Hadoop as a Service (HaaaS) auf Basis von Eucalyptus und Cloudera

Maximilian Hoecker

BACHELORARBEIT

MatrikelNr. 626023

eingereicht am

Bachelorstudiengang

Informatik

an der Hochschule Mannheim

im Juli 2010

Betreuer: MSc. Christian Baun, Dr. Marcel Kunze
Aufgabensteller: Prof. Günther Bengel

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hochschule Mannheim, am 15. Juli 2010

Maximilian Hoecker

Danksagung

Ich möchte mich an dieser Stelle bei Prof. Dipl.-Inform. Günther Bengel für seine Unterstützung und Betreuung im Rahmen dieser Bachelorthesis bedanken. Des Weiteren danke ich Dr. Marcel Kunze aus dem Karlsruher Institut für Technologie für die Unterstützung und Betreuung während der Thesis und die Bereitstellung der Infrastruktur.

Vielen Dank auch an Christian Baun für die Betreuung, insbesondere für schnelle Beantwortung meiner Fragen und die Unterstützung.

Ein besonderer Dank gilt meinen Eltern Dr. Kurt-Peter Hoecker und Michaela Hoecker, die mich immer während meines Studiums unterstützt haben sowie meiner Freundin Anja Uhrig für das schnelle Korrekturlesen meiner Bachelorthesis.

Kurzbeschreibung

Diese Bachelorarbeit beschäftigt sich mit der Konzeption und Realisierung einer Open Source Software zur Verteilung eines Hadoop Clusters in einem Infrastructure-as-a-Service. Als kommerzielles Vorbild dient der existierende Cloud-Dienst Elastic MapReduce, den Amazon in der eigenen Public Cloud betreibt. Der Zielinfrastrukturdienst für die mit dieser Arbeit entstandene Anwendung ist Eucalyptus.

Die im Rahmen der Arbeit entwickelte Implementierung besteht aus einer API und einer Weboberfläche, die sich mit Hilfe eines Archives auf einem Application Server installieren lässt. Die Weboberfläche in Verbindung mit der API ist ein Software-as-a-Service der in Public- und Private Clouds einsetzbar ist. Die Software fokussiert die Installation und Konfiguration eines Hadoop Clusters und bietet Vorteile gegenüber den bereits existierenden Lösungen, wie z.B. die parallele Konfiguration des Clusters, eine Weboberfläche oder Erweiterbarkeit der API.

Abstract

This bachelor thesis concentrates on the development and implementation of an open source solution for the deployment of a Hadoop cluster into an Infrastructure-as-a-Service. Elastic MapReduce, which is operated by Amazon in an own Public Cloud, is used as a commercial example. The target Infrastructure-as-a-Service solution of the implementation developed by this thesis, is Eucalyptus.

The developed software solution consists of an API and Web-based user interface, which is deployable via an archive on an application server. The webinterface combined with the API is a Software-as-a-Service solution, which is applicable for Private- and Public Clouds. The software focusses on the installation and configuration of the Hadoop Cluster and offers advantages over existing software solutions like parallel configuration of a cluster, a Web-UI or the expandablility of the API.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Industrielle und wissenschaftliche Anwendungen, beispielsweise aus den Bereichen Web-Indexierung, Datamining oder wissenschaftliche Simulation erzeugen und verarbeiten Daten im Bereich mehrerer Giga- oder Terabyte. Die Analyse und Verarbeitung von Datenmengen dieser Größenordnung ist, auf einem einzelnen Rechner, nicht in akzeptabler Zeit möglich. Eine denkbare Lösung des Problems ist die verteilte Verarbeitung der Daten.

Zur Entwicklung von verteilten Anwendungen existiert die von der Apache Software Foundation entwickelte Software Hadoop. Die Software implementiert das MapReduce Konzept sowie ein verteiltes Dateisystem, mit dem sich Daten verteilt verarbeiten lassen. Zur Ausführung einer Anwendung, die mit Hadoop Daten verteilt verarbeitet, ist ein Hadoop Cluster die Voraussetzung.

Beim Betrieb eines Hadoop Clusters auf physikalischen Maschinen, mit beispielsweise monatlicher Datenverarbeitung, überwiegen die Fixkosten für Anschaffung und Wartung des Clusters den Nutzen. Eine Möglichkeit der kosteneffizienteren Nutzung der Resourcen ist der Einsatz von Hadoop in Verbindung mit Cloud Computing. Es existiert Cloud Computing Software, die einen Infrastrukturdienst bietet, auf dem sich ein Hadoop Cluster installieren lässt.

Ziel dieser Bachelorthesis ist die Entwicklung einer Software, die ein Hadoop Cluster in unternehmsinternen oder öffentlich zugänglichen Clouds installiert bzw. konfiguriert, um damit einen Open Source Dienst on demand zur Verfügung stellen zu können. Die Software ist in Verbindung mit der Private Cloud Infrastructure-as-a-Service Lösung, Eucalyptus genannt, zu entwickeln. Diese Arbeit analysiert im ersten Kapitel die Grundlagen des Cloud Computing, von Hadoop und das kommerzielle Vorbild Elastic MapReduce, um daraus die Anforderungen herauszuarbeiten.

Kapitel 2 dieser Arbeit beschäftigt sich mit der Definition von Cloud Com-

puting und der Begriffsabgrenzung zu Grid Computing. Die Arbeit zeigt und beschreibt dabei die verschiedenen Cloud Architekturen und Cloud Service Kategorien.

Das Hadoop Framework bzw. die Hauptkomponenten von Hadoop sind in Kapitel 3 beschrieben, um die Arbeitsweise und den Aufbau von MapReduce und *Hadoop Distributed File System (HDFS)* zu zeigen. Die Arbeit geht zusätzlich auf die optional nutzbaren Hadoopkomponenten Hive und Pig ein.

Das 4. Kapitel beschäftigt sich mit dem kommerziellen Vorbild Amazon Elastic MapReduce. Der Aufbau sowie die Möglichkeiten Anwendungen in Elastic MapReduce zu starten, sind beschrieben.

Das Kapitel Realisierung (Kapitel 5) beschäftigt sich mit den Anforderungen, die eine Anwendung zur Installation und Konfiguration eines Hadoop Clusters hat. Die aus den Anforderungen entstandene Anwendung ist mit der Architektur, Design und deren Implementierung beschrieben. Die Arbeit vergleicht die entstandene Anwendung am Ende des Kapitels mit zwei bereits existierenden Lösungen.

Das vorletzte Kapitel 6 beschreibt die Evaluierung der in dieser Arbeit entstandenen Anwendung und geht dabei auf die verschiedenen topologischen Möglichkeiten ein, mit der die Anwendung evaluierbar ist.

Das letzte Kapitel gibt einen Ausblick auf denkbare Weiterentwicklungen der Anwendung sowie geplante Entwicklungen von Erweiterungen für Hadoop.

Kapitel 2

Cloud Computing

Cloud Computing ist ein Begriff, der sich momentan in vielen Bereichen der IT und bei vielen IT-Dienstleistern wiederfindet. Die Analysten von Gartner haben im „Hype Cycle for Emerging Technologies 2009“ das Cloud Computing als einen der dominierenden Hypes in der IT identifiziert [49]. Für 2010 wird Cloud Computing zu den Top-10 der strategischen Technologien gezählt [48].

2.1 Definition Cloud Computing

Eine allgemein gültige Defintion für den Begriff Cloud Computing exsistiert nicht. Der Begriff wird in vielen Kontexten verwendet, was die allgemeine Begriffsdefinition erschwert (siehe [13, Table 1]).

In [14] definieren die Autoren, dass eine Cloud ein Pool von virtualisierten Rechenresourcen ist. In [7] wird definiert, dass Cloud Computing eine automatisierte und standarditisierte Form einer dynamischen IT-Struktur ist. Die Autoren sind sich in den folgenden Punkten einig:

- Hinter dem Begriff Cloud Computing stehen verschiedenartige sogenannte *on demand Services* der Anbieter. Bei einem on demand Service bietet der jeweilige Dienstleister einem potentiellen Kunden einen Dienst (Service) an, den der Kunde sofort beziehen kann.
- Der Kunde kann den Dienst jederzeit wieder abbestellen und der Kunde bezahlt den Dienstleister, falls es kein kostenfreier Dienst ist, auch nur für die Zeit der Inanspruchnahme des Dienstes. (Prinzip des Pay-as-you-go)

- Wird der Dienst nicht in Anspruch genommen, ist er für den Benutzer kostenfrei.

Laut [2] wird in den allermeisten Fällen der Service über das Internet (bei sogenannten Public Clouds) oder Netzwerk der jeweiligen Organisation (bei sogenannten Private Clouds) in Form eines Web-Services erbracht. Die Konsolidierung der angebotenen Dienste erfolgt aufseiten der Dienstleister durch Virtualisierung. Durch die Virtualisierung ergibt sich für den Kunden eine dynamische IT-Struktur, die er exakt an seine Bedürfnisse anpassen kann. Der Anbieter hat den Vorteil, dass er seine eigene IT-Infrastruktur effizienter auslasten kann.

Die Vorzüge einer dynamischen IT-Struktur lassen sich anhand des folgenden Beispiels darstellen: Ein Unternehmen, das einen kostenfreien Emailservice anbietet, betreibt x Server, die außerhalb der Stoßzeiten stark ausgelastet sind. Zu den Stoßzeiten (z.B. jeweils 20 Minuten um 8 Uhr morgens, um 12 Uhr mittags und um 16 Uhr nachmittags) sind die Server überlastet und der Dienstleister kann den Dienst nicht mehr performant erbringen. Eine Lösung die Server zu entlasten ist, die Anschaffung und Installation neuer Server. Der Zukauf ist für das Unternehmen aber unrentabel, da diese Server nicht dauerhaft ausgelastet sind.

Setzt das Unternehmen Cloud Computing ein, kann es genau für den kurzen Zeitraum der Stoßzeiten zusätzliche Server mieten. Die Kosten sind in diesem Fall nur auf die Einsatzzeit beschränkt, wodurch Kosten gespart werden.

Kritiker des Cloud Computings führen Gefahren wie den sogenannten Vendor-Lock-In (siehe Kapitel 2.2.1) sowie generelle Sicherheits- und Datenschutzbedenken an. Fasst man die Gemeinsamkeiten vieler Cloud-Definitionen zusammen, ergibt sich die folgende Definition der Autoren aus [2]:

> Unter Ausnutzung virtualisierter Rechen- und Speicherressourcen und moderner Web-Technologien stellt Cloud-Computing skalierbare, netzwerk-zentrierte, abstrahierte IT-Infrastrukturen, Plattformen und Anwendungen als on demand Dienste zur Verfügung. Die Abrechnung dieser Dienste erfolgt nutzungsabhängig.

Aus dieser Definition ergibt sich aber noch keine eindeutige Unterscheidung zu verwandten Technologien, wie z.B. dem Grid-Computing.

2.1.1 Abgrenzung zu Grid Computing

Grids lassen sich mit der Checkliste [46] von Ian Forster identifizieren. Die Checkliste definiert ein Grid als ein System das

- dezentral gesteuert wird,
- offene, standardisierte und etablierte Schnittstellen und Protokolle nutzt,
- um nicht-triviale Dienstgüten bereitzustellen.

Grids, wie beispielsweise das D-Grid [43] oder das EGEE-Grid [44], sind über mehr als einen Standort verteilt. Die Grid-Middleware läuft unvirtualisiert auf dem Betriebssystem und ist Open Source Software. Die Einsatzgebiete von Grids sind primär Wissenschaft und Forschung. Durch die Einsatzgebiete sind Gridprojekte finanziell durch die öffentliche Hand gefördert.

Cloud Computing Dienste sind, im Gegensatz zu Grid-Systemen, durch eine zentrale Stelle kontrolliert. Sie verwenden sowohl proprietäre Schnittstellen (z.B. Amazon EC2 [21]) im Public Cloud-Bereich als auch offene Schnittstellen im Private Cloud-Bereich [2]. Das Einsatzgebiet von Cloud Computing ist primär die Industrie.

2.2 Cloud-Architekturen

Eine Unterscheidung der existierenden Clouds ist anhand der organisatorischen Architektur möglich. Die Architektur stellt den organisatorischen Kontext zwischen Endnutzer und Anbieter dar. Die Architektur einer Cloud beschreibt auf welche Art und Weise die Cloud ihre Dienste zur Verfügung stellt bzw. diese Dienste erbringt. Es existieren drei unterschiedliche Formen von Cloud-Architekturen. Public-, Private- und Hybrid Cloud unterscheiden sich bzgl. Kosten, Sicherheit und Datenschutz.

2.2.1 Public Cloud

Bei der Public Cloud Architektur gehören Dienst-Anbieter und Dienstnutzer nicht demselben Unternehmen an. Public Cloud Dienste, wie z.B. Amazon Elastic Compute Cloud (EC2), haben einen kommerziellen Hintergrund. Der Dienst wird nicht beim Kunden erbracht, sondern extern bei einem Anbieter (siehe Abbildung 2.1). Der Kunde zahlt nur die selbst genutzten Ressourcen, die der Dienst benötigt. Für den Kunden entfällt, die Notwendigkeit,

eigene Hardware für den benötigten Dienst zu betreiben sowie die damit verbundenen Anschaffungs- und Betriebskosten der Hardware.

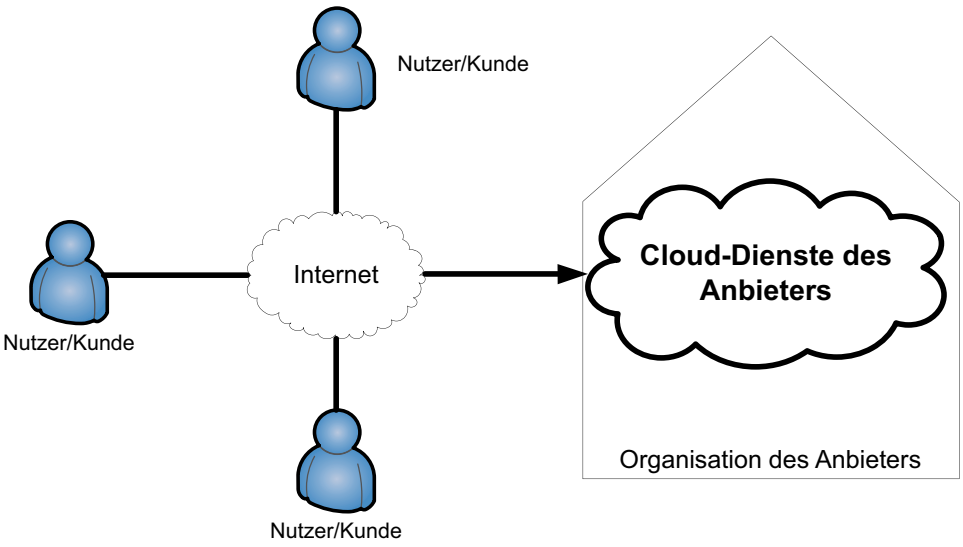

Abbildung 2.1: Architektur einer Public Cloud

Ebenso wie die Diensterbringung, ist die Datenhaltung im Aufgabenbereich des Dienstanbieters. Der Kunde muss daher, aus Sicherheitsgründen, auf sichere Verbindungen zu den Diensten des Anbieters achten. Beispielsweise in Form eines Virtual-Private-Network (VPN) oder einer Secure-Shell (SSH) Verbindung. Des Weiteren muss der Kunde die Datenschutzbestimmungen beachten, sobald er Public Cloud-Dienste in Anspruch nimmt. Ein Risiko für die Verfügbarkeit der Daten ist die Gefahr des sogenannten Vendor-Lock-in. Ein Vendor-Lock-In bedeutet für den Kunden, dass der Anbieter die Kundendaten speichert, der Kunde aber nicht mehr darauf zugreifen kann bzw. es ihm nicht möglich ist, die Daten zu einem anderen Anbieter zu übertragen. Ein denkbares Szenario dafür ist die Insolvenz des Public Cloud-Anbieters.

2.2.2 Private Cloud

Bei der Architektur einer Private Cloud befinden sich sowohl der Diensterbringer (Anbieter), als auch der Dienstnutzer (siehe Abbildung 2.2) in der gleichen Organisation. Der Service wird damit In-House erbracht und es bestehen keine Risiken hinsichtlich Datenschutz und Vendor-Lock-in. Die Organisation des Nutzers muss sich um die Anschaffung und Wartung der Hardware kümmern, was erhöhte Fixkosten bedeutet. Durch den Einsatz von Virtualisierung und der daraus resultierenden Konsolidierung lassen sich

dennoch Kosten, gegenüber einem herkömmlichen Rechenzentrum ohne automatisierte Cloud-Dienste, einsparen.

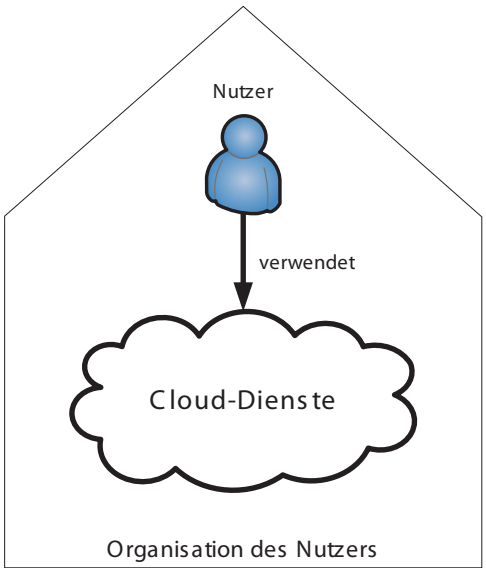

Abbildung 2.2: Architektur einer Private Cloud

Verwendet der Diensterbringer eine standardisierte oder frei verfügbare Schnittstelle zur Kontrolle seiner Private Cloud, ist es möglich, die Private Cloud bei einem kurzfristigen Resourcenengpass mit einem kompatiblen Dienst einer Public Cloud zu einer Hybrid Cloud zu erweitern.

2.2.3 Hybrid Cloud

Die Mischform der Public- und Private Cloud-Architekturen stellt die Hybrid Cloud dar. Bei dieser Organisationsform kommen sowohl Dienste aus Public- als auch aus Private Clouds zum Einsatz (siehe Abbildung 2.3). Da die eingesetzten Dienste sowohl In-House als auch extern erbracht werden, müssen die Clouds via VPN verbunden werden, um ein einheitliches, sicheres virtuelles Netzwerk bereitzustellen.

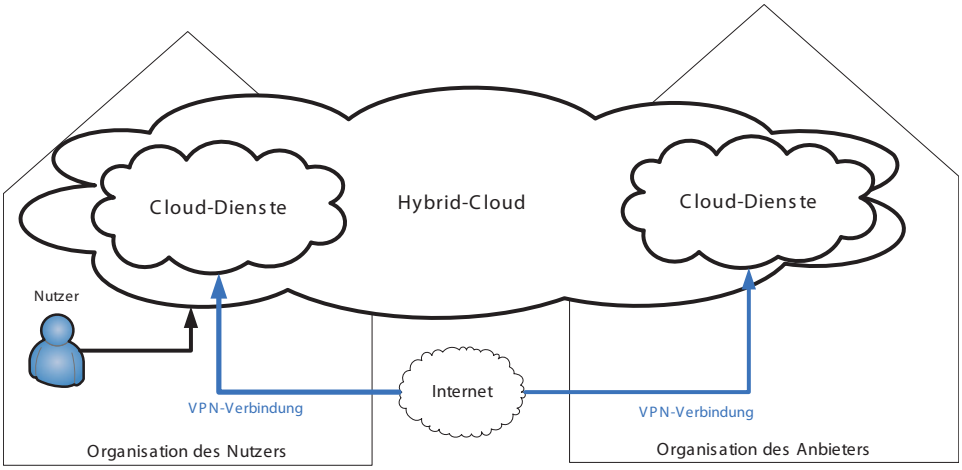

Abbildung 2.3: Architektur einer Hybrid Cloud

Die Gefahr eines Lock-Ins ist, bei Verwendung der Datenlagerung in der Private Cloud, nicht gegeben. Risiken bzgl. des Datenschutzes exisitieren ebenfalls nicht.

Es empfiehlt sich, Private Cloud-Lösungen einzusetzen, deren Schnittstellen zu den Schnittstellen etablierter Public Cloud-Dienstleister kompatibel sind: Die Architektur der Hybrid Cloud eignet sich beispielsweise für die temporäre Ergänzung der Private Cloud-Ressourcen zum Abfangen von Lastspitzen (Peaks).

Die Kosten einer solchen Architektur variieren je nach Benutzungsgrad der Dienste in einer Private- oder einer Public Cloud.

