagen	7
4.1	Emulatoren	7
4.2	Vollständige Virtualisierung	9
4.3	Para-Virtualization	13
5	Virtualisierungssoftware	15
5.1	KVM	15
5.2	QEMU	18
5.3	Libvirt	20
6	Cloud Computing - Theorie	22
6.1	Definition	22
6.2	Cloud Architekturen	24
6.3	Aufbau einer Cloud	26
6.4	Private Cloud im SCC	28
6.5	Wirtschaftlichkeitsbetrachtung	29
7	Eucalyptus	34
7.1	Allgemeiner Aufbau	34
7.2	Komponenten	36

7.3	Steuerung	38
III	Evaluation bestehender Systeme	42
8	Resource Monitoring	43
8.1	Libvirt	43
8.2	Euca2ools	49
8.3	Nagios	50
8.4	Nagios-Virt	51
8.5	Nagios-Eucalyptus	51
8.6	Fazit	52
9	Accounting	54
9.1	Amazon Web Services	54
9.2	RackSpace	63
9.3	GoGrid	63
9.4	Zimory	65
9.5	Fazit	66
10	Anforderungen für das KIT	68
10.1	Resource Monitoring	68
10.2	Accounting	71
IV	Implementierung	77
11	NC-Daemon	81
11.1	Programmbeschreibung	81
11.2	Der Einleitungsprozess	81
11.3	Erfassen der Verbrauchsdaten	83
11.4	Besitzer der Instanz zuweisen	84
11.5	Eintragen der Verbrauchsdaten in die Datenbank	86
11.6	Benachrichtigung bei terminierter Instanz	86
12	Zentrale Datenbank	90
12.1	Datenaggregation	90
12.2	Datenbanksystem	91

12.3 Datenbankmodell	92
13 Weboberfläche	95
14 Konfigurationsoberfläche	98
V Abschließendes	100
15 Fazit	101
16 Ausblick	103
VI Verzeichnisübersicht	i
A Abbildungsverzeichnis	ii
B Tabellenverzeichnis	v
C Quellcodeverzeichnis	vi
D Abkürzungsverzeichnis	vii
E Literaturverzeichnis	x

Teil I

Einleitung

ZUSAMMENFASSUNG

Deutsch

Diese Arbeit implementiert eine Accounting-Lösung für die Nutzung einer Private Cloud, angeboten vom Steinbuch Centre for Computing, einem Institut innerhalb des Karlsruher Institut für Technologie (KIT). Die Schwerpunkte dieser Arbeit liegen neben der Einführung in das Cloud Computing in der Evaluation bestehender Accounting-Lösungen. Auf der Basis dieser Evaluation wird ein Fazit für die Umgebung der Private Cloud gezogen. Dieses Fazit ist die Grundlage für das Konzept einer konkreten Implementierung der Accounting-Lösung. Für die Implementierung ist die Erfassung der Verbrauchsdaten und das Zuordnen dieser Daten auf die dazugehörigen Benutzer notwendig.

Englisch

This Bachelor-Thesis implements an accounting solution for the usage of a Private Cloud which has been established by the Steinbuch Centre for Computing, an institute of the Karlsruhe Institute of Technology. The main focus is the introduction to the basics of Cloud Computing especially to the benchmarking of already existing accounting solutions. Based on this, a conclusion for the Private Cloud environment is built. This is used for the conceptual implementation work of an accounting solution. Therefore, the resource monitoring and the match of those data to the users, that belong to each instance, has to be implemented.

MOTIVATION

Bezogen auf die aktuelle Gartnerstudie [17] ist das Auslagern von Rechenkapazitäten und Ressourcen in der Cloud das Top Thema in der IT. Cloud Computing bezeichnet eine Dienstleistung zur Nutzung von Rechenkapazitäten und Ressourcen über festgelegte Schnittstellen. Als Basis des Cloud Computings dient die Virtualisierung. Virtualisierung abstrahiert Betriebssysteme von der physikalischen Hardware. Im Gegensatz zur bisher bekannten Virtualisierung, der Bereitstellung virtueller Maschinen auf dem vorhandenen Rechner, erweitert sich das Angebot des Cloud Computings auf Kapazitäten eines kompletten Rechenzentrums. Die Bereitstellung dieser Kapazitäten erfolgt durch externe Dienstleister in Sekunden- beziehungsweise Minutenschnelle.

Die Virtualisierung beschränkt sich nicht auf einzelne Computer, sondern ermöglicht virtuelle Rechnernetze. Aus einem massiv vorhandenen und ortsunabhängigen Pool an Ressourcen werden je nach Bedarf Rechenkapazitäten gebündelt bereitgestellt. Die technische Infrastruktur ist für den Benutzer nicht sichtbar. Für ihn ist ausschließlich der Zugang und die Verfügbarkeit des genutzten Dienstes von Interesse. Dies ist ein Mehrgehalt im Vergleich zur einfachen Virtualisierung.

Bereits vorhandene Rechenkapazitäten können on-demand, das bedeutet dann, wenn sie gebraucht werden, in der Cloud gemietet werden. Abgerechnet wird ausschließlich die Nutzungsdauer der Kapazitäten. Nach Gebrauch werden die benötigten Ressourcen in der Cloud freigegeben. Es entsteht somit ein Hybrid aus bereits vorhandener Infrastruktur und Cloud Service.

Dass diese Bereitstellung besonders im Hinblick auf die genutzten Kapazitäten und die damit verbundenen Instandhaltungskosten nicht kostenlos

erbracht werden können, versteht sich von selbst. Es muss eine Möglichkeit geben, die Verbrauchsdaten der genutzten Kapazitäten zu erfassen und dem Kunden in angemessener Weise in Rechnung zu stellen.

Eucalyptus ist eine freie Software zur Erstellung einer Hybrid Cloud. Diese Arbeit ermöglicht ein Abrechnungssystem der genutzten Ressourcen in Eucalyptus. Diese Funktionalität ist bislang nicht vorhanden. Durch die Integration dieser Lösung kann Eucalyptus für ein kommerzielles Angebot von Cloud-Services eingesetzt werden, da es automatisiert dem Benutzer die jeweiligen Verbrauchsdaten in Rechnung stellt.

Eine Hürde wird das grundlegende Messen der Verbrauchsdaten, da es sich nicht um dediziert zugewiesene Hardware handelt, sondern um virtuelle Maschinen mit unterschiedlich großen Kapazitäten. Im zweiten Schritt muss eine Abrechnung dieser Messdaten geschehen. Erschwert wird dies durch den Mangel an fertigen Lösungen und der nicht vorhandenen Dokumentation.

Diese Arbeit ist in drei inhaltliche Bereiche geteilt. Zu Beginn (Teil II) wird eine Einführung in Begriffe rund um das Cloud Computing geliefert. Da es noch keine vollständigen Lösungen, sowohl was das Erfassen der Verbrauchsdaten von virtuellen Maschinen als auch das Abrechnen dieser Daten betrifft, für Eucalyptus gibt, wird in einem weiteren Teil (Teil III) eine Evaluation von möglichen Ansätzen vorgenommen. Auf Basis des Fazits dieser Evaluation wird abschließend (Teil IV) die im Rahmen dieser Arbeit realisierte Implementierung für solch ein System vorgestellt und diskutiert.

ZIELDEFINITION

Zur Realisierung einer Accounting-Lösung für die Private Cloud im SCC werden im Rahmen dieser Arbeit folgende Ziele definiert:

Die Themen Virtualisierung und Cloud Computing werden eingeführt. Es erfolgt eine Vorstellung mehrerer Virtualisierungsansätze sowie der Definition von Cloud Computing und dessen Eigenschaften. Eucalyptus wird als eingesetzte Softwarelösung zur Erstellung einer Private Cloud im SCC behandelt. Die geführte Wirtschaftlichkeitsbetrachtung zeigt einige Vorteile des Cloud Computings.

Die Realisierung bedarf einer Möglichkeit zum Erfassen der Verbrauchsdaten der bereitgestellten Ressourcen. Diese Messdaten müssen dem Benutzer in Rechnung gestellt werden.

Anhand bestehender Werkzeuge wird eine Möglichkeit zur Datenerfassung evaluiert auf die Umsetzbarkeit in der Eucalyptus-Umgebung geprüft.

Das Abrechnungssystem im SCC erfolgt anhand einer Analyse kommerzieller Cloud-Anbieter. Auf Basis dieser vorgestellten Preismodelle wird ein für das SCC passendes Modell erarbeitet. Das Preismodell zielt auf Flexibilität und einer genauen Abrechnung der Verbrauchsdaten anstelle eines Pauschalbetrags zur größtmöglichen Gewinnwirtschaft.

Abschließend wird ein Prototyp dieser Accounting-Lösung implementiert. Dazu werden zyklisch die Verbrauchsdaten erfasst, gespeichert und analysiert. Ist eine Instanz beendet, wird der Benutzer über die angefallenen Verbrauchsdaten und deren Kosten per E-Mail informiert. Die aktuellen Daten können jederzeit über eine implementierte Weboberfläche eingesehen werden.

Teil II

Grundlagen

Das Cloud Computing basiert auf der Virtualisierung von IT-Ressourcen. Die ersten beiden Kapitel dieses Teils diskutieren den Begriff der Virtualisierung als Grundlage der Cloud-Technologie. Ein Schwerpunkt ist die Virtualisierungssoftware Kernel based Virtual Machine (KVM). KVM dient bei der Hardwareumgebung dieser Arbeit als Virtualisierungssoftware für die Cloud-Infrastruktur mit Eucalyptus.

Die darauf folgenden Kapitel behandeln die Thematik des Cloud Computings. Dort werden sowohl die technischen als auch die wirtschaftlichen Aspekte betrachtet, da sich die Funktionalität dieser Arbeit auf beide Bereiche bezieht.

Eucalyptus ist die verwendete Open Source Implementierung zur Erstellung einer Cloud-Infrastruktur. Eucalyptus ermöglicht den Aufbau einer skalierbaren Hybrid Cloud und deren Rechenleistung und Speicherkapazität in Sekundenschnelle zu beziehen. Eine Accounting- beziehungsweise Billing-systematik für Eucalyptus ist bislang noch nicht vorhanden. Die Entwicklung solch eines Systems erfordert eine Möglichkeit zum Erfassen von Verbrauchsdaten der von Eucalyptus erstellten Instanzen.

VIRTUALISIERUNGSGRUNDLAGEN

Virtualisierung beschreibt per Definition die Entkopplung der statischen Verbindung zwischen Software(-Logik) und der physischen Hardware. Diese Verbindung wird auf eine Abstraktionsebene geführt, in der Anwendungen und Dienste gekapselt und getrennt voneinander kontextfrei laufen können; unabhängig von der zu Grunde liegenden Hardware [27].

Mit Hilfe dieser diversen Virtualisierungsmöglichkeiten kann das Cloud Computing sein flexibles Angebot ermöglichen. Drei populäre Ansätze der Virtualisierung sind:

- Emulatoren
- Vollständige Virtualisierung
- Para-Virtualisierung

4.1 Emulatoren

Wörtlich abgeleitet von dem lateinischen Verb *aemulari* bedeutet emulieren zu Deutsch: *nachahmen*. Ein Emulator ahmt die Hardware eines Rechners vollständig nach und abstrahiert diese in Software. Eine virtuelle Maschine innerhalb eines Emulators ist komplett losgelöst von der eigentlichen Hardware [2, 15, 27]. Dies bedeutet, dass keine Virtualisierung von Hardware stattfindet, da die angebotenen Hardwarekomponenten physikalisch nicht vorhanden sein müssen [1].

Ein Emulator greift sämtliche Befehle des Gastbetriebssystems, welche auf die Hardware zugreifen, ab und übersetzt diese in für die physikalisch vorhandene Hardware kompatible Befehle. Das Gastbetriebssystem kann zu keiner Zeit feststellen, dass es in einer virtuellen Umgebung läuft. Der Emulator muss mit seiner bereitgestellten Software oder Application Programming Interface (API) sämtliche Systembefehle (System Call) entgegennehmen, diese innerhalb seiner Applikationslogik mit der Hardware des Host-Systems bearbeiten und das Ergebnis anschließend an die Virtuelle Maschine (VM) liefern. Abbildung 4.1 zeigt die Funktionalität des Emulators. Dort wird das unterschiedliche Angebot an Systemarchitekturen dargestellt. Diese Architekturen übersetzt der Emulator mittels einer Software in die Architektur des Host-Systems.

Ein Vorteil von Emulatoren ist die Unabhängigkeit der Host-Plattform. Die emulierte Hardware muss nicht der Hardware des Host-System entsprechen. Daraus ergibt sich eine Flexibilität was beispielsweise das Testen von Applikationen auf unterschiedlichen Plattformen anbetrifft.

Ein Nachteil ist der Performance der Emulatoren. Durch die Emulation sämtlicher System Calls ist die Geschwindigkeit im Vergleich zu physisch vorhandenen Maschinen deutlich langsamer. Die von dem Gast-System abgesetzten System Calls greifen nicht direkt auf die Hardware zu, sondern durchlaufen zuvor den Bearbeitungsschritt des Emulators. Dieser Zwischenschritt erzeugt spürbare Geschwindigkeitseinbußen [15, 27].

Bekannte Emulatoren sind Hercules, damals noch für die IBM-Großrechner entwickelt, VirtualBox und VirtualPC.

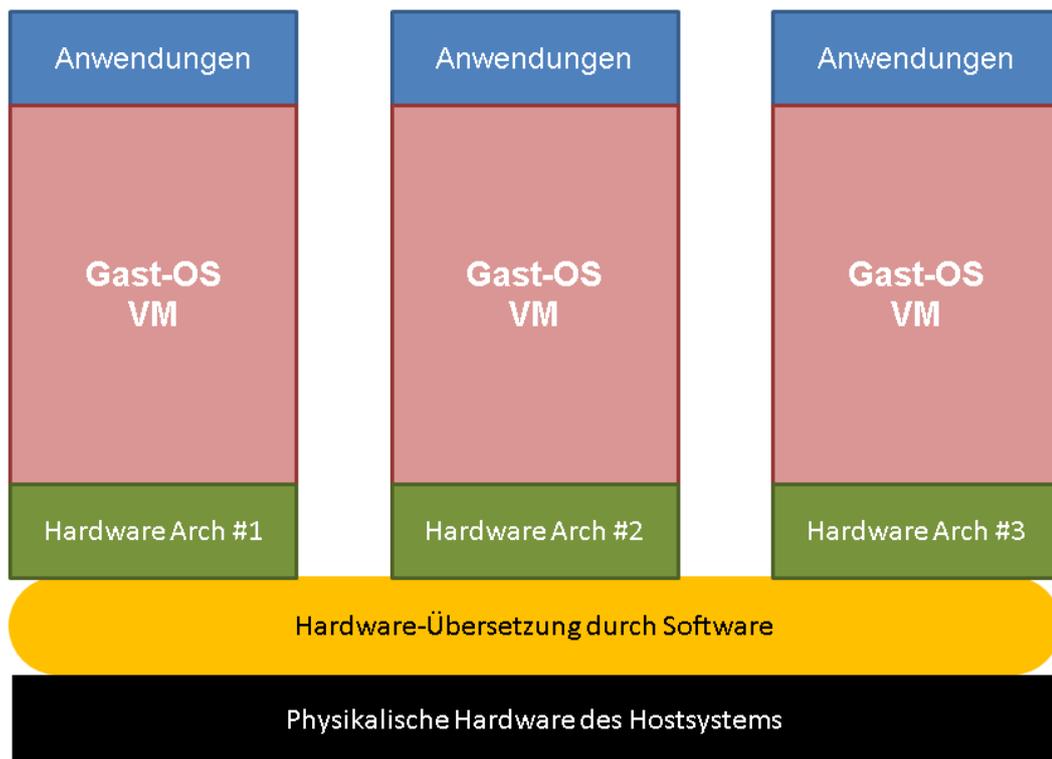


Abbildung 4.1: Unterschiedliche Hardwarearchitekturen können mittels Software in die physische Hardware übersetzt werden.

4.2 Vollständige Virtualisierung

Vollständige Virtualisierung wird auch als *Native-Virtualization*, *Hardware-Virtualization* oder *Full-Virtualization* bezeichnet. Um eine einheitliche Begriffswelt einzuführen, wird deshalb auf den Wortlaut aus [16, 27] zurückgegriffen.

Bedient der Scheduler des Host-Systems das Gast-System, stellt die Vollständige Virtualisierung durch die VM einen dedizierten Teil der Hardware des Host-Systems an die Gast-Systeme bereit. Die Gastbetriebssysteme müssen dementsprechend kompatibel zu der Host-Architektur sein. Wichtig ist die Unterscheidung zwischen VM und Gastbetriebssystem. Die VM ist lediglich eine Art Proxy für die Hardware, auf welchem das Gastbetriebssystem ausgeführt wird und entkoppelt von der physischen Hardware läuft.

Um mehrere Gäste gleichzeitig zu betreiben, muss jede Maschine für sich gekapselt und isoliert von anderen Gästen agieren können. Um dies zu erreichen, wird eine zusätzliche Virtualisierungskomponente, der Virtual Machine Monitor (VMM) eingeführt. Dieser hat die Aufgabe, systemkritische Befehle der VM entgegen zu nehmen, gezielt an die CPU weiterzureichen und der entsprechenden VM die Antwort zurückzuliefern. Dies ist notwendig, da das Gastbetriebssystem keine Informationen darüber enthält, dass es in einer virtuellen Umgebung läuft. Ausgeführt wie auf einem physikalischen Rechner beansprucht das Gastbetriebssystem die komplette CPU und Speicherverwaltung. Abbildung 4.2 zeigt wie der Hypervisor mit der integrierten Management Unit (Mgmt Unit) den Kontrollfluss zwischen den Gastsystemen und dem Host regelt. Im Gegensatz zur Emulation kann Vollständige Virtualisierung nur die Hardwarearchitektur des Host-Systems anbieten.

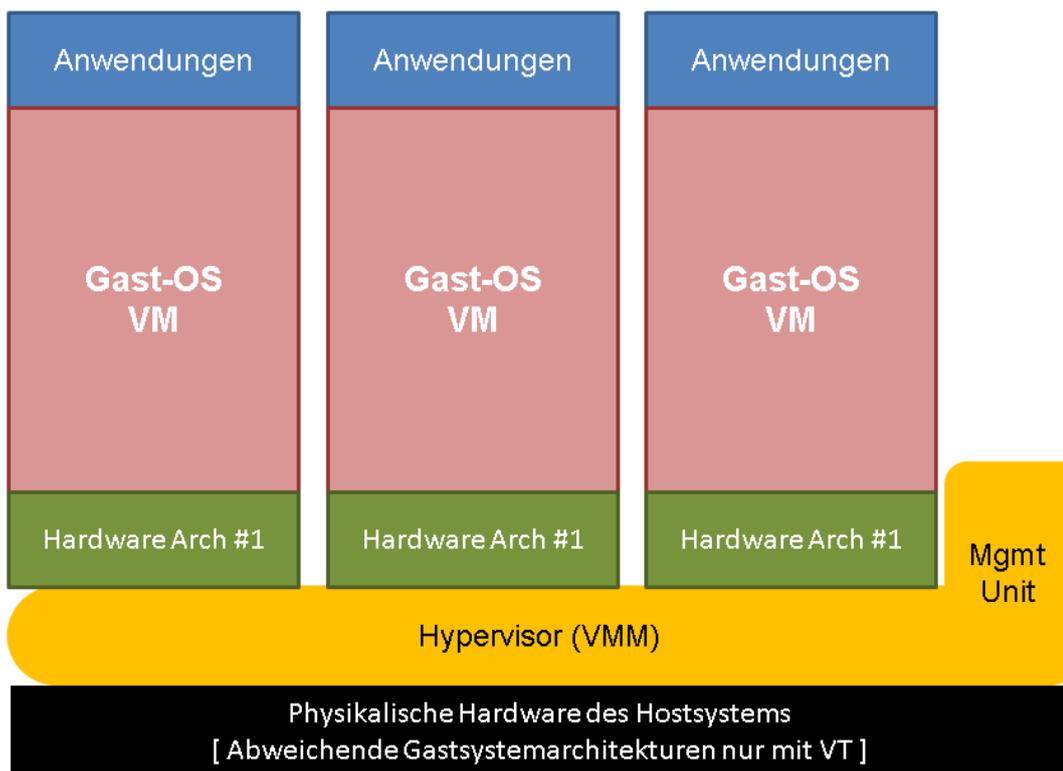


Abbildung 4.2: Vollständige Virtualisierung beinhaltet eine Managementfunktion, um die Ressourcen gezielt an die VMs zu adressieren.

Heutige Prozessoren arbeiten mit insgesamt vier Privilegstufen (Ringen). Diese Stufen oder Ringe symbolisieren eine Einschränkung in den verfügbaren Befehlen der CPU. Ring 0 hat vollen Zugriff auf die Hardware, Ring 3 keinen. Aktuelle Betriebssysteme verwenden lediglich Ring 0 und Ring 3. In Ring 3 laufen alle nicht-privilegierten Anwendungsprozesse. Diese Prozesse haben keinen Zugriff auf die Hardwareressourcen und bleiben gekapselt in dem ihm zugewiesenen Speicher. In diesem sogenannten User-Mode stehen einer Anwendung nur Teile des CPU-Befehlssatzes zur Verfügung.

Arbeitet ein solcher Prozess mit einem ihm nicht verfügbaren Befehl, wechselt er in den Kernel-Mode in Ring 0 und setzt einen System Call an die CPU ab. Dieser Ring ist essentiell für grundlegende Aufgaben des Betriebssystems [4]. Abbildung 4.3 zeigt diese Ringstruktur.

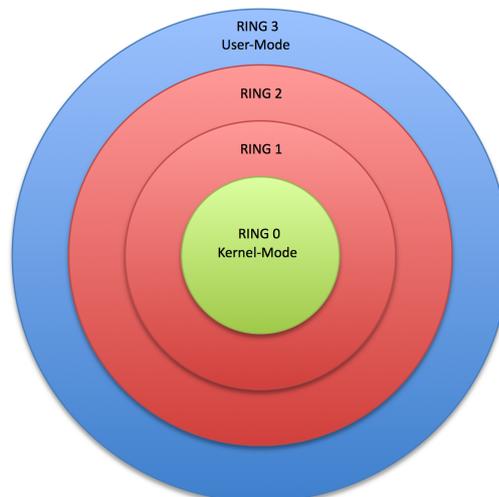


Abbildung 4.3: Ringstruktur eines x86-Prozessors

In der Ringstruktur befindet sich die Herausforderung der Vollständigen Virtualisierung. Aktuelle CPUs sind nicht in der Lage zu unterscheiden, welche VM den System Call abgesetzt hat. Dies ist Aufgabe des Hypervisors. Der Hypervisor verfolgt die von den einzelnen Maschinen ausgeführten Kernel-Modus Befehle, reicht sie an die CPU und leitet das Ergebnis an die gewünschte VM. Diese Funktionalität führt zu der Grundvoraussetzung, dass alle System Calls durch den Hypervisor führen. Weiter impliziert dies, dass das Gastbetriebssystem in einen minderwertigen Ring als der Hypervisor transformiert wird. In den meisten Fällen wird dazu der nicht verwendete

Ring 1 missbraucht (s. Abbildung.4.4). Der Hypervisor arbeitet in Ring 0 und kann somit alle System Calls abfangen und kontrolliert an die CPU durchreichen. Da ein Betriebssystem für gewöhnlich in Ring 0 läuft, muss der Hypervisor für jeden System Call, der in Ring 1 eine Ausnahme - in der Fachlektüre wird von Exception gesprochen - wirft, eine Funktion bereitstellen, die die Exception abfängt und aufbereitet an das Betriebssystem zurück gibt. Für das Betriebssystem der Gastmaschine ändert sich dabei nichts. Diese von VMWare eingeführte Technik nennt sich *Binary Translation* und erlaubt den Betrieb sämtlicher Betriebssysteme ohne Modifikation [16].

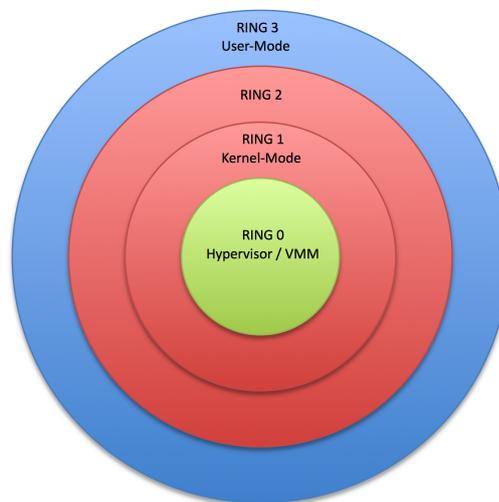


Abbildung 4.4: Ringstruktur eines x86-Prozessors mit Virtualisierungstechniken

Binary Translation bedarf sehr aufwendiger Softwarelösungen und kostet somit Performance. Ein Lösungsansatz für diese Virtualisierungsproblematik bringen die aktuellen CPUs mit Virtualisierungstechniken. Intel-VT oder AMD-V Prozessoren übernehmen das Weiterleiten der System Calls, was den Gastsystemen erlaubt, weiterhin in Ring 0 zu arbeiten. Hierbei übernimmt die CPU direkt die Aufgabe des Hypervisors. Realisiert wird dies durch die Einführung einer neuen Schicht in der CPU. Es existiert neben dem Root- und Usermode nun eine dritte Schicht; die Hypervisor-Schicht. Für die Ringstruktur bedeutet dies die Einführung eines neuen grundlegenden Rings -1. Dies macht den Einsatz von der software-getriebenen Binary Translation obsolet [13, 27].

Da jede VM direkten Zugriff auf die physikalische CPU hat, verschwindet der Overhead der im Vergleich der Emulation. Es entsteht nahezu native Geschwindigkeit für die VM. Der Hypervisor läuft dabei noch als Kontrolleinheit über die VM mit, reicht die CPU jedoch direkt an sie weiter.

Nachteile sind dabei vernachlässigbar unter der Annahme, dass die aktuellen CPUs Virtualisierungstechniken besitzen und finanziell erschwinglich sind. Ein Mangel an diesen CPUs erfordert eine Modifikation jedes Betriebssystems.

Bekanntesten Vertreter der Vollständigen Virtualisierung sind die ESX-Server von VMWare, sowie XEN und KVM, sofern Letztere auf Systemen mit hardwareseitigen Virtualisierungstechniken in Betrieb genommen werden. In anderen Fällen werden Emulationen beziehungsweise Para-Virtualization Ansätze gefahren.

4.3 Para-Virtualization

Para-Virtualization erfordert die Anpassung des Gastbetriebssystems, so dass alle System Calls der VMs direkt auf die Hardware zugreifen. Damit der Hypervisor die kritischen Befehle der einzelnen VMs verfolgen kann, muss die Modifikation des Gastbetriebssystems zur richtigen Zeit einen Kontextwechsel von der virtuellen Maschine in den Hypervisor tätigen. Da die CPU keine Informationen über diesen Wechsel enthält und das Betriebssystem nicht weiß, dass es in einer virtualisierten Umgebung läuft, muss dieses im Kernel angepasst werden, sodass es selbstständig einen Interrupt auslöst, wenn eine Aktion an den Hypervisor ausgeführt wird [26].