2.3 Cloud-Service Kategorien

Die verschiedenen Cloud-Services sind in vier unterschiedliche Kategorien unterteilt. Jede Kategorie stellt eine unterschiedlich starke Abstraktion von der Hardware dar (siehe Abbildung 2.4). Abstraktion bedeutet nicht, dass eine Kategorie die darunter liegende zwangsweise zur Abstraktion benötigt.

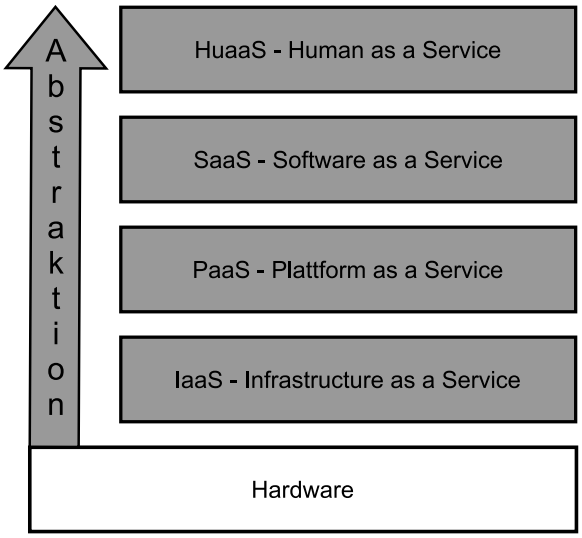

Abbildung 2.4: Abstraktion der Hardware von IaaS zu HuaaS

Um beispielsweise eine Software-as-a-Service Lösung zu realisieren, wird nicht zwangsweise eine Platform-as-a-Service oder eine Infrastructure-as-a-Service benötigt. Die Verwendung der darunter liegenden Dienste empfiehlt sich, da damit ein höherer Grad an Automatisierung möglich ist.

2.3.1 Infrastructure-as-a-Service (IaaS)

Eine IaaS, selten auch Hardware-as-a-Service (HaaS) genannt, stellt dem Dienstnutzer virtuelle Hardware zur Verfügung. Das Hardwareangebot der Anbieter beinhaltet u.a.: Datenspeicher, Netzwerke und komplette Rechner. Die virtuellen Maschinen des Anbieters starten mit fast beliebigen Betriebssystemen. Die eben erwähnten Einschränkungen in der Wahl des Betriebssystems sind vom Betreiber vorgenommen.

Der Dienstnutzer startet mittels einer Schnittstelle (Web-Oberfläche, Fat Client-Anwendung oder einer API) virtuelle Maschinen (Instanzen) in den Rechenzentren des Anbieters. Bei dem Start der Instanzen kann der Nutzer zwischen verschiedenen vorgegebenen Instanztypen wählen. Ein Instanztyp stellt eine Kombination von Hardwareresourcen unterschiedlicher Leistungsfähigkeit dar, die zusammen eine virtuelle Maschine ergeben.

Diese Instanzen kontrolliert der Dienstnutzer mit vollen Administratorrechten. Der Dienstnutzer muss aber, mit den Administratorrechten, sich selbst um die Sicherheit seiner virtuellen Maschinen kümmern. Dazu gehört beispielsweise die Administration der Firewall und das Einrichten bzw. Kontrollieren sicherer Passwörter. IaaS-Dienstleister wie GoGrid [50] oder Amazon

Webservices [21] bieten z.B. die Möglichkeit zum Einrichten und Administrieren einer Firewall für die Instanz direkt in der Kontrollschnittstelle.

2.3.2 Platform-as-a-Service (PaaS)

PaaS-Dienste sind eine weitere Abstraktionsschicht. Der Anbieter stellt eine Softwareplattform zur Verfügung. Die darunter liegende Anzahl der Maschinen bzw. die Infrastruktur ist virtualisiert und skaliert sich aus der Sicht der Nutzer automatisch [13]. Die Zielgruppe der PaaS-Dienste sind Entwickler.

Bekannte PaaS-Dienste, wie z.B. die Google App Engine [51] oder Salesforce Application Development [63], bieten Laufzeitumgebungen und teilweise integrierte Entwicklungsumgebungen oder Werkzeuge zur lokalen Entwicklung von Web-Anwendungen. Die fertigen Anwendungen laufen in der PaaS Cloud des Anbieters. Häufig sind Werkzeuge in Form von Bibliotheken für bestehende Programmiersprachen wie Java oder Phython vorhanden, um existierende Anwendungen leichter in die Cloud portieren zu können.

Die meisten Anbieter von PaaS-Diensten beschränken ihre Dienste dahingehend, dass nur bestimmte Möglichkeiten der Datenspeicherung und Netzwerkkommunikation zur Verfügung stehen. Die Google App Engine z.B. hat nur den Datastore[1] als Möglichkeit der Datenspeicherung und die Kommunikation zu anderen Server ist nur über wenige Ports und Protokolle möglich. Durch diese Einschränkungen ist mit dem Portieren einer Anwendung in eine PaaS Cloud das Risiko des Vendor-Lock-In verbunden.

2.3.3 Software-as-a-Service (SaaS)

Die am längsten etablierte Kategorie von Cloud Services ist Software-as-a-Service. Der SaaS-Anbieter stellt dem Nutzer eine bestimmte Software, in Form eines Webservice, zur Verfügung.

Unter den SaaS-Angeboten gibt es zwei Unterkategorien:

- Die Anwendungsdienste (Application Services), die einen Teil einer kompletten Anwendung als Webservice zur Verfügung stellen. Der Endanwender benutzt diese Teile direkt oder bindet diese in andere Anwendungen ein (beispielsweise Google Maps [53]) [2].
- Die zweite Unterkategorie bilden die Anwendungen (Applications). Dabei handelt es sich um komplette Anwendungen, die der Benutzer über eine Weboberfläche kontrolliert. Beispiele sind Online Email-Angebote wie z.B. Google Mail oder die Google Apps [52]

[1] Datastore ist ein persistenter assoziativer Speicher.

Die Datenhaltung bei SaaS-Diensten ist, wie die Diensterbringung, Aufgabe des Dienstanbieters. Damit besteht die Gefahr des Vendor-Lock-In und Risiken bezüglich des Datenschutzes.

2.3.4 Humans-as-a-Service (HuaaS)

Das Prinzip hinter HuaaS ist das Crowdsourcing [10]. Crowdsourcing ist die Resourcenauslagerung auf die Intelligenz und Arbeitskraft freiwilliger Menschen im Internet. auf Der Anbieter und Ressource beim Crowdsourcing ist der Mensch selbst. Es kann sich ein Mensch als Ressource, gegen geringe Kosten oder als Freiwilliger anbieten. Der Einsatz von HuaaS-Diensten eignet sich für Tätigkeiten, die ein Computer gar nicht, nur schlecht oder erst nach einer unverhältnismäßig hohen Entwicklungszeit erledigen könnte. Beispiele für den Einsatz für Human-as-a-Service-Dienste sind Bilderkennung, Produktbewertungen oder Übersetzungen. Der Schwerpunkt der Dienste beschäftigt sich mit Aufgaben, die auf menschlicher Kreativität basieren.

Ein Freiwilliger, der sich als HuaaS anbieten möchte, kann sich in einem Marktplatz für HuaaS-Dienste anbieten. Ein Markplatz für HuaaS-Dienste ist Amazon Mechanical Turk [23]. Dort kann ein Dienstnutzer Aufträge einstellen, die dann von den teilnehmenden Benutzern abgearbeitet werden können. Die Teilnehmer erhalten ein geringes Entgeld für ihre geleistete Arbeit.

Kapitel 3

Hadoop

Industrielle Vorhaben und wissenschaftliche Projekte produzieren täglich große Mengen an Daten. Das LHC am CERN z.B. produziert jährlich 15 Petabyte an Daten [5]. Die entstehenden Datenmengen müssen nicht nur übertragen und abgelegt, sondern auch verarbeitet werden.

Die New York Times hatte 2007 das Vorhaben, ihre eingescannten Zeitungen von 1851 bis 1980 in das PDF Format zu konvertieren. Die Ursprungsdaten hatten eine Gesamtkapazität von 4 TB und lagen im Dateiformat TIFF vor. Die Zeitungen mussten vor der Speicherung im PDF Format noch skaliert und zugeschnitten werden. Mitarbeiter der New York Times schrieben eine Hadoop MapReduce Anwendung und verteilten diese dann auf 100 Amazon EC2 Instanzen mit S3 als persistenten Datenspeicher. Dieses Vorgehen ermöglichte es, in unter 24 Stunden, 11 Millionen Artikel umzuwandeln [55].

Hadoop ist eine Open-Source Reimplementierung der Google Technologien *MapReduce* und *Google-File-System (GFS)* zur skalierbaren, verteilten Verarbeitung von großen Datenmengen. Hadoop hat seinen Ursprung im Projekt Apache Nutch [32]. In Nutch gab es Probleme die parallelen Berechnungen auf wenigen Rechnern zu kontrollieren. Nachdem Google in den Jahren 2003 und 2004 die Artikel zu MapReduce [6] und dem Google-File-System (GFS) [8] veröffentliche, änderten sich Implementierungsziele im Projekt Nutch [15]. Zwei Entwickler gründeten, zusammen mit einem Team von Yahoo!, das Lucene Subprojekt [31] mit dem Namen Hadoop.

Heute ist Hadoop ein eigenständiges Projekt der Apache Software Foundation und hat einige Subprojekte. Die Hauptbestandteile von Hadoop sind die (Re-)Implementierungen von MapReduce und GFS, namens *Hadoop-Distributed-File-System (HDFS)*. Die Subprojekte beschäftigen sich mit der Erweiterung der MapReduce Technologie. Beispielsweise HBase, das eine skalierbare verteile Datenbank auf Hadoopbasis bietet oder Chuwaka, mit

dem Hadoop Cluster bestehend aus mehreren tausend Knoten überwachbar sind [1].

Ein Ziel der Hadoop-Entwicklung ist Fehlertoleranz. Hadoop erkennt automatisch Ausfälle und reagiert in dem das System versucht, das Problem zu beheben. Die Zielplattform von Hadoop ist die sogenannte *Commodity Hardware*. Das bedeutet, es werden keine kostenintensive Server mit hoher Ausfallsicherheit bzw. Hochverfügbarkeit benötigt.

Hadoop ist in Java implementiert. Die MapReduce-Anwendungen werden von Hadoop gestartet und laufen ebenfalls innerhalb der Java-Virtual-Machine (JVM). Demnach benötigten die MapReduce-Anwendungen Java als Entwicklungssprache. Hadoop enthält auch Erweiterungen, wie das *Hadoop Streaming Utility* oder *Hadoop Pipes*. Mit dem Hadoop Streaming Utility ist es möglich, alle Shell-Skripte oder Anwendungen, als MapReduce-Anwendung zu starten. Die Hadoop Pipes stellen eine *Simplified Wrapper and Interface Generator* (SWIG)-kompatible API dar, mit der MapReduce-Anwendungen nativ in C++ implementierbar sind.

Um zu verstehen wie Hadoop aufgebaut ist, beschäftigen sich die folgenden beiden Unterkapitel mit dem allgemeinen Konzept MapReduce, dem verteilten Dateisystem und deren Zusammenarbeit. Das darauf folgende Kapitel 3.3 behandelt einige Hadoop-Erweiterungen.

3.1 MapReduce

MapReduce ist ein Konzept um das Entwickeln verteilter Anwendungen zu vereinfachen. Es besteht aus drei Teilen:

- Ein *Programmiermodell*, das vorgibt, wie eine Anwendung implementiert werden muss, damit diese im MapReduce Framework starten kann.
- Ein Framework, das die Laufzeitumgebung für MapReduce-Anwendungen darstellt.
- Eine Job-Konfiguration, die das Framework einstellt. Die Einstellungen sind die Parameter, mit denen das Framework agiert (z.B. Formate der Ein- und Ausgabedateien, Anzahl der Mapper bzw. Reducer, etc. Eine Übersicht über existierenden Optionen ist in [29] zu finden).

Um das Prinzip hinter MapReduce verständlicher zu machen, geht diese Arbeit genauer auf das Framework sowie das Programmiermodell ein.

3.1.1 MapReduce Programmiermodell

Das Programmiermodell von MapReduce ist simpel gehalten. Es besteht aus zwei Funktionen, mit deren Hilfe Entwickler die Anwendung abbilden müssen. Die beiden Funktionen sind *map* und *reduce*. Die *map* Funktion ist wie folgt definiert:

$$map(key1, value1) \rightarrow list(key2, value2)$$

Die Funktion benutzt die Schlüssel/Wert Paare aus der Eingabe und bildet eine Zwischenliste von Schlüssel/Wert Paaren als Ausgabe. Das Framework übernimmt die Aufgabe, alle enstandenen Zwischenlisten anhand der Schlüssel zu gruppieren. Diese Gruppierungen werden anschließend an die Reduce-Funktion weitergeleitet. Die *reduce* Funktion ist wie folgt definiert:

$$reduce(key2, list(value2)) \rightarrow list(value2)$$

Die *reduce*-Funktion hat das Ziel die einzelnen Gruppierungen wieder zusammenzuführen. Das Zusammenführen kann in Form von beliebigen Berechnungen stattfinden oder ganz weggelassen werden. Normalerweise liefert die *reduce*-Funktion entweder *einen* oder *keinen* Wert am Ende der Funktion [6].

Durch dieses *divide and conquer* Prinzip bei MapReduce lassen sich Aufgaben aufteilen, die bei einem traditionellen Vorgehen nicht in einen Speicher eines Rechners passen.

3.1.2 MapReduce Framework

Das Framework, in dem die Implementierung läuft, besteht aus einer Master-Slave-Architektur.

Der Master, genannt *Jobtracker*, übernimmt die Kontrolle für die sogannten MapReduce-*Jobs*. Ein MapReduce-Job ist eine MapReduce-Anwendung, die im Framework läuft. Der Jobtracker spaltet jeden Job in Map- und Reduce-*Tasks* auf, die die Slaves (sog. *Tasktracker*) abarbeiten. Er überwacht die Verfügbarkeit der einzelnen Tasktracker und reagiert auf eventuelle Fehler oder Abstürze mit dem Neuverteilen des fehlgeschlagenen Tasks. Der Jobtracker informiert sich zur Laufzeit bei den Tasktrackern, welchen Status (`idle`, `in progess` oder `completed`) die einzelnen Tasks haben [6].

Durch das Aufteilen der Jobs und Überwachen der Tasks, spart das MapReduce Konzept Laufzeit, da bei einem Fehler nur ein kleiner Teil eines Jobs wiederholt werden muss [9].

Das Framework kümmert sich um das Einlesen der Eingabedaten bzw. um das Schreiben der Ausgabedaten eines MapReduce Jobs. Die Implementie-

rung von Hadoop sieht dazu wenige vorgefertigte Datentypen vor, die das Framework von Haus aus einlesen und aufteilen kann.

3.1.3 Beispiel einer MapReduce-Anwendung

Die Funktionsweise der beiden MapReduce Komponenten ist hier anhand eines Beispiels dargestellt. Das Beispiel ist die, von den Hadoop-Entwicklern beigefügte Anwendung WordCount [28]. Ziel der Anwendung ist es, aus einer Textdatei, die Anzahl der Vorkommnisse der einzelnen Wörter zu zählen. Um die Implementierung nicht im Detail darzustellen, beschreibt diese Arbeit nur die Aufgaben der beiden Funktionen und geht anschließend auf den Ablauf eines MapReduce-Jobs ein:

- *Mapper*: Die Implementierung des Mappers erhält einen String geliefert, den der Mapper in die einzelnen Wörter aufspaltet. Um ein Schlüssel/Wert Paar zu erstellen, verknüpft der Mapper jedes Wort mit jeweils einem Integer mit dem Wert 1.
- *Reducer*: Der Reducer erhält als Eingabe das einzelne Wort als Schlüssel und eine Liste der dazugehörigen Werte aus dem Map-Vorgang. Im Wordcount-Beispiel ist jeder dieser Werte eine 1. Nach der Summierung der Integer verknüpft der Reducer, als Ausgabe, das Wort mit der Summe.

Weder der Mapper noch der Reducer haben als Aufgabe das Einlesen oder Schreiben von Daten. Nach dieser Beschreibung der Aufgaben von Mapper und Reducer folgt der Ablauf der Anwendung:

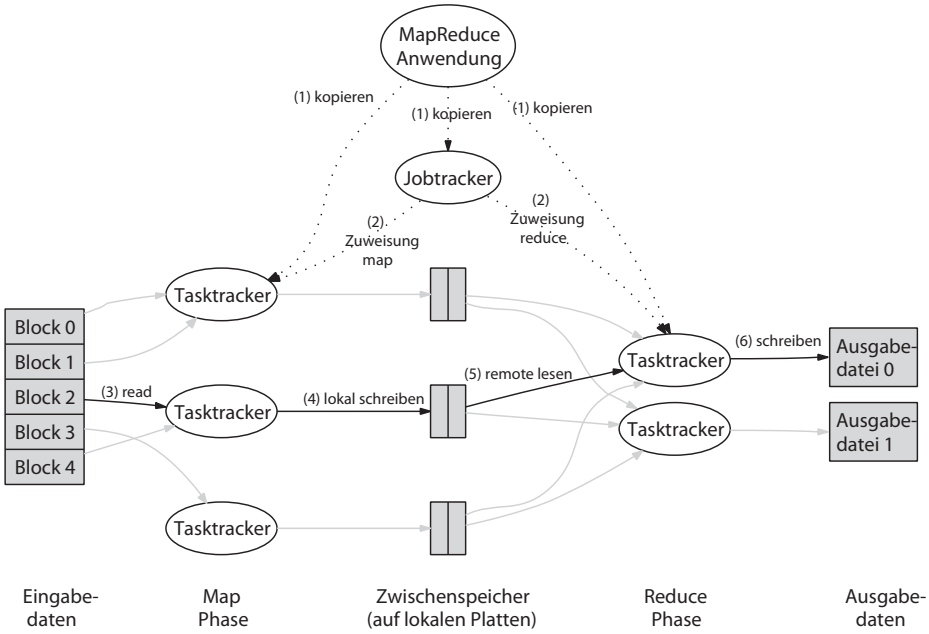

Abbildung 3.1: Ablauf einer MapReduce-Anwendung [6]

Abbildung 3.1 zeigt den Ablauf einer MapReduce-Anwendung im Framework, der aus dem Google Artikel [6] abgeändert übernommen wurde. Die eingeklammerten Zahlen im Bild entsprechen der Nummerierung in der folgenden Ablaufbeschreibung:

1. Zuerst teilt das Framework die Eingabedaten in M Blöcke auf, um die Eingabedaten auf einem Tasktracker verarbeiten zu können. Die Anzahl der Blöcke ist in der Job-Konfiguration einstellbar. Typischerweise ist ein Block zwischen 16 und 64 MB groß. Das Framework kopiert die MapReduce-Anwendung auf die verschiedenen Maschinen des MapReduce Clusters.

2. Mit Hilfe des Jobtrackers des Clusters weist das Framework den einzelnen Tasktrackern jeweils einen Map- bzw. einen Reduce-Task zu.

3. Die Tasktracker, die einen Map-Task erhalten haben, lesen jeweils ihren zugeteilten Block der Eingabedaten. Das Framework analysiert die Eingabedaten und leitet diese zur Map-Funktion weiter. Bei der Analyse teilt eine Hadoopkomponente den Eingabeblock in verschiedene Schlüssel/Wert Paare auf. Die Anzahl und Format der Paare ist implementierungsabhängig von der Java-Schnittstelle `InputFormat`. Die Implementierung dieser Schnittstelle gibt vor wie die Eingabedaten aufzuteilen sind, um damit die für die Map-Funktion nötigen Schlüssel-Werte

Paare zu erzeugen. Die im WordCount-Beispiel entstanden Wort/Zahl Paare aus der Map-Funktion sind im Arbeitsspeicher gepuffert.

4. Die gepufferten Daten schreibt das Framework in Intervallen auf die Festplatte zur Datensicherung. Der Tasktracker partitioniert diese Daten in R Stücke mit Hilfe eines Partitionsalgorithmus (dem sogenannten Partitionier). Die Art und Weise der Datenaufteilung hängt von der Implementierung des Partitioners (Implementierung der abstrakten Klasse `Partitioner`) ab. Die Anzahl R entspricht der Anzahl der in der Job-Konfiguration eingestellten Reducertasks. Der Tasktracker informiert den Jobtracker, welche gepufferten Paare er zur Verfügung stellt. Die Daten dieses Vorgangs benötigt der Reducer zu einem späteren Zeitpunkt.

5. Der vom Jobtracker gestartete Reducer benutzt Remote-Procedure-Calls (RPC) um die gepufferten Ergebnisdaten der Map-Funktion zu erhalten. Der Reducer sortiert und gruppiert die gepufferten Daten nach Wörtern mittels der Implementierung der Javaschnittstelle `Comparator`.

6. Die Gruppierungen durchlaufen die Reduce-Funktion, die diese aggregiert. Die Ausgabe wird anschließend in die definierte(n) Ausgabedatei(n) geschrieben.

3.2 Das verteilte Dateisystem HDFS

Der zweite Teil eines Hadoop Clusters ist das *Hadoop-Distributed-File-System (HDFS)*. Es ist eine Open-Source-Reimplementierung des Google-File-Systems. Bei diesem handelt es sich um ein fehlertolerantes, verteiltes Dateisystem für große Datenmengen ab dem Gigabyte-Bereich.

HDFS basiert, wie MapReduce, auf der Master-Slave Architektur. Der Master in einem HDFS-Cluster, *Namenode* genannt, ist die zentrale Verwaltungsstelle des HDFS Clusters. Er speichert keine Nutzdaten, sondern ausschließlich Metadaten (z.B. die Zuordnung, welcher Slave welche Daten speichert). Die Slaves in einem HDFS Clusters, die als *Datanodes* bezeichnet werden, speichern nur die Nutzdaten. Durch die Verwendung von Commodity Hardware als Basis ist eine häufige Ausfallquote der Hardware gegeben. Daher überwacht der Namenode, wie der Jobtracker auch, die Verfügbarkeit der einzelnen Datanodes und verteilt die Daten neu sobald ein Datanode ausfällt [8].

Die Speicherung der Daten erfolgt, genau wie bei einem nicht verteilten Dateisystem, in Form von Dateien und Verzeichnissen. Die Daten eines

HDFS Clusters sind in Blöcken gleicher Größe auf den Datanodes verteilt. Ein Daten-Block hat in der Standardeinstellung die Größe von 64 MB und ist drei mal im Cluster repliziert [15]. Die Anzahl der Replikationen lässt sich mittels eines Konfigurationsparameters einstellen.

Namenodes lassen sich mit einem sogenannten Master-File-Table eines nicht-verteilten Dateisystems vergleichen. Beide speichern nur Metadaten des Dateisystems. Ohne diese Metadaten ist das Dateisystem nicht funktionsfähig. Daher ergibt sich bei HDFS auch ein Single-Point-of-Failure: Fällt der Namenode eines HDFS Clusters aus oder verliert die Metadaten, ist das Dateisystem des Clusters zerstört [8].

Bei der Kommunikation zwischen einem Client, der Daten anfrägt oder sendet, agiert der Namenode nur als Verzeichnisdienst. Wie in Abbildung 3.2 zu sehen ist, sendet bzw. empfängt ein Client die Nutzdaten in Verbindung mit einer Datanode.

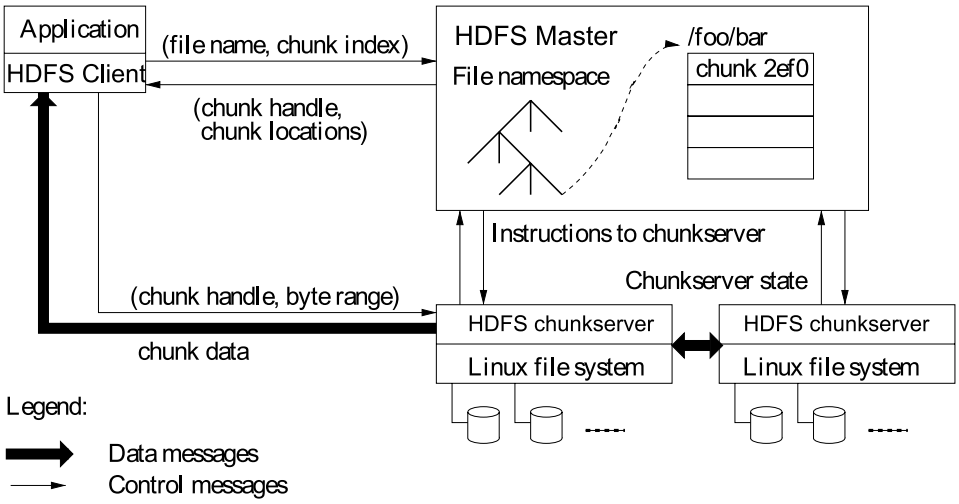

Abbildung 3.2: Architektur HDFS [8]

Wie zu Beginn des Abschnitts erwähnt, ist ein Entwicklungsziel von HDFS große Dateien zu verwalten. Dateien in einem HDFS Cluster sind typischerweise größer als 100 MB. Das Dateisystem verwaltet Dateien dieser Größe effizient. Das Ablegen und Auslesen kleinerer Dateien ist möglich, allerdings ist das System dazu nicht optimiert [15]. Der Grund für dies ist in Kapitel 3.2.1 erläutert.

3.2.1 Lese - und Schreibvorgänge in HDFS

Das Auslesen kleiner Dateien ist verhältnismäßig nicht effizient, wie das Auslesen großer Dateien. Die dazu führenden Gründe sind die Größe eines HDFS Blocks und das Vorgehen beim Zugriff auf die Daten, das im folgenden Abschnitt beschrieben ist.

Im Gegensatz zu normalen Dateisystemen belegt eine Datei, die kleiner ist als ein HDFS Block, nicht einen kompletten HDFS Block [15]. Der Prozess des Auslesens einer Datei in HDFS hat zwei Schritte. Die Erklärung der Schritte orientiert sich an Abbildung 3.2

1. Der HDFS Client versucht die Datei `/foo/bar` aus dem Cluster zu lesen. Der Client ruft einen Webservice auf der Namenode mit dem Dateinamen `/foo/bar` als Parameter. Der Namenode überprüft im sogenannten *Namespace*, der sich im Arbeitsspeicher befindet, auf welchen Datanodes sich die Blöcke (auch *chunks* genannt) befinden. Der Namenode liefert dem Client sogenannte *chunk-handles*, die Identifizierer eines Blocks auf den Datanodes darstellen sowie eine Liste von Datanodes, die die Blöcke speichern.
2. Der HDFS Client ruft die Schnittstelle eines oder mehrerer Datanodes, um die Nutzdaten zu erhalten. Der Datanode liest mit Hilfe der Identifizierer die HDFS Blöcke von der Festplatte und überträgt diese als Ergebnis zum Client.

Durch diese zwei Schritte dauert das Auslesen vieler kleiner Dateien aus folgenden Gründen länger als das Auslesen weniger großer Dateien [30]:

- Kleine Dateien bedeuten mehr Last und Laufzeit für der Namenode, da diese länger nach chunk-handles und Datanodes sucht.
- Das sequentielle Auslesen vieler kleiner Dateien bedeutetet eine höhere Netzwerklast, da die Clients mehr Datanodes ansprechen müssen.
- Der Namenode speichert im Arbeitsspeicher die Zuordnung zwischen Dateiname bzw. Ordnername, deren Datenblöcke und die Replikationsorte. Eine Zuordnung belegt laut [15, 30] ca. 150 byte. Bei 10 Millionen Einträgen für Dateien und Ordner benötigt HDFS auf dem Namenode für die Zuordnungen ca. 3 GB. Die Suche eines Eintrags sowie das automatisierte Backup der Metainformationen, dauern bei vielen kleinen Dateien länger, da das Betriebssystem bei zu geringer Arbeitsspeichergröße die Daten auf die Festplatte auslagert.

Die Schreibprozedur einer Datei in HDFS erfolgt in folgenden Schritten [15]:

1. Als erster Schritt kontaktiert der HDFS Client die Webservice-Schnittstelle des Namenode, um wie in Abbildung 3.2 dargestellt, die Datei `bar` im Ordner `/foo/` zu speichern. Der Namenode überprüft, ob der Client die Berechtigung hat, bzw. ob die Datei bereits existiert. Sind die Überprüfungen positiv, speichert der Namenode die Metainformation der Datei in Namespace und der HDFS Client führt den nächsten Schritt durch. Falls die Datei vorhanden ist oder der Client keine Schreibberechtigung hat, bricht der Namenode den Prozess durch eine Exception ab. Durch das Überprüfen des Namenode, ob eine bestimmte Datei bereits vorhanden ist, ist es in HDFS nicht möglich Dateien zu überschreiben. Ein Überschreiben der Datei ist nur mit dem Löschen und Wiedereinstellen der Datei möglich.

2. Der HDFS Client teilt die zu speichernde Datei in Blöcke auf und stellt diese in eine Queue. Der Client arbeitet die Queue iterativ ab und ruft bei jedem Block die Webservice-Schnittstelle des Namenode auf, der eine Liste von Datanodes zurückliefert, die den Datenblock speichern soll. Zusätzlich erhält der Client einen Identifizierer für den Block. Die Auswahl der Datanodes zur Speicherung eines Blocks ist konfigurationsabhängig.

   Es existieren Konfigurationsmöglichkeiten, wie beispielsweise die Definition von Racks, um physisch benachbarte Server zu einem virtuellen Rack zusammenzuschließen. Das damit verfolgte Ziel ist das Einsparen von Netzwerkverkehr und Laufzeiten über Rechenzentren hinweg.

3. Der HDFS Client überträgt den Block und die Liste der Datanodes zu einem Datanode, der diese mit dem Identifizierer lokal abspeichert.

4. Nach der erfolgreichen Übertragung des Blocks, leitet der Datanode den Block zur Speicherung zu einem weiteren Datanode der Liste. Die zuletzt angesprochene Datanode wiederholt den Vorgang mit einer weiteren Datanode aus der Liste, bis die konfigurierte Anzahl der Replikation eines Blocks erreicht ist.

Auf Lese- oder Schreibfehler reagiert HDFS automatisch mit der Ausgrenzung der Node aus dem Cluster und der Neureplikation der Blöcke, die auf der Node sind bzw. sein sollen.

3.2.2 Weitere Dateisysteme und Zugriffsprotokolle in Hadoop

Hadoop unterstützt, außer HDFS, weitere Dateisysteme bzw. Zugriffsprotokolle für MapReduce-Anwendungen, die im folgenden Abschnitt kurz beschrieben sind [15]. Jedes Dateisystem verwaltet die Daten mit Hilfe von

Dateien und Ordnern. Der Zugriff auf die Dateien in einem Dateisystem mit Hadoop erfolgt immer über einen *Uniform Resource Identifier (URI)*, der gleichzeitig den Dateisystemtyp identifiziert. Die Dateisysteme sind für spezielle Anwendungsfälle entwickelt.

Local File System

Das Local File System ist eine Implementierung, um direkt auf das, auf den Clients darunterliegende Dateisystem bzw. Festplatten, zuzugreifen. Das Dateisystem ist kein verteiltes Dateisystem wie das HDFS. Falls der Nutzer eines Local File Systems eine Datenredundanz benötigt, muss er diese auf Betriebssystemebene mit Hilfe von RAID oder verteilten Dateisystemen wie OCFS [60] konfigurieren. Das Local File System ist in MapReduce-Anwendungen mit dem URI Prefix `file://` anzusprechen.

Hadoop File Transfer Protocol (HFTP) und HSFTP

Das *Hadoop File Transfer Protocol (HFTP)* ist eine Zusatzimplementierung für HDFS. Mit Hilfe von HFTP können Anwendungen über eine HTTP Verbindung auf ein HDFS zugreifen. In der Praxis verwenden Benutzer HFTP für das Kopieren eines HDFS in ein anderes HDFS-Cluster mit Hilfe des `distcp`-Kommandozeilenwerkzeugs [27] von Hadoop, um beispielsweise verschiedene Versionen der HDFS Dateien zu speichern. Der Vorteil von HFTP gegenüber dem nativen HDFS Protokoll ist, dass es über HTTP kommuniziert und damit keine Firewallanpassungen nötig sind. HFTP bietet nur einen read-only Zugriff auf ein HDFS. Das *Hadoop Secure File Transfer Protocol (HSFTP)* ist, bis auf die Verwendung von HTTPS als Kommunikationsprotokoll, mit HFTP identisch. Der Zugriff auf HDFS mit dem H(S)FT Protokoll erfolgt über den URI Prefix `hftp://` bzw. `hsftp://`.

Hadoop Archive Filesystem (HAR)

Das *Hadoop Archive Filesystem (HAR)* ist entwickelt, um das in Kapitel 3.2.1 angesprochene Arbeitsspeicherproblem der Namenode bei kleinen Dateien zu lösen [30]. Die Implementierung von HAR baut auf HDFS auf. Das HAR Dateisystem speichert Archive, ähnlich ZIP Archiven in HDFS ab. Die Archive beinhalten die zu speichernden Dateien. Der Zugriff auf die Dateien erfolgt mit dem URI Prefix `har://` , wobei der Archivname nach dem Servernamen anzugeben ist. Beispielsweise: `har://meinserver:serverport/Archiv--pfad-und-Name/Dateipfad-und-Name-im-Archiv`.

KFS/CloudStore

CloudStore [41], früher bekannt als Kosmos File System, ist ein in C++ entwickeltes, verteiles Dateisystem. Das technische Design bzw. die Architektur von CloudStore ist vergleichbar mit der des HDFS bzw. GFS [15]. CloudStore speichert, wie HDFS, größere Blöcke auf Blockservern ab, die vergleichbar mit einer Datanode sind. Die Blockserver des CloudStore Dateisystems verwenden die Dateisysteme ZFS oder XFS auf Betriebssystemebene als Basis zur Speicherung der Blöcke. CloudStore ist eine eigenständige Entwicklung, die nicht von der Apache Software Foundation getrieben ist. Der Zugriff von MapReduce-Anwendungen auf Daten in einem CloudStore Dateisystem erfolgt über den URI Prefix `kfs://`.

FTP

Ein weiteres von Hadoop unterstütztes Protokoll in MapReduce-Anwendungen ist das FTP Protokoll. Hadoop bietet die Möglichkeit, dass MapReduce Anwendungen über das FTP auf HDFS Cluster zugreifen können. Dabei startet HDFS einen FTP Server im Hintergrund, der die Befehle des FTP in HDFS Befehle umwandelt. Der Prefix einer URI für eine Verbindung zu einem FTP Server ist `ftp://`

S3 btw. S3N

Hadoop unterstützt den Zugriff auf Daten in S3 über die Protokolle S3, der über HTTP bzw. HTTPS realisiert ist, sowie über das native S3N Protokoll. Bei der Implementierung des S3 Filesystems speichert Hadoop die Dateien und Ordner in Blöcken in den S3 Buckets ab, um das Limit der 5 GB Dateigröße zu umgehen. Der URL Prefix für eine Verbindung zu S3 ist `s3://` bzw. `s3n://`

3.3 Weitere Subprojekte

Es existieren mehrere Subprojekte zu Hadoop. Unter anderem Pig und Hive. Beide beschäftigen sich mit Datenbanken auf Hadoop-Basis.

3.3.1 Hive

Hive ist eine Hadoop-Erweiterung, die mit HDFS und einer relationalen Datenbank ein Data-Warehouse bildet.

Ein Data-Warehouse (dt: Datenlager) ist eine Datensammelstelle in Form einer Datenbank. Ein Data-Warehouse bezieht oder erhält die Daten von verschiedenen Quellen, beispielsweise aus weiteren Datenbanken. Ein Data-Warehouse ermöglicht eine Sicht auf heterogene, verteilte Datenbestände in einer konsistenten Datenbank. Die Konsistenz erreicht ein Data-Warehouse durch das Kopieren der Daten aus den verschiedenen Quellen in den eigenen Datenbestand mittels eines sogenannten *Loaders*. Die Hauptanwendung bei Data-Warehouses ist das sogenannte *Online-Analytical-Processing (OLAP)*.

Ein Anwender kann mittels OLAP mehrdimensionale, komplexe Abfragen auf den asynchronen Daten ausführen. Ein Beispiel für eine mehrdimensionale Abfrage ist die Fragestellung eines fiktiven Getränkehändlers, wie viel Umsatz er in den Jahren 2001 und 2002 mit Bier und Wasser in den Bundesländern Baden-Württemberg und Bayern gemacht hat. Die Dimensionen dieser Abfrage sind Zeit, Warengruppe, Region und Umsatz. OLAP Anfragen analysieren eine große Menge an Datensätzen und dauern im Verhältnis zu den sogenannten *Online-Transactional-Processing(OLTP)*-Anfragen der traditionellen relationalen Datenbanken länger.

Hive bietet einen sogenannten *Extract-Transform-Load (ETL)* Mechanismus, mit dem sich aus verschiedenen Datenquellen das Data-Warehouse füllen lässt. Die Kontrolle über das Data-Warehouse erfolgt über die *Hive-Shell*. Die Hive-Shell akzeptiert Befehle, die einer SQL-ähnlichen Sprache, der sogenannten *Hive-Query-Language (HiveQL)* entsprechen [35]. Die HiveQL-Anweisungen können entweder mittels interaktiver Eingabe in der Shell, mittels Skripting, via Webinterface oder mittels JDBC/ODBC-Schnittstelle erfolgen.

Ein Compiler nimmt die Befehle der Shell bzw. der Skripte entgegen und wandelt diese in MapReduce-Jobs, HDFS-Befehle bzw. Metadaten um. HiveQL unterstützt sowohl Befehle in der Data-Definition-Language (DDL), als auch in der Data-Manipulation-Language (DML).

Die HiveQL-DDL kennt primitive Datentypen wie `tinyint`, `smallint`, `int`, `bigint`, `boolean`, `float`, `string`, verschachtelte Datentypen `struc`, `map`, `array` und benutzerdefinierte Datentypen, die mittels einer Schnittstelle zur (De-)Serialsierung (`SerDe` genannt) implementiert sind.

Hive unterscheidet bei der internen Datenspeicherung zwischen zwei Datenarten. Zum einen speichert Hive die Metadaten des Warehouses in einer relationalen Datenbank, dem sogenannten *Metastore*. Zum anderen sind die Nutzdaten im HDFS abgelegt. Geeignete Anwendungsfelder von Hive sind beispielsweise Textmining, Indexierung von Dokumenten oder Hypothesentests [39].

3.3.2 Pig (Latin)

Pig ist eine Anwendung, die semi-strukturierte, struktierte oder relationale Daten in großen Dateien analysiert [47]. Pig ist mittels der Pig-Shell (*Grunt*) benutzbar. Alternativ lädt Grunt ein Skript, um die Befehle in einem Batch-Modus durchzuführen. Die Befehle, die Pig annimmt, entsprechen der deklarativen Sprache *Pig Latin*, mit deren Hilfe Pig die Daten analysiert.

Pig kennt die Datentypen `int`, `long`, `float`, `double`, `chararray` (analog in Java: String), `bytearray` (vgl. BLOB in relationalen Datenbanken), `tuple` (ein geordnete Liste an Feldern), `bag` (eine Collection aus tuples) und `map`.

Ziele der Pig-Entwickler sind unter anderem [26]:

- Pig liest alle Datenformate ein, unabhängig von deren Aufbau.
- Aktuell[1] baut Pig auf Hadoop auf. Die Entwickler haben zusätzlich das Ziel Pig auch in andere Umgebungen zu portieren.
- Ein weiteres Ziel ist die Einbettung von Benutzer-code. Der Benutzer bindet, wenn immer es möglich ist, seinen Code in Pig ein, um alle Ein- bzw. Ausgabeformat zu unterstützen.

Die Erweiterungen von Hadoop entstehen zum großen Teil bei Yahoo!. Yahoo! ist der Hauptcontributor bei der Hadoop-Entwicklung.

[1] Stand Mai 2010

Kapitel 4

Vorbild im kommerziellen Sektor Amazon Elastic MapReduce

Amazon Elastic MapReduce ist ein Service der Amazon Webservices [22]. Elastic MapReduce kann, mit Hilfe eines Hadoop Clusters, MapReduce Jobs abarbeiten. Amazon benutzt die Dienste EC2- bzw. S3 zum Installieren des Clusters und zum Ablegen der Eingabe- bzw. Ausgabedaten.

Amazon Elastic Compute Cloud (EC2) [21] ist ein IaaS der Amazon Webservices (AWS). EC2 bietet die Möglichkeit virtuelle Instanzen aus einem Betriebssystemimage zu starten. Jede Instanz hat einen sogenannten Instanztyp, der eine bestimmte, von Amazon definierte, Hardwarekonfiguration darstellt. Der Benutzer wählt den Instanztyp beim Start der Instanzen aus. Die Cloud kopiert das Image, der gestarteten Instanzen, auf eine virtuelle temporäre Festplatte. Nach dem Terminieren der Instanz löscht EC2 diese virtuelle Festplatte. Amazon bietet, für die Persistenz der Nutzerdaten einer Instanz, den sogenannten *Elastic-Block-Store* (EBS). Eine virtuelle Festplatte des EBS, *EBS Volume* genannt, ist mit maximal einer Instanz verbindbar und nicht außerhalb EC2 zugänglich.