Die Skizze in Abbildung 4.5 zeigt schematisch die Anpassung des Betriebssystems. Die Anpassung der einzelnen Betriebssysteme ermöglicht ein unterschiedliches Angebot an Hardwarearchitekturen. Eine für Virtualisierung ausgelegte Host-Architektur ist nicht notwendig.

Ein Vorteil der Para-Virtualization ist der Geschwindigkeitsgewinn auch bei fehlender Hardwareunterstützung. Der Hypervisor muss durch die Modifikation des Gastbetriebssystems deutlich weniger Exceptions abfangen.

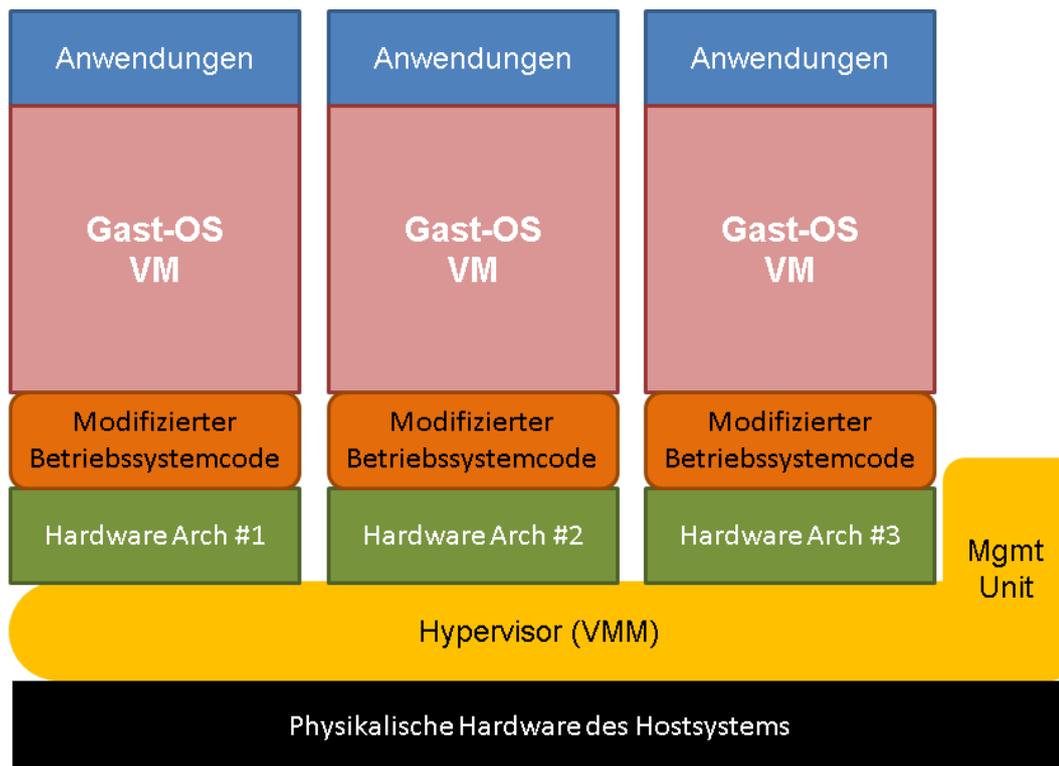


Abbildung 4.5: Eine Modifikation des Gastbetriebssystems ist bei Para-Virtualisierung notwendig.

Ein Nachteil ist, dass proprietäre Betriebssysteme nicht quell-offen sind. Somit ist eine Modifikation des Kernels nicht möglich. VMWare bietet dafür die VMWare-Tools an, die ausgeführt als Softwareapplikation, diese Modifikation übernehmen. Für die Linux-Distributionen wie CentOS und Debian gibt es bereits fertige Patche.

Bekannteste Vertreter für Para-Virtualization sind XEN und VMWare.

VIRTUALISIERUNGSSOFTWARE

Dieses Kapitel behandelt die für diese Arbeit eingesetzte Software zur Virtualisierung. Die Private Cloud im SCC ist realisiert mit einem Cluster, welches ausschließlich CPUs mit Hardwareunterstützung besitzt. Somit ist der Einsatz von KVM als Vollständige Virtualisierung möglich.

Dieses Kapitel geht näher auf KVM, der auf KVM aufbauende Prozessoremulator QEMU und die unabhängige Programmierschnittstelle Libvirt ein.

5.1 KVM

Kernel-based Virtual Machine (KVM) ist wie XEN eine Lösung zur Vollständigen Virtualisierung und benutzt die Unterstützung der Prozessoren von AMD und Intel. Bei der IT-Infrastruktur dieser Arbeit wurde KVM anstelle von XEN eingesetzt. Gründe hierfür sind:

- einfachere Installation
- KVM ist im Gegensatz zu XEN vollständig in den Linux-Kernel integriert und somit bei jeder Distribution vorhanden.
- höheres Augenmerk auf Full-Virtualization
- Xen besitzt zwei privilegierte Schichten für den Hypervisor¹ - KVM nur eine, welche leichter abzusichern ist

¹Die zwei Schichten sind der eigentliche Hypervisor und die privilegierte VM (Dom0), die als KontrollVM für die bereitgestellten Gastssysteme dient.

Die Ideologie von KVM ist simpel - *"don't reinvent the wheel!"*. Gängige Virtualisierungsprodukte implementieren zwei privilegierte Schichten. Den Hypervisor und eine VM in einem privilegierten Modus des Kernels, die alle ausgeführten VMs verwaltet. Die Aufgabe dieser VM ist ein Scheduling der einzelnen VMs an den Hypervisor und ihr CPU-Zeit zuzuweisen. Die Grundidee von KVM besteht darin, diese Scheduling-Aufgaben nicht zu implementieren, sondern diese dem Linux-Kernel zu überlassen.

Der Linux-Kernel verwaltet alle ausgeführten Prozesse. Die von KVM erstellte VM startet als eigener Prozess, was die Vergabe der CPU-Zeiten für jede VM dem Linux-Kernel zuweist. Somit sind Befehle wie `kill`, `stop`, `pause` möglich. Ein Vorteil ist, dass KVM Teilaufgaben des Hypervisors, die komplette Prozessverwaltung jeder einzelnen VM, dem Linux-Kernel überträgt.

KVM besteht aus insgesamt drei Bestandteilen. Darunter befinden sich zwei Kernelmodule, welche der Linux-Kernel zur Laufzeit lädt.

kvm.ko Kernelmodule für den Betrieb des Hypervisors

kvm-amd.ko , **kvm-intel.ko** gerätespezifische Kernelmodule zur Unterstützung der Full-Virtualization

/dev/kvm Gerät zur Verwaltung der VMs und der zugewiesenen Hardware

Jede VM bekommt eigene virtualisierte Peripheriegeräte wie Netzwerkkarten, Grafikkarten und Ähnliches. Als Voraussetzung, dass KVM nicht auf die x86 Architektur beschränkt ist, wird QEMU als Prozessoremulator eingesetzt. Auf diesen wird im späteren Verlauf des Kapitels eingegangen.

Da jede VM in einem eigenen Prozess läuft, der vom originalen Linux Scheduler bedient wird, hat jeder Prozess folglich zwei Ausführungsmodi:

- Anwendungsmodus (User-Mode)
- Root-Modus (Kernel-Mode)

KVM als Vollständige Virtualisierung bringt einen dritten Modus, den *Guest-Mode*. Dieser Mode läuft in einem eigenen Kernel- und Usermode, was die Unterstützung der Prozessoren verlangt.

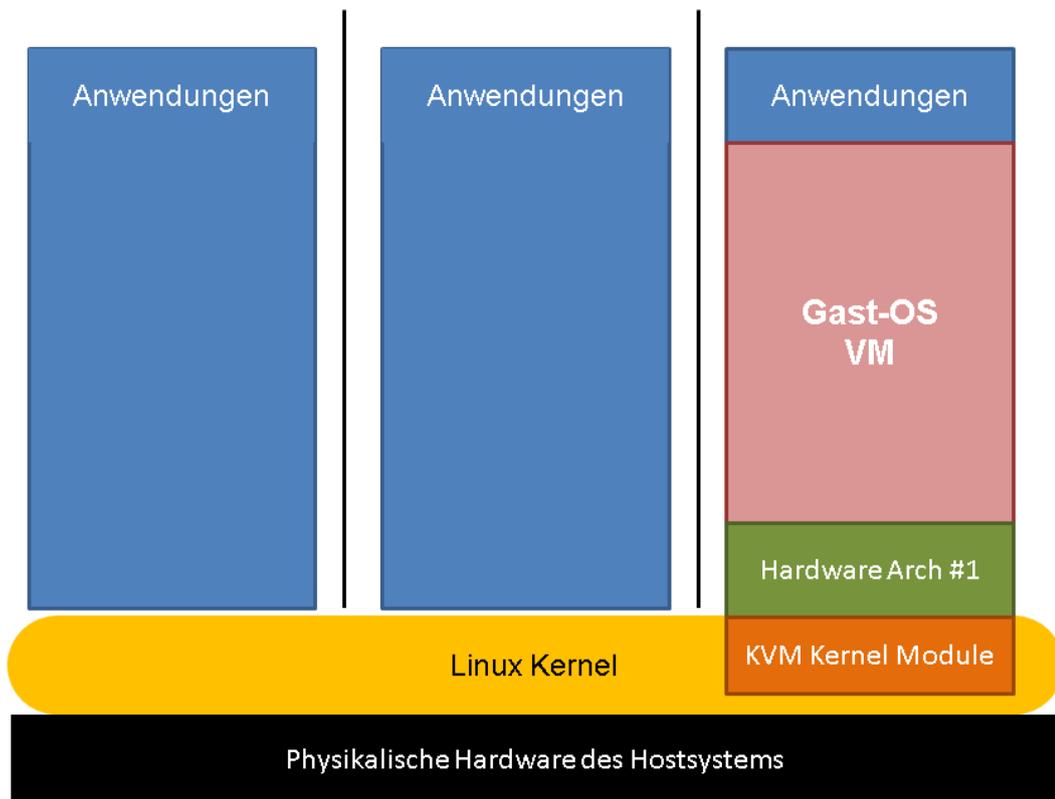


Abbildung 5.1: Skizze wie KVM in den Linux Kernel integriert ist und die dort vorhandene Umgebung nutzen kann.

KVM bringt als Verwaltungswerkzeug nur Kommandozeilenbefehle mit. Der folgende Befehl (siehe Listing 5.1) ist verantwortlich für die Erstellung einer neuen VM. Der Parameter `-m` setzt die Größe des Arbeitsspeichers. Der Parameter `-boot` kann entweder mit `a`, `c` oder `d` gesetzt werden. Die Buchstaben bedeuten die einzelnen bootfähigen Laufwerke: Floppydisk=`a`, HDD=`c`, CD=`d`.

```

1 # kvm -m <Ram in MB> \
2 -cdrom <Pfad/zum/CD-Image> \
3 -boot <a|b|c> \
4 -drive file=<Pfad/zur/HDD>

```

Listing 5.1: Starten einer Instanz

Weitere Parameter und Beispiele befinden sich in der dazugehörigen MAN-Page.

Der Start einer Maschine öffnet den QEMU-Monitor. Dies ist ein X-Fenster zur Administration der VM. Durch diesen Monitor lässt sich die Maschine ausschalten, Hardwareressourcen hinzufügen und Befehle für Sondertasten einfügen.

5.2 QEMU

QEMU ist ein kostenfreier Hardwareemulator und bietet die Möglichkeit, unterschiedliche CPU-Architekturen zu emulieren. So können für die jeweiligen Gastbetriebssysteme die folgenden Architekturen emuliert werden:

- x86
- x86_64
- AMD64
- PowerPC
- Sparc32/64

QEMU als Prozessoremulator ist ein Bestandteil von KVM. QEMU ist ein Hardwarebeschleuniger für zahlreiche Virtualisierungslösungen. QEMU ergänzt die fehlende Hardwareemulation des nativen KVM und ermöglicht die Nutzung unterschiedlicher Architekturen.

Trotz der Emulation verzögert QEMU den Betrieb von KVM nicht. Diese verwendete Technik nennt sich Dynamic Translation². QEMU bietet:

- Snapshot-Funktionalität,
- Live-Migration,
- Debugging-Funktionalität und
- Emulation von Hardwarefehlern

²nähere Informationen finden sich unter:
http://www.usenix.org/publications/library/proceedings/usenix05/tech/freenix/full_papers/bellard/bellard_html/index.html

Abbildung 5.2 zeigt beide Produkte - KVM, QEMU - in Kombination. Diese Kombination bedeutet, dass eine damit erstellte VM, sofern sie eine x86-Architektur aufweist, native CPU-Geschwindigkeiten erreicht und mit beliebig emulierter Hardware, wie zum Beispiel Netzwerkkarten etc., gestartet werden kann.

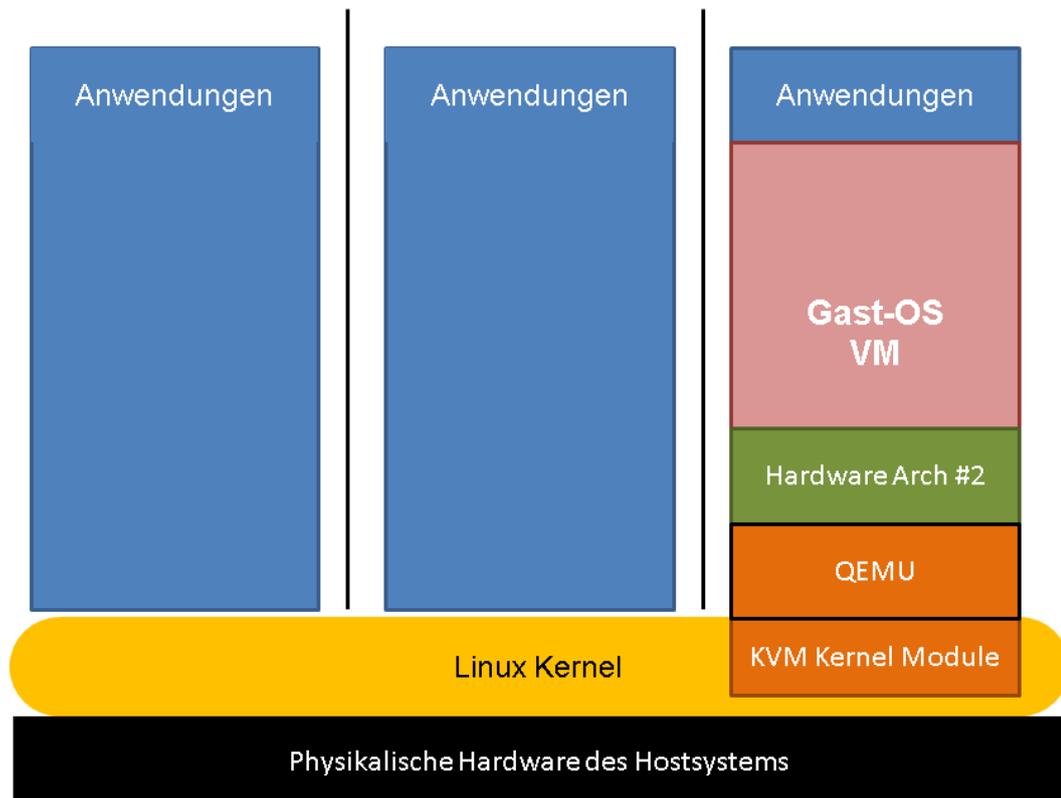


Abbildung 5.2: QEMU arbeitet auf der KVM Architektur und emuliert verschiedene Architekturen.

Ein hilfreiches Werkzeug ist *qemu-img*. Dieser Kommandozeilenbefehl ermöglicht die Erstellung von virtuellen Festplattendateien, die in zahlreiche Formate, wie z.B. *raw* oder *qcow2* konvertiert werden können. QEMU ist damit flexibel was die Kompatibilität zu anderen Virtualisierungslösungen betrifft. Unter anderem wird das von VMWare genutzte Format *vmdk* unterstützt.

Gesteuert wird QEMU analog zu KVM mittels Kommandozeile. Die Befehle von KVM identisch mit denen von QEMU. Wird eine VM gestartet, öffnet

sich diese innerhalb eines X-Fensters. Dies zeigt den QEMU-Monitor. Mittels dieses Werkzeuges kann die VM während des Betriebs verwaltet werden. Wichtige Funktionen zeigt folgende Auflistung:

- Wechsel und Auswerfen von Medien (DVD-ROMs, Floppies)
- Pausieren der VM
- Sichern oder Wiederherstellen von VM-Zuständen
- Statusüberprüfung und Debugging der VM

Zur Eingabeaufforderung des QEMU-Monitors gelangt man mittels STRG+ALT+2. Eine Auflistung aller möglichen Befehle erscheint mit Eingabe von `help`.

5.3 Libvirt

Die in C geschriebene Bibliothek Libvirt bietet Schnittstellen zur Verwaltung unterschiedlicher Virtualisierungslösungen. Heterogene Rechenzentren fahren oft verschiedene Virtualisierungsansätze, die nicht mit dem gleichen Werkzeug zu verwalten sind. Libvirt ist eine API und stellt einheitliche Methoden zur Kontrolle von mehreren Virtualisierungslösungen. Die Administration wird von dem Hypervisor abstrahiert. Derzeit werden unter anderem folgende Software unterstützt:

- QEMU
- KVM
- VirtualBox
- VMware ESX
- Xen
- OpenVZ
- OpenNebula

Es stehen Schnittstellen zu den Sprachen C#, Ruby, Python und Java bereit (siehe Abbildung 5.3).

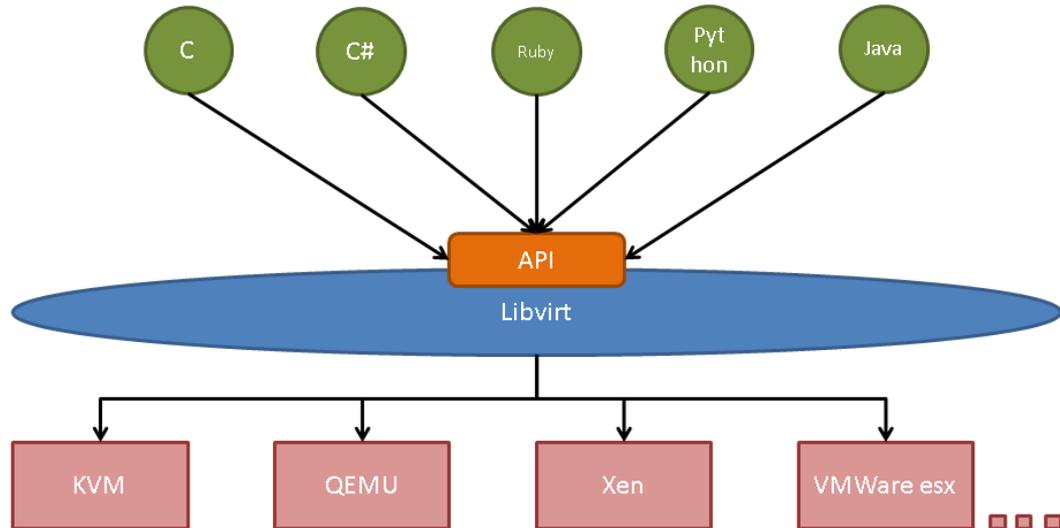


Abbildung 5.3: Funktionsweise der Libvirt

Libvirt verwaltet nicht nur die VMs, sondern auch sämtliche von der VM zu verwendenden Netzwerke beziehungsweise externen Medien. Die komplette Konfiguration der VM inklusive Netzwerk und Storage geschieht in Form von eXtensible Markup Language (XML) Dateien.

Folgend genannte Programme arbeiten mit der Libvirt. Beide manipulieren die XML-Datei, die zur Konfiguration der VM zuständig ist. Abgesetzte Steuerungsbefehle werden von der Libvirt gezielt an die darunterliegende Virtualisierungslösung gereicht [25].

Virt-Manager Ermöglicht die Verwaltung mit grafischer Oberfläche.

Virsh Ermöglicht die Verwaltung auf der Kommandozeile.

CLOUD COMPUTING - THEORIE

Cloud Computing dient dem Angebot von Diensten. Einzelne Dienste wie beispielsweise das Dokumentenverwaltungssystem Google Docs oder das Bereitstellen von kompletten Rechnerinstanzen auf Root-Ebene werden innerhalb der Cloud dem Endnutzer zur Verfügung gestellt. Das Bild der Wolke verdeutlicht die Abstraktionsebene zwischen Angebot der Services und der dafür benötigten Ressourcen. Diese Abstraktionsebene heißt Virtualisierung.

Dieses Kapitel liefert eine Einführung in die Architekturen des Cloud Computings. Dabei werden der allgemeine Aufbau sowie die weit-verbreitetsten Formen des Cloud Computings erläutert. Den Nutzen des Cloud Computings zeigt die am Ende dieses Kapitels aufbereitete Wirtschaftlichkeitsbetrachtung. Diese Betrachtung ist die Grundlage des Accounting-Systems.

6.1 Definition

Das Cloud Computing ist ein verteiltes System, vergleichbar dem Peer-to-Peer oder Grid Computing. Ein Merkmal von Cloud Computing ist die Auslagerung bestimmter Geschäftsmodelle zu Dienstleistern, die Speicher- und Rechenkapazität bereitstellen, die mit Hilfe fest definierter Schnittstellen und Service-Verträgen nutzbar sind. Details der Services, wie beispielsweise der Standort der Hardwareressourcen sind nicht bekannt.

Eine Analogie zur Verdeutlichung ist der Strom aus der Steckdosen. Die Steckdosen sind feste Schnittstellen und mit jedem passenden Stecker kann

Strom bezogen kann. Die Abrechnung beim Anbieter folgt den Richtlinien der Service-Verträge. Wo der Strom herkommt und die Art und Weise der Herstellung, ist vernachlässigbar. Wichtig ist die Dienstqualität, die sogenannte Quality of Service (QoS) des Stromanbieters:

- Er hat Sorge zu tragen, dass jederzeit der Strom verfügbar ist.
- Im Fehlerfall muss sich der Anbieter um die Wiederherstellung der Dienstleistung kümmern – nicht der Endverbraucher.

Abbildung 6.1 skizziert den Zugang zur Cloud. Die Erbringung der Dienste innerhalb der Cloud geschieht für die Benutzer transparent.

Aspekte der Verfügbarkeit und Datensicherheit liegen beim Cloud Computing im Aufgabenbereich der Dienstanbieter und nicht bei den Endbenutzern. Im konkreten Fall der Amazon Webservices (AWS) wird dieses verständlich. Amazon garantiert, dass eine bereitgestellte Instanz mit einer Verfügbarkeit von mindestens 99,95%¹ läuft. Der Servicevertrag spiegelt sich in den Nutzungsgebühren von aktuell 1 Cent pro Stunde und den Zugangsdaten über Secure Shell (SSH) oder Remote Desktop Protocol (RDP)². Probleme innerhalb der zugrunde liegenden IT-Infrastruktur tangieren die Benutzer der Dienste nicht.

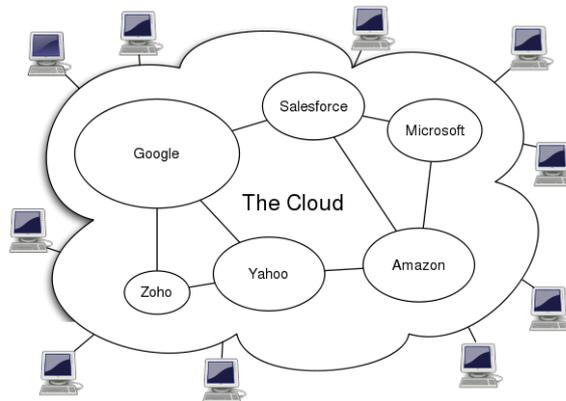


Abbildung 6.1: Skizzenhafte Auslagerung der Dienste in die Cloud. Als Schnittstelle dient ein Zugang in die Cloud. Genauere Informationen über die Art und Weise der Ressourcen bleiben verborgen.

¹für aktuelle Zahlen siehe <http://aws.amazon.com/ec2/>

²Dies variiert je nach Betriebssystem.

6.2 Cloud Architekturen

Cloud Computing beinhaltet mehrere Formen der angebotenen Dienste. Zwischen den einzelnen Architekturen gibt es keine klare Abgrenzung; der Übergang zwischen ihnen ist fließend. Auf den einzelnen physikalischen Rechnern läuft die Cloud-Software, die je nach Art des Servicelevels abstrakte Dienste für die Außenwelt anbietet.

Abbildung 6.2 zeigt die Hierarchie der einzelnen Cloud-Architekturen. Die Sichtbarkeit auf die darunterliegenden Technologie nimmt für den Anwender zur Basis der Pyramide ab.

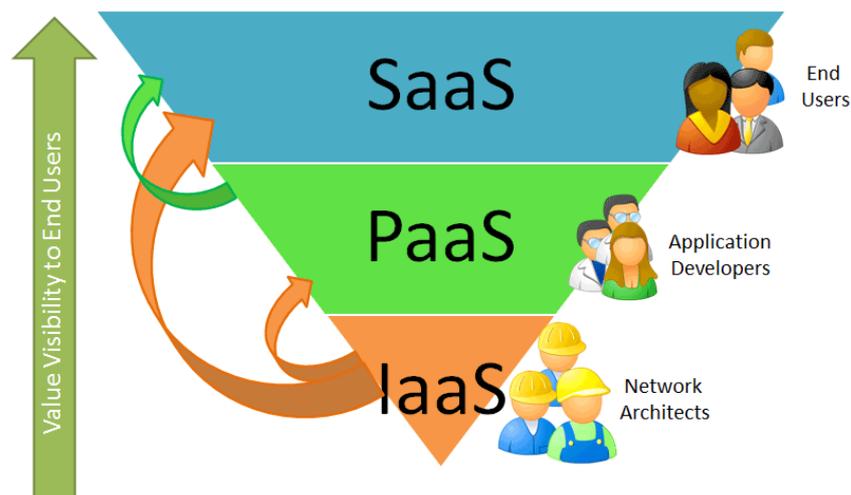


Abbildung 6.2: Stack der Cloud-Architekturen

Infrastructure as a Service (IaaS) Diese Form stellt ein Betriebssystem bereit. Der Anwender kann eine vollständige Instanz und übernimmt die Konfiguration für sämtliche Anwendungen und Dienste. Zugang bekommt er durch die Protokoll RDP und SSH. Er hat Administratorrechte.