Ein anderer Dienst der AWS ist der *Simple Storage Service (S3)* [24]. S3 ist ein persistenter Datenspeicher, der Dateien oder Order jeweils in einem sogenanntem *Bucket* speichert. Der Nutzer kann jedes Bucket mit einem von ihm definierten Schlüssel auslesen oder verändern. Ein Nutzer kann mit S3 über eine native oder HTTP(S) Verbindung Daten in, aus, bzw. innerhalb der Amazon Cloud übertragen.

Die Anzahl der Instanzen des zu startenden EMR Clusters legt der Nutzer selbst fest. Amazon berechnet für diesen Dienst die Laufzeit der EC2-

Instanzen sowie das genutzte Datenvolumen in den S3 Buckets bzw. den Datentransfer in die und aus der Amazon Cloud [22]. Abbildung 4.1 enthält eine Übersicht über ein typisches Hadoop Cluster in EC2.

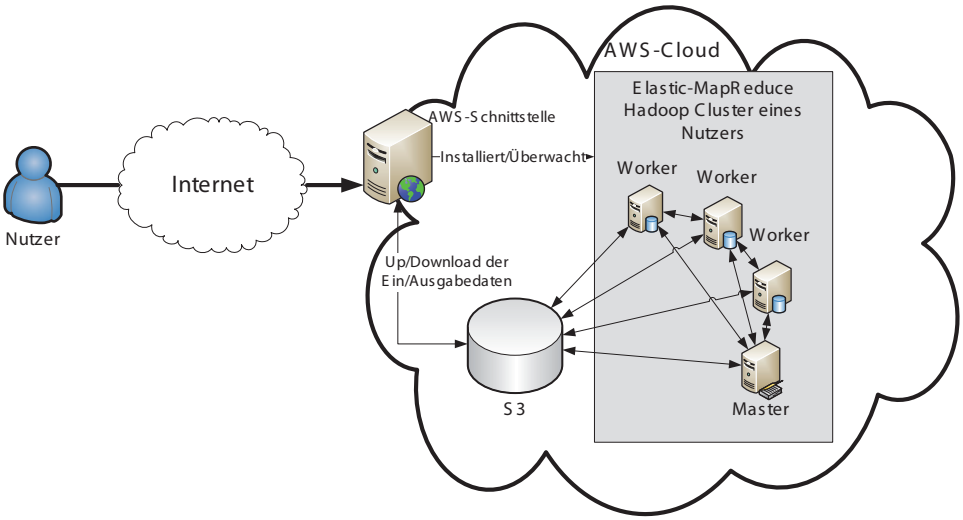

Abbildung 4.1: Aufbau des Dienstes Amazon Elastic MapReduce

Die Abbildung zeigt wie der Nutzer indirekt auf ein Hadoop Cluster zugreift. Der Zugriff erfolgt über die AWS-Schnittstelle, mit Hilfe der Nutzer Daten in S3 hochlädt oder MapReduce-Anwendung startet.

Um eine MapReduce-Anwendung bei Amazon MapReduce zu starten muss ein sogenannter *Job-Flow* definiert sein. Ein Job-Flow ist eine Konfiguration eines Hadoop Clusters, die in EC2 ausgeführt wird. Die Konfiguration beinhaltet z.B. den Instanztyp oder MapReduce relevante Parameter. Jeder Job-Flow in Elastic MapReduce ist in sogenannte *Job-Steps* bzw. *Steps* aufgeteilt. Ein Step ist entweder eine MapReduce-Anwendung, die Elastic MapReduce ausführt, ein Konfigurationsskript oder ein Konfigurationsbefehl. Elastic MapReduce führt alle Steps sequentiell aus. Bei Steps unterscheidet Elastic MapReduce zwei Steptypen:

- *Konfigurationssteps* (Konfigurationsskripte oder Konfigurationsbefehle), die EC2-Instanzen konfigurieren,
- und *MapReduce-Steps*, die den eigentlichen Job darstellen.

Elastic MapReduce führt die Konfigurationssteps vor allen andern Steps durch [19], um zuerst das Cluster zu konfigurieren und anschließend die MapReduce-Anwendungen durchzuführen.

4.1 Definieren und Starten eines Job-Flows

Das Anlegen bzw. Starten eines Job-Flows in Elastic MapReduce ist auf drei verschiedenen Wegen bzw. drei verschiedenen Schnittstellen möglich [18–20]:

- Mit der von Amazon bereitgestellten Weboberfläche, *https://console.aws.amazon.com/elasticmapreduce/home*
- mit den Elastic MapReduce Kommandozeilen-Werkzeugen,
- oder mit den Elastic MapReduce SOAP and Query APIs.

Alle drei Wege haben die gleichen Funktionalitäten bzw. das gleiche Prinzip im Ablauf. Der Ablauf ist, zuerst einen Job-Flow zu definieren, um diesen dann zu starten. Die Definition eines Job-Flows besteht aus vier Informationensbereichen [19]:

- *Jobart*: Mit der Jobart ist definiert, ob der Job eine eigene JAR-Datei, ein Pigscript, ein Hivescript oder ein Hadoop Streaming Programm sein soll. Die Jobarten Pig und Hive können auch interaktiv gestartet werden. Interaktiven Jobarten stellen nach dem Start des Clusters einen SSH Port bereit, auf dem die Pig oder Hive Shell kontrolliert werden kann.
- *Jobparameter*: Die Jobparameter fallen nur bei nicht-interaktiven Jobs an. Die Parameter definieren die Ein- und Ausgabe Buckets sowie den Speicherort des Skripts bzw. der JAR-Datei. Im Falle des Hadoop Streamings sind zusätzlich noch Angaben für Mapper- und Reducerprogramm zu machen.
- *Instanzparameter*: Die Performance des Clusters lässt sich durch die Instanzparameter beeinflussen. Elastic MapReduce bietet die Möglichkeit, die Typen der zu startenden EC2-Instanzen zu verändern. Zusätzlich lassen sich Debugging-Optionen konfigurieren. Mit aktiver Debugging-Option schreibt Hadoop die `controller-`, `stderr-`, `stdout-` und `syslog`-Ausgaben nach dem Ausführen in den angegebenen S3-Bucket.
- *Bootstrap*: Amazon bietet an, Konfigurationsoptionen für Hadoop, dessen Hintergrunddienste (*daemons*) oder ein eigenes Skript einzustellen bzw. auszuführen.

Nachdem die jeweils nötigen Parameter definiert sind, lässt sich der Job-Flow mit Hilfe der beschriebenen Schnittstellen starten und Elastic MapReduce installiert das Cluster.

4.2 Ablauf eines Job-Flows

Sind die Programme/Skripte und die Eingabedaten in die jeweils eingestellten S3 Buckets geladen, kann ein Job-Flow gestartet werden. Der Ablauf eines Job-Flows ist anhand der Job-Flow-Definition von Elastic MapReduce gesteuert. Während der Laufzeit kann ein Job-Flow die folgenden sieben Zustände erreichen, die Auskunft über den aktuellen Status geben [18]:

- *Completed*: Das Hadoop Cluster ist heruntergefahren, nachdem alle Steps abgeschlossen sind.
- *Failed*: Nachdem ein Fehler in einem Step aufgetreten ist, ist das Hadoop Cluster heruntergefahren.
- *Running*: Ein Step ist in Bearbeitung.
- *Shutting Down*: Das Cluster fährt herunter und die Instanzen werden terminiert.
- *Starting*: Elastic-MapReduce startet, installiert und konfiguriert ein Cluster.
- *Terminated*: Der Nutzer hat den Job-Flow abgebrochen.
- *Waiting*: Das Cluster wartet auf einen weiteren Job.

Die Zustände sind in der Benutzeroberfläche angezeigt bzw. sind mittels der API abrufbar. Durch die fehlende Kontrolle über die einzelnen EC2 Instanzen des Hadoop Clusters in Elastic MapReduce, ist die Statusabfrage mit Hilfe der drei Schnittstellen die einzige Möglichkeit den Zustand des Clusters zu überwachen.

Ein erfolgreicher Ablauf eines Job-Flows besteht aus vier Schritten [18]:

1. *Clusterstart*: Elastic MapReduce startet die im Job-Flow konfigurierte Anzahl von EC2-Instanzen. Die Anzahl der Worker Nodes eines Elastic MapReduce Clusters ist immer entsprechend der konfigurierten Anzahl der EC2-Instanzen. Eine Ausnahme bildet der Fall mit einer einzelnen EC2 Instanz. Im Ausnahmefall sind Master und Worker auf einer Instanz gestartet. Die Verteilung lässt sich anhand der Logfiles eines EMR-Clusters (bei aktiviertem Debugging) herausfinden. Das gestartete Image ist ein von Amazon gebündeltes Image mit aktuell installiertem `hadoop-0.18.3`[1]. Nach dem Start der Instanzen installiert Elastic MapReduce, abhängig von der Job-Flow-Definition, Erweiterungen von Hadoop (beispielsweise Pig).

[1] Stand Mai 2010

2. *Eingabedatendownload*: Wie Abbildung 4.1 darstellt, lädt das gestartete Hadoop Cluster die Eingabedaten aus dem definierten S3 Bucket in das konfigurierte, verteilte Dateisystem. Mögliche Dateisysteme sind: HDFS und S3.

3. *Hadoop-Jobausführung*: Das Hadoop Cluster führt den MapReduce Job aus.

4. *Ausgabedatenupload*: Die Ausführung des MapReduce Jobs ist beendet und das Cluster kopiert die Ausgabedaten in das konfigurierte S3 Bucket.

Die Steps, die Elastic MapReduce innerhalb eines Job-Flows ausführt, sind ebenfalls mittels verschiedener Zustände überwachbar. Die Logs bzw. die Ausgaben der einzelnen Steps sind nach der Ausführung abrufbar. Der Download der Ausgaben ist nur bei aktivierter Debugging-Option in der Job-Flow-Definition möglich.

Kapitel 5

Realisierung

Die Apache Software Foundation bietet die Quellen von Hadoop zum Download an. Das Konfigurieren und Kompilieren der Quellen und das Installieren von Hadoop ist sehr komplex. Das Unternehmen Cloudera bietet zur Vereinfachung der Installation fertige Installationspakete für Debianderivate sowie für RPM-fähige Linuxdistributionen [40]. Die Pakete sind auf Ebene der Softwarekomponenten von Hadoop granularisiert. Beispielsweise lässt sich der Jobtracker von der Namenode getrennt installieren. Cloudera bietet Pakete für die folgenden Komponenten und Subprojekte:

- MapReduce
- HDFS
- die Hadoop-Dokumentation
- Hadoop Pipes
- Pig
- Hive

Aktuell[1] existieren keine weiteren Pakete für andere Hadoop-Subprojekte wie z.B. HBase oder Zookeeper.

Aktuell existiert keine Software, die ein Hadoop Cluster in unternehmsinternen oder öffentlich zugänglichen Clouds installiert bzw. konfiguriert, um damit einen Open Source Dienst on demand zur Verfügung stellen zu können. Unternehmen, die aktuell Hadoop einsetzten, müssen Hadoop auf physikalischen Cluster installieren und konfigurieren, um die Software einsetzen

[1] Stand April 2010

zu können. Beim Betrieb eines Hadoop Clusters auf physikalischen Maschinen mit beispielsweise monatlich anfallender Datenverarbeitung, überwiegen die Kosten für Anschaffung und Wartung den Nutzen. Eine Möglichkeit der kosteneffizienteren Nutzung der Ressourcen ist der Einsatz von Hadoop in Verbindung mit Cloud Computing. Es existiert Cloud Computing Software, die einen Infrastrukturdienst bietet, auf dem sich ein Hadoop Cluster installieren lässt.

Die mit dieser Arbeit entstandene Applikation ist eine Implementierung zur Installation und Konfiguration eines Hadoop-Clusters unter Zunahme eines IaaS-Dienstes. Der IaaS-Dienst kann in einer Public- oder Private Cloud angeboten sein. Die Applikation heisst ***Ha**doop-**a**s-**a**-**S**ervice (HaaaS).*

5.1 Anforderungsanalyse

Die Anforderungen an eine Applikation, die ein Hadoop-Cluster in einer Infrastructure-as-a-Service Cloud installiert und konfiguriert, sind in diesem Unterkapitel beschrieben. Die Anforderungen orientieren sich an den Konfigurationsmöglichkeiten von Hadoop bzw. an den Möglichkeiten virtuelle Maschinen in einer IaaS Cloud zu installieren, konfigurieren (zusammengefasst: deployen) bzw. kontrollieren.

Das Deployment eines Hadoop-Clusters in einer IaaS Cloud ist in zwei Arbeitsvorgänge zu unterteilen. Schritt 1 ist das Starten der virtuellen Maschinen. Schritt 2 beschäftigt sich mit der Hadoop-Installation und dessen Konfiguration. Die Konfiguration eines Hadoop-Clusters hängt im Allgemeinen von folgenden Faktoren ab:

- *Clustergröße*: Die Clustergröße entscheidet über die Art der Konfiguration. Hadoop hat 3 Konfigurationstypen:
  - Die alleinstehende Konfiguration (*Standalone*), die bei Hadoop MapReduce direkt ausführt, ohne einen Server mit Schnittstelle zu starten.
  - Die pseudoverteilte Konfiguration (*Pseudo-Distributed*), die bei Hadoop ein Cluster auf einer Maschine startet.
  - Eine vollverteilte Konfiguration (*Fully-Distributed*), startet Hadoop als Cluster auf mehreren physikalischen oder virtuellen Maschinen.
- *Einsatzzweck*: Es existieren 3 denkbare Szenarien, die jeweils einen Einsatzzweck beschreiben. Die Hadoop-Komponenten MapReduce und HDFS lassen sich getrennt starten. Daraus ergeben sich die 3 Szenarien:

  - *Nur-HDFS-aktiv* (Nur das verteilte Dateisystem),
  - *Nur-MapReduce-aktiv* (Für MapReduce Jobs, die das HDFS eines anderen Clusters nutzen)
  - und *Beide-aktiv* (das Standardszenario).

- *Ressourcen der darunterliegenden Maschinen*: Ein Cluster mit ressourcenreicheren Knoten benötigen andere Konfigurationsoptionen in Hadoop als resourcenärmere. Optimierungspotential bietet unter anderem eine andere Heap-Größe für MapReduce-Jobs (Konfigurationsoption: `mapred.child.java.opts`).

- *Fehleranfälligkeit/Ausfallrate eines Cluster-Mitglieds*: In Hadoop gibt es Konfigurationsoptionen, mit deren Hilfe sich eine erhöhte Fehleranfälligkeit von Knoten in einem Cluster ausgleichen lässt. Hilfreich ist die Konfigurationsoption `dfs.replication`, die angibt, wie oft ein Datenblock in einem HDFS-Cluster repliziert wird.

- *Aufteilung des Clusters in Master/Worker Knoten*: Je nach Verwendung von einem oder mehreren Masterknoten, existieren verschiedene Konfigurationswege.

- *Größe der Verarbeitungsdaten*: Die Größe der einzelnen Dateien in einem HDFS System ist entscheidend für die Einstellung diverser HDFS-Parameter, z.B. der Blockgröße eines HDFS-Replikationsblocks.

Die Faktoren Clustergröße, Aufteilung und Einsatzzweck sind entscheidend für ein Programm, das ein Hadoop-Cluster installiert und konfiguriert, da diese die Art und Weise des Clusterstarts beeinflussen. Die anderen Faktoren sind für das Deployment nicht relevant. Die unrelevanten Faktoren beeinflussen Laufzeitkomponenten von Hadoop und sind dadurch nicht relevant für den Start des Clusters.

5.1.1 Anforderung 1: Starten von Instanzen eines Images in einer Private Cloud IaaS basierend auf Eucalyptus 1.6

Vor der Konfiguration der Systeme sind Instanzen in einer Infrastructure-as-a-Service Cloud zu starten. Als IaaS Cloud ist Eucalyptus [45] mit der Version 1.6 zu verwenden. Zum Start der Instanzen verwendet die Applikation, eine vom Benutzer definierte Konfigurationsdatei. Die Konfigurationsdatei beinhaltet die folgenden Parameter:

- SSH-Key-Name
- Image-ID

- einen Instanztyp der virtuellen Maschinen

Das zu startende Image beinhaltet eine aktuelle Version des Ubuntu-Betriebssystems, für die Cloudera Hadoop-Pakete anbietet.

5.1.2 Anforderung 2: Parallele Konfiguration der Hadoop-Systeme

Die Konfiguration der Maschinen eines Hadoop Clusters ist parallel durchzuführen. Die Parallelität ist mittels einer Konfigurationseinstellung einzuschränken. Das heißt, die Konfiguration bestimmt, ob eine parallele Konfiguration durchzuführen ist.

5.1.3 Anforderung 3: Starten des Hadoop Clusters

Die Kernanforderung ist das Starten eines Hadoop Clusters. Die Anwendung benutzt zum Start des Clusters, die von der Cloud zur Verfügung gestellten Images. In einem Standard-Ubuntu-Image ist Hadoop nicht vorinstalliert. Die Installation von Hadoop ist zu zwei Zeitpunkten möglich:

- *Vor dem Start* der Instanzen mit Hilfe einer Imageerstellung.
- *Nach dem Start* der Instanzen durch Skripte oder der direkten Ausführung der Befehle mittels SSH.

Die Installation von Hadoop vor dem Start kostet weniger Startzeit und damit weniger Geld in einer kostenpflichtigen Public Cloud. Die im Rahmen dieser Arbeit entwickelte und implementierte Anwendung HaaaS verfolgt diesen Ansatz.

5.1.4 Anforderung 4: Wiederverwendbarkeit der Anwendung als API

Die Architektur und das Design der Anwendung ist dahingehend zu gestalten, dass diese als API wiederverwendbar ist. Durch die API ist der Konfigurationsstatus einer einzelnen Maschine überwachbar.

5.1.5 Anforderung 5: Erstellung eines Clusters, der hinter einer Firewall liegt

In einer IaaS Private Cloud ist es möglich, dass die Instanzen mit Hilfe einer Firewall gesichert sind. Die Firewall bietet die Möglichkeit des SSH-

Tunneling, wobei die Firewall mittels eines SSH-Tunnels eine Verbindung zu dein Instanzen hinter der Firewall ermöglicht. Die Anwendung unterstützt diesen Weg der Kommunikation.

5.1.6 Anwendungsfälle des Systems

Die Anforderungen ergeben die im Use-Case Diagramm aus Abbildung 5.1 dargestellten Anwendungsfälle für das Deploymentsystem und den Nutzer. Es existieren vier Anwendungsfälle, wobei der Nutzer mit einem Anwendungsfall agiert. Die Deployment-Applikation agiert mittels drei Anwendungsfällen mit den virtuellen Maschinen des Clusters.

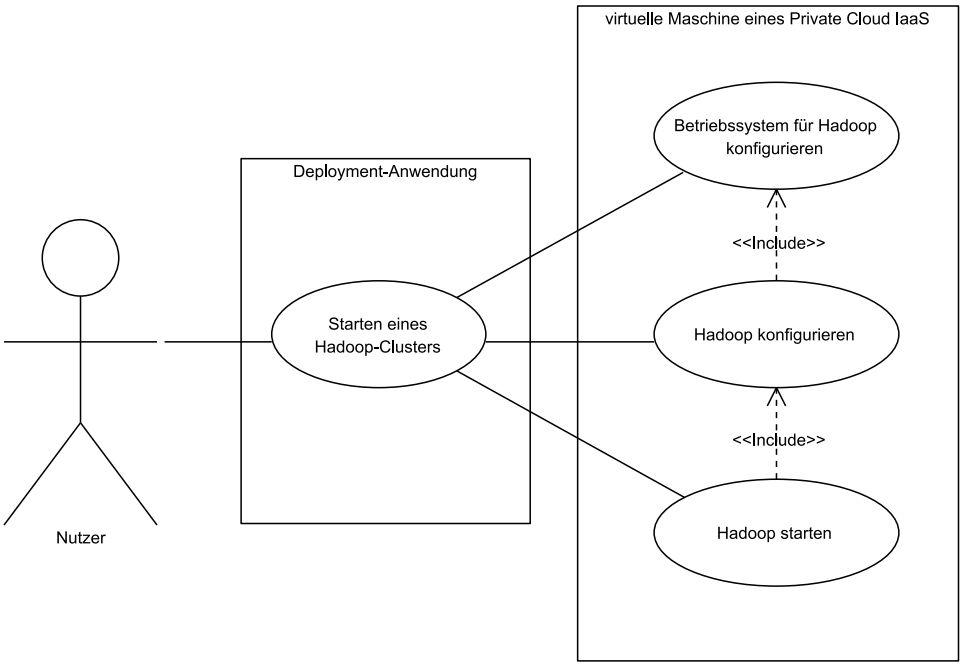

Abbildung 5.1: Anwendungsfälle der Applikation und der beteiligen Systeme

Der Nutzer startet die Applikation bzw. den Deploymentvorgang. Die Deployment-Applikation konfiguriert die Betriebssysteme der virtuellen Maschinen sowie die installierten Hadoop Pakete. Anschließend startet die Anwendung das konfigurierte Hadoop Cluster und teilt dem Nutzer den Zugang mit. Wie in Abbildung 5.1 zeigt, hängt der Start von Hadoop indirekt von der Konfiguration des Betriebssystems ab.

5.2 Architektur und Design der Applikation

Die Erstellung eines Images mit Hadoop bzw. das Deployment des Clusters stellen zwei unabhängige Schritte dar. Die Imageerstellung dauert im Verhältnis zum Deployment des Cluster länger und wird im Verhältnis seltener durchgeführt. Daher besteht Aufbau der Anwendung aus zwei Elementen. Einerseits ein Shellskript zur Imageerstellung und andererseits eine Anwendung zum Deployment. Die folgenden Unterkapitel beschreiben den Aufbau des Skripts bzw. den Aufbau der Deployment-Applikation.

5.2.1 Architektur der Deployment-Applikation

Die Applikation ist in zwei Teile aufgeteilt. Ein Teil ist die API, der andere Teil ist eine Benutzeroberfläche in Form einer Anwendung im Webbrowser. Die API ist isoliert entwickelt. Dadurch ist sie von der Benutzeroberfläche unabhängig und in andere Applikationen integrierbar, womit die Anforderung aus Kapitel 5.1.4 erfüllt ist.

Die API bietet, mit der implementierten Kommandozeilenschnittstelle, die Möglichkeit, ohne die eigene Benutzeroberfläche und ohne ein vom Nutzer implementiertes Programm, ein Hadoop Cluster zu deployen. Die Kommandozeilenschnittstelle erwartet als einzigsten Startparameter eine Konfigurationsdatei, die beschreibt wie und wo das Hadoop Cluster zu Installieren und Konfigurieren ist. Die Architektur der Anwendung, die ein Nutzer mit der Kommandozeile benutzt, ist in Abbildung 5.2 abgebildet.

Der Nutzer der Kommandozeile, in Abbildung 5.2, startet die Applikation, die mittels HTTP(S)-Verbindungen die Instanzen erstellt und mittels SSH die Instanzen konfiguriert. Da die Anwendung die Konfiguration über das Internet durchführt, ist eine sichere Verbindung zu den Instanzen nötig. Die Applikation nutzt SSH2, um eine sichere Kommunikation mit den Instanzen in der Cloud zu ermöglichen. Ein alternatives Anwendungsszenario für die Nutzung der Applikation mittels Kommandozeile ist die Ausführung innerhalb einer virtuellen Maschine in der Cloud, um Datenverkehr über das Internet zu sparen.

Der andere Teil der Applikation, die Benutzeroberfläche im Webbrowser, ist eine Webanwendung, die auf einem sogenannten *Application Server* (beispielsweise dem Apache Tomcat Application Server [33] der Apache Software Foundation) zu installieren ist. Die Benutzeroberfäche besteht aus zwei voneinander abhängigen Teilen.

- Der Hauptteil ist die Implementierung der Servlets auf der Seite des Applicationservers. Die Servlets greifen auf die API zu und initialisie-

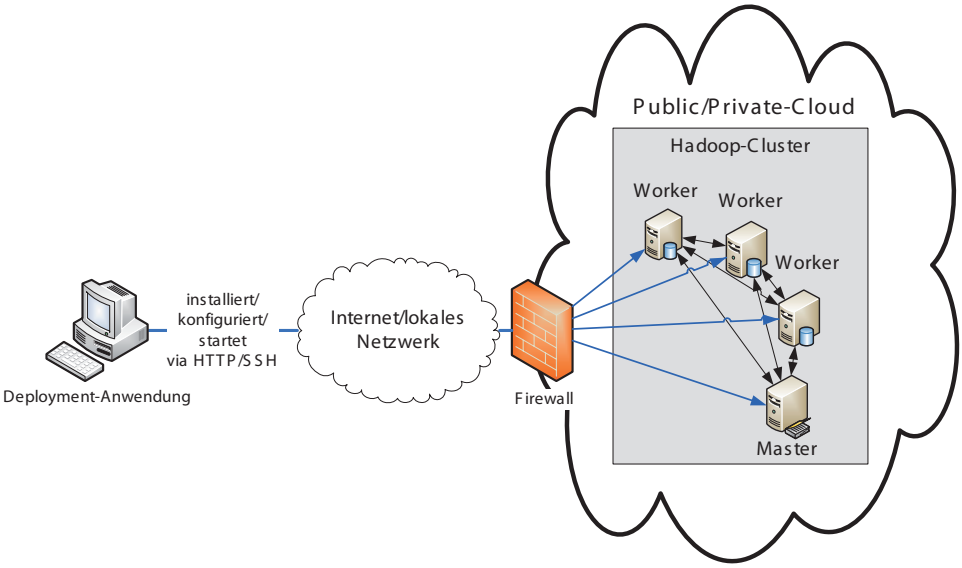

Abbildung 5.2: Aufbau und Funktionsweise der Kommandozeilen-Deployment-Applikation aus netzwerk-topologischer Sicht

ren das Deployment des Clusters. Zusätzlich bieten die Servlets eine Schnittstelle zur Überwachung des Deploymentprozesses. Der serverseitige Code ist das sogenannte *Backend*.

- Der clientseitige Code, *Frontend* genannt, besteht aus aus HTML, CSS, JavaScript und stellt die für den Benutzer sichtbare Oberfläche im Browser dar. Die Oberfläche beinhaltet Elemente zum Start des Deployments und zur Überwachung des serverseitigen Deploymentprozesses. Die Oberfläche bietet ein Textfeld zur Eingabe der Konfiguration und einen Knopf, der das Deployment startet. Zur Überwachung zeigt die Oberfläche eine Tabelle, bei der die gestarteten virtuellen Maschinen in der Cloud einzeln überwachbar sind.

Mit der angebotenen Weboberfläche ist ein Szenario, der Einsatz der Applikation als SaaS, möglich. Durch die Teilung der Applikation in eine API und ein Weboberfläche hat die Applikation insgesamt zwei Architekturen, die in Abbildung 5.2 und Abbildung 5.3 dargestellt sind.

- Die Applikation ist, wie in Abbildung 5.2 dargstellt, ein sogenannter *Fat Client*, da die Applikation sich komplett auf einem Client ausführen lässt und keinen Server benötigt, der die Installation und Konfiguration des Hadoop Clusters übernimmt.
- Die Weboberfläche der Applikation auf der Clientseite zusammen mit

der serverseitigen Implementierung sind eine Architektur mit den drei folgenden Schichten.

- Eine *Visualisierungs*-Schicht, innerhalb der die Client-Software auf dem Browser läuft.
- Die *Präsentations*-Schicht ist serverseitig integriert und ist die Rendering- und Zugriffsschnittstelle der Visualisierungsschicht. Die Hauptaufgabe dieser Schicht ist das Rendering der Oberfläche sowie Session-Management und die Umleitung der Webserviceaufrufe der Visualiserungsschicht zur Anwendungslogik.
- Die *Anwendungslogik*-Schicht ist, wie die Präsentationsschicht, serverseitig integriert und stellt die API dar.

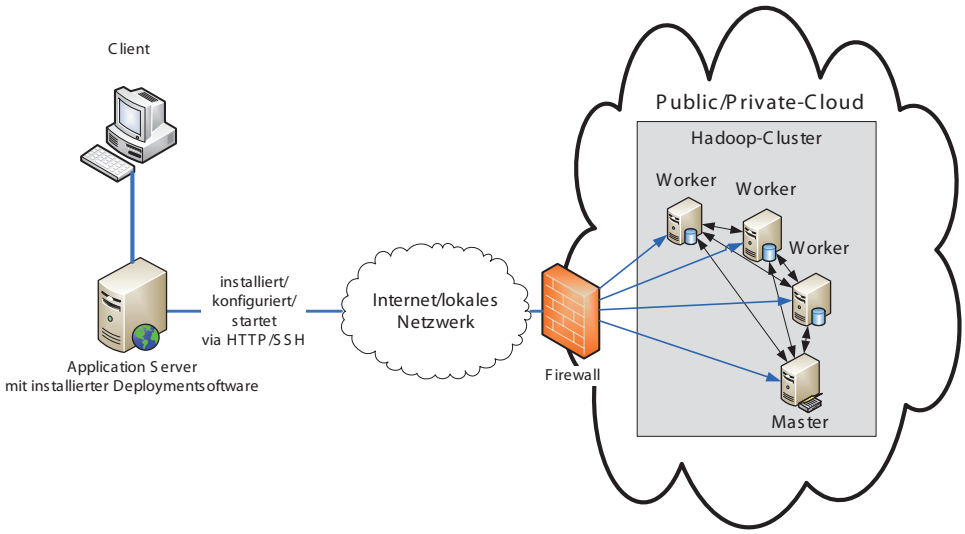

Abbildung 5.3: Aufbau und Funktionsweise der Deployment-Applikation incl. Weboberfläche aus netzwerk-topologischer Sicht

Die Nutzung der Anwendung über die Weboberfläche, die in Abbildung 5.3 dargestellt ist, bietet die Möglichkeit, die Anwendung vielen Nutzern zur Verfügung zu stellen, ohne eine Installation auf der Clientseite vornehmen zu müssen, wie es bei einer Fat Client-Anwendung der Fall ist.

Der Application Server bietet den Nutzern mit JavaScript-fähigem Browser im Mobiltelefon die Möglichkeit, ein Hadoop Cluster zu deployen, da der Server die Installations- und Konfigurationsschritte übernimmt und damit das Handy und dessen Mobilfunkverbindungen nicht stark belastet. Die Schritte und Kommunikationswege sind identisch zu den Schritten und Kommunikationswegen bei der Nutzung der Applikation über die Weboberfläche.

Der Application Server kann sich entweder außerhalb der Cloud oder innerhalb der Cloud als virtuelle Maschine befinden. Befindet sich der Application Server innerhalb der Cloud oder an der Schnittstelle Internet/Cloud, wie bei Amazon Elastic MapReduce, lässt sich der Datenverkehr über das Internet reduzieren, da die SSH Verbindungen aus dem Internet in die Cloud für die Konfiguration entfallen.

5.2.2 Design der Deployment-Applikation-API

Die API der Deployment-Applikation hat mehrere Teilaufgaben die Komponenten übernehmen, die der folgende Abschnitt erläutert. Das Deployment besteht aus dem Start der virtuellen Maschinen sowie deren Konfiguration. Abbildung 5.4 zeigt ein vereinfachtes, nicht vollständiges Klassendiagramm der Anwendung. Das Diagramm stellt ausschließlich die für das Deployment relevanten Klassen und deren Beziehungen zueinander dar. In den Implementierungen der Klassen exisitieren mehr Methoden und Variablen, als in der Abbildung dargestellt ist. Auf weitere existierende Klassen geht das Kapitel 5.3 ein. Die für das Deployment relevante API ist in drei Komponenten aufgeteilt, die in der Abbildung farblich unterschieden sind.

Deployer

Eine Klasse in Abbildung 5.4 stellt den Deployer dar, der die Hauptkomponente der Anwendung ist. Der Deployer kennt und benutzt zwei weitere Komponenten: den Instance-Manager und einen Configurator. Die Komponente Deployer startet mittels Instance-Manager die virtuellen Maschinen in einer Cloud.

Jede gestartete Instanz in einer Cloud entspricht einem Instance-Objekt in dieser Applikation. Der Deployer überwacht die gestarteten Instances und übergibt diese an einen Configurator.

InstanceManager

Der InstanceManager ist die Umsetzung der Anforderung beschrieben in Kapitel 5.1.1. Der InstanceManager startet Instanzen in einer IaaS Cloud und verwaltet die Verbindung zur Schnittstelle der Cloud. Die Überwachung des Instanzstarts sowie die Verwaltung der Instanzen erfolgt über die gleiche HTTP-Verbindung mit der Cloud. Jedes Instance-Objekt hat einen `RunningState` der zeigt, ob das Betriebssystem der virtuellen Maschine bereits gestartet ist.

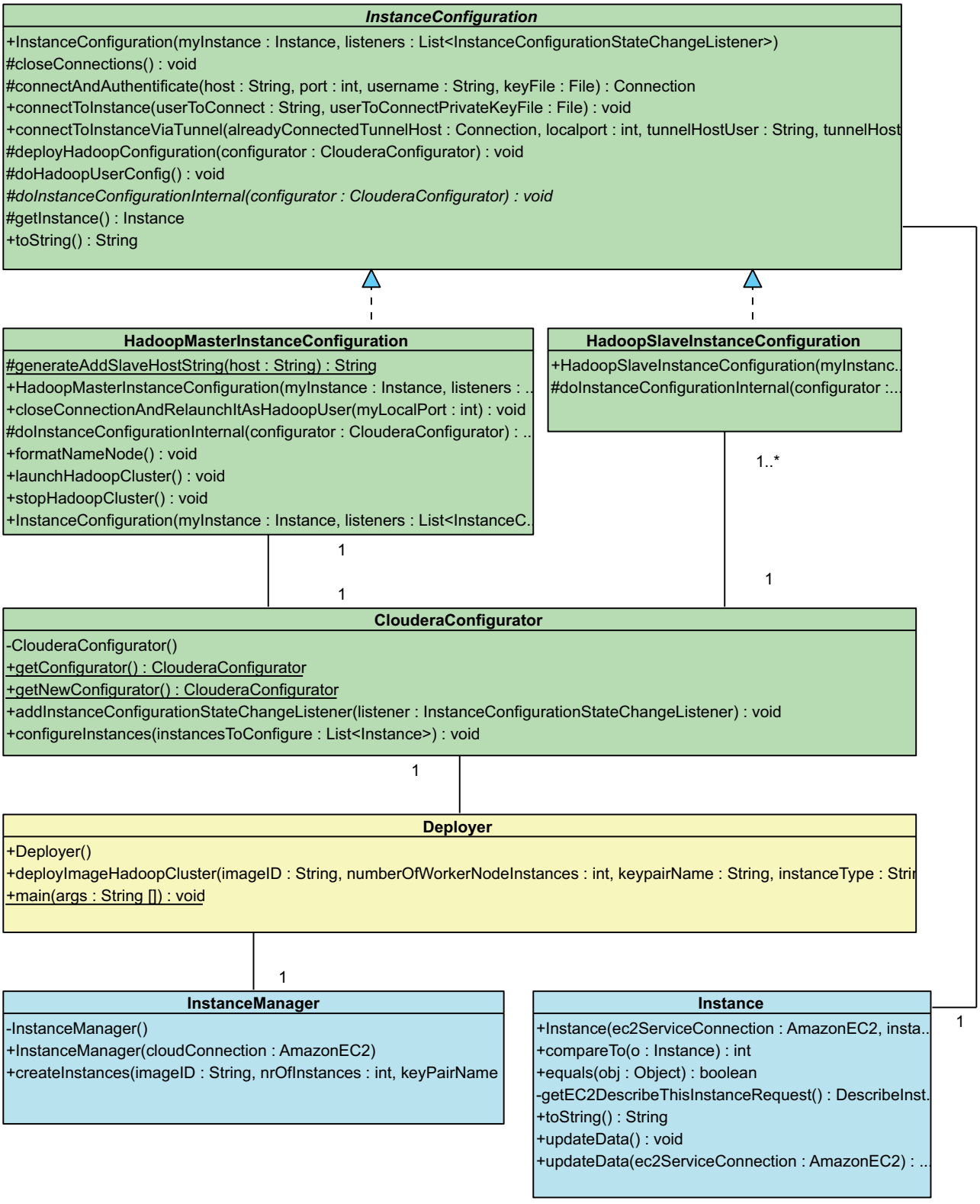

Abbildung 5.4: Klassendiagramm der Deployment-Applikation

Configurator

Die Schnittstelle *InstanceConfigurator* setzt die Anforderung aus den Kapiteln 5.1.2 bzw. 5.1.3 um und ist eine Komponenteschnittstelle zum Start der Konfigurationsprozesse. Die Komponente ist in Abbildung 5.4 ebenfalls vorzufinden. Eine Implementierung der Schnittstelle ist der `ClouderaConfigurator`, der die Konfiguration eines Hadoop Clusters vornimmt.

Die Komponente Configurator benutzt sogenannte *InstanceConfigurations*. InstanceConfigurations sind Implementierungen der abstrakten Klasse `InstanceConfiguration` und stellen die verschiedenen möglichen Konfigurationen einer Instanz dar. Eine Konfiguration verbindet sich zu einer virtuellen Maschine in der Cloud und konfiguriert diese selbstständig. Zur Überwachung der Konfiguration existieren verschiedene `InstanceConfigurationStates`, die mittels Observer-Pattern einem Listener mitteilen, wie weit die Konfiguration einer Instanz fortgeschritten ist. Zum Aufbau eines Hadoop Clusters existieren zwei spezielle InstanceConfigurations, die die Klasse `ClouderaConfigurator` zum Deployment nutzt:

- `HadoopMasterInstanceConfiguration`: Die Master-Configuration konfiguriert Master Instanzen von Hadoop und startet das Cluster nach einer erfolgreichen Konfiguration.
- `HadoopSlaveInstanceConfiguration`: Diese Implementierung konfiguriert die Tasktracker bzw. die Datanodes eines Hadoop-Clusters. Jede Maschine mit dieser Konfiguration startet nach der Konfiguration sowohl einen Tasktracker, als auch eine Datanode auf ihrer Java-Virtual-Maschine.

Die API bietet die Möglichkeit, die SSH-Verbindungen, die die InstanceConfigurations benötigen, über einen SSH Tunnel aufzubauen. Das Aufbauen des Tunnels sowie das Aufbauen der SSH-Verbindungen ist Aufgabe der API, womit die Anforderung aus Kapitel 5.1.5 erfüllt ist. Der Einsatz von SSH Tunneln ist in der Konfiguration des jeweiligen Deployments anzugeben.

5.2.3 Aufbau und Implementierung des Shellskripts

Die Kernfunktionalität des Shellskripts ist die Erstellung eines in der IaaS Cloud verwendbaren Images. Weitere Funktionen sind das Hochladen des Images sowie die Registrierung des Images in der Cloud. Voraussetzung für das Benutzen des Skripts ist das installierte Paket `euca-tools`.

Das Shellskript ist in drei Dateien aufgeteilt, die unterschiedliche Funktionen besitzen.

- Eine Datei mit Funktionen (`buildUploadRegisterImage.functions`), die das Skript benötigt.
- Die Datei `buildUploadRegisterImage.conf` beinhaltet die Konfiguration, von der der Inhalt des zu erstellenden Images abhängt.
- Das hauptsächliche Skript `buildUploadRegisterImage`, führt der Nutzer aus, um das Image zu erstellen.

Das Hauptskript liest die Konfigurationsdatei und startet die vom Benutzer angegebene Operation. Der Benutzer kann folgende Operationen starten:

- `Auto`: Erstellt das Image ohne es hochzuladen.
- `Upload`: Lädt ein erstelltes Image in die Cloud.
- `Mount`: Bindet (mountet) ein Image ein.
- `Umount`: Entbindet (unmountet) ein Image.

Die Konfigurationsdatei `buildUploadRegisterImage.conf` bietet die folgenden Parameter, die die Erstellung des Images beeinflussen. Das Skript benötigt alle Konfigurationsparameter der Datei.

- Mit Hilfe der *Softwarerepositories* und deren *Repositorykeys* lassen sich verschiedene non-standard Softwarerepositorys für Ubuntu einbinden. Innerhalb der Konfigurationsdatei sind mit Hilfe eines Arrays (`sources`) verschiedene Repositorys anzugeben, die das Skript vor der Installation der angegebenen Softwarepakete in das System einpflegt. Die Repositorykeys sind im Array (`keys`) zu pflegen.
- Die Konfigurationsdatei bietet ein Array zur Eingabe der *Softwarepakete* (`packages`), die das Skript während der Erstellung des Images für die Cloud installieren soll.
- Mit Hilfe eines weiteren Arrays (`postcommands`) sind auszuführende *Post-Commands* anzugeben, die das Skript nach der Installation der Softwarepakete ausführt. Beispielsweise kann ein Nutzer diese Option verwenden, um einem Softwarepaket eine andere Konfiguration zur Verfügung zu stellen. Alterantiv kann der Nutzer Array verwenden um weitere Software zu konfigurieren, die das Skript mit Hilfe des `packages` Array installiert.
- In der Konfiguration ist ein Verzeichnis anzugeben, in dem das Skript die Dateien herunterlädt bzw. in dem das Skript die Erstellung des Images durchführt. Das Verzeichnis ist mit der Konfigurationseinstellung `WORKDIR` anzugeben.