Bsp: Amazon EC2

Platform as a Service (PaaS) Bei PaaS werden Laufzeitumgebungen und Entwicklungsumgebungen bereitgestellt, die vom Anwender ohne zusätzliche Einrichtungen genutzt werden können. Die Konfiguration und Administration der kompletten Instanz ist nicht vom Anwender zu übernehmen. Die benötigte Rechenleistung verteilt sich auf die kompletten Kapazitäten der Cloud-Umgebung.

Bsp: Google App Engine

Software as a Service (SaaS) Die abstrakteste Form von Cloud-Modellen ist SaaS. Der Anwender ist auf die Nutzung bereitgestellter Dienstleistungen beschränkt. Diese Dienstleistungen sind im Gegensatz zu PaaS voll implementiert.

Bsp: Google Mail

Diese drei Varianten sind populär. Weitere werden unter dem Punkt *Everything as a Service* gekapselt. Unter anderem auch der von Amazon angebotene Dienst *Human as a Service*. Unter diesem Punkt verstecken sich kleine Aufgaben, die zu aufwendig für Computerprogramme sind, jedoch für Menschen keine Probleme darstellen.

Die Abbildung 6.3 zeigt das Zusammenspiel der Architekturen und verdeutlicht, dass es keine klare Abgrenzung zwischen IaaS, PaaS und SaaS gibt. Des Weiteren zeigt die Abbildung die bekanntesten Anbieter jeder Architektur.

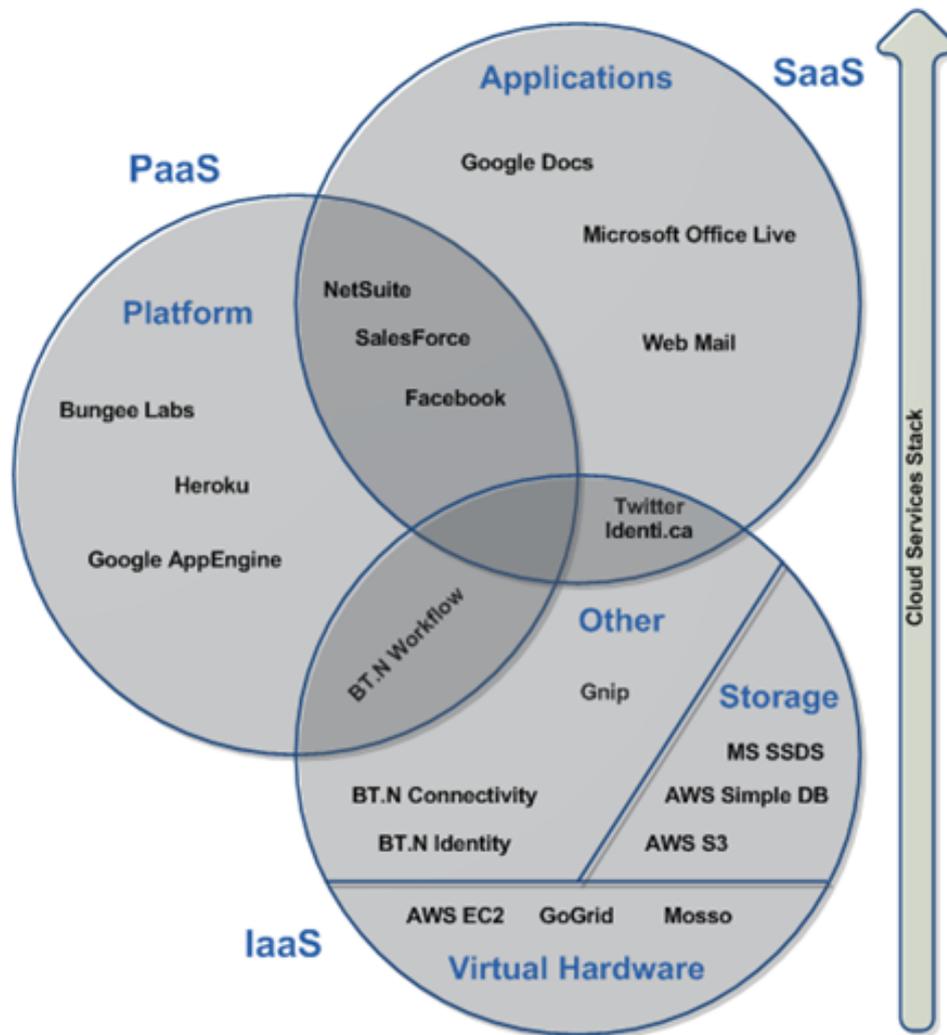


Abbildung 6.3: Übersicht der einzelnen Architekturen

6.3 Aufbau einer Cloud

Der Aufbau wird Bottom-Up erklärt, am Beispiel von IaaS. Von der untersten Schicht der physikalischen Hardware bis zu den einzelnen Anwendungen, die innerhalb der Cloud laufen können.

Physikalische Hardware

Die Grundlage einer jeden IT-Umgebung ist die Hardware. Im Gegensatz zum Grid-Computing bedarf es nicht zwingend homogenen Ressourcen. Es werden mehrere Rechenzentren mit großen Clustern eingesetzt, die unter zentraler Kontrolle stehen. Auf dieser Hardware werden virtuelle Instanzen erstellt und skalierbar zur Verfügung gestellt. Trotz der Tatsache, dass sämtliche Anwendungsgebiete der Cloud sich hinter in dem Modell der Wolkapseln, bestimmt die zu Grunde liegende Hardware das Maß der Skalierbarkeit und Performance.

Virtualisierung

Die Instanzen von Infrastructure as a Service (IaaS), die Systeme von Platform as a Service (PaaS) und die Software von Software as a Service (SaaS) werden stets virtualisiert. Im Falle von IaaS arbeitet der Endanwender mit VMs. Der Hypervisor ist die wichtigste Komponente der Cloud-Infrastruktur. Der Hypervisor erstellt virtuelle Instanzen auf Grund von vorgefertigten Images. Er muss die Anforderung erfüllen, dass er über einfache Kommandozeilenbefehle oder Webservices steuerbar ist. Dies stellt die Grundlage für die verteilte Anwendung und die Steuerung an zentraler Stelle.

Cloud-Software

Die Cloud-Software, wie die AWS oder die Open Source-Entwicklung Eucalyptus, ist die zentrale Steuereinheit einer Cloud-Infrastruktur. Diese steuert die Bereitstellung der Ressourcen beziehungsweise das Skalieren der Services on-demand. Diese Komponente beinhaltet gleichzeitig einen Load-Balancer, um die Verteilung der Rechenlast für den Kunden, aber auch für den Betreiber der Cloud zu optimieren.

Die verwendete Virtualisierungskomponente und die Cloud-Software stellen ihre Dienste via Webservices bereit. Das ermöglicht einen programmatischen Zugriff zur Außenwelt. Ein Anwender kann aus jeder Anwendung, von jeder Homepage, virtualisierte Dienste der Cloud nutzen - und zwar zur Laufzeit.

Anwendungen

Das Cloud-Computing beschränkt sich nicht auf einzelne Anwendungsgebiete. Eine Applikation hat die Chance, virtuelle Instanzen bereitzustellen und gewisse Ressourcen innerhalb des Programmlebenszyklus auf die entfernte Rechenleistung auszulagern. PaaS lagert Anwendungen aus, die auf entfernte Rechner innerhalb der Cloud zurückgreifen können. Eigens entwickelte Applikationen können Anwendungen wie GoogleMail und ähnliche Cloud-Services nutzen. Eine Cloud-Infrastruktur ist modular, was sich in der Dimension der Skalierbarkeit widerspiegelt.

6.4 Private Cloud im SCC

Bei der Umsetzung der Cloud-Infrastruktur am SCC handelt es sich um eine so genannte *Private Cloud*. Der Begriff ist auf die Zugangsbeschränkung beziehungsweise die örtliche Lage der Hardwareressourcen zurückzuführen. In einer Private Cloud nutzen Firmen ihre eigenen Rechenzentren als Grundlage für ihre firmeneigene Cloud. Ein Vorteil ist die Kontrolle über die eigenen Ressourcen.

Allerdings kann die Private Cloud die Ideologie des Cloud Computings verletzt werden, da die Ressourcen nicht flexibel und unabhängig von dem Ort für Jedermann bei Bedarf zugänglich gemacht werden³.

³Eine öffentliche Gegenstimme: James Urquhart - „Urquhart on Barriers to Exit“, Seite 16.

Die Implementierung dieser Arbeit basiert auf der Private Cloud des SCC. Nutzer innerhalb des KIT sollen in der Lage sein, Virtuelle Instanzen der Private Cloud zu mieten. Das zu implementierende Accounting-System rechnet diesen Anwendern die Nutzungsgebühren ab.

6.5 Wirtschaftlichkeitsbetrachtung

Neben den technischen Aspekten werden bei der betriebswirtschaftlichen Betrachtung einige Vorteile des Cloud Computings deutlich. Unter dem Ausdruck *Service on-demand* versteht man das Nutzen der Cloud-Services ausschließlich bei Bedarf. Sobald kein Bedarf an die Dienste mehr besteht, können sie minutengenau gekündigt werden.

Besser ersichtlich wird dies an dem Beispiel eines einfachen Unternehmens. Das einfache Unternehmen verkauft ein Produkt und ist etabliert am Markt. Das hat zur Folge, dass im Idealverhalten die Auslastung der vorhandenen physikalischen Ressourcen auf 100 Prozent läuft und die Kapazität voll ausgeschöpft wird. Das Jahresbudget des Unternehmens wird vollständig ausgenutzt, um die Auslastung auf 100 Prozent zu halten.

Abbildung 6.4 zeigt das Wirtschaftsverhalten des Unternehmens im Idealfall bei konstanter Auslastung. Die Diagramme verdeutlichen, dass die Kapazitäten des Unternehmens genau auf die Anforderungen des Alltagsgeschäfts zugeschnitten sind. Größere Abweichungen führen zu Änderungen in den Ressourcen.

Kommen auf das Unternehmen temporäre Hochauslastungen (engl. Peaks) zu, muss das Unternehmen zusätzliche Hardwareressourcen beschaffen um den Auftrag abzuarbeiten. Nutzt sie diese neuen Ressourcen, indem sie neue physikalische Hardware kauft, erhöhen sich die Fixkosten für das Unternehmen. Entscheidend an diesem Punkt ist, dass die Fixkosten für die Instandhaltung und Anschaffung der Zusatzkapazitäten auch nach Beendigung des Ansturms erhalten bleiben (siehe Abbildung 6.5). Diese zusätzlichen Kosten können für kleine Unternehmen zu ernstzunehmenden Problemen führen, da teure und Kosten verursachende Kapazitäten für die Alltagssituation nicht ausgelastet sind.

Service on-demand bietet eine Kosten effiziente Lösung dieses Problems. Die Nutzung eines Dienstes der Cloud wird nur für die aktive Inanspruchnahme in Rechnung gestellt. Für das Unternehmen bedeutet dies, dass genügend Hardwareressourcen in der Form von IaaS genutzt werden können, um die Lastspitze zu überwinden. Allerdings ohne Steigerung der Fixkosten, da bei Beendigung der Hochauslastung der Dienst minutengenau gekündigt wird. Abbildung 6.6 zeigt den Unterschied der Budgetierung unter Zuhilfenahme von Cloud-Services.

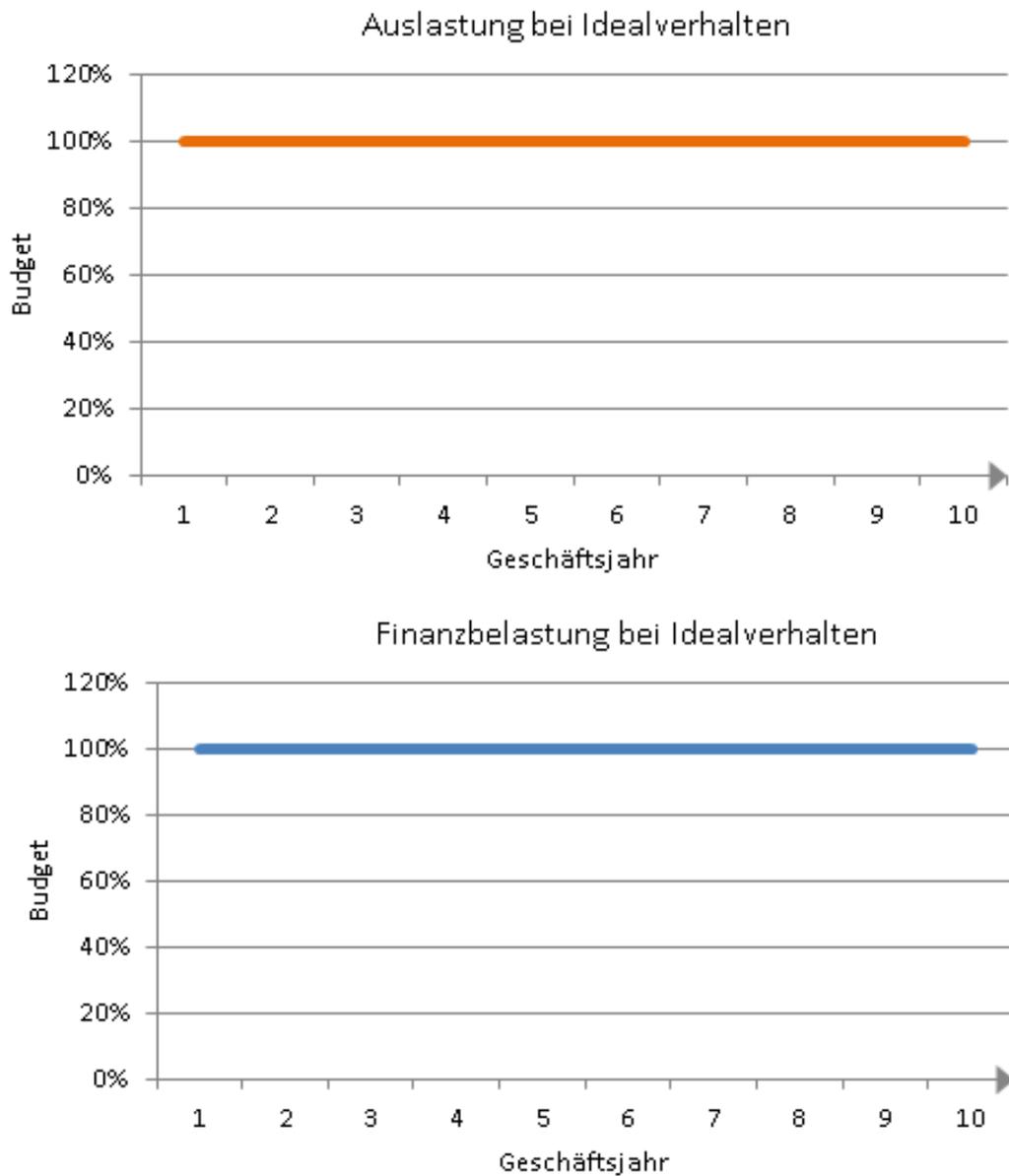


Abbildung 6.4: Idealverhalten in Auslastung und Budgetierung

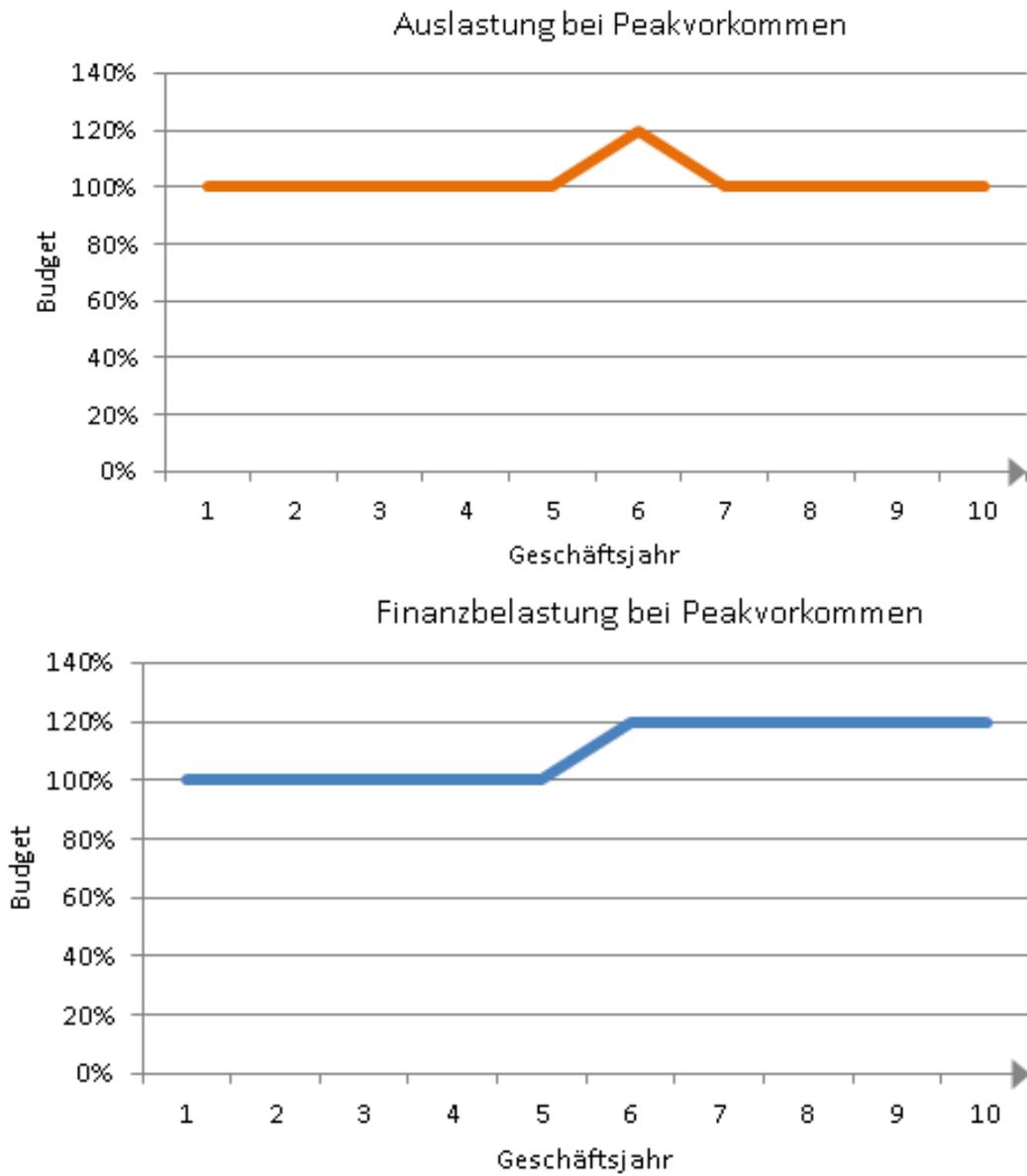


Abbildung 6.5: Verhalten in Auslastung und Budgetierung im Falle von Hochauslastung

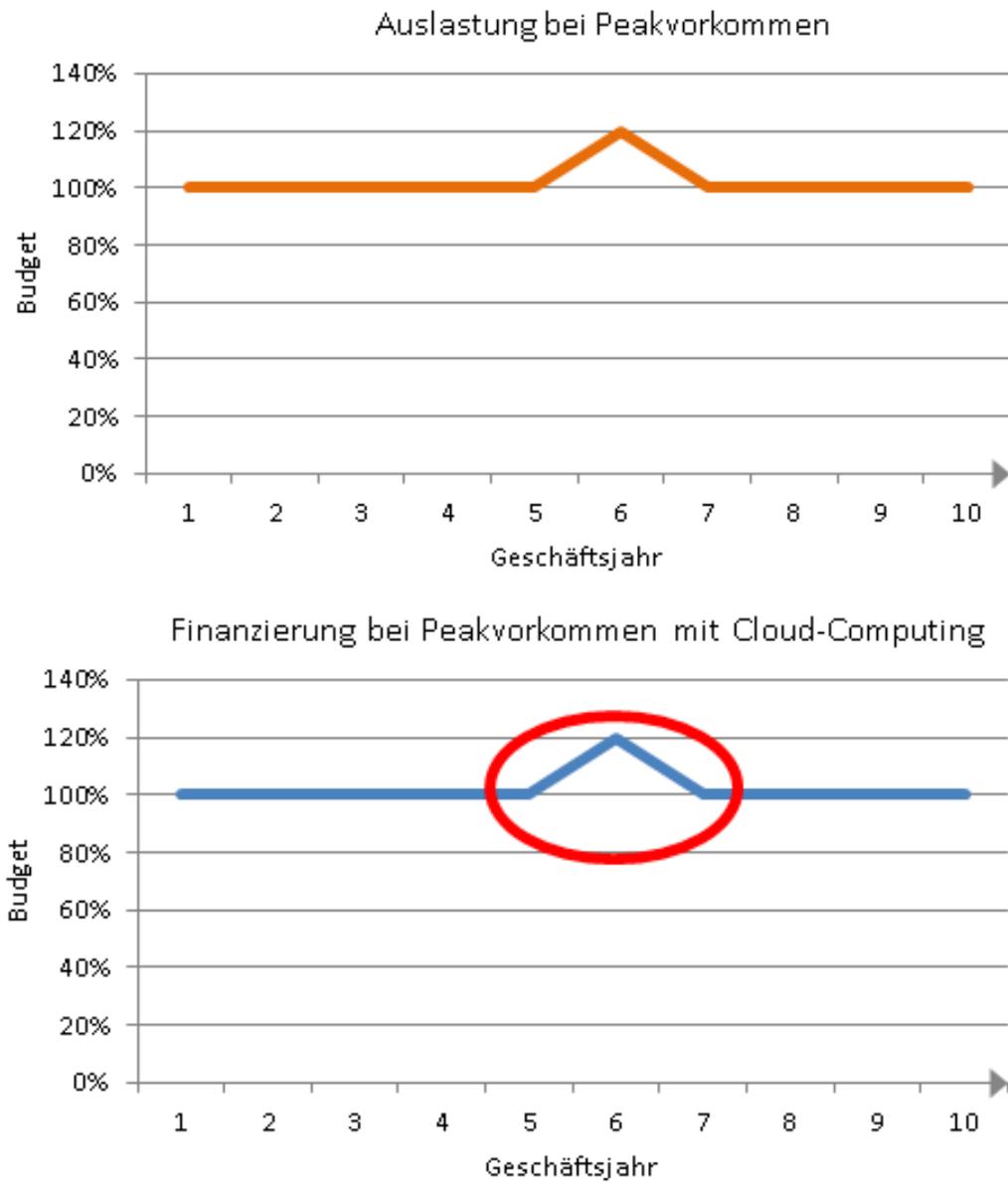


Abbildung 6.6: Verhalten in Auslastung und Budgetierung unter Einsatz von Cloud Computing

EUCALYPTUS

Eucalyptus ist die Abkürzung für *Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*. Dabei handelt es sich um die Software zur Erstellung einer Private Cloud. Eucalyptus bietet in Verbindung mit unterstützter Virtualisierungssoftware wie KVM oder XEN ein Cloud-Angebot in Form der IaaS-Architektur. Virtuelle Instanzen können auf Basis von vorgefertigten Images bereitgestellt werden. Diese Instanzen verteilen sich abhängig von ihrem Ressourcenbedarf auf die vorhandene Hardware. Eucalyptus ist frei verfügbar (Open Source)¹, aktuell in der Version 1.6.2 vorhanden und so in dieser Arbeit eingesetzt.

Ein wichtiges Merkmal an Eucalyptus ist die Kompatibilität zu den AWS. Dies ermöglicht ein Zusammenspiel aus Private und Public Cloud. Der Nutzen einer Hybrid Cloud kann voll ausgeschöpft werden. Ein Unternehmen kann mit seinen vorhandenen Hardwareressourcen eine Private Cloud mittels Eucalyptus realisieren. Dieses kann in seine bestehende Struktur zusätzliche Ressourcen der AWS integrieren, um temporäre Lastspitzen an Aufträgen zu bearbeiten.

7.1 Allgemeiner Aufbau

Der Aufbau von Eucalyptus ist modular. Sämtliche Komponenten sind für sich unabhängig und kommunizieren über Simple Object Access Protocol

¹Es gibt neben dem Open Source Angebot auch ein kommerzielles, welches besseren Support und einige zusätzlichen Features mitbringt. Der Aufbau innerhalb des SCC ist jedoch dabei komplett auf die Open Source Teile beschränkt.

(SOAP) beziehungsweise Representational State Transfer (REST) miteinander. Die Skalierbarkeit der Cloud ist dadurch gewährleistet. Zusätzliche Hardwarekomponenten können dynamisch hinzugefügt werden².

Abbildung 7.1 zeigt den allgemeinen Aufbau von Eucalyptus mit seinen einzelnen Komponenten. Auf die einzelnen Komponenten wird im Abschnitt 7.2 näher eingegangen. Die Management Platform steuert Eucalyptus an zentraler Stelle. Dort können einzelne Rechenkapazitäten – sowohl interne Kapazitäten als auch externe – komponentenweise hinzugefügt werden.

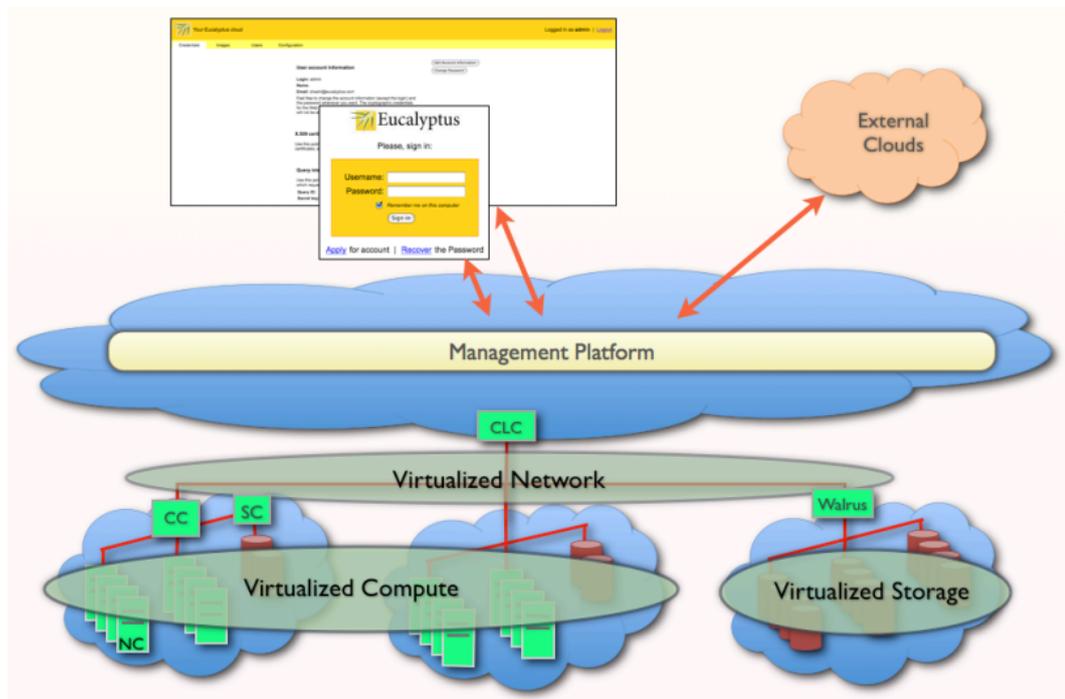


Abbildung 7.1: Technischer Aufbau von Eucalyptus [12]

Eine Einheit innerhalb des *Virtualized Compute* ist für sich eine modular betriebene Hardwareressource³. Ob die Ressourcen von Eucalyptus oder von Amazon verwaltet werden, ist irrelevant [6], da die Management-Komponente von Eucalyptus mit den Webservices von Amazon harmoniert.

²Google inc. finanziert mittlerweile ausschließlich voll mit Servern bestückte Container. Diese werden lediglich mit einem Strom- beziehungsweise FibreChanel-Kabel angeschlossen und sind auf diese Weise dynamisch integriert.

³Meist diverse Cluster innerhalb eines Rechenzentrums.

7.2 Komponenten

Im Folgenden werden alle in Abbildung 7.1 gezeigten Komponenten diskutiert. Für nähere Informationen wird auf den offiziellen Eucalyptus Guide [10, 12] verwiesen.

Cloud Controller (CLC) ist der Einstiegspunkt in die Private Cloud. Der Cloud Controller sammelt Informationen über die Ressourcen und übernimmt Scheduling-Aufgaben. Jede Anfrage zum Erstellen einer Instanz wird vom CLC zu dem Cluster Controller weitergereicht. Wie die Abbildung 7.1 zeigt, stellt der Cloud Controller eine API für die Management Plattform bereit. Der CLC ist für die Administration und Bereitstellung virtueller Hardware zuständig. Innerhalb des Cloud Controller können mehrere Cluster Controller eingehängt werden. Daraus bilden sich verschiedene Regionen.

Cluster Controller (CC) stellt das Frontend zu dem darunterliegenden Cluster. Der Cluster Controller nimmt die Anfragen des CLC entgegen und reicht diese an die zuständigen Node Controller weiter. Die Hauptaufgabe des CC ist das Scheduling der Anfragen auf die einzelnen Nodes innerhalb des Clusters und diese für die Anwender zugänglich zu machen.

Node Controller (NC) ist ein einzelner Knoten innerhalb eines Clusters. Dort werden die Anfragen direkt bearbeitet. Es ist somit die unterste Schicht der Private Cloud. Als einzige Komponente beinhaltet jeder Node Controller eine Virtualisierungssoftware. Die Hauptaufgabe des NC ist die Verwaltung der einzelnen Virtuellen Instanzen innerhalb der jeweiligen Virtualisierungssoftware. Eine Virtuelle Instanz wird mit der Ramdisk, Betriebssysteminstallation und einem Kernelimages gestartet, welches vom Cluster Controller kopiert wird. Aus Performance-Gründen werden die einzelnen Images einmalig kopiert und dort im Cache behalten. Das erstmalige Erstellen einer Instanz eines bestimmten Images ist zeitaufwändig. Die Erstellung der zweiten Instanz dieses Typs dauert dementsprechend kürzer.

Storage Controller (SC) stellt ein im Netzwerk verfügbares Storage Device zur Verfügung. Dieses Blockdevice lässt sich mit *Amazon Elastic Block Storage* vergleichen. Diese Laufwerke lassen sich dynamisch den einzelnen Instanzen hinzufügen. Walrus speichert dieses Block Device.

Walrus erlaubt den Anwendern ihre benötigten Daten innerhalb der Private Cloud persistent zu speichern. Walrus ist kein Dateisystem. Einzelne Objekte werden in sogenannten Bukkits gespeichert. Ein Verschachteln dieser Bukkits ist nicht möglich.

Management Platform ist eine zentrale Oberfläche zur Verwaltung der Private Cloud. Es bietet unter anderem Möglichkeiten zur Administration der Instanzen, des Speichers oder des allgemeinen Monitorings⁴.

Die *External Clouds* verdeutlichen die reibungslose Integration von weiteren Cloud-Betreibern (AWS kompatibel) in die Eucalyptus Infrastruktur.

Der Aufbau von der Eucalyptus Infrastruktur hat einen Nachteil, was die Fehlersuche erheblich einschränkt. Es ist nicht ohne aufwendige Analyse der Log-Dateien herauszufinden, welche Instanz auf welchem NC läuft. Die Entscheidung wird anhand der Verfügbarkeit der Ressourcen gefällt. Der CLC hat keine Information darüber. Dieser Nachteil kommt beim Resource Monitoring erheblich zum Tragen, da nicht an zentraler Stelle wie dem CLC, sondern auf jedem NC getrennt, die Informationen gesammelt und zusammengeführt werden müssen.

⁴Die Implementierung dieser Arbeit könnte dort einen Integrationspunkt in die Eucalyptus Infrastruktur finden.

7.3 Steuerung

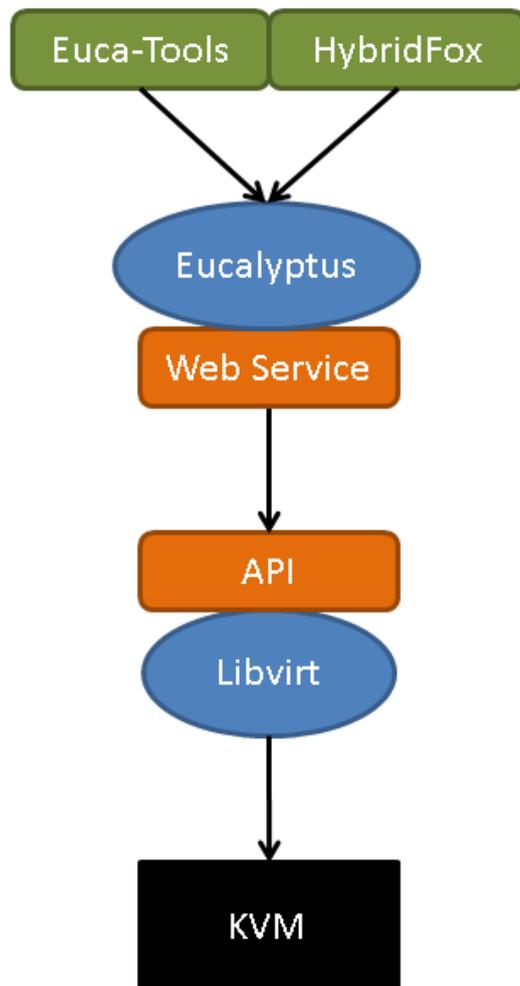


Abbildung 7.2: Eucalyptus wird mittels den Euca2ools gesteuert. Die Webservices sprechen die Libvirt an.

Die Steuerung von Eucalyptus erfolgt über drei verschiedene Methoden:

- Über die Kommandozeile mit *Euca2ools*. Für ausführlichere und genaue Bedienungsanleitungen wird auf [11] hingewiesen.
- Von Eucalyptus bereitgestellte Weboberfläche zur *first-time* Konfiguration und Benutzerverwaltung
- Ein Firefox-Plugin namens HybridFox, mit welchem sich alle Befehle der Kommandozeile auf einer GUI ausführen lassen

Abbildung 7.2 zeigt wie sowohl Euca2ools als auch HybridFox die Cloud-Umgebung steuern. Beide Werkzeuge sprechen die Eucalyptus Webservices an, welche die Libvirt auf den einzelnen NCs ansprechen.

Kommandozeilensteuerung

Aus Gründen der Übersicht beschränkt sich dieser Abschnitt auf wenige Befehle. Falls ein Befehl an dieser Stelle nicht beschrieben sein sollte, wird auf die dazugehörige und ausführliche *man-page* verwiesen.

Für das **Imagemanagement** sind folgende Befehle maßgeblich.

euca-bundle-image bündelt ein Image, teilt es in kleinere Pakete auf und erstellt eine dazugehörige Konfigurationsdatei. Das Image kann in Eucalyptus und auch in AWS verwendet werden.

Rückgängige Aktion: euca-unbundle

euca-upload-bundle lädt das gebündelte Image auf den Cluster Controller. Von dort werden die einzelnen Imagepaketen bei Bedarf auf die jeweiligen NCs kopiert.

Rückgängige Aktion: euca-download-bundle

euca-register registriert das gebündelte und bereits hochgeladene Image. Nach diesem Schritt ist das Image in der Eucalyptus-Umgebung bekannt und kann instantiiert werden.

Rückgängige Aktion: euca-delete-bundle

Die **Steuerung** der einzelnen VMs geschieht mit Hilfe der folgenden Befehle. Die VMs werden aus den mit den vorhergehenden Befehlen registrierten Images gestartet.

euca-describe-instances zeigt die aktuell vorhandenen Instanzen an und gibt Auskunft über deren Status (siehe Abbildung 7.3).

euca-describe-images zeigt die aktuell registrierten Images.

euca-run-instances startet eine Virtuelle Instanz anhand angegebener Images.

euca-terminate-instances beendet eine gestartete Instanz.

Der Zugriff auf die Instanzen ist abhängig vom Betriebssystem. Bei Linux-Distributionen wird mittels Public Key Infrastructure (PKI) über SSH Zugriff auf die Maschinen gewährt. Bei der Erstellung von Windows-Instanzen ist den RDP Verkehr in der Windows-Firewall zu erlauben. Ansonsten ist kein Kontakt zu den Windows-Instanzen möglich.

```

root@scchpblade01a:~# euca-describe-instances
RESERVATION r-49A30766 admin default
INSTANCE i-4CE208C9 emi-15AC1163 141.52.167.101 172.19.1.3 running 0
m1.large 2010-07-25T10:14:24.947Z SCC_Cloud eki-99B01718 eri-E06217F9
RESERVATION r-612A0AE8 admin default
INSTANCE i-52C50838 emi-15AC1163 141.52.167.100 172.19.1.2 running 0
m1.large 2010-07-25T00:30:12.117Z SCC_Cloud eki-99B01718 eri-E06217F9
root@scchpblade01a:~#
    
```

Abbildung 7.3: Der Befehle `euca-describe-instances` zeigt eine Auflistung aller existierender Instanzen.

Benutzermanagement

Eucalyptus bietet eine Weboberfläche zur Benutzerverwaltung innerhalb der Private Cloud. Der Administrator hat die Möglichkeit, Benutzer für die Cloud-Services freizuschalten, und zu regeln, welche Images der jeweilige Benutzer starten darf. Erreicht wird diese Oberfläche unter `https://your.front.end.hostname:8443/`

Abbildung 7.4 zeigt die Startseite der Oberfläche. Die Registerkarten in der Kopfzeile lassen die Konfigurationsmöglichkeiten erkennen. Möglich ist eine Konfiguration der Images und Benutzer, sowie weitere Konfigurationen, wie die Größe der einzelnen Images.

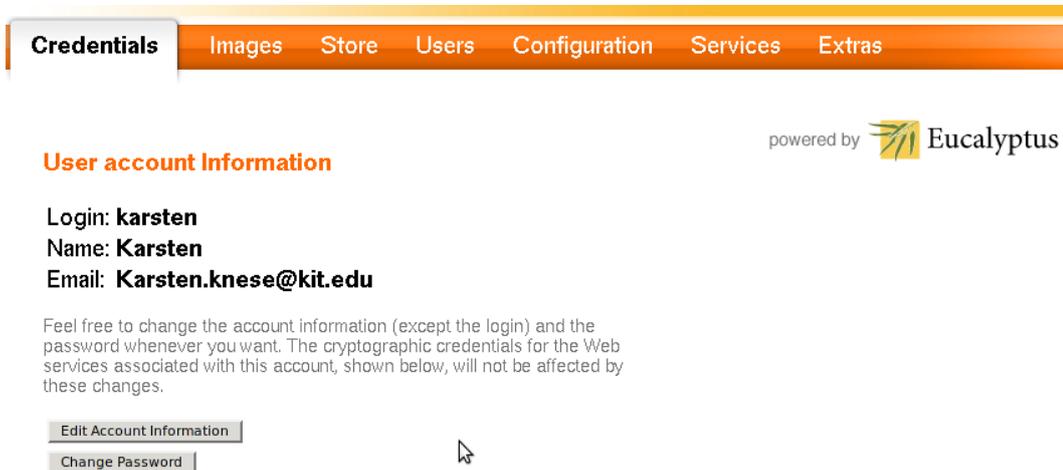


Abbildung 7.4: Startseite der Eucalyptus Weboberfläche

HybridFox

HybridFox ist der Name eines Firefox Plugins zur Steuerung der Cloud Umgebung. Dieses Plugin erlaubt die Steuerung der angegebenen Region der Cloud nach Eingabe der dazugehörigen Zugangsdaten. Der Funktionsumfang von HybridFox ist identisch zu dem der Euca2ools. Beide Werkzeuge benutzen die Dienste von Eucalyptus durch die bereitgestellten WebserVICES.

Abbildung 7.5 zeigt einen Screenshot des Firefox-Plugins. Dabei sind die zum Aufnahmezeitpunkt aktuell laufenden Instanzen innerhalb der Cloud sichtbar.

The screenshot shows a window titled "Your Instances" with a search bar and a checkbox "Don't show Terminated Instances". Below is a table with the following columns: Reservat..., Owner, Instance ID, AMI, Public DNS, Private DNS, Groups, and Local Launch Time.

Reservat...	Owner	Instance ID	AMI	Public DNS	Private DNS	Groups	Local Launch Time
r-49BA0832	admin	i-2F070558	emi-15AC1163	141.52.167.108	172.19.1.34	ssh	2010-06-15 17:55:40
r-49BA0832	admin	i-31E606DA	emi-15AC1163	141.52.167.109	172.19.1.35	ssh	2010-06-15 17:55:40
r-49BA0832	admin	i-3EAB0764	emi-15AC1163	141.52.167.113	172.19.1.39	ssh	2010-06-15 17:55:40
r-49BA0832	admin	i-44430812	emi-15AC1163	141.52.167.110	172.19.1.36	ssh	2010-06-15 17:55:40
r-49BA0832	admin	i-45480836	emi-15AC1163	141.52.167.114	172.19.1.40	ssh	2010-06-15 17:55:40
r-49BA0832	admin	i-4A0707CC	emi-15AC1163	141.52.167.115	172.19.1.41	ssh	2010-06-15 17:55:40
r-49BA0832	admin	i-4A420932	emi-15AC1163	141.52.167.111	172.19.1.37	ssh	2010-06-15 17:55:40
r-49BA0832	admin	i-60900B55	emi-15AC1163	141.52.167.112	172.19.1.38	ssh	2010-06-15 17:55:40
r-348C0682	admin	i-33B5065B	emi-15AC1163	141.52.167.116	172.19.1.10	default	2010-06-16 02:45:10
r-348F0629	karsten	i-4B1E08D8	emi-BC8C13F7	141.52.167.105	172.19.1.66	default, default,...	2010-06-16 08:11:10

Abbildung 7.5: Screenshot des Firefox Plugins HybridFox

Teil III

Evaluation bestehender Systeme

Grundlage eines Accounting-Systems ist die Zuordnung zwischen den Benutzern und der von ihnen verbrauchten Ressourcen des Host-Systems. Die Schwerpunkte dieser Arbeit bestehen auf dem Erfassen der Verbrauchsdaten sowie der Bildung eines passenden Preismodells für diese Daten.

In diesem Teil findet eine Evaluation bestehender Systeme für das Resource Monitoring sowie des Accountings statt. Das Resource Monitoring wird entlang existierenden Programmen und Werkzeugen evaluiert; das Accounting entlang kommerzieller Cloud-Anbieter.

Aus dem daraus folgendem Fazit schließen sich die brauchbaren Ansätze für das Resource Monitoring und Accounting für die Anforderungen der Private Cloud innerhalb des SCC.

RESOURCE MONITORING

Das Abgreifen der Verbrauchsdaten gestaltet sich schwierig, da es sich bei den erstellten Instanzen um virtuelle Maschinen handelt, die nicht exklusiv physikalische Hardware zugewiesen bekommen. Dieses Kapitel untersucht mehrere bestehende Ansätze zum Erfassen der Verbrauchsdaten von Virtuellen Maschinen. Da es für die Cloud-Infrastruktur von Eucalyptus bislang keine Möglichkeit des Resource Monitorings gibt, werden die Ansätze darangehend bewertet, wie praktikabel sich diese beim Umgang mit Eucalyptus erweisen. Die Anforderungen an das Resource Monitoring erstrecken sich auf Werte wie *Gesamtlaufzeit*, *CPU-Zeit*, *Netzwerkverkehr* und *Festplattenzugriffe*. Die Gesamtlaufzeit ist zwingend notwendig für die Accounting-Lösung. Die weiteren Kategorien sind optional.

Die im Folgenden vorgestellten Programme und Werkzeuge werden gegeneinander abgewogen, gemessen an der Flexibilität, Einsatzfähigkeit und Produktivität. Das daraus resultierende Fazit ist Hauptbestandteil der Implementierungsarbeit in Teil IV.

8.1 Libvirt

Aufbauend auf der Bibliothek Libvirt(vgl. Abschnitt 5.3) befinden sich mehrere Werkzeuge zum Umgang mit Virtuellen Maschinen. Die Libvirt bietet eine Funktionalität zum Erfassen von Verbrauchsdaten der Virtuellen Maschinen. Eucalyptus verlangt zwingend die Installation der Bibliothek auf jedem Node Controller, da die Webservices von Eucalyptus nicht direkt mit der zu

Grunde liegenden Virtualisierungslösung kommunizieren, sondern die Libvirt als unabhängige Schnittstelle benutzen. Dies ermöglicht den Einsatz von Eucalyptus mit mehreren Virtualisierungslösungen. Drei Methoden für das Erfassen von Verbrauchsdaten werden in diesem Abschnitt diskutiert.

Ein Nachteil jedes Ansatzes der Libvirt, ist die notwendige Ausführung des Resource Monitorings auf jedem NC. Das Abgreifen der Ressourcen aller Instanzen ist somit abhängig vom physikalischen Ausführungsort und kann nicht zentral für alle Instanzen ausgeführt werden. Dies führt zu einem Daemon pro NC, der die gemessenen Daten an einen zentralen Ort ablegt, wie beispielsweise einer Datenbank.

Als weiterer Nachteil gilt, dass Eucalyptus keine Möglichkeit bietet, Informationen über den Ausführungsort einer Instanz zu sammeln. Das bedeutet, dass für ein Resource Monitoring mit Libvirt alle NCs und der CLC involviert sind.

Auf Grund dieser beiden Einschränkungen wird diese Methode mit folgenden Kriterien entwickelt.

- Es wird ein auf dem NC ausgeführter Daemon entwickelt. Dieser sammelt regelmäßig Statusinformationen der auf dem NC ausgeführten Instanzen.
- Die gesammelten Daten der einzelnen Instanzen müssen mit den Daten aus Eucalyptus (wie Besitzer, Instanzgröße und Erstellungsdatum) abgeglichen werden. Eine Instanz hat dabei genau einen Besitzer. Ein Besitzer kann jedoch mehrere Instanzen besitzen.
- Jeder Daemon muss regelmäßig seine Daten einheitlich an einen zentralen Ort persistent speichern. Dieser Ort dient als Quelle des Abrechnungssystems.
- Zusätzlich zu den beiden Daemon startet an zentraler Stelle eine weiteres Skript, welches innerhalb einer Abrechnungseinheit¹ ausgeführt wird, die aggregierten Daten von dem zentralen Persistenzort aufbereitet, gegebenenfalls grafisch darstellt und den Besitzer über seine Kosten informiert.

¹ Der Zeitraum wird im Fazit dieses Kapitels festgelegt, in dem entschieden wird, welche Anforderungen an das Cloud-Angebot innerhalb des SCCs gestellt werden.

Der letzte Punkt wird nicht exklusiv dieser Methode zugeordnet. In allen Fällen muss der Besitzer die Verbrauchsdaten und seine finanzielle Belastung einsehen können. Diese Aufbereitung kann so aussehen, dass die Benutzer über eine Weboberfläche eine aktuelle Datenübersicht präsentiert bekommen (vergleich mit Online-Banking) oder per E-Mail informiert werden.

Das Resource Monitoring kann mit der Libvirt auf drei Arten geschehen. Unter Zuhilfenahme der angebotenen API kann eine Anwendung, zugeschnitten auf die eigenen Anforderungen, entwickelt werden. Es gibt bereits fertig entwickelte und auch weiterhin gepflegte Applikationen, die den Anforderungen an das Resource Monitoring entsprechen. Zwei dieser Werkzeuge werden evaluiert.

Libvirt API

Die Libvirt stellt eine Vielzahl an Methoden, Structs und Makros zur vollständigen Entwicklung eigener Anwendungen. Der Übersicht wegen, werden an dieser Stelle nur drei Methoden (siehe Listing 8.1) vorgestellt, die ein Messen der Werte für die jeweiligen Ressourcenkategorien ermöglichen.

```

1 int virDomainGetBlockInfo (virDomainPtr domain,
2     const char * path,
3     virDomainBlockInfoPtr info,
4     unsigned int flags)
5 int virDomainInterfaceStats (virDomainPtr dom,
6     const char * path,
7     virDomainInterfaceStatsPtr stats,
8     size_t size)
9 int virDomainGetVcpus (virDomainPtr domain,
10    virVcpuInfoPtr info,
11    int maxinfo,
12    unsigned char * cpumaps,
13    int maplen)
    
```

Listing 8.1: Libvirt Methoden zum Resource Monitoring

Alle Methoden geben Structs zurück, deren Eigenschaften die gewünschten Messdaten beinhalten. Für eine vollständige Auflistung aller Datenty-

pen, Funktionen und Makros wird auf die online API-Referenz verwiesen. Nähere Informationen darüber finden sich unter [19].

Die API stellt eine mächtige Lösung dar, um in C geschriebene Applikationen auf Basis der Libvirt zu entwickeln. Dabei können die Anwendungen, passend auf die eigenen Anforderungen, entwickelt werden.

Virsh

Beschreibung

Virsh (Virt Shell) wurde entwickelt zur konsolenbasierten Steuerung aller Instanzen. Es stellt eine Alternative zur grafischen Implementierung *Virt-manager*. Virsh ist soweit entwickelt, dass es neben den Befehlen zum Starten und Stoppen von Virtuelle Maschinen (VMs) auch Statusinformationen über die jeweiligen Instanzen bereitstellt. Der dazugehörige Parameter heißt `dominfo <id>`. Darüber können folgende Eigenschaften über die mittels der `<id>` adressierten Instanz abgelesen werden:

id Eine von der Libvirt geführte Identifikation zur indexierten Adressierung von Instanzen.

Name Ein vom Ersteller gewählter Name für die Instanz. Der Name dient auch zur Adressierung der Instanz.

UUID Diese UID ist einmalig für jede Instanz. Im Vergleich zur ID kann diese niemals doppelt vorkommen.

OS Type Gibt Auskunft darüber, in welchem Kontext von Virtualisierung die Instanz geführt wird. Dies kann variieren zwischen *para* und *hvm* (= *Full-Virtualization*)

State Diese Eigenschaft gibt Auskunft über den Ausführungszustand. Dort wird ersichtlich, ob die Instanz noch läuft (*running*), pausiert (*paused*) oder gestoppt ist (*shut off*).

CPU Anzahl der virtualisierten Prozessoren

CPU-Time Summe der aktuell verbrauchten CPU-Zeit

Max memory Maximal der VM zugeordneter Arbeitsspeicher

Used memory Aktuell genutzter Arbeitsspeicher. Dies wird bei KVM stets gleich mit Max memory sein.

Weitere Eigenschaften sind an dieser Stelle nicht relevant. Alle Eigenschaften können durch den Befehl `virsh dominfo <id>` erhalten werden.

Programmablauf

Das Vorgehen dieses Ansatzes untergliedert sich in drei Teile:

- Der erste Teil ist das Parsen des Befehls `virsh list --all`, um eine Liste aller auf dem NC vorhandenen Instanzen zu generieren. Anhand dieser Liste wird sukzessive der bereits beschriebene Befehl `virsh dominfo <id>` ausgeführt. Die gewonnenen Daten beschränken sich auf die Laufzeit und CPU-Zeit. Diese Daten werden in einer Datenbank gespeichert. Die Wiederholungsrate des Daemon liegt bei einem Durchlauf pro variablen n Sekunden.
- Zusätzlich zu diesem NC-seitigen Skript läuft ein Skript auf dem CLC. Dies hat lediglich die Aufgabe, regelmäßig auf neu erstellte Instanzen zu achten und diese mit samt Ersteller in die Datenbank einzutragen.
- Laufen diese beiden Skripte, kann mittels einer separaten Anwendung die Aufbereitung und Darstellung der Daten vorgenommen werden. Dies stellt einen Abgleich zwischen den gesammelten Daten aus dem NC-Skript und dem CLC-Skript und ordnet jedem Benutzer die verbrauchten Daten zu. Als Schlüssel für diesen Abgleich dient dabei die Instanz ID.

Virt-Top

Beschreibung

Eine weitere Anwendung auf Basis der Libvirt ist *Virt-Top*. Es erinnert stark an das in den Linux Kernel integrierte Monitoring Tool *top* [18]. Es ist in der

Lage, Statistiken über CPU-Zeiten, Netzwerkverkehr und Festplattenzugriffe der einzelnen Instanzen zu erstellen². Abbildung 8.1 zeigt eine beispielhafte Analyse der CPU innerhalb von Virt-Top:

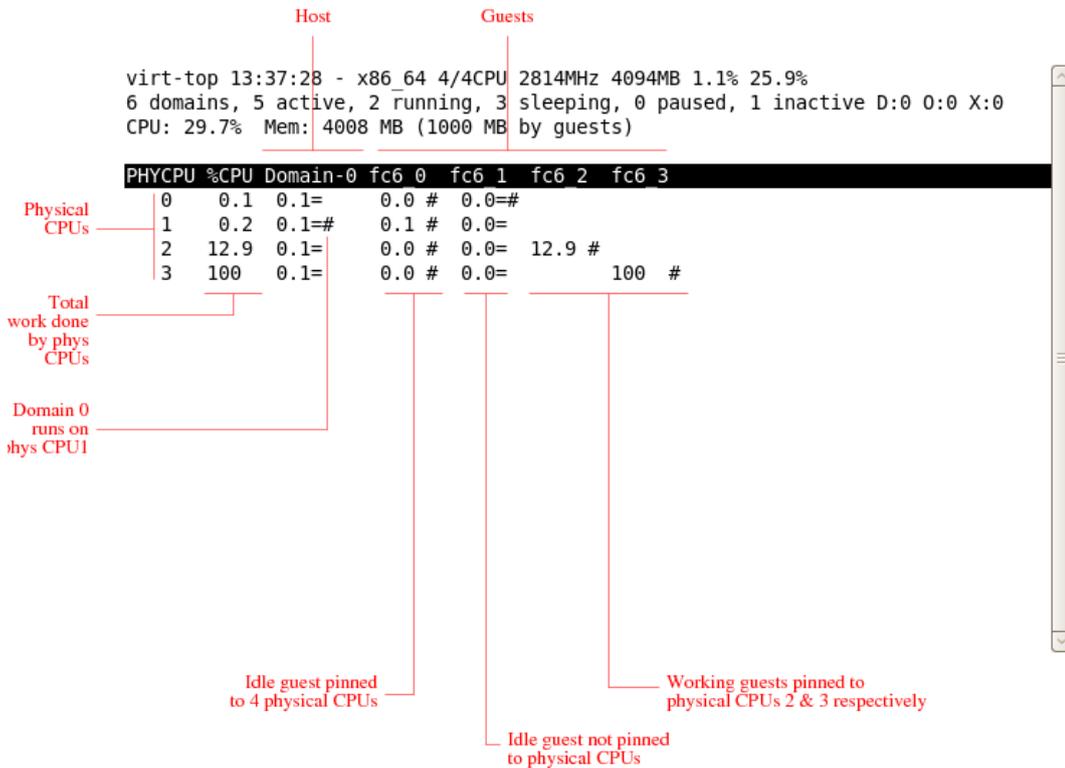


Abbildung 8.1: Beispielhafte Analyse der CPU Auslastung für verschiedene Instanzen [18]

Virt-top wurde wie das ursprüngliche Programm top ebenfalls als Daemon entwickelt. Eine Funktionalität von Virt-Top ist die Exportfunktion der Daten in eine externe Comma Separated Values (CSV)-Datei. Dort können zusätzliche Skripte ansetzen, um die Daten aufzubereiten.