- Falls die Größe eines heruntergeladenen Images nicht ausreicht um die angegebenen Softwarepakete zu installieren, kann der Nutzer mit Hilfe einer Konfigurationsoption eine Imagegröße angeben. Das Skript passt die Größe des Images entsprechend der Konfigurationsoption an. Die Konfigurationsvariable `RESIZE` aktiviert die Option sowie Konfigurationsoption `RESIZE_SIZE`, die eine finale Größe der Imagedatei und des darin enthaltenen Dateisystems angibt.

Der Ubuntu-Hersteller Canonical bietet ein Standard Cloud Image zum Download an. Das Shellskript verwendet dieses als Basis zur Erstellung des Images. Während der Imageerstellung bindet das Skript das Basis-Image mittels Loop-Device und führt die Installation, der in der Konfiguration angegebenen Software, mit Hilfe der `chroot`-Umgebung durch.

Die Upload-Funktion lädt ein fertig erstelltes Image mit Hilfe der `euca-tools` in die Cloud und registriert es dort. Der Benutzer erhält, nach dem Upload bzw. nach der Registierung, die ID des Images. Der Nutzer benötigt die ID des Images zur Konfiguration der Deployment-Applikation.

5.3 Implementierung der Deployment-Applikation

Die Applikation ist in Java implementiert. Eine Implementierung in Java erfüllt die Anforderung aus Kapitel 5.1.4, da aktuell[2] Java APIs in vielen Middlewares, wie beispielsweise in einem Application Server wie Apache Tomcat, einsetzbar ist.

5.3.1 Verwendete Bibliotheken in der API

Die Implementierung der API der Deployment-Applikation verwendet zwei externe Bibliotheken um die Kommunikation mit der IaaS Cloud, bzw. den Instanzen in der Cloud aufzubauen.

- Die Bibliothek zur Kommunikation mit der Cloud ist die Amazon EC2-API-Tools [25]. Sie verfügt über Interfaces, mit Hilfe deren die EC2-Schnittstelle selbst implementierbar ist. Die bereitgestellten Implementierungen dieser Schnittstelle von Amazon sind, für die Verwendung mit Eucalyptus, konfigurierbar. Bei der Konfiguration ist darauf zu achten, dass die Implementierung die Protokoll-Version 1 verwendet, da ansonsten die Kommunikation zwischen Eucalyptus und der Bibliothek nicht funktioniert.

[2] Stand April 2010

- Die zweite Bibliothek [38] ist eine Open Source-Implementierung einer SSH2 Schnittstelle. Die ETH in Zürich hat diese ursprünglich im Rahmen eines wissenschaftlichen Projekts entwickelt. Die Implementierung bietet sogenannte *Connections* an, die eine SSH2-Verbindung zu einem Server herstellen. Eine Connection kann eine oder mehrere *Sessions* bzw. *Secure-Copy-Verbindungen (SCP)* herstellen. Mit Hilfe einer Session schickt die Anwendung einzelne Shell-Befehle zu der jeweiligen Instanz. Die SCP-Verbindungen dienen zum Kopieren der Hadoop-Konfigurationsdateien auf die zu konfigurierende Instanz.

5.3.2 Verwendete Bibliotheken in der Weboberfläche

Die Implementierung der Weboberfläche ist mit Hilfe von zwei Bibliotheken realisiert, die beide in diesem Abschnitt nacheinander beschrieben sind.

- Zum einen mit dem Google Web Toolkit (GWT) [54]
- und zum anderen mit der *Ext GWT* Bibliothek [64].

Das Google Web Toolkit ist ein Framework, mit dem sich Webapplikationen bzw. Weboberflächen in Java entwickeln lassen. Die Entwicklung der Webapplikationen mit dem GWT ähnelt dem Konzept der Entwicklung, der in Java etablierten Oberflächentechnologie `Swing` [61].

Eine in GWT entwickelte und kompillierte Anwendung besteht aus einer HTML Seite und der generierten JavaScript Datei. Die HTML Datei bindet die JavaScript Datei ein und beide bilden zusammen die Anwendung im Browser des Nutzers. Das GWT arbeitet mit Oberflächenelementen, sogenannten *Widgets*, wie bespielsweise ein Knopf oder ein Eingabefeld. Das Konzept von GWT sieht vor, dass GWT alle Oberflächenelemente zur Laufzeit von JavaScript in das sogenannte *Document-Object-Model (DOM)* des Browsers einfügt. GWT kann mit diesem Vorgehen alle Oberflächenelemente, die ein Nutzer im Browser sieht, zur Laufzeit konfigurieren und kontrollieren.

Ein sogenannter *Java to JavaScript Compiler* des GWT Framework kompiliert den entwickelten Javacode vor dem Deployment auf dem Application Server in HTML- und JavaScriptcode, der auf einem Application Server zu installieren ist [12].

Mit Hilfe von GWT lassen sich sogenannte *Remote Procedure Calls (RPC)* realisieren [11]. GWT kann Objekte, die das Interface `Serializable` implementieren übertragen. Objekte, deren Klassen dieses Interface nicht implementieren, kann GWT nicht übertragen. Ruft der Client eine entfernte Methode auf, übernimmt die Implementierung des GWT Frameworks die Aufgabe des Transports der Daten. Die Anwendung verhält sich damit für

den Entwickler, ab dem Aufruf der entfernten Methode bis zur Ankunft der Daten am Server bzw. bis zum Aufruf der Methode der *Servletklasse* auf dem Server, transparent.

Die *Ext GWT* Bibliothek ist ein sogenanntes *Rich Internet Application (RIA)* Framework für GWT Anwendungen. RIA sind Internetanwendungen in einem Browser, die von der Optik und mit deren Bedienungskonzept mit Desktopanwendungen vergleichbar sind [12]. Die Bibliothek Ext GWT erweitert GWT mit Widgets, die Entwickler zur Erstellung einer RIA benötigen. Zusätzlich zu den Widgets bietet Ext GWT Layoutmöglichkeiten wie z.B. virtuelle Fenster in einem Browsertab.

5.3.3 Ablauf eines Deploymentvorgangs der API

Die Deployment API ist, wie in Kapitel 5.2.2 beschrieben, in mehrere unabhängige Komponenten unterteilt. Abbildung 5.4 zeigt den Zusammenhang der Komponenten und Klassen untereinander. Der Deploymentvorgang, den die API mit Hilfe aller Komponenten durchführt, ist in diesem Kapitel beschrieben.

Das Deployment eines Hadoop Clusters mit der API in einer Public IaaS Cloud ist in drei voneinander abhängige Schritte aufgeteilt, die im Detail in Abbildung 5.5 dargestellt sind. In der Beschreibung der Schritte sind in Klammern die Schrittzahlen des Sequenzdiagramms aufgeführt.

1. Der erste Schritt ist das Starten der virtuellen Instanzen in der Cloud. Nachdem der Nutzer das Deployment initiiert hat (*1*), überprüft der `Deployer` die Konfiguration des Nutzers und stellt diese für alle anderen Komponenten bereit. Das `Deployer` Objekt initialisiert die Klassen bzw. Komponenten `InstanceManager` (*2*) und `Configurator` (*3*) und startet mit Hilfe des `Instance-Managers` die in der Konfiguration angegebene Anzahl an virtuellen Maschinen in der Cloud (*4.1*). Der Instance Manager liefert dem `Deployer` eine Liste der erfolgreich gestarteten virtuellen Maschinen zurück (*4.1.2*).

2. Schritt 2 ist die Konfiguration der virtuellen Instanzen. Vor der Konfiguration der Instanzen überwacht der `Deployer` (*5*) den Zustand der Instanzen, um festzustellen, ob eine Instanz erfolgreich gestartet ist. Ist der Start einer Instanz fehlgeschlagen, startet der `Deployer` mit Hilfe des Objekts der Klasse `InstanceManager` eine weitere Instanz in der Cloud. Sind alle Betriebssysteme der Instanzen erfolgreich gebootet, startet der `Deployer` mit Hilfe des `Configurator` die Konfiguration des Hadoop Clusters (*6*).

   Der `Configurator` erhält vom `Deployer` eine Liste von virtuellen

Maschinen, die er konfigurieren soll (*6*). Ist in der Konfiguration der Parameter für das Multithreading auf `true` gesetzt, startet der `Configurator` pro virtueller Maschine einen Thread zur Konfiguration (*6.1*). Ist das Multithreading abgeschaltet, findet die Konfiguration sequentiell statt.

Die Konfiguration der einzelnen Maschinen übernehmen die Objekte der Klasse `Instance Configuration`, die der `Configurator` pro virtueller Maschine initialisiert (*6.2*). Jedes `Instance Configuration` Objekt erhält eine virtuelle Maschine zugewiesen, die es konfigurieren muss. Die Konfiguration einer virtuellen Maschine läuft wie folgt ab:

Schritt 1: Zuerst verbindet sich das `Instance Configuration` Objekt, mit Hilfe der SSH API, zu der zugewiesenen virtuellen Maschine in der Cloud.

Schritt 2: Das `Instance Configuration` Objekt kopiert den in der Konfiguration angegebenen *SSH Public Key* in das `.ssh` Verzeichnis des `hadoop` Benutzers, um ein Clusterstart über SSH zu ermöglichen.

Schritt 3: Das Objekt bereitet die Dateien im Hadoopkonfigurationsordner vor, der vom Nutzer in der Deploymentkonfiguration angegeben ist. Das Objekt passt die Dateien im Arbeitsspeicher so an, dass sie den korrekten Master und die korrekten Slaves beinhalten, um einen korrekten Clusterstart zu ermöglichen. Nach der Vorbereitung lädt das Objekt die Dateien, über SCP, auf die virtuelle Maschine, in den von Hadoop vorgesehenen Ordner für Konfigurationsvarianten.

Schritt 4: Als letzten Schritt der Konfiguration einer einzelnen virtuellen Maschine legt das Objekt die in der Konfiguration angegebenen Hadoop Nutzdatenordner an, damit die Hadoop Prozesse korrekt starten können.

3. Sind alle virtuellen Maschinen fertig konfiguriert, startet die API im dritten und letzten Schritt des Deployments das Cluster mit Hilfe der virtuellen Maschine, auf der der Namenode und Jobtracker laufen. Der Clusterstart erfolgt mit Hilfe des `start-all` Skripts, das die Hadoop Installationspakete zur Verfügung stellen (*6.4*). Das Skript verbindet sich mit Hilfe von SSH und dem Benutzer `hadoop` zu den in der Konfiguration angegebenen Slave-Instanzen des Clusters und startet lokal die benötigten Hadoopprozesse.

Die Deployment API verwendet eigene Exceptions, die bei Problemen während des Deployments auftreten können. Die API bricht das Deployment, mit Hilfe der Exceptions, in den folgenden Situationen ab.

- Die SSH Verbindung zur virtuellen Maschine ist unterbrochen und die API kann die Verbindungen nach fünf Versuchen mit jeweils 5 Sekunden Wartezeit nicht erneut herstellen.
- Während der Konfiguration einer einzelnen Maschine liefert eine virtuelle Maschine nicht `null` als *Exit-Code* zurück. Ein *Exit-Code*, der nicht `null` ist, bedeutet, dass ein Fehler bei der Ausführung eines Shellbefehls auf der virtuellen Maschine aufgetreten ist.

Die Deployment API starten virtuelle Instanzen *vor der Konfiguration* in der Cloud neu, die beim Start abgestürzt sind. Es ist in bestimmten Situationen in einer Eucalyptus 1.6.2 Private IaaS Cloud möglich, dass virtuelle Maschinen nicht korrekt hochfahren oder während dem Start terminieren, da in der Cloud ein kurzfristiges Problem besteht.

5.3.4 Starten eines Deployments über die Kommandozeile

Es existieren zwei Möglichkeiten um ein Deployment mit HaaaS zu starten. Eine der Möglichkeiten ist, die Anwendung über die Kommandozeile lokal zu starten. Ein Anwender startet sie mit Hilfe der `.jar` Datei der API und einem Parameter. Der Parameter ist ein String, der den Dateinamen inklusive optionaler Pfadangabe zur Konfigurationsdatei darstellt.

Nach dem Start der `.jar` Datei, startet eine Instanz der `Deployer` Klasse mit dem Deployment des Clusters, das in Kapitel 5.3.3 beschrieben ist. Der Anwender kann den Fortschritt des Deployments, auf der Kommandozeile, mit Hilfe der Ausgaben der API in der Kommandozeile, überwachen.

Die Granularität der Ausgabe lässt sich mit einer Textdatei, die im *Java Properties* Format geschrieben ist, festlegen. In dieser Datei ist mit dem Parameternamen `de.hsmannheim.hoecker.bsc.clouderadeployer.level` und den Standard Logging Level Werten (`ALL`, `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINER`, `FINEST`, `NONE`) die Granularität eingestellt. In Abhängigkeit von diesem Parameter schreibt die API detaillierte bzw. weniger detaillierte Informationen in die Kommanndozeilenausgabe.

Am Ende des Deployments erscheint eine Ausgabe, mit welcher URL der Namenode bzw. Jobtracker erreichbar ist. Die API bzw. Anwendung läuft, wie in Abbildung 5.2 dargestellt, lokal auf dem Rechner des Nutzers. Diese Fat Client-Architektur ist nicht für die mobile Anwendung auf einem Smartphone oder einem Notebook, das über eine UMTS Verbindung verfügt, geeignet. Aus diesem Grund ist in dieser Arbeit eine Weboberfläche entstanden, das HaaaS Deployer Frontend, deren Nutzung im nächsten Kapitel beschrieben ist.

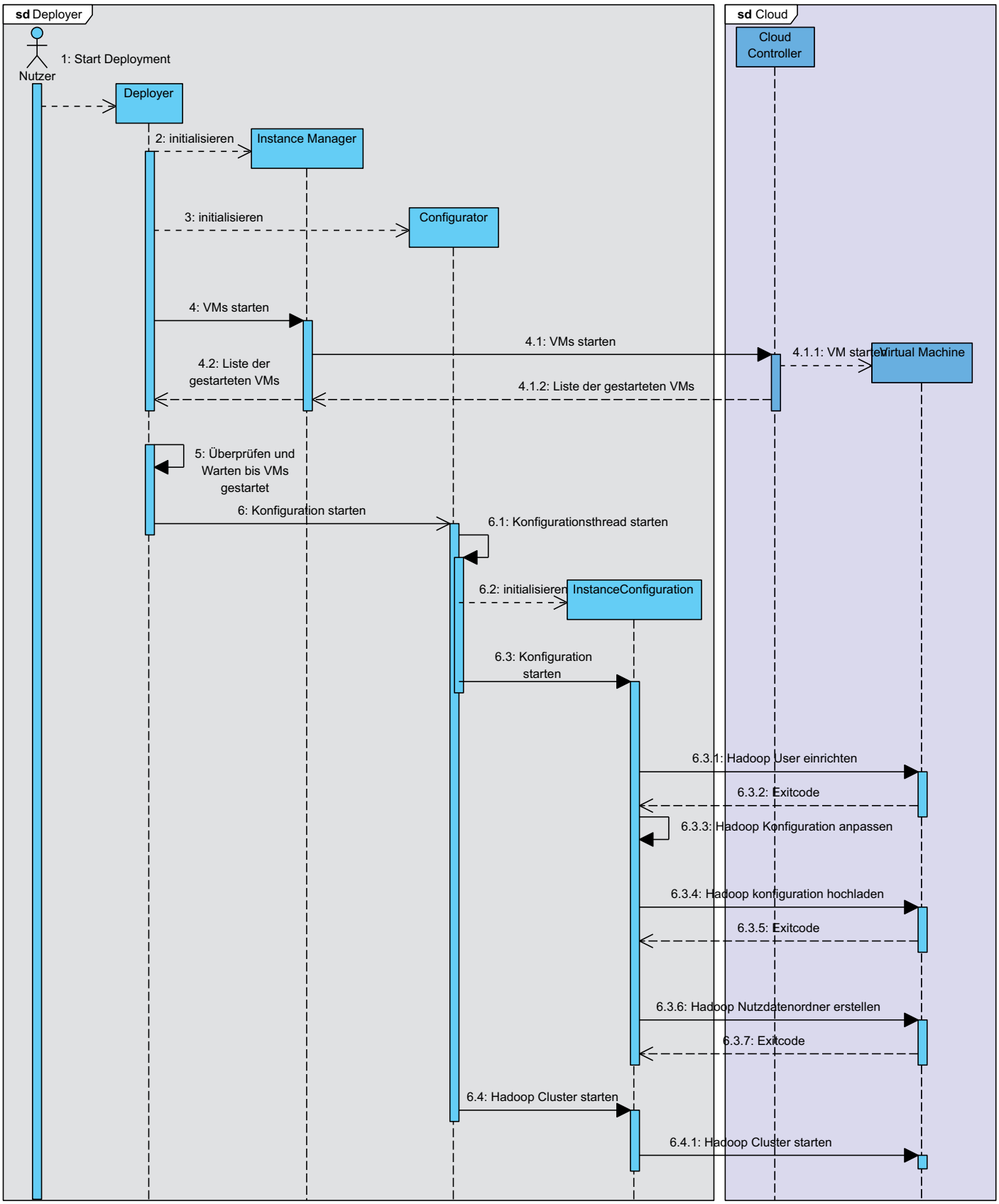

Abbildung 5.5: Sequenzdiagramm eines Deploymentvorgangs

5.3.5 Starten eines Deployments über die Weboberfläche

Das Deployment eines Hadoop Clusters kann neben dem Start über die Kommandozeile auch mit der Weboberfläche erfolgen. Die Weboberfläche besteht aus einer `.war` Datei, die ein Administrator auf einem Application Server installieren kann.

Ein Anwender kann die installierte Applikation über die konfigurierte URL auf dem Application Server mit Hilfe eines Browsers öffnen. Die Applikation im Browser zeigt eine Kurzbeschreibung des Ablaufs der Applikation, ein Textfeld für die Konfiguration und ein Button für den Start des Deployments. Der Anwender fügt den Inhalt einer Konfigurationsdatei der API in das dargestellte Textfeld und kann anschließend den Button drücken, um das Deployment zu starten.

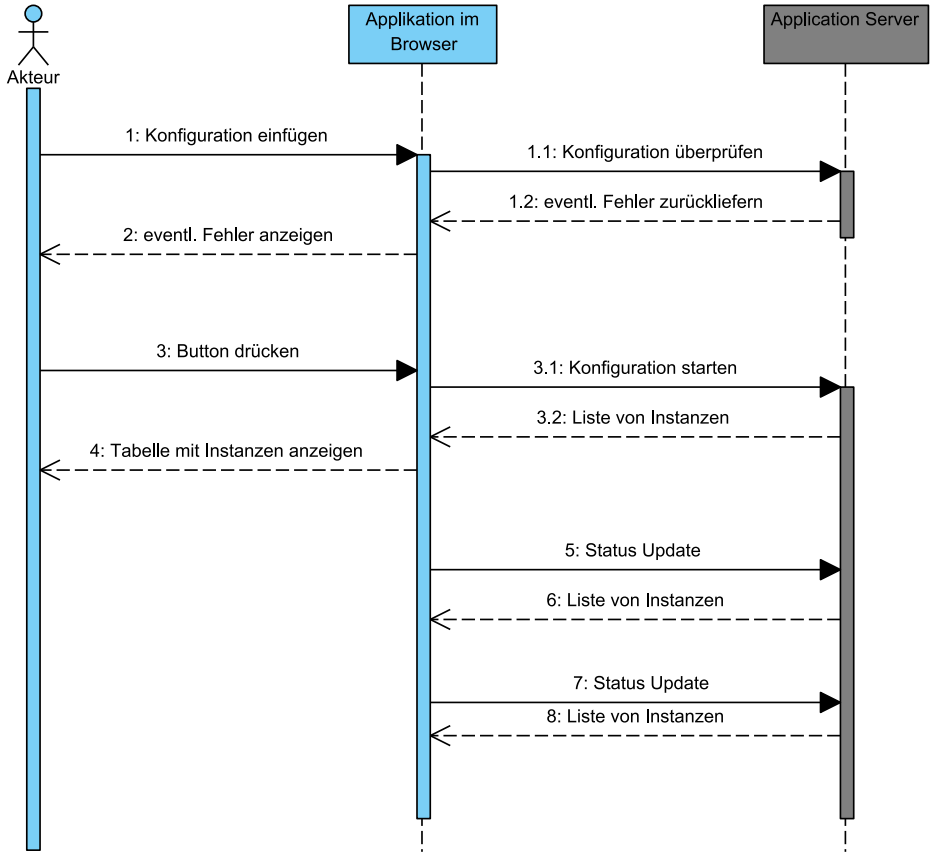

Abbildung 5.6: Ablauf eines Deployments über die Weboberfläche der Applikation

Der Ablauf eines Deployments und die Kommunikation zwischen Weboberfläche im Browser und dem Application Server ist in Abbildung 5.6 darge-

stellt. Der Deploymentvorgang auf der Seite des Application Servers ist auf den Punkt *Konfiguration starten* gekürzt, da der reine Deploymentvorgang in Abbildung 5.5 dargestellt ist.