²Die Voraussetzung dafür ist jedoch, dass die Instanzen via Libvirt Tools wie *virsh* oder *virt-manager* erstellt wurden und somit Virt-Top bekannt sind. Instanzen, die direkt über *kvm* Kommandozeilenbefehle erzeugt wurden sind daher nicht messbar mit Virt-Top.

Programmablauf

Der Programmablauf ähnelt dem Ablauf mittels Virsh. Ebenfalls läuft ein Daemon, der auf dem NC regelmäßig Informationen über die laufenden Instanzen sammelt. Anstelle von Virsh wird bei diesem Ansatz die Ausgabe von Virt-Top ausgelesen.

Virt-Top auslesen ist im technischen Sinne nicht ganz korrekt. Es wird initial ein Shell-Skript aufgerufen. Als Parameter enthält es den Zeitintervall, in welchem die Informationen über die Instanzen gesammelt werden. Sämtliche Informationen schreibt Virt-Top in eine parametrierbare CSV-Datei. Schließlich liest der Daemon nicht die Konsolenausgabe von Virt-Top, sondern lediglich die CSV-Datei. Dies hat den Grund, dass es aus technischer Sicht einfacher ist, eine CSV-Datei auszulesen, als regelmäßig die Ausgaben auf der Konsole zu parsen.

Sobald der Inhalt der CSV-Datei analysiert wurde, ist der Vorgang analog zu Virsh zu betrachten. Auch bei dieser Methode müssen fehlende Daten von Eucalyptus - vor allem der Besitzer - von dem CLC erfragt und der Instanz zugeordnet werden. Sind alle Daten gesammelt, werden diese in die zentrale Datenbank gespeichert, aufbereitet und zur weiteren Bearbeitung angeboten.

8.2 Euca2ools

Bei der Analyse der Libvirt-Werkzeuge wurde ersichtlich, dass bei diesen Methoden wichtige Daten, wie z.B. der Besitzer oder die Instanzgröße, fehlen. Für das Erfassen dieser Daten ist der Kommandozeilenbefehl `euca-describe-instances` notwendig. Die Euca2ools sind ausschließlich auf dem CLC installiert.

Ausgeführt als Daemon auf dem CLC kann mit dieser Methode die Gesamtlaufzeit zentral an einer Stelle gemessen werden. Wenn ausschließlich die Gesamtlaufzeit für das Preismodell in Betracht gezogen wird, ist dies ein plausibler Ansatz, da dies auf die Ausführung eines Daemon auf jedem NC verzichtet.

Ein Nachteil ist der Verzicht auf weitere Verbrauchsdaten wie CPU-Zeit, Netzwerkverkehr und Festplattenzugriffe. Von dem CLC ist mittels der Euca2ools kein Zugriff auf die einzelnen Instanzen auf dem NC möglich.

Für weitere Informationen über die Euca2ools, wird auf [11] verwiesen.

8.3 Nagios

Nagios ist ein Werkzeug für das Überwachen von unterschiedlicher Komponenten in einem Intranet. Eine zentrale Stelle, der Nagios-Server, beobachtet alle erreichbaren Komponenten (Client, Server, Router, ...). Der Zustand dieser Komponenten in den Merkmalen wie *Status* und sonstigen Services steht unter Kontrolle. An zentraler Stelle kann ein komplettes Intranet nach Ausfällen von Rechnern oder fehlerhaften Diensten protokolliert werden. Abbildung 8.2 zeigt wie eine Übersicht von Rechnern aussieht.

Host 	Service 	Status 	Last Check 
iwrazubi10.ka.fzk.de	C:\ Drive Space	CRITICAL	07-18-2010 19:44:14
	CPU Load	CRITICAL	07-18-2010 19:41:13
	Explorer	CRITICAL	07-18-2010 19:38:13
	Memory Usage	CRITICAL	07-18-2010 19:41:10
	NSClient++ Version	CRITICAL	07-18-2010 19:42:10
	Uptime	CRITICAL	07-18-2010 19:43:10
	W3SVC	CRITICAL	07-18-2010 19:46:06
localhost	Current Load	OK	07-18-2010 19:46:10
	Current Users	OK	07-18-2010 19:44:54
	HTTP	 OK	07-18-2010 19:41:42
	PING	OK	07-18-2010 19:41:44
	Root Partition	OK	07-18-2010 19:42:40
	SSH	 OK	07-18-2010 19:43:44
	Swap Usage	OK	07-18-2010 19:44:42
	Total Processes	OK	07-18-2010 19:45:37

Abbildung 8.2: Eine Übersicht von zwei Computern. Der obere ist kritisch, da er nicht erreichbar ist. Localhost läuft fehlerfrei.

Nagios ist modular aufgebaut und durch so genannte *Templates* erweiterbar. Zwei Templates zum Überwachen von Virtuellen Maschinen behandeln die beiden folgenden Unterabschnitte.

8.4 Nagios-Virt

Ein Template ist *Nagios-Virt*. Dies ermöglicht eine Überwachung der virtuellen Maschinen auf einem NC. Als Voraussetzung gilt, dass die Instanzen von der Libvirt verwaltet werden. Die direkt von KVM per Kommandozeile erzeugten Instanzen stehen nicht unter der Überwachung von Nagios-Virt.

Nagios-Virt überwacht die virtuellen Maschinen nach den selben Kriterien wie die Hauptanwendung Nagios. Somit können die CPU-Last und Speicherverbrauch erfasst werden. Abbildung 8.3 zeigt eine ähnliche Übersicht wie bereits in Abbildung 8.2 zu sehen ist, jedoch mit dem Unterschied, dass es sich um virtuelle Maschinen handelt.

Der Einsatz von Nagios-Virt setzt eine Installation des Nagios-Servers auf jedem NC voraus. Des Weiteren erfordert Nagios eine Modifikation von Windows-Betriebssystemen. Beim Einsatz innerhalb der Cloud muss jedes Windows-Betriebssystem angepasst werden.

8.5 Nagios-Eucalyptus

Zur Evaluation einer zentrale Monitoring-Lösung wurde ein weiteres Nagios-Template untersucht. Das Skript überwacht die von Eucalyptus ausgeführten Instanzen. Gegenüber Nagios-Virt hat dies den Vorteil, alle Instanzen direkt an einer Stelle zentral verwalten zu können.

Die Evaluierung dieses Skripts schlug jedoch fehl. Gründe dafür waren eine empfohlene Installation von Nagios auf dem CLC, was ein zu hohes Risiko auf der Produktivumgebung darstellte. Zugleich erwies sich das von Eucalyptus angebotene Skript als fehlerhaft.

Virtual hosts (virt-hostgroup)

Host	Status	Services	Actions
centos5qax64fv	UP	<u>1 OK</u>	
debian32fv	UP	<u>1 OK</u>	
f764pv	UP	<u>1 OK</u>	
fc6_0	UP	<u>1 CRITICAL</u>	
fc6_1	UP	<u>1 CRITICAL</u>	
freebsd32fv	UP	<u>1 CRITICAL</u>	
gentoo32fv	UP	<u>1 CRITICAL</u>	
rhel5qax32fv	UP	<u>1 OK</u>	

Abbildung 8.3: Eine Übersicht der virtuellen Maschinen, die auf dem NC ausgeführt werden.

8.6 Fazit

Die verschiedenen Skripte, aufbauend auf der Libvirt wie *Virsh* und *Virt-Top* eignen sich gut zum Resource Monitoring, da kaum Einschränkungen im Programmfluss entstehen. Diese Anwendungen sind frei verfügbar und werden offiziell gepflegt. Die Verarbeitung der erfassten Daten kann in einer beliebigen Programmiersprache (in diesem Fall mit Java) geschehen, was den Einsatz von weiteren Anwendungen wie etwa dem Open Source Projekt Business Intelligence and Reporting Tools (BIRT)³ zur Darstellung der gewonnenen Daten ermöglicht.

Ein Nachteil davon ist der große Aufwand zur Gewinnung der Daten. Dazu gehört das Auslesen der eingesetzten Shell-Skripte und das Zusammenführen der Daten von jedem Node Controller zu einem zentralen Speicherort. Die Umsetzung als Daemon ist vorteilhaft, da auf diese Weise keine zeitli-

³Nähere Informationen dazu siehe [9].

chen Inkonsistenzen entstehen. Der Einsatz von insgesamt sechs Daemon (fünf davon auf den Node Controllern, einen auf dem Cloud Controller) für diese Anforderungen erzeugen einen großen Overhead. Die einmalige Ausführung eines Daemon an zentraler Stelle bietet mehr Performance und weniger Fehlermöglichkeiten.

Ersichtlich ist, dass der auf dem Cloud Controller ausgeführte Befehl `euca-describe-instances` unverzichtbar ist, da sämtliche Libvirt-Tools vollständig getrennt und unabhängig von Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems (Eucalyptus) agieren und sich rein auf die jeweiligen Hypervisor (in diesem Fall KVM) beziehen. Ausschließlich mit diesem Befehl ist das Erfassen der Instanzgröße und des zugeordneten Besitzers möglich. Um eine Analyse zwischen den Benutzern der Cloud und der dazugehörigen Verbrauchsdaten herzustellen, wird eine Mischform aus den Libvirt-Werkzeugen und den Euca2ools implementiert. Dies erlaubt eine vollständige Analyse von sämtlichen Verbrauchsdaten, ermittelt den Benutzer der Cloud und bietet die Möglichkeit einer flexiblen Weiterentwicklung des Accounting-Systems.

Der Einsatz von Nagios-Systemen sieht bei erster Betrachtung als das am besten geeignete Werkzeug aus. Die grafische Darstellung ist implementiert, die Eingliederung in bestehende Nagios-Installation ist möglich und das Erfassen der Ressourcen ist bereitgestellt.

Bei genauerer Betrachtung stellt sich heraus, dass diese Möglichkeiten des Resource Monitorings mit sehr viel Aufwand zur Einarbeitung verbunden sind, einen sehr hohen Overhead durch die komplette Installation von Nagios erzeugen und dadurch an Flexibilität verlieren.

Um die Daten gesammelt an zentraler Stelle zu speichern, müssten auch beim Nagios-Einsatz weitere Skripts ausgeführt werden. Dies führt letztendlich zu einer Entscheidung gegen Nagios. Hinzukommt eine Anpassung der Windows-Systeme, um von Nagios überwacht zu werden. Dies erfordert eine Modifikation eines jeden Windows-Image vor dem Einsatz in der Cloud.

ACCOUNTING

Accounting bedeutet im groben Umfang die Kundenverwaltung. Im Bezug auf das Cloud Computing bedeutet es die Aktivierung eines SLA. Dieser Service-Vertrag bindet unter anderem¹ den Benutzer an die von ihm erstellten Instanzen innerhalb der Private Cloud im SCC. Diese Bindung impliziert eine Bezahlung jener Instanzen.

Es gibt mehrere kommerzielle IaaS Cloud-Anbieter. Eine Wirtschaftlichkeitsbetrachtung wird anhand von vier Cloud-Anbietern vorgenommen, um im darauf folgenden Kapitel als Fazit daraus die Anforderungen, Leistungen und Preise für die Private Cloud innerhalb des Steinbuch Centre for Computing (SCC) vorzunehmen. Die Preismodelle und Angebotsvarianten von jedem Anbieter werden diskutiert.

9.1 Amazon Web Services

Federführend im Bereich von IaaS Angeboten ist Amazon. Amazon hat im Vergleich zu anderen Anbietern das am stärksten ausgereifte Angebot am Markt. Somit wird Amazon sehr ausführlich behandelt. Die weiteren Anbieter werden an dem hier Dargebotenen gemessen und die Unterschiede aufgezeigt.

¹Natürlich entstehen auch für den Dienstleister gewisse Pflichten.

Funktionsweise und Servicemerkmale

Amazons IaaS Angebot ist bequem via Weboberfläche für den Kunden nutzbar. Dabei können Instanzen mit unterschiedlichen Betriebssystemen gestartet, konfiguriert und wieder beendet werden. Durch die freie Netzwerkkonfiguration der Instanz ist diese vollständig in die private Umgebung integrierbar.

elastisch Dabei ist die erstellte Instanz vollkommen *elastisch*. Das heißt, eine Modifikation der bereitgestellten Kapazitäten kann binnen kürzester Zeit durchgeführt werden. Diese Elastizität skaliert bis hin zur Bereitstellung von 1000 Instanzen.

flexibel Amazons Angebot beschränkt sich nicht auf einen Instanztyp und ein dazugehöriges Betriebssystem, sondern der Benutzer hat die Wahl zwischen mehreren Instanztypen (siehe Abschnitt 9.1) mit den Betriebssystemen Linux oder Windows.

kompatibel EC2 ist vollständig modular und arbeitet mit anderen Amazon Web Services zusammen wie zum Beispiel Amazon Simple Storage Service (S3) als persistenter Speicher.

zuverlässig Für jede von Amazon bereitgestellte Region bietet Amazon aktuell eine 99.95% Zuverlässigkeit ihrer Dienste an, was jedoch immerhin auf das gesamte Jahr einen Ausfall von 438 Stunden macht.

sicher Das Thema Sicherheit ist in der Cloud ein heikles Thema². Amazon bietet frei konfigurierbare Firewall-Einstellungen und kann via IPSec ein VPN-Gateway zur Verfügung stellen.

Service Level Agreement

Im Grundlagenteil in Abschnitt 6.1 wurde bereits die Analogie zwischen den Cloud-Anbietern und dem Stromanbieter gezogen. An dieser Stelle wurde der Service Vertrag behandelt. Der Servicevertrag von Amazon beschreibt

²Das Thema Sicherheit in der Cloud ist sehr umfangreich. Daher wird nicht näher auf dieses Thema eingegangen, da es sonst den Rahmen dieser Arbeit sprengen würde.

die Nutzungsbedingungen für Elastic Cloud Compute (EC2) und die weiteren Webservices.

” AWS will use commercially reasonable efforts to make Amazon EC2 available with an Annual Uptime Percentage of at least 99.95% during Service Year. In the event Amazon EC2 does not meet the Annual Uptime Percentage commitment, you will be eligible to receive a Service Credit.”

Das obige Zitat stammt aus dem offiziellen Servicevertrag von Amazon. Dort wird klar dargestellt, welche Leistungen Amazon als Dienstleister bringt, jedoch auch, was im Fehlerfalle passiert. Amazon nennt ebenfalls Absicherungspunkte für das eigene Unternehmen.

Ist eine Wirtschaftlichkeit gegeben, garantiert Amazon eine 99.95 prozentige Ausfallsicherheit pro Jahr. Der Servicevertrag schließt sämtliche externe Fehlerquellen aus. Im Falle der Nicht-Erreichbarkeit einer Instanz auf Grund von Ausfällen des Internetproviders übernimmt Amazon keinen Schadensersatz.

Ausschließlich eine Nicht-Verfügbarkeit von Instanzen, verursacht von Amazon selbst, wird anerkannt und daran gemessen der Schadensersatz in Form einer Gutschrift durchgeführt.

Instanztypen

Amazon EC2 bietet mehrere Typen von Instanzen an. Diese Unterscheidung dieser Typen findet in der Ausstattung der Ressourcenkapazitäten wie Arbeitsspeicher und CPU-Leistung statt. Eine Instanz besitzt eine gewisse von Amazon bereitgestellte Menge an dedizierter Rechenleistung. Amazon rechnet die einzelnen Instanzen stundengenau ab³.

³für genauere Informationen zur Preisbildung wird auf den Abschnitt 9.1 verwiesen

Amazon bietet drei Instanztypen an:

- Standard
- High-Memory
- High-CPU

Die folgende Auflistung diskutiert die Instanztypen. Die Auswahl für den jeweiligen Instanztyp wird von dem Benutzer je nach Bedarf getroffen. Amazon bietet Testinstanzen an, um die Anforderungen an die Instanz zu evaluieren und sich auf dessen Fazit für den passenden Typ zu entscheiden.

Des Weiteren werden die Ein- und Ausgangsleistung (E/A, Netzwerkleistung) in *mittel* und *hoch* kategorisiert. Amazon führt für die Berechnung der CPU-Leistung eine eigens gewählte Einheit ein: *EC2 Compute Unit*. Diese entspricht in etwa der Leistung eines Opteron- beziehungsweise Xeon-Prozessors vom Jahr 2007 mit 1,0 bis 1,2 Gigahertz (GHz). Eingeführt wurde diese Einheit, da sämtliche virtuelle Instanzen auf physischer Hardware basieren. Das bedeutet, dass die zu Grunde liegende Hardware sich im Laufe des Betriebs ändern kann. Um jedoch ein einheitliches Modell zur CPU-Leistung, unabhängig von der Hardware, zu erschaffen, wurde das abstrakte Modell der Compute Unit erstellt.

Die folgende Auflistung erläutert die einzelnen Instanzen [Stand 28.6.2010]:

Standard Instanzen

Komponente	Größe
Arbeitsspeicher	1.7 GB
CPU	1 EC2 Compute Unit
Persistenzspeicher	160 GB
Plattform	32-bit
E/A Leistung	Mittel
API	m1.small

Tabelle 9.1: Small Instance (Standard)

Komponente	Größe
Arbeitsspeicher	7.5 GB
CPU	4 EC2 Compute Unit
Persistenzspeicher	850 GB
Plattform	64-bit
E/A Leistung	Hoch
API	m1.large

Tabelle 9.2: Large Instance

Komponente	Größe
Arbeitsspeicher	157 GB
CPU	8 EC2 Compute Unit
Persistenzspeicher	1690 GB
Plattform	64-bit
E/A Leistung	Hoch
API	m1.xlarge

Tabelle 9.3: Extra Large Instance

High-Memory Instanzen

Komponente	Größe
Arbeitsspeicher	17.1 GB
CPU	6.5 EC2 Compute Unit
Persistenzspeicher	420 GB
Plattform	64-bit
E/A Leistung	Mittel
API	m2.xlarge

Tabelle 9.4: High-Memory Extra Large Instance

Komponente	Größe
Arbeitsspeicher	34.2 GB
CPU	13 EC2 Compute Unit
Persistenzspeicher	850 GB
Plattform	64-bit
E/A Leistung	Hoch
API	m2.2xlarge

Tabelle 9.5: High-Memory Double Extra Large Instance

Komponente	Größe
Arbeitsspeicher	68.4 GB
CPU	26 EC2 Compute Unit
Persistenzspeicher	1690 GB
Plattform	64-bit
E/A Leistung	Hoch
API	m2.4xlarge

Tabelle 9.6: High-Memory Quadruple Extra Large Instance

High-CPU Instanzen

Komponente	Größe
Arbeitsspeicher	1.7 GB
CPU	5 EC2 Compute Unit
Persistenzspeicher	350 GB
Plattform	32-bit
E/A Leistung	Mittel
API	c1.medium

Tabelle 9.7: High-CPU Medium Instance

Komponente	Größe
Arbeitsspeicher	7 GB
CPU	20 EC2 Compute Unit
Persistenzspeicher	1690 GB
Plattform	64-bit
E/A Leistung	Hoch
API	c1.xlarge

Tabelle 9.8: High-CPU Extra Large Instance

Preisgestaltung

Amazon entwickelte unabhängig von den einzelnen Instanztypen ein Modell für verschiedene Finanzierungen. Es kann ausschließlich die direkte Nutzung gezahlt oder eine Instanz über mehrere Jahre hinweg finanziert werden. Es sind aktuell drei Finanzierungsarten verfügbar:

On-Demand Instances Dabei werden komplett frei von Grundkosten ausschließlich die Nutzung der Instanz bezahlt. Solange die Instanz hochgefahren ist und Arbeit verrichtet, fallen Kosten für den Besitzer an. Sobald sie abgeschaltet wird, kostet die Instanz kein Geld.

Die On-Demand Instances Variante ist mit Abstand die häufigst genutzte, da die komplette von Amazon angepriesene Flexibilität verfügbar ist. Es entstehen nur sehr geringe Kosten, die stundengenau abgerechnet werden. Es fallen keine Vorabkosten an und die Instanz ist jederzeit kündbar. Diese Variante wird häufig genutzt, um kurzzeitige Lastspitzen zu überbrücken.

Reserved Instances Bei dieser Art können Instanzen über ein oder drei Jahre reserviert genutzt werden. Es fällt einmalig eine Vorabgebühr zur Reservierung an. Der Stundensatz zum Betrieb der Instanz fällt ist deutlich niedriger.

Reserved Instances ist für Kunden gedacht, deren Aufträge in einer deterministischen Zeit fertigstellbar sind. Dort sind keine Faktoren flexibel und somit über einen bestimmten Zeitraum wie einem oder drei Jahren günstig und mit harten Garantien zu finanzieren.

Spot Instances Dieses Angebot stellt keine harten Garantien zur Ausführung. Amazon legt abhängig von dem Instanztyp und der umgebenen Region einen Spot Price fest. Kunden können selbst entscheiden, bis zu welchem Preis die gewünschten Instanzen gestartet werden sollen. Ist der Spot Price höher als der Maximalwert des Kunden wird die Instanz beendet. Dies hat für Amazon den Charme, nicht genutzte Rechenleistung noch (wenn auch günstiger) zu verkaufen.

Die Spot Instances sind für Kunden gedacht, deren Aufträge mit variablen Start- und Endterminen erledigt werden können. Abhängig von dem Spot Price starten und stoppen die benötigten Instanzen.

Preisliste

Bei Amazon muss ausschließlich die aktive Nutzung bezahlt werden. Bei ausgeschalteten Instanzen fallen keine Gebühren an. Alle folgenden Preise sind Stundenpreise und abhängig von der Region, in der die Instanz ihren Host hat.

Die Preise variieren anhand der Instanztypen, der Größe der Rechenleistung und der Art der Finanzierung. Abbildungen 9.1 bis 9.3 zeigen die Preisliste für das komplette Angebot. Das letzte Bild (Abbildung 9.4) zeigt die Preise für die aus dem Netz transferierten Daten⁴.

US – N. Virginia	US – N. California	EU – Ireland	APAC – Singapore
Standard On-Demand Instances		Linux/UNIX Usage	Windows Usage
Small (Default)		\$0.095 per hour	\$0.12 per hour
Large		\$0.38 per hour	\$0.48 per hour
Extra Large		\$0.76 per hour	\$0.96 per hour
High-Memory On-Demand Instances			
Extra Large		\$0.57 per hour	\$0.62 per hour
Double Extra Large		\$1.34 per hour	\$1.44 per hour
Quadruple Extra Large		\$2.68 per hour	\$2.88 per hour
High-CPU On-Demand Instances			
Medium		\$0.19 per hour	\$0.29 per hour
Extra Large		\$0.76 per hour	\$1.16 per hour

Abbildung 9.1: Pricing: On-Demand Instances

⁴Alle Preise sind vom Stand 28.6.2010 und gültig für die Region EU-Irland. Die Preise für die anderen angebotenen Regionen können abweichen.

US – N. Virginia	US – N. California	EU – Ireland	APAC – Singapore	
One-time Fee				
Standard Reserved Instances	1 yr Term	3 yr Term	Linux/UNIX Usage	Windows Usage
Small (Default)	\$227.50	\$350	\$0.04 per hour	\$0.06 per hour
Large	\$910	\$1400	\$0.16 per hour	\$0.24 per hour
Extra Large	\$1820	\$2800	\$0.32 per hour	\$0.48 per hour
High-Memory Reserved Instances				
Extra Large	\$1325	\$2000	\$0.24 per hour	\$0.32 per hour
Double Extra Large	\$3185	\$4900	\$0.56 per hour	\$0.69 per hour
Quadruple Extra Large	\$6370	\$9800	\$1.12 per hour	\$1.38 per hour
High-CPU Reserved Instances				
Medium	\$455	\$700	\$0.08 per hour	\$0.145 per hour
Extra Large	\$1820	\$2800	\$0.32 per hour	\$0.58 per hour

Abbildung 9.2: Pricing: Reserved Instances

US – N. Virginia	US – N. California	EU – Ireland	APAC – Singapore	
Standard Spot Instances	Linux/UNIX Usage		Windows Usage	
Small (Default)	\$0.04 per hour		\$0.07 per hour	
Large	\$0.158 per hour		\$0.262 per hour	
Extra Large	\$0.333 per hour		\$0.526 per hour	
High-Memory Spot Instances	Linux/UNIX Usage		Windows Usage	
Extra Large	\$0.23 per hour		\$0.331 per hour	
Double Extra Large	\$0.557 per hour		\$0.726 per hour	
Quadruple Extra Large	\$1.133 per hour		\$1.403 per hour	
High-CPU Spot Instances	Linux/UNIX Usage		Windows Usage	
Medium	\$0.078 per hour		\$0.166 per hour	
Extra Large	\$0.328 per hour		\$0.636 per hour	

Abbildung 9.3: Pricing: Spot Instances

Data Transfer In	US & EU Regions	APAC Region
All Data Transfer	Free until Nov 1, 2010 *	Free until Nov 1, 2010 *

Data Transfer Out **	US & EU Regions	APAC Region
First 1 GB per Month	\$0.00 per GB	\$0.00 per GB
Up to 10 TB per Month	\$0.15 per GB	\$0.19 per GB
Next 40 TB per Month	\$0.11 per GB	\$0.15 per GB
Next 100 TB per Month	\$0.09 per GB	\$0.13 per GB
Over 150 TB per Month	\$0.08 per GB	\$0.12 per GB

Abbildung 9.4: Pricing: Internet Data Transfer

9.2 RackSpace

Als weiterer Anbieter von Cloud-Services in Form von Servern (IaaS) ist RackSpace⁵. RackSpace bietet verschiedene Formen der Abrechnung und Instanztypen an. RackSpace liefert eine Unterscheidung zwischen einer stündlichen und monatlichen Abrechnung. Das Angebot erstreckt sich über insgesamt sieben unterschiedliche Instanztypen, das heißt eine Kombination aus Arbeitsspeicher und Festplattengröße. Angaben über CPU-Kapazitäten werden nicht gemacht. Die Bandbreitennutzung wird in Input und Output unterschieden. Die folgende Abbildung 9.5 zeigt das Angebot von RackSpace.

RackSpace bietet ohne Abschluss weiterer Verträge (SLA) ein vollständiges Backup der Instanzen. Des Weiteren beginnen die Instanzen mit deutlich kleineren Kapazitäten im Vergleich zu Amazon. Dies bietet den Vorteil, dass keine unnötig gebrauchten Ressourcen bezahlt werden.

9.3 GoGrid

Der dritte Referenzanbieter von Cloud Services als IaaS ist GoGrid. Abbildung 9.6 zeigt die verschiedenen Größen der angebotenen Serverinstanzen.

⁵www.Rackspacecloud.com

Linux Server Size (RAM / Disk)	Hourly Cost *	Monthly Cost
256 MB / 10 GB	1.5c	\$10.95
512 MB / 20 GB	3c	\$21.90
1024 MB / 40 GB	6c	\$43.80
2048 MB / 80 GB	12c	\$87.60
4096 MB / 160 GB	24c	\$175.20
8192 MB / 320 GB	48c	\$350.40
15872 MB / 620 GB	96c	\$700.80

*Add 3c/hour for Red Hat Enterprise Linux.

Bandwidth Out	22c / GB
Bandwidth In	8c / GB

Abbildung 9.5: Pricing-Angebot von Rackspace

CPU (cores)	RAM (GB)	Storage Space (GB)
0.5	0.5	30
1	1	60
2	2	120
4	4	240
8	8	480

Abbildung 9.6: Die verschiedenen Größen der Instanztypen des Anbieters GoGrid

Die Abrechnung der Serverkapazitäten erfolgt ausschließlich nach dem Arbeitsspeicher. Neben den Instanztypen bietet GoGrid dem Benutzer bereits vordefinierte Finanzierungspläne (siehe Abbildung 9.7). Die Bezahlung erfolgt entweder monatlich oder abhängig vom Verbrauch. Nur der ausgehende Netzwerkverkehr wird in Rechnung gestellt. Als Serviceleistung bietet GoGrid die ersten 10 GB als im Preis enthalten. Jedes weitere Gigabyte wird mit je \$ 0.15 berechnet.

Plan	Monthly Cost	Server RAM Hours	Effective Hourly Cost	Overage Rate*	Effective Server Cost (per 0.5 GB RAM)
Professional Cloud	\$199.00	2500	\$0.08	\$0.09	\$29.90/mo
Business Cloud	\$999.00	14,500	\$0.07	\$0.08	\$25.55/mo
Corporate Cloud	\$3999.00	67,000	\$0.06	\$0.07	\$21.90/mo
Enterprise Cloud	\$9999.00	200,000	\$0.05	\$0.05	\$18.25/mo
Pay As You Go	N/A	N/A	\$0.19	\$0.19	\$69.35/mo

* Server RAM Hour usage beyond any pre-paid plan allotment is billed at the specified overage unit price for the particular plan. For example, usage beyond the 2,500 Server RAM Hours allotted on the Professional Cloud plan is billed at \$0.09 per GB/hr.

Note: All inbound data transfer is free.

Plan	Monthly Cost	Outbound Transfer (GB)	Effective Unit Cost	Overage *
Pay As You Go	N/A	N/A	\$0.29	N/A
Transfer 500 GB	\$99	500	\$0.20	\$0.29
Transfer 3.6 TB	\$499	3,600	\$0.14	\$0.20
Transfer 20 TB	\$1999	20,000	\$0.10	\$0.14
Transfer 57 TB	\$3999	57,000	\$0.07	\$0.07

Abbildung 9.7: Das Abrechnungssystem von GoGrid

9.4 Zimory

Als einen weiteren Anbieter ist *Zimory* zu nennen, die als anfängliche Tochterfirma von der deutschen Telekom, IaaS anbietet. Leider konnten im Verlauf dieser Arbeit kaum Informationen zum Vergleich des Angebots gesammelt werden.

Zimory erfasst vier Merkmale für ihre Accounting-Strategie:

- Laufzeit
- Speicher
- Netzwerkverkehr eingehend
- Netzwerkverkehr ausgehend

Abgerechnet wird nach dem Prinzip *pay as you use*. Berechnet wird ausschließlich der Verbrauch der tatsächlich benutzten Ressourcen. Der Preis für die Laufzeit pro Stunde erfolgt nach Höhe des Arbeitsspeichers, der Zuverlässigkeit der Instanz und dem Ausführungsort.

Neben Amazon bietet auch Zimory eine abstrakte Einheit um die heterogenen Hardwareumgebungen zu abstrahieren.

9.5 Fazit

Amazon bietet ein weitreichendes Angebot; von dem privaten Laien, der die Spot-Instanzen bestellt über kleine Firmen, die ihre Lastspitzen in der Leistungsanforderung on-demand befriedigen, bis hin zur B2B-Lösung für die Reserved-instances.

Die drei anderen Anbieter, die hier diskutiert wurden, haben interessante Aspekte, was das Accounting und die Preisbildung angeht. Sodass zum Beispiel bei diesen Anbietern die CPU keine wesentliche Rolle spielt, sondern lediglich der für jede Instanz reservierte Arbeits- und Persistenzspeicher. Dass die CPU vernachlässigt wird, kann auf den Mangel einer künstliche Referenzeinheit (vgl. Amazon Compute Unit) zurückzuführen sein. Der Aspekt von RackSpace, ein Angebot von deutlich kleinere Instanzen, ist ein wichtiger Aspekt. Das Größenverhältnis der einzelnen Instanzen ist innerhalb von Eucalyptus variabel konfigurierbar. Eine exakte Größe für die einzelnen Typen muss abhängig von dem Angebot an Ressourcen gebildet werden.

Im Rahmen dieser Arbeit wird das Accounting stark an Amazon angelehnt. Besonders im Hinblick darauf, was die Instanztypen angeht. Das Preismodell muss für das SCC angepasst werden, da es sich bei dieser Arbeit um das Angebot einer Private Cloud handelt, deren Services innerhalb des Unternehmens genutzt werden und maximal sich selbst tragen soll. Das heißt, mit dieser Arbeit soll keine Gewinnwirtschaft betrieben werden.

Das Preismodell ist abhängig von allen gemessenen Verbrauchsdaten wie CPU-Zeit, Netzwerkverkehr und Festplattenzugriffe. Da für diese Ausarbeitung versucht wird, möglichst viele Informationen zu sammeln, anstelle von

Pauschalbeträge anzubieten, fiel eine Entscheidung gegen die Modelle von RackSpace und GoGrid. Ein genaues Preismodell wird im nächsten Kapitel eingeleitet.

Des Weiteren soll an dieser Stelle der Aspekt der Compute Unit, wie sie bei Amazon und Zimory eingeführt wurde, nochmals aufgegriffen werden. Es ist ein essentieller Punkt, da es sich bei den physikalischen Ressourcen der Cloud um heterogene Umgebungen handeln kann. Eine solch fiktive Einheit zieht einen Vergleich zu einer Referenzmaschine. Eine Instanz kann nun auf unterschiedlicher Hardware erstellt werden, solange die zugewiesenen Ressourcen die Anforderungen der Referenzmaschine erfüllen.

Da es sich bei der Umgebung für diese Arbeit um eine absolut homogene handelt, ist dieser Schritt in der ersten Implementierung vernachlässigbar. Für eine spätere Weiterentwicklung sollte dieser Punkt jedoch auf alle Fälle in Betracht gezogen werden.

ANFORDERUNGEN FÜR DAS KIT

Die Erstellung einer Accounting-Lösung für das KIT innerhalb einer Private Cloud Umgebung mit Eucalyptus basiert auf den Ergebnissen der Evaluation der vorhergehenden Kapitel. Die Vorteile der einzelnen Ansatzes werden kombiniert und zugeschnitten auf die eigenen Bedürfnisse angewandt. Das KIT wird von öffentlicher Hand bezahlt und soll sich selbst tragen, jedoch keine Gewinnwirtschaft betreiben. Die Anforderungen erstrecken sich auf die Ausschöpfung des Möglichen, nicht auf die größtmögliche Ausbeute durch die Preisbildung.

Das Resource Monitoring stützt sich auf die Anwendung Virt-Top in Kombination mit dem Befehl `euca-describe-instances`. Das Accounting lehnt sich stark an den Ansatz von Amazon an, da die verwendete Cloud-Software Eucalyptus den AWS sehr stark ähnelt. Die Lösungen für das Resource Monitoring sowie das Accounting werden in den beiden folgenden Abschnitten näher diskutiert.

10.1 Resource Monitoring

Das Resource Monitoring wird auf Basis der Libvirt mittels Virt-Top gelöst. Die Gründe für den Einsatz sind sowohl die einfache Weiterverarbeitung der von der Anwendung erzeugten Daten, als auch ein Maximum an gewonnenen Verbrauchsdaten, die von der Anwendung selbst, bereits aufbereitet werden.

Virt-Top liefert alle geforderten Messdaten wie CPU-Zeit, Netzwerkverkehr und Festplattenzugriffe. Für diesen Einsatz müssen mehrere Vorbereitungen getroffen werden:

CSV Virt-top muss die Ausgabe in eine CSV-Datei schreiben, um zur Weiterverarbeitung zur Verfügung zu stehen. Der ausgeführte Befehl dafür ist `virt-top --csv output.csv`.

Kopiervorgang Die Ausführung des Befehls wird in einem Shell-Skript gekapselt, welches in einem parametrisierten Zeitintervall ausgeführt wird. In jedem Zyklus kopiert das Skript erzeugte CSV-Datei. Dies ist notwendig, um den Arbeitsvorgang der Anwendung nicht zu unterbrechen und die Daten abgekoppelt von Virt-Top für die Weiterverarbeitung bereitzustellen.

Euca2ools Das Fehlen der Daten seitens Eucalyptus erfordert den Einsatz des Befehls `euca-describe-instances`, um die restlichen Informationen über die laufenden Instanzen zu erfassen. Dort werden Daten wie den Instanztyp und -größe, sowie der Benutzer der Instanz hinzugefügt.

Die Entwicklung einer eigenen Anwendung auf Basis der Libvirt API ist möglich, verhindert gegebenenfalls das Auslesen der erzeugten CSV-Datei, jedoch steht der Aufwand der Einarbeitung in die API in keinem Verhältnis zu dem einfachen Einsatz eines Java-basierenden CSV-Readers¹.

Das Tool Virsh erwies sich zunächst als praktikabel, da regelmäßig die Befehle `virsh list --all` und mit dessen Ausgabe `virsh dominfo <id>` aufgerufen werden musste. Dieser Ansatz erfordert die Behandlung von nur einer Schleife und der Einsatz eines kapselnden Shell-Skriptes wurde obsolet. Das K.O.-Kriterium für diesen Ansatz ist, dass nur die CPU Informationen erfassbar sind. Informationen über Netzwerkverkehr oder Festplattenzugriffe werden ignoriert.

Der Einsatz von Nagios-Systemen, auf dem CLC sowohl als auf den einzelnen NCs schied aus Aufwands- und Erhaltungsgründen aus. Die Installation

¹ Trotz dass ein Filehandler erzeugt werden muss, geschehen dadurch keine Geschwindigkeitseinbußen, da bei der Implementierung ein Cache erzeugt wird. Dieser Cache ist dafür zuständig, dass das Resource Monitoring ohne Unterbrechung und völlig getrennt von der Verarbeitung der Daten weiterarbeiten kann.

und die damit verbundene Instandhaltung von Nagios ist zu aufwändig. Die Daten werden zwar grafisch aufbereitet, bleiben jedoch in Nagios gekapselt. Eine modulare Flexibilität wie das Halten der Daten innerhalb einer zentral geführten Datenbank ist nicht ohne Weiteres möglich. Das auf der Eucalyptus angebotene Nagios Template ist nicht funktionsfähig.

Für den ersten Implementierungsansatz wurden die nachstehenden Kriterien festgelegt:

CPU Die CPU-Ressource wird als Summe der aktiven CPU-Zeit gemessen. Aktiv heißt, dass effektiv nur die Zeit gemessen wird, in der die CPU direkt der VM zugewiesen ist. Die Zeit, in der die VM im Leerlaufmodus läuft, wird nicht gemessen.

Netzwerkverkehr Der gesamte Netzwerkverkehr wird unterteilt in *gesendet* und *empfangen*. Gemessen wird diese Ressource in Byte.

Festplattenzugriffe Analog zu dem Netzwerkverkehr werden die Festplattenzugriffe unterteilt in *geschrieben* und *gelesen*. Diese Ressource wird in Byte gemessen.

Laufzeit Mit jedem Messdatum wird der aktuelle Ausführungsstatus (laufend, pausiert oder terminiert) der Instanz und der aktuelle Zeitstempel gespeichert. Auf Grund dieser Daten ergibt sich die Gesamtlaufzeit als Differenz zwischen dem ersten und letzten Zeitstempel.

Besitzer Durch das Parsen der Euca2ools wird der Benutzer der Instanz ermittelt. Anhand von zugeordneten Kontaktdaten informiert das System den Benutzer über seine aktuelle Abrechnung.

Instanzgröße Die Euca2ools erfassen den Instanztyp gespeichert. Dieses Datum ist notwendig, um die Preisbildung für die unterschiedlichen Instanzgrößen zu differenzieren.

Der variable Zeitintervall legt die Häufigkeit der Messung der Verbrauchsdaten und die daraus entstehende Datenmenge fest. Sämtliche ausgeführten Skripte sind parametrierbar mit dem Zeitintervall. Der genaue Umgang, der je nach Intervallgröße anfallenden Datenmenge, ist dabei Implementierungsdetail und wird in dem Teil IV näher beschrieben.

10.2 Accounting

In diesem Abschnitt wird die Berechnungsgrundlage für eine erste Implementierung der Accounting-Lösung in Eucalyptus geschaffen. Als Anmerkung sei erwähnt, dass es sich um eine prototypische Auseinandersetzung handelt. Mehrere Faktoren werden in die Preisbildung miteinbezogen. Für eine Produktivschaltung des Systems muss eine genaue Festlegung der Preise von den zuständigen Finanzexperten geschehen.

Preisbildung

Die Bildung eines Preises für Dienstleistungen innerhalb einer Landesanstalt wie dem KIT gestaltet sich als schwierig, da keine Gewinnwirtschaft betrieben werden soll. Ein exakter Preis kommt an dieser Stelle nicht zu Stande. Vielmehr wird eine Diskussion erbracht, welche Faktoren für die Preisbildung notwendig sind. Diese Faktoren müssen zuständige Finanzexperten konkret werten, um einen exakten Preis zu bilden.

Für die Berechnung der Instanzen werden folgende Faktoren in Betracht gezogen, um eine effiziente Abrechnung aller Verbrauchs- und Anschaffungskosten in Abhängigkeit der Instanzgröße zu ermöglichen:

Instanzgröße Abhängig von dem gewählten Instanztyp besitzt die Instanz unterschiedliche Leistung. Da eine Instanz mit doppelter Leistung, doppelte Kosten produziert ², kann der Instanztyp als einfacher numerischer Faktor hinzugenommen werden. Für jede Konfiguration ist der Faktor an einem Referenztyp (zum Beispiel der kleinsten Instanz) zu bestimmen.

Einheit:

Numeric

²Unter der Voraussetzung, dass es sich um homogene Hardware handelt und die Auslastung gleich ist.

Anschaffungskosten Diese Komponente der Preisbildung ist als Grundgebühr zu betrachten. Die Instandhaltung der Hardwareressourcen ist nicht zu vernachlässigen, da dies Teil der finanziellen Belastung für den Cloud-Betreiber darstellt.

Einheit:

$$\frac{\text{€}}{\text{Instanzgrösse}}$$

CPU-Faktor Die Kosten für die CPU sind aus dem Produkt des CPU-Faktors und der CPU-Zeit zu bilden.

Einheit:

$$\frac{\text{€}}{\text{s}}$$

Net-Faktor Der anfallende Netzwerkverkehr berechnet sich analog zur CPU durch einen Net-Faktor. Die Kosten sind abhängig von dem verbrauchten Verkehr (in Byte) zu berechnen. Dieser Faktor kann in weiterer Gestaltung in ausgehenden und eingehenden Netzwerkverkehr unterschieden werden.

Einheit:

$$\frac{\text{€}}{\text{Byte}}$$

DiskIO-Faktor Analog zu dem Netzwerkverkehr wird auch für die Festplattenzugriffe ein fiktiver Faktor eingeführt, der in Abhängigkeit der Zugriffe (in Byte) die Kosten berechnet. Dieser Faktor kann in weiterer Gestaltung in lesenden und schreibenden Zugriff unterschieden werden.

Einheit:

$$\frac{\text{€}}{\text{Byte}}$$

Für die möglichst flexible Gestaltung des Preises wird zunächst eine Unterscheidung zwischen Grund- und Nutzungsgebühr stattfinden. Dies äußert sich in einem Pauschalbetrag für die Nutzungsdauer abhängig von

der Instanzgröße oder detailliert auf jede Verbrauchseinheit abgerechnet. Daraus ergibt sich die folgende Gleichung:

$$\begin{aligned}
 & \text{Grundgebuehr} && + \text{Verbrauchsdaten} \\
 & [\text{Instanzgroesse} * \text{Anschaffungskosten}] && + [\text{CPU - Faktor} * \text{CPU - Zeit} \\
 & && + \text{Net - Faktor} * \text{Net - Byte} \\
 & && + \text{Disk - Faktor} * \text{Disk - Byte}]
 \end{aligned}$$

Die Abrechnung der Instanzen, angelehnt an diese Formel, bietet eine flexible Preisgestaltung. Diese Formel ermöglicht folgende Abrechnungen:

Grundgebühr Für eine Pauschalabrechnung der Grundgebühr müssen die einzelnen Faktoren der Verbrauchsdaten auf 0 gesetzt werden. Die beeinflussenden Faktoren für die Rechnung sind die Instanzgröße im Verhältnis zum Anschaffungspreis der Host-Hardware.

Verbrauchsdaten Für eine Abrechnung der Verbrauchsdaten wird die Grundgebühr ignoriert. Dazu muss der Faktor der Anschaffungskosten auf 0 gesetzt werden. Weiter kann eine Abrechnung gezielt auf ausgewählte Ressourcen erfolgen, indem die Faktoren der nicht gewünschten Ressourcen auf 0 gesetzt werden.

Grundgebühr + Verbrauchsdaten Die Formel ermöglicht eine Kombination beider genannten Abrechnungsmodelle. Bei der Kombination steht weiterhin eine Filterung auf bestimmte Ressourcen zur Verfügung.

Um diese Formel mit konkreten Zahlen zu füllen, wurde eine Konfigurationsoberfläche erstellt. Diese erlaubt den zuständigen Experten, ohne großen Aufwand diese Formel zu bedienen. Eine Erläuterung befindet sich in Abschnitt 14.

Die aktuelle Konfiguration der Instanztypen zeigt Abbildung 10.1. Diese Konfiguration erzwingt eine Unterscheidung zwischen den einzelnen Instanzen. Eine stark ausgestattete Instanz muss im Vergleich zu einer leistungsschwächeren Instanz proportional mehr Kosten verursachen.

VM Types:

Name	CPUs	Memory (MB)	Disk (GB)
m1.small	1	384	4
c1.medium	1	768	6
m1.large	1	1280	10
m1.xlarge	1	2048	16
c1.xlarge	2	2048	16

Abbildung 10.1: Pricing: On-Demand Instances

Die Hardwareressourcen des physikalischen Hosts (NC) beinhalten insgesamt 8 CPU-Cores und 16 GB RAM. Man kann annehmen die Anzahl der VMs hänge ausschließlich von dem verfügbaren Arbeitsspeicher ab, da 8 Kerne mit je 2.33 GHz ausreichend für den Betrieb der Instanzen sind. Daraus ergibt sich folgende Herleitung für die Gesamtanzahl der Instanzen pro NC und Instanz:

Instanztyp	Arbeitsspeicher [MB]
Gesamtpeicher	16.000
m1.small	384
c1.medium	768
m1.medium	1.280
m1.xlarge	2.048
c1.xlarge	2.048

Tabelle 10.1: Small Instance (Standard)

$\frac{\text{Gesamtspeicher}}{m1.small}$	$\approx 41,6667$	= 41 Instanzen (Typ <i>m1.small</i>)
$\frac{\text{Gesamtspeicher}}{c1.medium}$	$\approx 20,8333$	= 20 Instanzen (Typ <i>c1.medium</i>)
$\frac{\text{Gesamtspeicher}}{m1.medium}$	$\approx 12,5$	= 12 Instanzen (Typ <i>m1.medium</i>)
$\frac{\text{Gesamtspeicher}}{m1.xlarge}$	$\approx 7,8125$	= 7 Instanzen (Typ <i>m1.xlarge</i>)
$\frac{\text{Gesamtspeicher}}{c1.xlarge}$	$\approx 7,8125$	= 7 Instanzen (Typ <i>c1.xlarge</i>)

Die Instanztypen sind für unterschiedliche Zwecke ausgerichtet. Kein Typ objektiv betrachtet einer anderen vorzuziehen. Daraus folgt eine gleichmäßige Verteilung der Instanztypen, was den Einsatz eines arithmetischen Mittels zur Bestimmung der durchschnittlichen Gesamtanzahl an Instanzen pro NC unter voller Auslastung des Arbeitsspeichers ermöglicht.

Unter der Annahme, dass zwei Instanzen für den Betrieb des physikalischen Hosts abgezogen werden können, macht das eine Gesamtanzahl von 15 Instanzen pro NC.

$$\frac{(41 + 20 + 12 + 7 + 7)}{5} - 2 \approx 15,4 = 15 \text{ Stück}$$

Dieses arithmetische Mittel ist für die Bestimmung von exakten Werten zwingend notwendig. Alle anfallenden Anschaffungskosten, sowie Betriebskosten werden auf die durchschnittliche Anzahl an Instanzen umgelegt.

Für eine genaue Kalkulation der einzelnen Faktoren für die Verbrauchsdaten, sowie der Anteile für die Anschaffungskosten, bedarf es genauerer Analysen der Rechnungen und Stromkostenabrechnungen. Da diese nicht ohne Weiteres ersichtlich sind, wurde diese Abrechnung flexibel genug gestaltet, um von qualifizierteren Personen angepasst zu werden.

Eine bereits sehr strukturierte Abrechnung der Instandhaltungskosten für das GridKA-Umfeld, welches aus ähnlicher Hardwarekonfiguration besteht, befindet sich in [5]. Dieses Dokument ist nicht darauf ausgelegt einen konkreten Preis für die Anforderungen der Cloud-Services innerhalb des SCC

zu bilden. Innerhalb dessen erfolgt jedoch eine genaue Aufzählung unterschiedlicher Faktoren, die anfallende Kosten verursachen.

Für die Anschaffungskosten gelten folgende Faktoren:

- Hardwarekosten für Server
- Hardwarekosten für Router (allg. Netzwerkinfrastruktur)
- Mietkosten für Inter- bzw. Intranetverbindungen
- Personalkosten

Diese Kosten verursachende Faktoren betreffen die physikalische Instandhaltung der Cloud-Services. Der Anschaffungsfaktor der Grundgebühr kann anteilig im Bezug auf die durchschnittliche Auslastung des Hosts (15 Stück pro NC) durch die virtuellen Instanzen berechnet werden:

$$\begin{aligned} \text{Hardwarekosten}(\text{Gesamt}) &= \text{Hardwarekosten}(\text{Server}) \\ &+ \text{Hardwarekosten}(\text{Netzwerk}) \\ &+ \text{Mietkosten}(\text{Netzwerk}) \\ &+ \text{Personalkosten} \end{aligned}$$

$$\frac{\text{Hardwarekosten}(\text{Gesamt})}{15} = \text{Faktor}(\text{Anschaffungskosten})$$

Die Faktoren der einzelnen Ressourcen beeinflusst folgende Auflistung:

- Stromkosten für Server
- Stromkosten für Storage
- Stromkosten für Netzwerkinfrastruktur
- Ausfallrate von Speichergeräten

Wie diese einzelnen Ressourcenfaktoren konkret gewertet werden, liegt im Aufgabengebiet der Finanzexperten.

Teil IV

Implementierung

Die Implementierung stützt sich auf die Libvirt Anwendung Virt-Top. Angewandt wird diese in einem Daemon, welcher auf jedem Node Controller ausgeführt wird. Da Eucalyptus zwingend die Libvirt auf jedem NC fordert, kann es ohne großen Overhead installiert und genutzt werden.

Die Implementierung ist in der Hochsprache *java* umgesetzt. Java wurde gewählt, da es durch seine plattformunabhängige Architektur ohne Einschränkungen auf dem im SCC eingesetzten Linux-Cluster ausgeführt werden kann. Lediglich das Sun-JDK muss als Grundvoraussetzung für die Inbetriebnahme auf jedem der Knoten installiert sein. Unter Ubuntu lassen sich die folgenden Java-Pakete mit dem Befehl `apt-get install` installieren.

- `sun-java6-jdk`
- `sun-java6-source`
- `sun-java6-demo`
- `sun-java6-fonts`

Werden die Pakete nicht gefunden, fehlen die dazu gehörigen Einträge in der Datei `/etc/apt/sources.list`. Diese muss um folgende Einträge ergänzt werden.

```

1 deb http://archive.canonical.com/ lucid partner
2 deb-src http://archive.canonical.com/ lucid partner
    
```

Listing 10.1: Eintrag der Sourcen für die Installation von Java

Mit `apt-get update` sind die neuen Softwarequellen bereitgestellt. Anschließend können mit dem `apt-get install` Befehl die oben aufgezählten Pakete installiert werden.

Das komplette Projekt ist in drei Komponenten untergliedert. Sie wurden auf die drei Bereiche Cloud Controller, Node Controller und einer externen Datenbank verteilt. Die NC Komponente ist dabei die zentrale Komponente, die in ihrem Programmfluss an notwendiger Stelle die anderen Komponenten anspricht und bedient.

NC Die NC-Komponente ist ein Daemon, der regelmäßig Informationen über die Verbrauchsdaten der einzelnen Instanzen unter Zuhilfenahme von Virt-Top ermittelt. Diese werden nach dem Sammeln in einer passenden Datenstruktur, einer sogenannten Hashmap, im Speicher behalten. Über eine SSH Verbindung wird die CLC Komponente hinzugefügt, um die fehlenden Daten zu sammeln. Sind alle Daten komplett, werden diese als Datensatz in die Datenbank eingetragen. Ist eine Instanz beendet, wird der entsprechende Benutzer davon informiert und zur Abrechnung gebeten.