Abbildung 5.7: Startbildschirm der Deployment Weboberfläche

Der Anwender fügt in das Textfeld (siehe Abbildung 5.7) die Konfiguration ein, die die Weboberfläche nach der Eingabe automatisch überprüft und bei einem Fehler eine Fehlermeldung ausgibt und das Textfeld rot markiert. Ist die Eingabe erfolgreich, kann der Nutzer den Button drücken, der die Konfiguration an den Server sendet und der das Deployment startet.

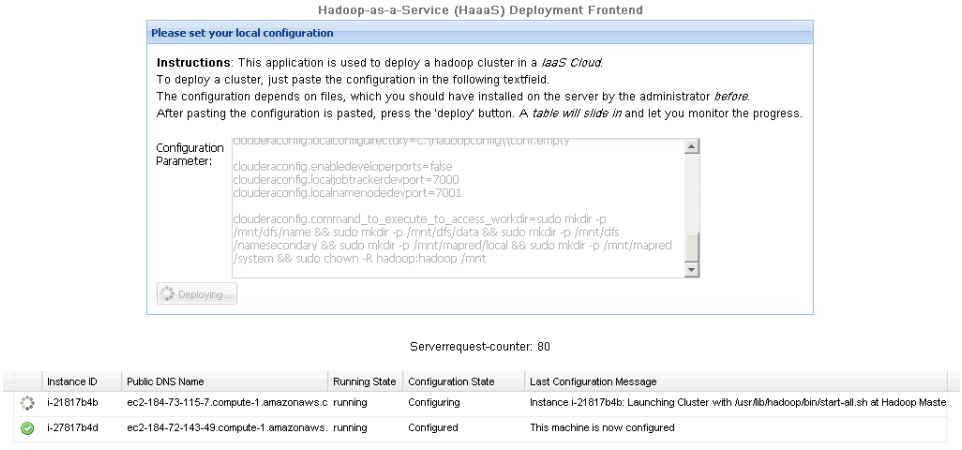

Abbildung 5.8: Anzeige eines Deploymentvorgangs in der Deployment Weboberfläche mit einer abgeschlossenen Konfiguration

Nach dem Start der virtuellen Maschinen in der Cloud liefert der Application Server eine Liste von virtuellen Maschinen, die der Deployer konfigurieren kann. Die virtuellen Maschinen zeigt die Weboberfläche in einer Tabelle an,

die sie bei einem erfolgreichen Start des Deployments einblendet (siehe Abbildung 5.8). Die Tabelle zeigt die Details der virtuellen Maschinen, wie den *Configuration State*, der anzeigt in welchem Konfigurationsstatus sich eine virtuelle Maschine befindet. Ein Icon in der ersten Spalte der Tabelle zeigt, ob die Konfiguration eines Knotens noch andauert, ob sie beendet ist oder ob es einen Fehler gab. Die letzte Spalte in der Tabelle zeigt die letzte Konfigurationsinformation einer einzelnen virtuellen Maschine.

Das Deployment ist an die Session des Webservers gebunden. Solange der Browser des Nutzers offen ist, bleibt die Session erhalten. Ist das Deployment erfolgreich abgeschlossen oder fehlgeschlagen, kann der Nutzer die Webanwendung mit der `F5` Taste neu starten. Ist der Deploymentvorgang noch aktiv, erhält der Nutzer die Nachricht, dass der Deploymentvorgang noch aktiv ist und die Webanwendung zeigt dem Nutzer den aktuellen Fortschritt an.

Die Weboberfläche ist, durch die Architektur einer Webanwendung, von der Durchführung der Konfiguration auf Seite des Application Servers entkoppelt. Die Konfiguration findet, nach dem Start des Deployments, vollständig auf der Serverseite ohne Eingriff eines Clients statt. Ein Client erhält, durch das HTTP Protokoll und der Architektur eines Webservers, keine Methodenaufrufe der Deployment API. Die Weboberfläche fordert, wie in Abbildung 5.6 dargestellt, in einem festen Intervall, erneut die Liste der Instanzen, um die aktuellsten Informationen darstellen zu können.

5.4 Vergleich der Implementierung mit anderen Entwicklungen

Aktuell[3] existieren mehrere Softwarelösungen um ein Hadoop Cluster auf mehreren Rechnern bzw. virtuellen Maschinen zu deployen. Drei dieser Lösungen, *Hadoop on Demand*, die Cloud-Skripte von Cloudera und Amazon Elastic MapReduce sind in dieser Arbeit in den folgenden Unterkapiteln beschrieben.

5.4.1 HOD - Hadoop on Demand

Hadoop on Demand (HOD) [34] ist eine Erweiterung für Hadoop, die die Apache Software Foundation zur Verfügung stellt. Hadoop on Demand kann, mit Hilfe mehrerer physikalisch oder virtuell existierenden Rechner, ein Hadoop Cluster aufbauen.

[3] Stand: April 2010

HOD benötigt für den Einsatz den Torque Resource Manager [42] von der Firma Cluster Resources Inc. Die Software bietet die Möglichkeit, innerhalb eines Torque Clusters, verschiedene *Batch Jobs* auszuführen. Die Batch Jobs sind Shell Skripte, die der Torque Resource Manager als entfernten Befehl auf einem Knoten im Cluster ausführt.

Ein Torque Cluster besteht aus den folgenden Komponenten [42].

- Die zentrale Verwaltung eines Torque Clusters übernimmt die sogenannte *Master Node*. Auf einer Master Node läuft der `pbs_server`, der das Cluster verwaltet. Neu gestartete Knoten registrieren sich bei der Master Node. Je nach Clustergröße muss der Master Node auf einem dedizierten Rechner installiert sein.
- Knoten in einem Torque Cluster, die *Submit/Interactive Node* sind, stellen die Knoten in einem Cluster dar, auf denen ein Nutzer die Jobs startet bzw. absendet. Zusätzlich überwachen diese Knoten den Fortschritt der Jobs auf den Compute Nodes.
- Eine *Compute Node* in einem Torque Cluster sind die Knoten, die einen Job ausführen. Die Compute Nodes registrieren sich an einem Torque Master Node. Eine Compute Node besteht aus dem `pbs_mom` Dienst, der die erhaltenen Jobs startet, überwacht und beendet.
- *Resources* sind optionale Knoten, die eine Sammlung verschiedener physikalischer oder virtueller Resourcen für die anderen Knoten zur Verfügung stellen. Resourcen können beispielsweise persistente Datenspeicherung im Netzwerk oder Lizenzmanager sein.

Hadoop on Demand nutzt den Torque Resource Manager zur Installation und Konfiguration von Hadoop auf den physikalischen oder virtuellen Maschinen. Das bedeutet, die Grundvoraussetzung um HOD nutzen zu können ist ein fertig installiertes und konfiguriertes Torque Cluster mit mindestens einer Master Node, einer Compute Node und einer Submit/Interactive Node. Die Knoten können auf einer oder mehreren Maschinen installiert sein.

Um HOD in einer Private oder Public IaaS Cloud nutzen zu können, müssen die gestarteten virtuellen Instanzen zuerst ein Torque Cluster bilden. Beim Start des Clusters muss eine Software die Konfiguration der einzelnen Knoten verteilen.

HaaaS bzw. die HaaaS API hat gegenüber HOD folgende Vorteile.

- Sie kommt ohne zusätzliche Software wie Torque aus, da sie ausschließlich Standardsoftware wie SSH nutzt.

- HaaaS bietet eine Weboberfläche zur Nutzung des Dienstes mit einem Browser.
- Der Start eines HOD Clusters dauert länger, da erst ein Torque Cluster aufgebaut werden muss um danach erst Hadoop zu installieren, konfigurieren und zu starten.
- Die HaaaS API benötigt keine dedizierte Maschine im Cluster bzw. in der Cloud, die das Hadoop Cluster verwaltet. Die Verwaltung des Hadoop Clusters übernehmen Namenode und Jobtracker.
- Ein Nutzer muss sich, bei der Nutzung der HaaaS API, nicht in die Software Torque Resource Manager einarbeiten.

5.4.2 Cloud Skripte von Cloudera

Cloudera bietet eine Sammlung von Skripten an, um ein Hadoop Cluster in EC2 zu deployen. Die Cloud Skripte bestehen aus Shell- und Python Skripten, die von Cloudera entwickelt sind. Das Deployment eines Hadoop Clusters mit Hilfe der Cloud Skripte läuft folgendermaßen ab.

1. Das Hauptskript startet mit der Hilfe der Boto Bibliothek [16] und der Anzahl der Hadoop Knoten als Parameter, die angegebene Zahl an virtuellen Maschinen in Amazon EC2. Das gestartete Image ist ein Ubuntu Standard Image für EC2 von Canonical. Die folgenden Schritte erfolgen sequentiell pro virtueller Maschine. Bei dem Start markiert das Skript eine virtuelle Maschine als Master.
2. Die Skripte kopieren die `hadoopec2initremote.sh` Datei auf die einzelnen virtuellen Maschinen und führen das Skript aus.
3. Im dritten Schritt installiert das Skript den Hadoop User, die Repositorys, Hadoop und einen thttpd Webserver auf der virtuellen Maschine. Der Webserver startet auf Port 80 und enthält nur eine HTML Seite, die Links zu der Weboberfläche des Namenode bzw. Jobtracker des Hadoop Clusters darstellt. Vor der Hadoop Konfiguration stellt das Skript mit Hilfe des Metawebservers (URL: `http://169.254.169.254/latest/meta-data/public-hostname`) auf dem Cloudcontroller von EC2 fest, ob die virtuelle Maschine als Master gekennzeichnet ist, um dem Master eine andere Konfiguration zu installieren.
4. Zuletzt startet das Skript das Hadoop Cluster, in dem die Skripte auf jeder einzelnen Maschine Hadoop starten.

Die Cloud Skripte von Cloudera können im Gegensatz zu HaaaS, nur Amazon EC2 als IaaS Cloud nutzen. Damit ein Nutzer eine andere IaaS Cloud nutzen kann, müsste er die Implementierung der Skripte ändern. Die HaaaS API kann mit Hilfe eines Konfigurationsparameters den IaaS Dienst eines weiteren Anbieters nutzen, vorausgesetzt die genutzte IaaS Cloud hat eine EC2 kompatible Schnittstelle. Falls diese Schnittstelle nicht vorhanden ist, lässt sich die API mit der Implementierung einer Schnittstelle einfach erweitern.

Die Cloud Skripte nutzten nur einen Thread für die Installation und Konfiguration von Hadoop in EC2. Die API konfiguriert die Maschinen in einem Hadoop Cluster parallel, womit die API bei einer größeren Anzahl von Maschinen einen Zeitvorteil hat. HaaaS bzw. die HaaaS API hat gegenüber den Cloud Skripten von Cloudera folgende Vorteile.

- HaaaS konfiguriert die virtuellen Maschinen in der Cloud parallel.
- Eine mit HaaaS gestartete Cloud ist schneller verfügbar, da Hadoop vor dem Start im Image der virtuellen Maschinen installiert ist.
- Mit HaaaS ist es möglich, das Hadoop Cluster in allen IaaS Clouds mit kompatibler EC2 Schnittstelle zu deployen.
- Die HaaaS API lässt sich mit anderen verfügbaren Cloudschnittstellen wie z.B. der *Open Cloud Computing Interface (OCCI) [59]* erweitern
- Die HaaaS Anwendung kann auf einem Application Server installiert sein, um beispielsweise intern in einem Unternehmen Hadoop als SaaS anzubieten. Die Cloud Skripte sind nur in der Kommandozeile ausführbar.

Cloudera arbeitet aktuell an einer kostenpflichtigen Erweiterung der *Cloudera Hadoop Disitribution (CDH)* Pakete und der Cloud Skripte namens *Cloudera Enterprise*. Die Erweiterung besteht aus Werkzeugen, um ein Hadoop Cluster zu überwachen und mit LDAP Authentifizierung abzusichern.

5.4.3 Amazon Elastic MapReduce

Amazon Elastic MapReduce (siehe Kapitel 4) installiert und konfiguriert Hadoop Cluster transparent für den Nutzer. Durch die Transparenz ist es für den Nutzer nicht möglich das Cluster mit Hilfe der Namenodes, Datanodes, Jobtracker oder Tasktracker zu steuern.

Das Ziel von Amazon Elastic MapReduce ist die Ausführung von MapReduce Anwendungen. Elastic MapReduce startet das Cluster, führt den Job-Flow

aus und schaltet das Cluster danach ab. Der Nutzer kann nur mit Hilfe der Elastic MapReduce Kommandozeilenwerkzeugen bzw. mit der Weboberfläche einen Job-Flow anlegen und starten.

HaaaS fokussiert die Installation und Konfiguration eines Hadoop Clusters und bietet folgende Vorteile gegenüber Elastic MapReduce.

- Es ist möglich beliebige Hadoop-Konfigurationen und Erweiterungen in das Deployment zu integrieren, falls diese in der Standardkonfiguration einstellbar bzw. aktivierbar sind.
- Durch die Aufteilung der API in Komponenten existiert eine Flexibilität, um beispielsweise weitere Hadoopkomponenten wie HBase, Hive, Pig oder Chuwaka einzubinden. Die Einbindung der Hadoopkomponenten ist mit Hilfe der Erweiterung der Implementierungen der `InstanceConfiguration` Klassen möglich.
- Die Flexibilität der API bietet die Voraussetzungen für eigene Deploymentlösungen, die weitere clusterfähige Software in der Cloud installiert und konfiguriert.

5.5 Vergleich von on demand Hadoop Clustern mit physikalischen Hadoop Clustern

In klassichen physikalischen Hadoop Clustern muss ein Nutzer die Anwendung in eine Queue einstellen, die das Cluster abarbeitet. Der Nutzer muss unter Umständen längere Zeit warten, bis ein Zeit-Slot im Cluster verfügbar ist, um die MapReduce Anwendung durchzuführen.

HaaaS bietet den Vorteil ein vollständig nutzbares on demand Hadoop Cluster aufzubauen. Mit der Verwendung einer IaaS Cloud als Basis, die die on demand Eigenschaft ermöglicht, verringert sich die Gesamtlaufzeit (Summe aus Wartezeit bis zur Anwendungsdurchführung und Laufzeit) einer kleineren Hadoop MapReduce Anwendung gegenüber eines physikalischen Hadoop Clusters.

HaaaS verwendet die in einer IaaS Cloud eingesetzten virtuellen Maschinen zur Installation und Konfiguration des Hadoop Clusters. Die virtuellen Maschinen sind in der Regel mit einem Hypervisor wie XEN [37] oder KVM [62] virtualisiert. Der Hypervisor, der eine Abstraktionsschicht zwischen virtualisiertem Betriebssystem und physikalischen Rechner ist, verwaltet die Hardware der physikalischen Maschine und verteilt die zur Verfügung gestellten Ressourcen an die virtuellen Maschinen.

Die Verwaltung der Ressourcen und virtuellen Maschinen auf einer physikalischen Maschine kostet Laufzeit, da der Hypervisor alle Zugriffe auf Laufwerke oder Geräte der virtuellen Maschinen koordinieren muss. Durch den Koordinationsaufwand des Hypervisors eignen sich physikalische Hadoop Cluster bei I/O intensiven Anwendungen besser als on demand Hadoop Cluster.

Die weiteren Vor- bzw. Nachteile von virtualisierten Clustern gegenüber Cluster, die auf physikalischen Maschinen installiert sind, sind die Vor- bzw. Nachteile der Virtualisierung von Servern, die in [3] diskutiert sind.

Kapitel 6

Evaluierung der Anwendung

Die Weboberfläche und die API der Anwendung sind mit Hilfe der Anforderungen aus Kapitel 5.1 evaluiert. Die Hauptanforderung, beschrieben in Kapitel 5.1.3, ist der Start eines Hadoop Clusters in einer IaaS Cloud. Die Evaluierung der Anforderungen 1, 2, 3, 5 ist in den folgenden Unterkapiteln beschrieben.

Der Deployment Konfigurationsparameter `clouderaconfig.doparallel configurations` ist die Umsetzung der Anforderung 2 (siehe Kapitel 5.1.2). Die API nimmt für diesen Parameter die Werte `true` oder `false` an und bestimmt damit, ob die API den gesamte Deploymentvorgang in einem oder mehreren Thread(s) ausführt.

Die für Hadoop-as-a-Service verwendete SSH Bibliothek startet bei der Nutzung von SSH Tunneln für jeden Tunnel einen eigenen Thread, um jeden Tunnel jederzeit für Verbindungen offen zu halten. Durch große Anzahl an entstehenden Threads bei der SSH Tunnel Nutzung, beschränkt die API die maximale Anzahl an paralleler Konfigurationen auf 10. Das bedeutet, dass die API zu einem Zeitpunkt maximal 10 Maschinen gleichzeitig konfiguriert.

Die Anforderung 4, beschrieben in Kapitel 5.1.4 ist mit der Implementierung der Weboberfläche evaluiert. Die Weboberfläche nutzt die HaaaS API entkoppelt und tätigt nur einen Methodenaufruf in die API, um das Deployment zu starten.

6.1 Evaluierung mit Eucalyptus

Aus Anforderung 1 (siehe Kapitel 5.1.1), entstand die Anwendung auf Basis von Eucalyptus 1.6.2. Das Steinbuch Centre for Computing (SCC) [57]

im Karlsruher Institut für Technologie (KIT) stellte zur Entwicklung eine Eucalyptus Cloud zur Verfügung.

Eucalyptus [45] steht für „**E**lastic **U**tility **C**omputing **A**rchitecture for **L**inking **Y**our **P**rograms **T**o **U**seful **S**ystems“ und ist eine IaaS Private Cloud Software. Die University of California in Santa Barbara (UCSB) entwickelte die Software als Open Source Reimplementierung der Amazon Webservice Dienste EC2 und S3. Die UCSB hat eine Ausgründung gebildet, die Eucalyptus Systems, Inc., die Eucalyptus weiter entwickelt.

Die von Eucalyptus angebotenen Schnittstellen der Webservices sind kompatibel zu den etablierten EC2 und S3 Schnittstellen von Amazon. Die Private IaaS Cloud des SCC besteht aus einem Cloud Controller und acht Nodes. Die Hardware der Cloud besteht aus fünf HP Blade ProLiant BL2x220c G5 Blade-Systemen, die jeweils zwei Server mit zwei Quad-Core Intel Xeon 2,33 GHz Prozessoren und 16 GB Arbeitsspeicher enthalten.

In der Cloud ist ein durch das Imageskript angefertigtes Image gespeichert. Das Image beinhaltet alle nötigen Pakete der *Cloudera's Distribution for Hadoop Version 2 (CDH2)*, vom März 2010.

Es existieren zwei topologisch unterschiedliche Evaluierungsaufbauten, mit deren Hilfe Anforderung 5 (siehe Kapitel 5.1.5) evaluierbar ist. Beide Evaluierungsaufbauten sind in Abbildung 6.1 dargestellt.