CLC Die CLC Komponente erfasst Daten über die erstellten Instanzen seitens Eucalyptus. Die vorhanden Daten in der Hashmap werden durch Benutzer, Instanztyp und Erstellungsdatum ergänzt. Dieser Schritt ist notwendig, da auf der NC-Seite keine Informationen darüber erfasst werden können.

Datenbank Eine zentrale Datenbank stellt einen externen Zugriffspunkt auf die erfassten Verbrauchsdaten. Alle Daten, die innerhalb eines Zyklus des NC-Daemon gemessen werden, werden dort persistent gespeichert. Gleichzeitig bietet sie eine Schnittstelle, um mittels weiteren

Werkzeugen Informationen über die Daten aufzubereiten und darzustellen. Bei Speichern erfolgt keine Aufbereitung der gemessenen Daten. Sämtliche Aggregationsfunktionen oder andere Aufbereitungen der Daten erfolgen abhängig von der jeweiligen externen Anwendung zur Darstellung der Verbrauchsdaten.

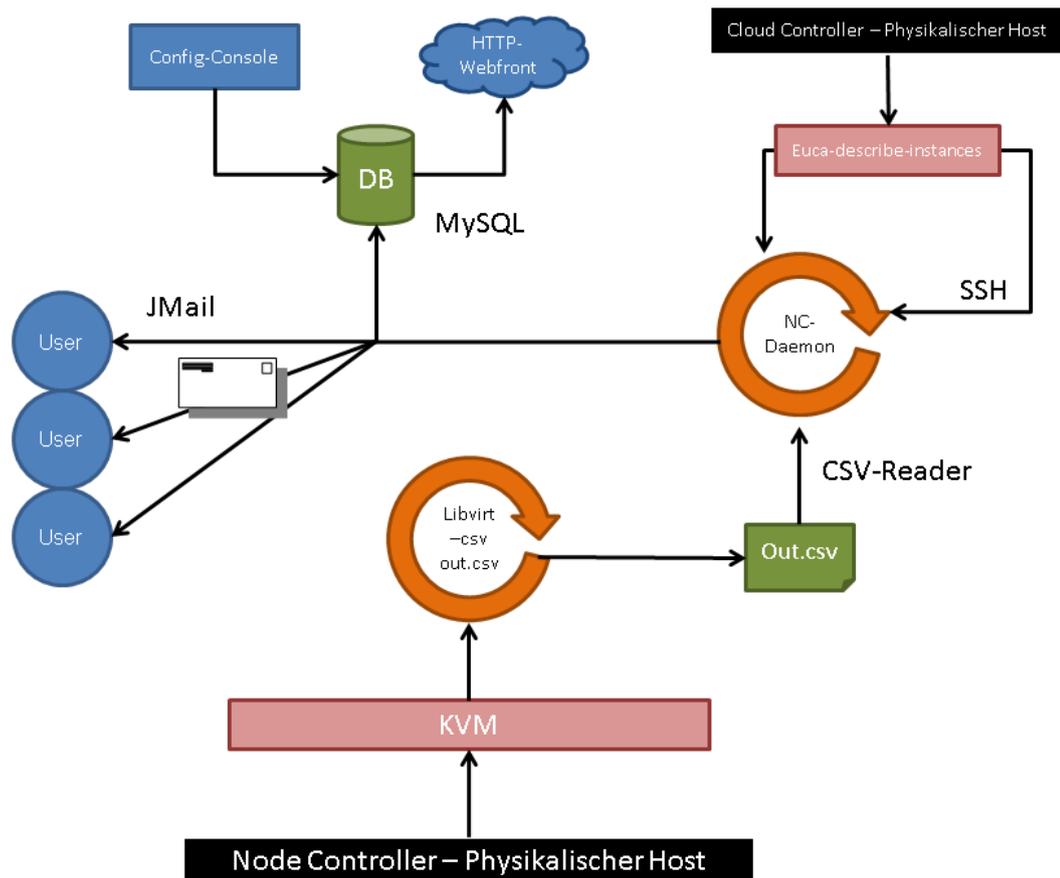


Abbildung 10.2: Gesamtaufbau der Implementierung

Abbildung 10.2 zeigt den kompletten Aufbau der Implementierung. Dort wird das Zusammenspiel der Komponenten ersichtlich. Initial wird ein Skript gestartet, welches auf dem NC Virt-Top ausführt und so parametrisiert, dass die gemessenen Daten in eine CSV-Datei geschrieben werden. Diese CSV-Datei wird von dem NC-Daemon ausgelesen und analysiert. Ergänzt werden diese Daten durch die Informationen des Befehls `euca-describe-instances`. Um diese Daten zu gewinnen, wird ein SSH-Tunnel aufgebaut. Die vollständigen Daten werden in der Datenbank dau-

erhaft gespeichert. Dieser Daten können sich weitere Anwendungen bedienen, um auf geeignete Weise eine Analyse der Daten anzubieten. Innerhalb des NC-Daemons wird in einem weiteren Schritt festgestellt, welche Instanzen beendet wurden. Die Besitzer der beendeten Instanzen werden per E-Mail darüber informiert. Die folgenden Kapitel behandeln diesen Ablauf genauer.

NC-DAEMON

Der entwickelte NC-Daemon ist die zentrale Steuerkomponente, was das Resource Monitoring und das damit verbundene Benachrichtigen der Benutzer über terminierte Instanzen betrifft. Ausgehend von diesem Daemon werden die Ressourcen gemessen, Informationen über Instanzen ergänzt und überprüft, ob eine Instanz terminierte. Bei einer Terminierung wird der Besitzer per E-Mail darüber benachrichtigt.

11.1 Programmbeschreibung

Ein Daemon läuft per Definition im Hintergrund als Dienst ohne grafische Oberfläche in einer Dauerschleife. Innerhalb dieser Schleife prüft der NC-Daemon regelmäßig den Status aller auf dem NC vorhandenen Instanzen. Abbildung 11.1 zeigt das Ablaufdiagramm für den NC-Daemon. Die nachstehenden Abschnitte behandeln die einzelnen Schritte.

11.2 Der Einleitungsprozess

In einem initialen Schritt wird ein eigens implementiertes Shell-Skript gestartet. Dies ist ein weiterer Daemon, welcher in einem Zehn-Sekundenintervall alle Instanzen nach Status und aktuellen Verbrauchsdaten abfragt. Kern des Shell-Skripts ist der in Listing 11.1 gezeigte Befehl.

```
1 Virt-Top --csv output.csv --script -b -d 10 &
```

Listing 11.1: Virt-Top

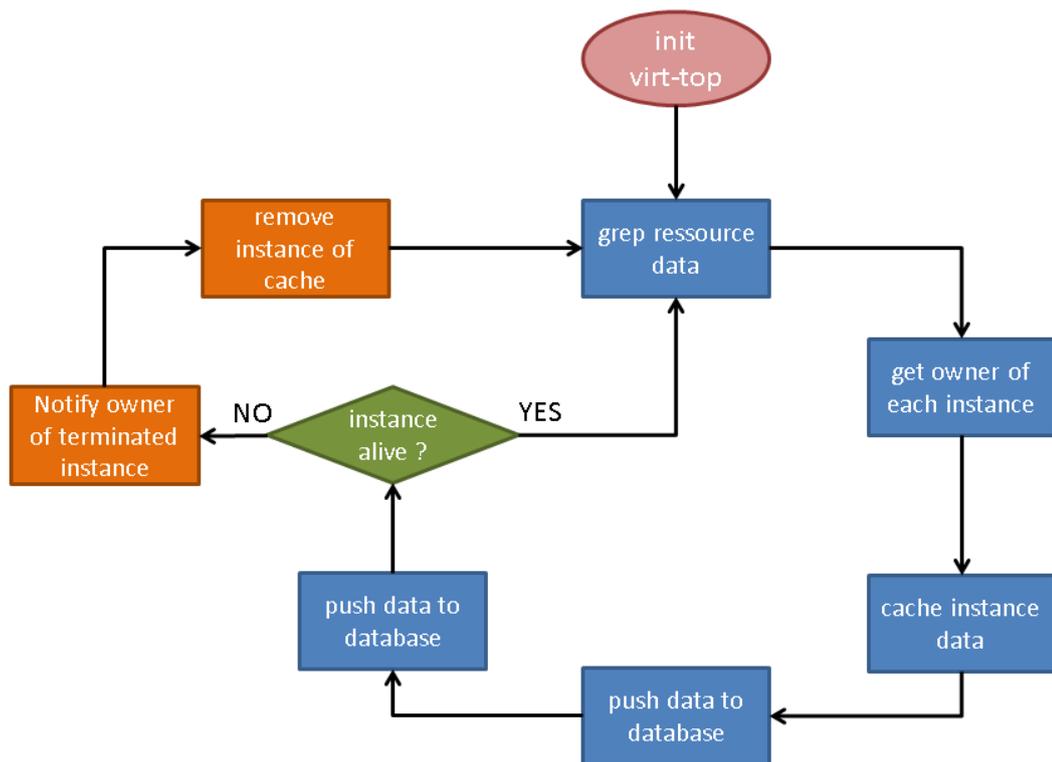


Abbildung 11.1: Ablaufdiagramm des NC-Daemons

Virt-Top wurde bereits in Abschnitt 8.1 eingeführt, jedoch ohne die dazugehörigen Parameter:

- csv <filename>.csv** Virt-Top erlaubt die Umleitung der Konsolenausgabe zusätzlich in eine CSV-Datei. Diese Datei ist Komma-separiert und reiht innerhalb einer Zeile zunächst die globalen Host-Informationen gefolgt von den einzelnen Verbrauchsdaten jeder laufenden Instanz aneinander. Ausgeschaltete Instanzen werden ignoriert.
- script** Mittels dieses Parameters wird die Konsolenausgabe abgeschaltet. Dies hat nur in Kombination mit der CSV-Umleitung Sinn, sobald der Befehl skriptbasiert ausgeführt wird.
- b** Im Batch-Mode werden sämtliche Tastatureingaben ignoriert. Dieser Befehl ist ausschließlich zur skriptbasierten Ausführung der Anwendung geeignet.
- d <time in seconds>** Das 'd' steht für Delay (engl. Verzögerung). Abhängig von der angegeben Zeit in Sekunden kann damit die Zeit ange-

passt werden, über welche die Verbrauchsdaten aggregiert werden. Im Falle einer CSV-Umleitung impliziert dies eine Zeile pro angegebener Zeitspanne.

Alle weiteren Informationen über die Parametrierung finden sich in der dazugehörigen Man-Page¹.

Das Shell-Skript schreibt zusammenfassend alle zehn Sekunden einen aktuellen Status über laufende Instanzen in eine externe CSV-Datei. Diese wird von dem NC-Daemon weiterverarbeitet.

11.3 Erfassen der Verbrauchsdaten

Das Messen der Verbrauchsdaten wird von *Virt-Top* übernommen. Dieses schreibt, wie im vorhergehenden Abschnitt erwähnt, die Messdaten in eine externe CSV-Datei. Die Aufgabe des NC-Daemons ist das Auswerten der Daten innerhalb dieser CSV-Datei.

Mit dem selben Intervall wie auch das initiale Shell-Skript gestartet wurde (z.B. zehn Sekunden) liest der Daemon die CSV-Datei aus und analysiert alle dort gemessenen Instanzen. Diese gemessenen Instanzen beinhalten Informationen über die Verbrauchsdaten wie Netzwerkverkehr, Festplattenzugriffe und CPU-Zeit. Da ein Node Controller keine Informationen über die Daten des Cloud Controller besitzt, können dort ausschließlich Informationen über KVM oder die jeweiligen Hypervisor erfasst werden.

Die analysierten Instanzen werden innerhalb des Programmzyklus in einer globalen Hashmap gespeichert. Dort bleiben sie über den gesamten Ausführungsprozess erhalten. Terminiert eine Instanz, wird sie aus der Hashmap entfernt.

Die Hashmap macht einen Abgleich zwischen ID und der gesamten Instanz, sodass die Instanz indiziert angesprochen werden kann. Diese Indizierung wird zwingend benötigt, um festzustellen, ob neue Instanzen erzeugt oder bestehende Instanzen erneut gemessen wurden. Weiterhin wird diese ID benötigt, um den Besitzer der Instanz zuzuweisen.

¹Das Zeichen '&' impliziert eine Ausführung des Befehls im Hintergrund. Es wird an dieser Stelle als bekannt vorausgesetzt.

11.4 Besitzer der Instanz zuweisen

Ein großes Manko des Eucalyptus-Aufbaus ist, dass die jeweiligen Node Controller keinerlei Kommunikation mit dem Cloud Controller betreiben. Um nun den Besitzer einer auf dem NC gemessenen Instanz herauszufinden, muss auf Informationen des CLC zurückgegriffen werden.

Dazu wird ausgehend von dem NC ein SSH-Tunnel zu dem CLC aufgebaut und der Befehl `euca-describe-instances` ausgewertet. Dies erzeugt einen unschönen Overhead, jedoch muss kein weiteres Skript auf dem CLC ausgeführt werden. Der Abgleich zwischen den Informationen auf dem NC und denen des CLC muss nicht bei jedem Durchgang stattfinden, sondern lediglich dann, wenn für mindestens eine Instanz noch kein Besitzer zugewiesen wurde. Unter der Annahme, dass die Instanzen über längere Zeit betrieben werden, wird die Frequenz der SSH-Verbindungsaufbauten nahezu null.

Beim Auswerten des `Euca2ools` wird ein neues Instanzmodell eingeführt. Dieses Modell beinhaltet im Gegensatz zu dem Instanzmodell von `Virt-Top` Informationen über den Besitzer, die Größe und den Typ der Instanz. Die beiden Instanzmodelle werden über das gemeinsame Merkmal, die ID der Instanz, zusammengeführt.

Somit beinhaltet am Ende das Instanzmodell folgende Eigenschaften (s. Listing 11.2):

```

1  private final String instanceID;
2  private final String instanceName;
3  private double instanceCPUTime;
4  private double instanceCPUPercentage;
5  private double instanceBlockWritten;
6  private double instanceBlockRead;
7  private double instanceNetReceived;
8  private double instanceNetTransmitted;
9  private String state;
10
11 private String instanceType;
12 private String instanceOwner = "UNKNOWN";
13 private String instanceEmi;
14 private String instanceStartingTime;
15 private String instanceEndingTime;
    
```

Listing 11.2: Instanzmodell des NC-Daemons

Auf zwei Besonderheiten des Listings 11.2 sind:

final Die Eigenschaften *instanceID* sowie *instanceName* wurden als *final* deklariert, um diese als konstant zu markieren. Diese Eigenschaften sind während der Ausführungszeit einer Instanz nicht variabel.

UNKNOWN Der Besitzer einer Instanz wird als UNKNOWN initialisiert. Von Eucalyptus nicht verwaltete Instanzen (sogenannte Geisterinstanzen) werden sichtbar und bleiben als unbekannt gekennzeichnet.

11.5 Eintragen der Verbrauchsdaten in die Datenbank

Nachdem Erfassen aller Informationen über die einzelnen Instanzen, werden die Daten in die Datenbank eingetragen und dort persistent gespeichert. Der Eintrag entspricht den erfassten Daten von Virt-Top. Es erfolgt keine Modifikation der Daten.

Die Daten werden zusätzlich mit dem aktuellen Zeitstempel versehen. Dies bietet die Möglichkeit Statistiken über die Verbrauchsdaten in eine zeitliche Abhängigkeit zu setzen. Ein Beispiel hierfür wären die Zeitpunkte, an denen die CPU-Last am höchsten war.

11.6 Benachrichtigung bei terminierter Instanz

Um eine Terminierung von Instanzen innerhalb eines Intervalls zu erkennen, werden zu Beginn des Intervalls alle bereits im Speicher enthaltenen Instanzen mit dem Status `terminated` deklariert. Danach werden die aktuell ausgeführten Instanzen wie in Abschnitt 11.3 erfasst. Diese modellierten Instanzen besitzen den Status `running`. Der Speicher wird mit diesen Instanzdaten abgeglichen und aktualisiert.

Die Instanzen, die sich bereits im Speicher befanden, deren Status auf `terminated` gesetzt wurde, sind daraufhin wieder als laufend deklariert. Alle Instanzen, die nicht erneut gefunden wurden, bleiben auf `terminiert`. Dies ermöglicht folglich ein Filtern der im aktuellen Zyklus terminierten Instanzen.

Listing 11.3 zeigt die Hauptkonstrukte, die für diesen Vorgang essentiell sind.

```
1  for (Instance inst : parsedInstances.values()) {
2      inst.setState("terminated");
3  }
4
5  [...]
6
7  if (runningInstance != null){
```

```
8 //Will overwrite existing instances in HashMap
9 parsedInstances.put(runningInstance.getInstanceName(),
10                     runningInstance);
11 }
12
13 [...]
14
15 for (Instance inst : parsedInstances.values()) {
16     if (inst.getState.equals("terminated")){
17         inst.notifyOwner();
18     }
19 }
```

Listing 11.3: Quellcodeauszüge zur Analyse von terminierten Instanzen

Bei der Benachrichtigung werden alle Daten, die zur jeweiligen Instanz gehören, aggregiert. Die Aggregation geschieht auf unterschiedliche Art und Weise:

Sum() Aus allen Datensätzen wird die Summe gebildet, um den Gesamtverbrauch von CPU-Zeit, Netzwerkverkehr und DiskIO festzustellen. Verwendet wird eine Addition aller Datensätze.

Avg() Analog zur Summe können aus allen Datensätzen mit Hilfe des arithmetischen Mittels Durchschnittswerte berechnet werden. Dies ist für Unternehmen interessant, um den Verlauf der Verbrauchsdaten über längere Zeit zu beobachten, Tendenzen herauszufinden, sowie Prognosen zu erstellen.

Min(),Max() Ein weiterer wichtiger Auswertungspunkt sind Tief- und Hochpunkte der gemessenen Daten. In Kombination mit dem zu jedem Datensatz eingetragenen Zeitpunkt der Messung, können Höchstleistungen zeitlich analysiert und gegebenenfalls den einzelnen Aufträge eines Unternehmens zugeordnet werden.

Diff() Unabhängig von den verschiedenen Kategorien der Messdaten kann mit einer einfachen Differenz des Zeitstempels des ersten Eintrags und dem letzten einer Instanz die Gesamtlaufzeit gemessen werden. Diese Berechnung wird benötigt, wenn die Abrechnung der Instanzen nicht nach verbrauchten Ressourcen, sondern pauschal nach Laufzeit geschieht.

Je nach Kontaktdaten werden diese aggregierten Statistiken dem Besitzer der Instanz zugänglich gemacht. In dieser ersten Implementierung werden diese Daten per E-Mail versandt.

Die Berechnung der anfallenden Kosten wird mittels der flexibel gestaltbaren Formel geschehen. Ein erster Ansatz der versandten E-Mail zeigt Listing 11.4.

```

1 Your instance was terminated
2
3 Following the data of the instance:
4
5 Global information:
6 id:          i-504508F7
7 owner:       karsten
8 emi:         emi-AA1113DB
9 status:      terminated
10 type:        m1.small
11 startingtime: 2010-06-24T08:01:54.15Z
12 endingtime:   2010-07-12T10:12:13.15Z
13
14 Detailed usage information:
15 CPU-time:     20000000.00 ns
16 CPU-percentage: 0.02 %
17
18 net-received: 1880 Byte
19 net-transmitted:126 Byte
20
21 disk-read:    0.00 Byte
22 disk-written: 3.00 Byte
23
    
```

```

24 Applied statistic information:
25 CPU-Max:      xxx ns
26 Net-Max:      xxx Byte
27 Disk-Max:     xxx Byte
28
29 Effective cost information:
30 Instance-type factor: 1
31 CPU:          20000000.00 * CPU-factor
32 NET:          2006 Byte * Net-factor
33 DISK:         3.00 Byte * Disk-factor
34
35 charges:      xxx \$
    
```

Listing 11.4: Generierte E-Mail zur Benachrichtigung der Besitzer über terminierte Instanz

Nachdem die E-Mail an den Besitzer geschickt wurde, ist der darauf folgende Schritt das Herauslöschchen der Instanz aus der globalen Hashmap. Der Daemon darf keine Informationen über die terminierte Instanz im Cache haben. Die Gründe dafür sind unter anderem das Befreien von nicht benötigtem Speicherplatz und die Restriktion, dass keine weitere E-Mail versandt wird. Vergisst man diesen Löschvorgang, werden in jedem Intervall die Besitzer über ihre eventuell schon längst vergangene Instanz benachrichtigt².

Bis zum jetzigen Stand wird diese E-Mail bei Terminierung einer Instanz generiert. In einem weiteren Schritt kann allerdings ähnlich einem Online-Banking eine Weboberfläche angeboten werden, auf der der Besitzer jederzeit die Chance hat, seine aktuelle Belastung nachzusehen.

²In einer Testphase mit einem provisorischen Intervall von einer Sekunde führte diese Vernachlässigung sehr schnell zu einer Flutung des eigenen Postfachs.

ZENTRALE DATENBANK

Um unabhängig und flexibel von dem Resource Monitoring zu sein, entschied man sich für eine externe, zentrale Datenbank. Wie Abschnitt 11.5 erklärt, werden die Daten vollständig gesammelt und in die Datenbank geschrieben. Dies hat den Vorteil, dass sämtliche Weiterentwicklungen und grafischen Werkzeugen auf die Datenbank zugreifen können ohne den Programmfluss der Verbrauchsdatenmessung zu stören. Dort können die Daten geladen und anwendungsorientiert aggregiert und verarbeitet werden. All diese Anwendungen sind komplett unabhängig von dem Daemon und dem damit verbundenen Laufzeitzyklus.

12.1 Datenaggregation

Für das Arbeiten mit der Datenbank im Hinblick auf die Aggregation der gesammelten Daten wurde eine prägnante Entscheidung bei der Implementierung dieser Arbeit getroffen. Pro Laufzeitintervall (in den vorhergehenden Beispielen waren es zehn Sekunden) werden alle laufenden Instanzen mit den Messdaten, wie sie von Virt-Top bereitgestellt werden, abgelegt. Die Daten werden ohne zusätzliche Aufbereitung oder Aggregation in die Datenbank gespeichert. Der Grund dafür ist, dass Virt-Top innerhalb des angegebenen Intervalls bereits aggregiert¹. Eine erste Überlegung war, pro Instanz nur einen Eintrag zu generieren, der aktualisiert wird.

¹Für die CPU-Zeit bedeutet dies, dass bei zehnfach größerem Intervall, eine zehnfach größere CPU-Zeit gemessen wird, der prozentuale Anteil der CPU jedoch mitunter gleich bleiben kann.

Dies hätte den Vorteil, dass weit weniger Datensätze zu speichern wären. Ein Nachteil ist der Verlust von zeitlichen Verläufen der Verbrauchsdaten. Eine Statistik zu welcher Zeit die Auslastung am höchsten war, ist auf diese Weise nicht möglich.

Trotz der Tatsache, dass bei dieser Entscheidung deutlich mehr Einträge in der Datenbank enthalten sind, entsteht dabei kein Nachteil der Qualität der Daten. Durch die intensive Datenmenge können beliebige Statistiken gezogen werden. Die Genauigkeit der Daten hängt von dem gewählten Zeitintervall ab. Ein weiterer Vorteil an dieser Stelle ist, dass die Umsetzung und konkrete Anwendung beliebiger Aggregationsfunktionen, wie beispielsweise eine Summen- `[select Sum(*)]`, `[Maximal- select Max(*)]`, `[Minimal- select Min(*)]`, oder Durchschnittsfunktionen `[select Avg(*)]`, bereits von den meisten Datenbanksystemen angeboten werden und somit die Umsetzung dieser Funktionen als gegeben betrachtet werden kann.

12.2 Datenbanksystem

Als Datenbanksystem entschied man sich im ersten Ansatz für eine MySQL Datenbank. Die Gründe dafür waren das simple Design und die einfache Umgangsweise. Die folgende Berechnung umfasst die ungefähren Leistungsanforderungen an eine zentral gestaltete Datenbank wie die hier gewählte MySQL:

<i>NodeController:</i>	5
<i>Delay (in sec):</i>	10
<i>Ausgeführte Instanzen pro NC:</i>	15

Das macht eine Transaktionsfrequenz (Zugriffe pro Sekunde) von:

$$\frac{5 * 15}{10} = 7,5 \frac{\text{Zugriffe}}{\text{Sekunde}}$$

Dies erzeugt an einem Tag (Einträge pro Tag) etwa:

$$7,5 * 3.600 * 24 = 648.000 \frac{\text{Eintraege}}{\text{Tag}}$$

Dies macht in einem Monat (Einträge pro Monat) etwa:

$$648.000 * 31 = 20.088.000 \frac{\text{Eintraege}}{\text{Monat}}$$

Gestützt auf das MySQL Referenzbuch der Herausgeber und mehrere Praxistests sind diese Daten für eine MySQL Datenbank, ausgeführt auf einem Mehrkernsystem ohne Probleme handhabbar. Die Datenmengen wachsen nicht exponentiell, sodass eine Explosion der Datenmenge nicht vorkommen kann.

Unter der Annahme, dass eine Abrechnung der Verbrauchsdaten bei längeren Laufzeiten monatsgenau, anstelle von minutengenau geschieht, können nach einem Monat sämtliche Verbrauchsdaten gesammelt und zusammengefasst archiviert werden. Dies erlaubt keine sekundengenaue Statistiken mehr, die älter als ein Monat sind, jedoch können die aggregierten Monatsdaten erneut gesammelt und für Jahresstatistiken verwendet werden².

12.3 Datenbankmodell

Das gewählte Datenbankmodell beinhaltet insgesamt drei Tabellen. Um alle Anforderungen an das Accounting-System zu erfüllen, wird in einer ersten Tabelle der Benutzer erfasst. Neben dem Benutzernamen, welcher als Primary Key dient, werden die Kontaktdaten, wie E-Mailadresse, Telefonnummer oder ähnliches erfasst. Als letzte Spalte werden alle ihm belasteten Kosten, unabhängig von den einzelnen Instanzen, zugewiesen.

Die zweite Tabelle erfasst die Verbrauchsdaten. Diese Daten sind *Disk read*, *Disk written*, *Net received*, *Net transmitted*, *CPU-Time*, *CPU-Percentage*. Als Primary Key dient an dieser Stelle die ID der Instanz.

²Für die konkrete Anwendung in der Praxis ist eine sekundengenaue Statistik der Verbrauchsdaten wohl eher uninteressant, da eine Finanzierungsbilanz monats-, wenn nicht jahrgenau sein muss und die Hochleistungen auf einzelne Projekte und Aufträge heruntergebrochen werden.

Alle anderen Daten, wie der Typ der Instanz, wurden auf eine dritte Tabelle ausgelagert. Dies dient im Hinblick auf die dritte Normalform der Duplikatsvermeidung. Die dritte Tabelle macht einen Abgleich zwischen dem Benutzer (referenziert über den Benutzernamen) und der Instanz (referenziert über die Instanz-ID). Zusätzlich kommen konstante Daten der Instanz hinzu. Diese sind zum Beispiel der Instanztyp oder die Erstellungszeit. Eine weitere Spalte enthält die aktuellen Kosten für den Benutzer, verursacht von der jeweiligen Instanz.

Eine Beschränkung auf zwei Tabellen bedeutet das Speichern von Daten wie Instanztyp und Besitzer für jeden Eintrag. Die Beschränkung hätte den Vorteil, dass keine rechenintensiven Zusammenführungen von Tabellen (Joins) geführt werden müssten. Dieser Vorteil verliert an Bedeutung, wenn nach der Aggregation der großen Instanzverbrauchsdaten eine Zusammenführung der Tabellen gemacht wird. Die Laufzeit bleibt dementsprechend performant.

Abbildung 12.1 zeigt die Beschreibung der drei verwendeten Tabellen. Die Abbildung stammt direkt aus der MySQL Kommandozeile.

```
mysql> describe instance;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | varchar(20)   | YES  |     | NULL    |       |
| netReceived | float(100,2)  | YES  |     | NULL    |       |
| netTransmitted | float(100,2) | YES  |     | NULL    |       |
| diskRead   | float(100,2)  | YES  |     | NULL    |       |
| diskWritten | float(100,2)  | YES  |     | NULL    |       |
| cpuTime    | float(150,2)  | YES  |     | NULL    |       |
| cpuPercentage | float(150,2) | YES  |     | NULL    |       |
| hostname   | varchar(50)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> describe user;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| username   | varchar(50)   | NO   | PRI | NULL    |       |
| email      | varchar(50)   | YES  |     | NULL    |       |
| notes      | varchar(200)  | YES  |     | NULL    |       |
| charges    | float(100,2)  | YES  |     | 0.00    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> describe user_instance;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| username   | varchar(50)   | NO   | PRI | NULL    |       |
| instance   | varchar(20)   | NO   | PRI | NULL    |       |
| charges    | float(100,2)  | YES  |     | 0.00    |       |
| startingTime | varchar(50)   | YES  |     | NULL    |       |
| endingTime | varchar(20)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> 
```

Abbildung 12.1: Datenbankdesign mit drei Tabellen *User*, *Instance*, *User_Instance*

WEBOBERFLÄCHE

Eine bislang noch nicht behandelte Webanwendung ist Karlsruhe Open Application (for) cCloud Administration (KOALA). KOALA ist eine Open Source Entwicklung zur Steuerung unterschiedlicher Cloud-Infrastrukturen wie AWS, Eucalyptus, Nimbus oder Open Nebula.

KOALA selbst läuft in der Cloud als PaaS, entweder unter der Google App Engine (GAE) oder der Open Source Nachentwicklung Appscale. Abbildung 13.1 zeigt den allgemeinen Aufbau des KOALA Werkzeugs. Interessant ist die Kombination der verschiedenen Cloud-Architekturen (s. 6.2). KOALA verbindet PaaS und IaaS und zeigt gleichzeitig eine Integration mehrerer IaaS-Anbieter in einer Anwendung.

Neben der Entwicklung des NC-Daemon wird eine Weboberfläche in KOALA integriert. Diese Weboberfläche stellt dem Besitzer verschiedener Instanzen innerhalb der Private Cloud im SCC eine Übersicht seiner Verbrauchsdaten zusammen. Innerhalb der Oberfläche hat er die Möglichkeit, seine Daten nach den vorgegebenen Aggregationsfunktionen (zum Beispiel Maximalwerte der CPU) statistisch auswerten zu lassen. Zusätzlich hat er den Überblick seiner aktuell anfallenden Kosten. Die Oberfläche ist analog zu einem Online-Banking zu betrachten.

Eine vollständige Integration in die bestehende KOALA Umgebung findet im Rahmen dieser Arbeit nicht statt. Für die erste Implementierung wurde eine unabhängige Weboberfläche (siehe Abbildung 13.2) erstellt, welche die wichtigsten Funktionen zeigt. Implementiert wurde eine Übersicht der persönlichen Daten und aller Instanzen, die dem eingeloggten Benutzer zugeordnet sind.

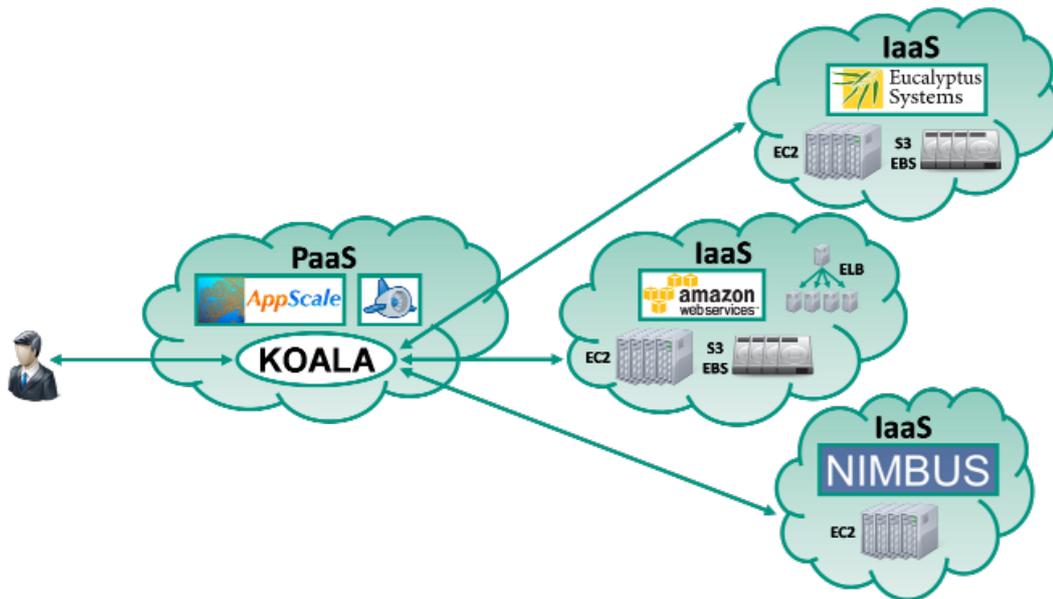


Abbildung 13.1: Der allgemeine Aufbau von KOALA zeigt die Kombination von PaaS und IaaS sowie die Integration mehrerer Cloud Infrastrukturen in eine Anwendung.

Willkommen zur Übersicht der Verbrauchsdaten der Private Cloud im SCC

Übersicht Ihrer persönlichen Daten:
 Name: karsten
 Beschreibung: Karsten ist ein Held!
 Kontaktdaten: karsten.knese@kit.edu
 Aktuelle Belastung: 0

Übersicht Ihrer aktuell verfügbaren Instanzen:

id	netRechnet	trans	diskRead	diskWritten	cpuTime	cpuPercentage
i-480D0707	0	0	0	0	70000000	0,011667
i-4CCB0940	23023	630	0	10	120000000	0,023333

Übersicht der nicht zugeordneten Instanzen:

user	id	netRechnet	trans	diskRead	diskWritten	cpuTime	cpuPercentage
UNKNOWN	None-391	23653	0	0	0	220000000	0,043333

Abbildung 13.2: Eine erste Implementierung der Weboberfläche

Der Screenshot zeigt drei Bereiche:

Persönliche Daten Die letzte Zeile des ersten Bereiches ist besonders relevant für den Benutzer, da er dort eine Übersicht über seine bisher

anfallenden Kosten sieht. Des Weiteren werden ihm dort all seine persönlichen Daten angezeigt.

Zugeordnete Instanzen Der zweite Bereich zeigt eine Auswertung der Instanzen, die der Benutzer erstellt hat. Dort wird aktuell die Aggregationsfunktion Sum() über die Ressourcen CPU-Zeit, Netzwerkverkehr und DiskIO angeboten. Auf Basis der Datenbank werden im Laufe der Produktionsphase weitere Statistikausgaben entwickelt.

Nicht zugeordnete Instanzen (Geisterinstanzen) Der dritte Bereich zeigt Instanzen ohne zugeordnetem Besitzer. Diese Anzeige dient in erster Linie den Administratoren von Eucalyptus, um festzustellen, welche Instanzen nicht ordnungsgemäß laufen.

Das verwendete SQL-Statement zur Ausgabe der Instanzen ist recht simpel (siehe Listing 13.1).

```

1 select user_instance.username as user ,
2     id, sum(netReceived) as netRec ,
3     sum(netTransmitted) as netTrans ,
4     sum(diskRead) as diskRead ,
5     sum(diskWritten) as diskWritten ,
6     sum(cpuTime) as cpuTime ,
7     avg(cpuPercentage) as cpuPercentage
8 from instance join user_instance
9 on user_instance.instance=instance.id
10 where user_instance.username='karsten'
11 group by id ;
    
```

Listing 13.1: SQL-Statement für den Gesamtverbrauch aller Instanzen vom Benutzer Karsten

Durch den Einsatz einer externen zentralen Datenbank kann eine beliebige Darstellung - in diesem Fall das Web - der aufbereiteten Daten angeboten werden. Des Weiteren wird eine Plattformunabhängigkeit geschaffen. Der NC-Daemon ist in Java implementiert. Als Implementierungssprache für die gezeigte Homepage wurde deshalb bewusst .net gewählt.

KONFIGURATIONSOBERFLÄCHE

Die komplette Arbeit wurde unter dem Hintergrund entwickelt, unabhängig von einer Umgebung zu sein. Das Wunschziel ist eine universale Lösung, mit der auf jeder Cloud-Umgebung die Accounting-Lösung eingerichtet und produktiv eingesetzt werden kann.

Die Unabhängigkeit für das Resource Monitoring wurde durch die Libvirt Bibliothek und die externe Datenbank geschaffen. Diese Unabhängigkeit wurde für das Abrechnungssystem ebenfalls initiiert durch die Einführung einer universalen Abrechnungsformel (siehe Abschnitt 10.2). Die konkreten Werte dieser Formel befinden sich in einer zusätzlichen Tabelle in der Datenbank. Abbildung 14.1 zeigt diese Tabelle. Diese beinhaltet drei Spalten; Den Namen der Ressource, den dazugehörigen Berechnungsfaktor und ein Flag, ob diese Komponente Teil der Abrechnung ist.

```
mysql> describe accounting;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(50)   | YES  |     | NULL    |       |
| value | int(11)       | YES  |     | NULL    |       |
| enabled | tinyint(1)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from accounting;
+-----+-----+-----+
| name    | value | enabled |
+-----+-----+-----+
| hardware | 0     | 1       |
| cpu-time | 0     | 1       |
| net     | 0     | 1       |
| disk    | 0     | 1       |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

Abbildung 14.1: Die separate Tabelle für die Konfiguration des Abrechnungssystems

Mit dieser Tabelle ist eine flexible Gestaltung der Preisbildung ermöglicht. Finanzexperten, welche zuständig für eine konkrete Preisbildung sind, sollen

keinen Quellcode zur Aktualisierung des Preises modifizieren müssen. Aus diesem Grund wurde in einem weiteren Schritt eine Oberfläche generiert, die solch eine Konfiguration mittels Mausklicks ermöglicht. Diese Oberfläche (s. Abbildung 14.2) ist intuitiv bedienbar.

Billing System Configuration Form

SCC
Steinbuch Centre
for Computing

Financial workers can submit a new configuration for the billing system implemented for the Private Cloud in the Steinbuch Centre for Computing.

Reset to default

Submit new configuration

Billing Configuration Eucalyptus Systems

cost factors

First you can control the hardware cost factors. It depends on the underlying hardware of each host system.

Hardware 0

Next, you can modify each resource cost factor individually.

cpu 0

net 0

disk i/o 0

charge dependencies

You can enable / disable each of the following dependencies. All of the enabled items will be a component of the billing system.

- Hardware**
- cpu**
- net**
- disk i/o**

Abbildung 14.2: Eine intuitiv bedienbare Konfigurationsoberfläche zur Anpassung der Preisbildung.

Die gewünschten Parameter zur Berechnung der Preises können in den oberen Teil des Formulars eingetragen werden. In dem unteren Teil können die Felder aktiviert werden, die zur Preisbildung hinzugefügt werden sollen.

Teil V

Abschließendes

Cloud Computing ist eines der Top Themen in der IT-Welt [17]. Die Virtualisierung macht auch vor ganzen virtuellen Rechenzentren keinen Halt. Das Denken in Diensten die quer über die Welt angeboten werden, revolutioniert die Vorstellung einer technischen Infrastruktur und erweitert diese global und ortsunabhängig. Damit eine adäquate Abrechnung funktionieren kann, müssen die Anforderungen an das genaue Messen von Verbrauchsdaten sowie an eine ordnungsgemäße Benutzerverwaltung erfüllt werden.

Diese Arbeit beinhaltete zwei primäre Herausforderungen. Zum Einen musste eine Möglichkeit zum Erfassen der Verbrauchsdaten gefunden werden. Die Schwierigkeit bestand darin, dass nicht die Verbrauchsdaten des Host-Systems gemessen werden konnten. Begründet wird das durch das Angebot von virtuellen Instanzen. Durch das Angebot von unterschiedlichen Instanzgrößen muss eine Messung gezielt jede einzelne VM stattfinden. Um eine Unabhängigkeit der eingesetzten Betriebssysteme zu ermöglichen, wurde darauf geachtet das Betriebssystem nicht zu modifizieren.

Virt-Top misst drei Kategorien der Verbrauchsdaten. Mittels dieser Anwendung können CPU-Zeit, Netzwerkverkehr und Festplattenzugriffe gezielt für jede Instanz gemessen werden. Durch den Einsatz der Libvirt und die Speicherung der Daten in eine externe Datenbank wurde eine Flexibilität geschaffen, um die konkrete Implementierung nicht abhängig von der Hardwareumgebung und dem eingesetzten Hypervisor produktiv einzusetzen. Zur flexiblen Weiterverarbeitung der Daten, speichert Virt-Top die Daten in eine externe CSV-Datei. Um die Daten auch für eine spätere Analyse bereitzustellen, werden sie in eine Datenbank gespeichert. Auf diese Datenbank

kann jede weitere Anwendung, wie beispielsweise die implementierte Weboberfläche, zugreifen.

Die zweite Herausforderung war die Erstellung einer adäquaten Preisbildung. Da im Eucalyptus Umfeld noch keine Accounting-Lösung existiert, fand eine Evaluation von drei bestehenden Anbietern (Amazon, Rackspace und GoGrid) statt. Sie unterschieden sich darin, welche Ressourcen zur Berechnung miteinbezogen werden. Die Schwierigkeit bestand bei der Anwendung eines solchen Abrechnungsmodells innerhalb des Angebots der Private Cloud im SCC. Da dies nicht zum Ziel hat, einen Gewinn zu erwirtschaften, sondern lediglich sich selbst zu tragen, liegt der Schwerpunkt bei diesem Modell auf der flexiblen Anpassung des Preismodells.

Um solch eine Anpassung an das Preismodell für das SCC zu ermöglichen, wurde eine flexible Formel zur Erstellung eines konkreten Preises pro Instanz entwickelt. Diese Formel ermöglicht eine Unterscheidung zwischen Grundgebühr und Verbrauchsdaten. So kann für jede Kategorie von Verbrauchsdatum ein Preis festgelegt werden. Nach der Festlegung der Gebühr für jede Ressource kann entschieden werden, welche für die Bestimmung des Preises relevant ist. Um diese Formel den zuständigen Finanzbeauftragten zugänglich zu machen, wurde dafür eine einfache Konfigurationsoberfläche erstellt.

Um den Benutzer über seine anfallenden Kosten zu informieren, entstanden zwei Implementierungen. Zum Einen wird direkt bei Terminierung der Instanz eine E-Mail versandt. Diese E-Mail beinhaltet Informationen über die Menge der Verbrauchsdaten und die dementsprechenden Kosten. Als weitere Methode wurde eine Weboberfläche erstellt, die dem Benutzer jederzeit Informationen über seine aktuellen Instanzen liefert.

AUSBLICK

Diese Arbeit stellt eine grundlegende Implementierung für das Resource Monitoring und Accounting dar. Aufbauend auf die gesammelten Daten in der Datenbank können verschiedene Anwendungen entwickelt werden.

Die in einer prototypischen Weise implementierte Weboberfläche kann dahingehend erweitert werden, dass in einer ausführlichen Darstellung sämtliche Statistiken über die Verbrauchsdaten der einzelnen Instanzen angezeigt werden. Diese können Diagramme und Zeitpunkte über Höchstauslastungen oder Durchschnittswerte darstellen. Eine Integration dieser Weboberfläche in KOALA ist mögliche Weiterentwicklung, um Instanzen innerhalb einer Webanwendung zu verwalten und abzurechnen.

Bei einer Produktivschaltung des Systems ist abzuwägen, ob eine in die Javaanwendung integrierte Datenbank (embedded db) für jeden NC-Daemon, die sich mit einer Masterdatenbank repliziert, performanter ist, als alle NC-Daemon direkt auf eine Datenbank schreiben zu lassen. Des Weiteren ist zu evaluieren, in welchem Rahmen eine Aggregation der gesammelten Daten erfolgen soll, sodass die Datenmenge die eingesetzte Datenbank nicht überlastet.

Für einen Produktionsbetrieb ist eine Anpassung der Abrechnungsformel an die lokalen Gegebenheiten und Einflussfaktoren notwendig. Die variablen Faktoren der Formel müssen von Finanzexperten konkret besetzt werden.

Teil VI

Verzeichnisübersicht

ABBILDUNGSVERZEICHNIS

4.1	Unterschiedliche Hardwarearchitekturen können mittels Software in die physikalische Hardware übersetzt werden.	9
4.2	Vollständige Virtualisierung beinhaltet eine Managementfunktion, um die Ressourcen gezielt an die VMs zu adressieren.	10
4.3	Ringstruktur eines x86-Prozessors	11
4.4	Ringstruktur eines x86-Prozessors mit Virtualisierungstechniken	12
4.5	Eine Modifikation des Gastbetriebssystems ist bei Paravirtualisierung notwendig.	14
5.1	Skizze wie KVM in den Linux Kernel integriert ist und die dort vorhandene Umgebung nutzen kann.	17
5.2	QEMU arbeitet auf der KVM Architektur und emuliert verschiedene Architekturen.	19
5.3	Funktionsweise der Libvirt	21
6.1	Skizzenhafte Auslagerung der Dienste in die Cloud. Als Schnittstelle dient ein Zugang in die Cloud. Genauere Informationen über die Art und Weise der Ressourcen bleiben verborgen.	23
6.2	Stack der Cloud-Architekturen	24
6.3	Übersicht der einzelnen Architekturen	26

6.4	Idealverhalten in Auslastung und Budgetierung	31
6.5	Verhalten in Auslastung und Budgetierung im Falle von Hochauslastung	32
6.6	Verhalten in Auslastung und Budgetierung unter Einsatz von Cloud Computing	33
7.1	Technischer Aufbau von Eucalyptus [12]	35
7.2	Eucalyptus wird mittels den Euca2ools gesteuert. Die Webservices sprechen die Libvirt an.	38
7.3	Der Befehle <code>euca-describe-instances</code> zeigt eine Auflistung aller existierender Instanzen.	40
7.4	Startseite der Eucalyptus Weboberfläche	40
7.5	Screenshot des Firefox Plugins HybridFox	41
8.1	Beispielhafte Analyse der CPU Auslastung für verschiedene Instanzen [18]	48
8.2	Eine Übersicht von zwei Computern. Der obere ist kritisch, da er nicht erreichbar ist. Localhost läuft fehlerfrei.	50
8.3	Eine Übersicht der virtuellen Maschinen, die auf dem NC ausgeführt werden.	52
9.1	Pricing: On-Demand Instances	61
9.2	Pricing: Reserved Instances	62
9.3	Pricing: Spot Instances	62
9.4	Pricing: Internet Data Transfer	63
9.5	Princing-Angebot von Rackspace	64
9.6	Die verschiedenen Größen der Instanztypen des Anbieters GoGrid	64
9.7	Das Abrechnungssystem von GoGrid	65
10.1	Pricing: On-Demand Instances	74

10.2 Gesamtaufbau der Implementierung	79
11.1 Ablaufdiagramm des NC-Daemons	82
12.1 Datenbankdesign mit drei Tabellen <i>User</i> , <i>Instance</i> , <i>User_Instance</i>	94
13.1 Der allgemeine Aufbau von KOALA zeigt die Kombination von PaaS und IaaS sowie die Integration mehrerer Cloud Infra- strukturen in eine Anwendung.	96
13.2 Eine erste Implementierung der Weboberfläche	96
14.1 Die separate Tabelle für die Konfiguration des Abrechnungs- systems	98
14.2 Eine intuitiv bedienbare Konfigurationsoberfläche zur Anpas- sung der Preisbildung.	99

TABELLENVERZEICHNIS

9.1	Small Instance (Standard)	57
9.2	Large Instance	58
9.3	Extra Large Instance	58
9.4	High-Memory Extra Large Instance	58
9.5	High-Memory Double Extra Large Instance	58
9.6	High-Memory Quadruple Extra Large Instance	59
9.7	High-CPU Medium Instance	59
9.8	High-CPU Extra Large Instance	59
10.1	Small Instance (Standard)	74

QUELLCODEVERZEICHNIS

5.1	Starten einer Instanz	17
8.1	Libvirt Methoden zum Resource Monitoring	45
10.1	Eintrag der Sourcen für die Installation von Java	78
11.1	Virt-Top	81
11.2	Instanzmodell des NC-Daemons	85
11.3	Quellcodeauszüge zur Analyse von terminierten Instanzen . .	86
11.4	Generierte E-Mail zur Benachrichtigung der Besitzer über ter- minierte Instanz	88
13.1	SQL-Statement für den Gesamtverbrauch aller Instanzen vom Benutzer Karsten	97

D

ABKÜRZUNGSVERZEICHNIS

KVM Kernel based Virtual Machine	6
Eucalyptus Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems	53
SCC Steinbuch Centre for Computing	54
KIT Karlsruher Institut für Technologie	2
IaaS Infrastructure as a Service	27
PaaS Platform as a Service	27
SaaS Software as a Service	27
VM Virtuelle Maschine	8
VMs Virtuelle Maschinen	46
VMM Virtual Machine Monitor	10

API Application Programming Interface	8
QoS Quality of Service	23
AWS Amazon Webservices	23
SSH Secure Shell	23
RDP Remote Desktop Protocol	23
API Application Programming Interface	8
XML eXtensible Markup Language	21
SOAP Simple Object Access Protocol	34
REST Representational State Transfer	35
CLC Cloud Controller	36
NC Node Controller	36
CC Cluster Controller	36
SC Storage Controller	37
PKI Public Key Infrastructure	39

BIRT Business Intelligence and Reporting Tools	52
CSV Comma Separated Values	48
EC2 Elastic Cloud Compute	56
GHz Gigahertz	57
S3 Simple Storage Service	55
KOALA Karlsruhe Open Application (for) cCloud Administration	95
GAE Google App Engine	95

LITERATURVERZEICHNIS

- [1] *Virtualisierung - Proseminar Linux für Umsteiger SS 2008*, Juli 2008.
- [2] Frank Meyer Andrej Radonic. *XEN3*. Franzis, 1. auflage edition, Oktober 2006.
- [3] Brooks Moses Carsten Heinz. *The Listings Package*, Februar 2007.
- [4] Christian Baun. *Servervirtualisierung*. Informatik-Spektrum, Mai 2009.
- [5] Alexander Voss Christian Baun, Marcel Kunze. Analyse der kosten des grid-computing. Technical report, Forschungszentrum Karlsruhe, Mai 2008.
- [6] Chris Grzegorzcyk Graziano Obertelli Sunil Soman Lamia Youseff Dmitrii Zagorodnov Daniel Nurmi, Rich Wolski. The eucalyptus open-source cloud-computing system. Technical report, Computer Science Department University of California, Santa Barbara, 2009.
- [7] Stephan Dlugosz. *Dokumentation zur Latex-Vorlage*, Mai 2007.
- [8] Uwe Domaratius. Diplomarbeitvorlage mit latex. Master's thesis, Technische Universität Chemnitz, Januar 2010.
- [9] Eclipse. Birt project.
- [10] Inc. Eucalyptus Systems. *Eucalyptus Administrator's Guide (1.6)*. <http://open.eucalyptus.com/wiki/eucalyptus-administrators-guide-16>.
- [11] Inc. Eucalyptus Systems. *Eucalyptus User's Guide (1.6)*. http://open.eucalyptus.com/wiki/EucalyptusUserGuide_v1.6.

- [12] Inc. Eucalyptus Systems. Eucalyptus open-source cloud computing infrastructure - an overview. Technical report, Computer Science Department University of California, Santa Barbara, August 2009.
- [13] Marcus Fischer. *Ubuntu GNU/Linux*. Galileo Computing, 4. auflage edition.
- [14] George A. Gratzer. *Math into LaTeX: an introduction to LaTeX and AMS-LaTeX*. Birkhauser Boston, 1. auflage edition, 1996.
- [15] Hochschule Mannheim. *13. Vorlesung Systemsoftware*. Christian Baun, Januar 2008.
- [16] Chris Horne. Understanding full virtualization, paravirtualization and hardware assist. Technical report, VMWare, Oktober 2007.
- [17] Gartner inc. Gartner identifies the top 10 strategic technologies for 2010. Technical report, Gartner inc., Oktober 2009.
- [18] Richard Jones. *Virt Top*. Redhat, <http://people.redhat.com/rjones/virt-top/>, September 2009.
- [19] Redhat libvirt. *libvirt API reference*. <http://libvirt.org/html/libvirt-libvirt.html>.
- [20] Lars Madsen. *Various chapter styles for the memoir class*, Mai 2010.
- [21] Arnold Robbins Nelson H.F. Beebe. *Classic Shell Scripting*. O'Reilly, 1. auflage edition, Mai 2005.
- [22] Tobias Oetiker. *An Acronym Environment for Latex*, Oktober 2009.
- [23] Scott Pakin. *The Comprehensive Latex Symbol List*, November 2009.
- [24] Dr. Wolfgang Riedel. *Latex für Fortgeschrittene*. TU Chemnitz, Januar 2010.
- [25] Thomas Ritzau Robert Warnke. *qemu-kvm & libvirt: 2010*. Books on Demand, 4 edition, März 2009.
- [26] Technische Universität München. *Paravirtualisierung und Virtualisierungstechnologie*. Christian Kern, 2006.

-
- [27] Fabian Thorns. *Das Virtualisierungsbuch*. C & I Computer- U. Literaturverlag, 2. auflage edition, September 2007.
- [28] University of Groningen. *BibTEX Tutorial*, Februar 2009.
- [29] Piet van Oostrum. *Page Layout in Latex*. University Utrecht, März 2004.