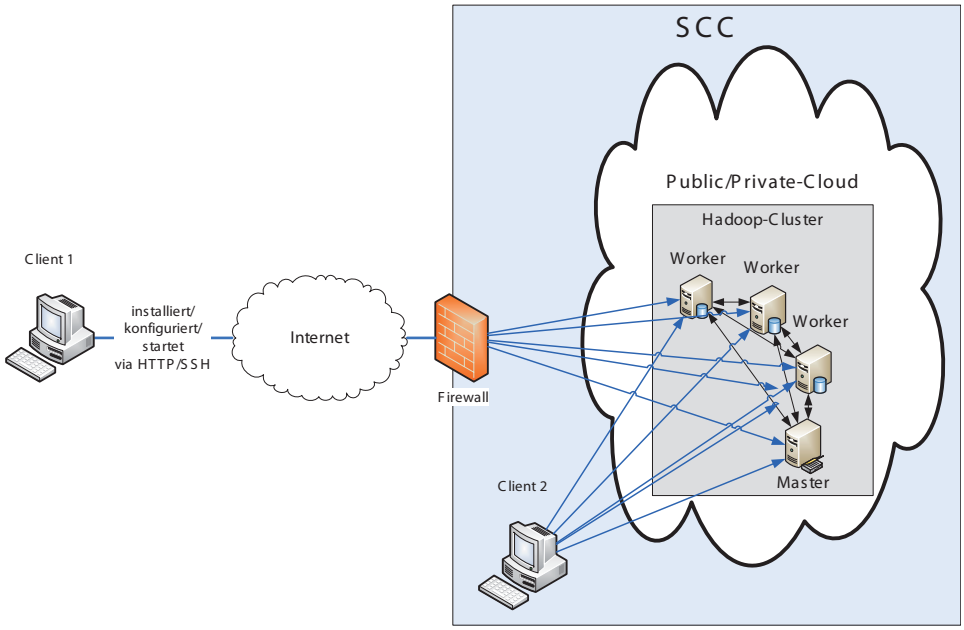

Abbildung 6.1: Evaluierung von HaaaS mit Eucalyptus im SCC

Die Abbildung zeigt zwei Clients mit unterschiedlichen Zugangsmöglichkeiten zu den virtuellen Maschinen in der Cloud.

- Client 1 ist außerhalb des internen Netzwerks im SCC und kann daher nur über die Firewall, die als Proxy dient, mit den virtuellen Maschinen kommunizieren. Die einzige Kommunikationsmöglichkeit ist ein SSH Tunnel durch die Firewall. Mit Client 1 existiert der Aufbau speziell für die Evaluierung von Anforderung 5.
- Client 2 befindet sich innerhalb des internen SCC Netzwerks und kommuniziert ohne SSH Tunnel mit den virtuellen Instanzen in der IaaS Cloud.

Beide Clients starten ein Deployment eines Hadoop Clusters mit einer einfachen Konfiguration für Hadoop und einer einfachen Konfiguration für das Deployment.

Die Imageerstellung vor dem Deployment, die in Anforderung 3 (siehe Kapitel 5.1.3) beschrieben ist, ist mit Hilfe einer virtuellen Maschine in der SCC Cloud evaluierbar. Der Nutzer muss die virtuelle Maschine mit einem Linux Image starten, dass die Pakete der `euca-tools` enthält. Alternativ kann der Nutzer die Pakete der `euca-tools` nachträglich installieren und konfigurieren.

Der Nutzer passt die Konfigurationsdatei, die im Archiv des mit dieser Arbeit entstandenen Skripts enthalten ist, an und führt danach das Skript mit der `auto` Option aus. Nach der Imageerstellung lädt der Nutzer das Image des Betriebssystems mit den installierten Hadooppaketen in die Cloud. Dies ist mit der Skriptoption `upload` durchführbar, die nach dem Upload des Images die *Image ID* ausgibt.

6.2 Evaluierung mit Amazon EC2

Hadoop-as-a-Service ist mit Hilfe von EC2 evaluiert. Die Evaluierung ist, durch die Verwendung der Amazon EC2 Java Bibliothek zum Zugriff auf die Dienste einer IaaS Cloud, möglich. Der topologische Aufbau der Evaluierung ist in Abbildung 6.2 dargestellt. Der topologische Aufbau ist mit dem Aufbau der Evaluierung mit Eucalyptus vergleichbar. Es besteht ein Unterschied, in den möglichen Kommunikationswegen. Eine virtuelle Maschine in EC2 ist für einen Nutzer direkt erreichbar, da das EC2 Netzwerkgateway die Verbindungen weiterleitet.

Damit die HaaaS API die HTTP Verbindungen für Hadoop und die SSH Verbindungen zur Konfiguration aufbauen kann, müssen die *Security Groups*

in EC2 die jeweilen Ports durchlassen. *Security Groups* stellen eine vom Nutzer konfigurierbare Firewall in EC2 dar. Der Nutzer erstellt eine Security Group, in der er Ports definiert, die bei einer virtuellen Maschine außerhalb EC2 zugänglich sind. Beim Start einer virtuellen Maschine kann der Nutzer eine oder mehrere Security Groups für die Maschine zuweisen.

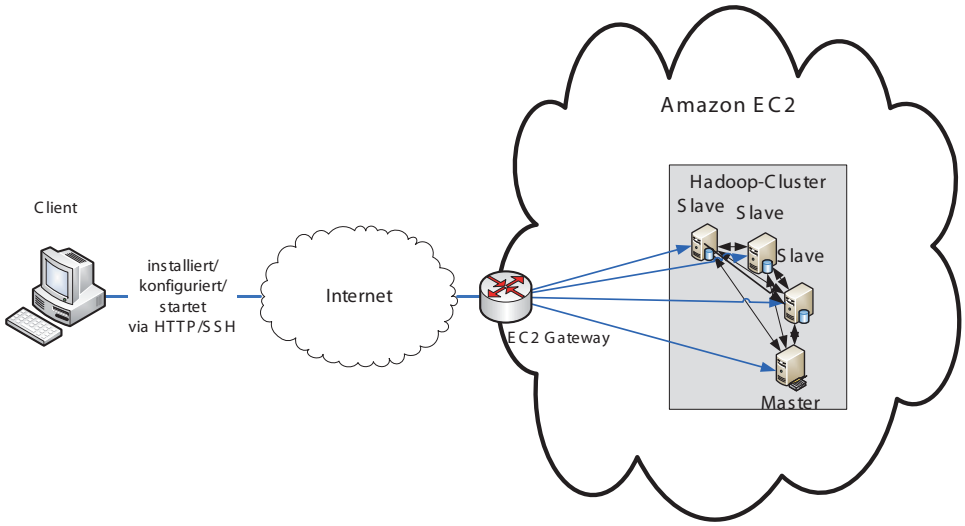

Abbildung 6.2: Evaluierung von HaaaS in EC2

6.3 Evaluierung im Power-Grid-Exist (PGE)

Power-Grid-Exist (PGE) [4] ist ein Cluster der Hochschule Mannheim, der als Plattform für wissenschaftliche Projekte dient. Die Architektur des Clusters folgt dem Master-Worker-Schema. Der Cluster besteht aus 14 Servern. Zwei Server erfüllen die Rolle des Masters im Cluster, die jeweils sechs Worker kontrollieren. Die Master Knoten verteilen die von außen ankommenden Jobs an die Worker, die diese lokal ausführen.

Das PGE Cluster ist vom internen Netzwerk der Hochschule Mannheim isoliert. Die beiden Master Knoten stehen für das Hochschulnetz und für die Knoten des Clusters als Zugangspunkt bzw. Gateway zur Verfügung. Jeder Master ist mit den sechs Workern über einen 48 Port Switch verbunden. Die beiden Switches sind vierfach miteinander Verbunden, um genug Bandbreite zur Verfügung stellen zu können, falls z.B. ein Master ausfällt. Durch dieses Setup besteht die Möglichkeit, dass ein Master den anderen bei einem Ausfall ersetzt.

Die installierte Software im PGE [4] Cluster, wie beispielsweise MPICH2 [36] oder LAM/MPI [56], ist für den Betrieb eines Grids konzipiert. Grids sind,

wie in Kapitel 2.1.1 aus der Architektur- und Konzeptionssicht von Clouds zu unterscheiden. Die installierte Software im PGE stellt aktuell[1] keinen IaaS Dienst zur Verfügung, der für die Hadoop-as-a-Service API Voraussetzung ist, um die virtuellen Maschinen zu instantiieren.

Der Einsatz von Hadoop-as-a-Service ist in der nächsten Ausbaustufe [4] des PGE, dem Power Cloud Exist (PCE) denkbar, falls die PCE Cloud ein IaaS Dienst mit einer EC2 kompatiblen Schnittstelle anbietet, beispielsweise auf Basis von Eucalyptus.

[1] Stand: Mai 2010

Kapitel 7

Ausblick

Die Firmen und Projekte aus [17] nutzen Hadoop Cluster. Die Größe der Cluster variiert von vier Knoten bis zu mehreren Tausend Knoten bei Yahoo!. Die kleineren Projekte nutzen Hadoop, für seltene Datenanalysen größerer Datenmengen (beispielsweise die Monatliche Analyse der Zugriffe einer Website).

Durch die seltene Nutzung ist der Aufbau eines firmeninternen Hadoop Clusters für kleinere Projekte betriebswirtschaftlich uninteressant, da die Fixkosten für die Anschaffung und Wartung von Hard- und Software nicht im Verhältnis zur Nutzungsdauer stehen. Die Nutzung eines Public Cloud Hadoop Dienstes ist für bestimmte Anwendungsgebiete, wie z.B. die Analyse von geheimen firmeninternen Daten, aus Sicherheits bzw. Datenschutzgründen nicht geeignet.

HaaaS erstellt, mit Hilfe einer bestehenden Private Cloud einer Organisation, ein On Demand Hadoop Cluster. Die Entkopplung von API und Weboberfläche sowie die Entkopplung der Komponenten innerhalb der API ermöglicht eine Erweiterbarkeit. Eine denkbare Ergänzung in der Software wäre die Unterstützung von persistentem Speicher (z.B. mit Elastic Block Storage bei Instanzen in EC2) einer virtuellen Maschine in einem Hadoop Cluster, um einen Neustart des Clusters zu ermöglichen. Durch den persistenten Speicher ist es möglich, dass ein Nutzer die Instanzen eines Hadoop Cluster herunterfährt und zu einem anderen Zeitpunkt wieder startet, ohne die Daten in HDFS zu verlieren.

Die Entkopplung der API in Komponenten bietet weitere Erweiterunsgmöglichkeiten. Der Configurator verteilt die Konfigurationen (`InstanceConfiguration`) an die Maschinen. Die Aufteilung bzw. Entkopplung bietet die Möglichkeit, um andere Konfigurationen zu entwicklen und deployen, beispielsweise clusterfähige Sofware wie MPI [36].

Ein weiteres denkbares Szenario für die Nutzung des Webservices der Weboberfläche auf dem Application Server, ist die Implementierung einer Android oder iPhone Applikation, die mit dem Webservice kommuniziert. Der Start und die Überwachung des Deployments könnte von dem Smartphone ausgehen.

In der aktuellen HaaaS Implementierung befinden sich die Daten des Deploymentvorgangs in einer Session, die der Application Server verwaltet. Schließt der Nutzer den Browser, ist die Session für den Nutzer nicht mehr verfügbar. Eine Erweiterung der Webanwendung um ein Benutzerkonzept mit Authentifizierung löst dieses Problem, da das System die Session mit dem Benutzer verknüpft und damit für den Benutzer immer erreichbar ist. Durch die Erweiterung entstehen zusätzliche Möglichkeiten für Erweiterungen wie beispielsweise eine Historie oder Accounting bzw. Billing.

Aktuell unterstützt die Applikation Public- oder Private IaaS Clouds, die eine EC2 kompatible Schnittstelle bieten. Durch eine erweiterte Implementierung kann die Applikation weitere IaaS Clouds einbinden.

Yahoo hat mit dem Hadoop Summit [65] im Juni 2010 gezeigt, die Entwicklung von Hadoop auf die Marktreife zu fokussieren. Yahoo entwickelt z.B. aktuell eine Hadoop Erweiterung namens *Hadoop with Security*. Hadoop stellt aktuell, bis auf ein Usernamemapping, keine Sicherheitsfunktionen zur Verfügung. Die Erweiterung integriert Kerberos [58] in Hadoop, um Authentizierung für die Hadoopkomponenten HDFS und MapReduce zur Verfügung zu stellen.

Literatur

[1] *Chukwa, a large-scale monitoring system.* Chicago, IL, Oktober 2008 . – 1–5 S.

[2] Baun, C. : *Cloud Computing : Web-basierte dynamische IT-Services.* Springer, 2010 (Informatik im Fokus)

[3] Baun, C. ; Kunze, M. ; Ludwig, T. : Servervirtualisierung. In: *Informatik Spektrum* 32 (2009), Nr. 3, S. 197–205

[4] Bengel, G. ; Senst, S. ; Szkutnik, M. ; Schmitt, M. ; Abu-Rab, S. ; Defiebre, D. ; Tauer, L. ; Alkuns, B. : Power Grid Exist: Ziele, Architektur und Aufbau / Hochschule Mannheim. 2009. – Forschungsbericht

[5] Berlich, R. ; Kunze, M. : Das Grid für den LHC. In: *Welt der Physik* (2008). http://www.weltderphysik.de/de/6181.php

[6] Dean, J. ; Ghemawat, S. : MapReduce: Simplified Data Processing on Large Clusters / Google Inc. Version: 2004. http://static.googleusercontent.com/external_content/untrusted_dlcp/labs.google.com/de//papers – Forschungsbericht

[7] Dueck, G. : Cloud - über die Wolke des IT-Himmels. In: *Informatik Spektrum* 32 (2009), Nr. 3, 260-266. http://dblp.uni-trier.de/db/journals/insk/insk32.html#Dueck09b

[8] Ghemawat, S. ; Gobioff, H. ; Leung, S.-T. : The Google File System / Google Inc. Version: 2003. http://labs.google.com/papers/gfs-sosp2003.pdf. – Forschungsbericht

[9] Jetter, O. ; Dinger, J. ; Hartenstein, H. : Leistungsbewertung der Cloud-Plattform Apache Hadoop am Anwendungsbeispiel verteilter Simulationen. In: *Praxis der Informationsverarbeitung und Kommunikation (PIK)* 33 (2010), S. 3644

[10] Papsdorf, C. : *Wie Surfen zu Arbeit wird : Crowdsourcing im Web 2.0.* Frankfurt [u.a.] : Campus, 2009. – ISBN 978–3–593–39040–6

[11] Seemann, M. : *Das Google Web Toolkit: GWT.* 1. O'Reilly, 2008. – ISBN 978–3–89721–719–5

[12] Slender, G. : *Developing with Ext GWT: Enterprise RIA Development.* 1. Apress, 2009. – ISBN 1430219408

[13] Vaquero, L. M. ; Rodero-Merino, L. ; Caceres, J. ; Lindner, M. : A Break in the Clouds: Towards a Cloud Definition. In: *SIGCOMM Comput. Commun. Rev.* 39 (2009), Nr. 1, S. 50–55

[14] Weinhardt, C. : Cloud-Computing Eine Abgrenzung, Geschäftsmodelle und Forschungsgebiete. In: *Wirtschaftsinformatik* 5 (2009), S. 453–462

[15] White, T. : *Hadoop: The Definitive Guide*. O'Reilly, 2009

Web-Quellen

[16] *Boto Python Library*. http://code.google.com/p/boto/. Version: April 2010

[17] *Hadoop Powered By*. http://wiki.apache.org/hadoop/PoweredBy. Version: April 2010

[18] Amazon Webservices: *Amazon Elastic MapReduce – API Guide*. http://s3.amazonaws.com/awsdocs/ElasticMapReduce/latest/emr-api.pdf. Version: September 2009

[19] Amazon Webservices: *Amazon Elastic MapReduce – Developer Guide*. http://s3.amazonaws.com/awsdocs/ElasticMapReduce/latest/emr-dg.pdf. Version: September 2009

[20] Amazon Webservices: *Amazon Elastic MapReduce – Getting Started Guide*. http://s3.amazonaws.com/awsdocs/ElasticMapReduce/latest/emr-gsg.pdf. Version: September 2009

[21] Amazon Webservices: *Amazon EC2 Home Page*. http://aws.amazon.com/de/ec2/. Version: April 2010

[22] Amazon Webservices: *Amazon Elastic MapReduce Home Page*. http://aws.amazon.com/de/elasticmapreduce/. Version: April 2010

[23] Amazon Webservices: *Amazon Mechnanical Turk Home Page*. http://aws.amazon.com/mturk/. Version: April 2010

[24] Amazon Webservices: *Amazon S3 Home Page*. http://aws.amazon.com/de/s3/. Version: April 2010

[25] Amazon Webservices: *EC2 API Tools*. http://developer.amazonwebservices.com/connect/entry.jspa?externalID=351&categoryID=88. Version: April 2010

[26] Apache Software Foundation: *Pig Philosophy*. http://hadoop.apache.org/pig/philosophy.html. Version: September 2007

[27] Apache Software Foundation: *Hadoop Distributed Copy Manual*. http://hadoop.apache.org/common/docs/current/distcp.html. Version: Feburar 2009

[28] Apache Software Foundation: *Hadoop Word Count Beispiel*. http://wiki.apache.org/hadoop/WordCount. Version: September 2009

[29] Apache Software Foundation: *JavaDoc: Hadoop Map Reduce Job Configuration*. http://hadoop.apache.org/common/docs/r0.18.3/api/org/apache/hadoop/mapred/JobConf.html. Version: September 2009

[30] Apache Software Foundation: *The Small Files Problem*. http://www.cloudera.com/blog/2009/02/the-small-files-problem/. Version: Feburar 2009

[31] Apache Software Foundation: *Apache Lucene Home Page*. http://lucene.apache.org/. Version: April 2010

[32] Apache Software Foundation: *Apache Nutch Home Page*. http://nutch.apache.org/. Version: April 2010

[33] Apache Software Foundation: *Apache Tomcat*. http://tomcat.apache.org/. Version: April 2010

[34] Apache Software Foundation: *Hadoop on Demand*. http://hadoop.apache.org/common/docs/r0.20.2/hod_user_guide.html. Version: April 2010

[35] Apache Software Foundation: *Hadoop Subprojekte: Hive*. http://hadoop.apache.org/hive/. Version: April 2010

[36] Argonne National Labratory: *MPICH2*. http://www.mcs.anl.gov/research/projects/mpich2/index.php. Version: April 2010

[37] Citrix Systems Inc.: *XEN Hypervisor*. http://www.xen.org/. Version: Juli 2010

[38] Cleondris GmbH: *Ganymed SSH-2 for Java*. http://www.cleondris.ch/opensource/ssh2/. Version: April 2010

[39] Cloudera: *Introduction to Hive*. http://www.cloudera.com/wp-content/uploads/2010/01/6-IntroToHive.pdf. Version: November 2009

[40] Cloudera: *Cloudera – Apache Hadoop for the Enterprise*. http://www.cloudera.com. Version: April 2010

[41] Cloudstore: *Cloudstore*. http://kosmosfs.sourceforge.net/. Version: Feburar 2009

[42] Cluster Resources Inc.: *Torque Resource Manager*. http://www.clusterresources.com/products/torque-resource-manager.php. Version: April 2010

[43] D-Grid Initiative: *D-Grid Initiative*. http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf. Version: April 2010

[44] EGEE: *EGEE - Enabling Grids for E-science*. http://www.eu-egee.org/. Version: April 2010

[45] Eucalyptus Systems Inc.: *Eucalyptus*. http://open.eucalyptus.com/. Version: April 2010

[46] Foster, I. : *What is the Grid? A Three Point Checklist*. http://www.d-grid.de/. Version: Juli 2002

[47] Foundation, A. S.: *Welcome to Pig!* http://hadoop.apache.org/pig/. Version: September 2007

[48] Gartner Inc.: *Gartner Identifies the Top 10 Strategic Technologies for 2010*. http://www.gartner.com/it/page.jsp?id=1210613. Version: Oktober 2009

[49] Gartner Inc.: *Gartner's Hype Cycle Special Report for 2009*. http://www.gartner.com/DisplayDocument?ref=g_search&id=1108412&subref=simplesearch. Version: Juli 2009

[50] GoGrid: *GoGrid*. http://www.gogrid.com/. Version: Juli 2010

[51] Google Inc.: *Google App Engine Home Page*. http://code.google.com/intl/de-DE/appengine/docs/whatisgoogleappengine.html. Version: April 2010

[52] Google Inc.: *Google Apps Home*. http://www.google.com/apps/intl/de/business/index.html. Version: April 2010

[53] Google Inc.: *Google Maps Home*. http://maps.google.de/. Version: April 2010

[54] Google Inc.: *The Google Web Toolkit*. http://code.google.com/intl/de-DE/webtoolkit/. Version: April 2010

[55] Gottfrid, D. : *The New York Times Archives + Amazon Web Services = TimesMachine*. http://open.blogs.nytimes.com/2008/05/21/the-new-york-times-archives-amazon-web-services-times Version: Mai 2008

[56] Indiana University: *LAM MPI*. http://www.lam-mpi.org/. Version: April 2010

[57] Karlsruher Institut für Technologie: *Steinbruch Computing Center (SCC) Karlsruher Institut für Technologie*. http://www.scc.kit.edu/. Version: April 2010

[58] Massachusetts Institute of Technology: *Kerberos*. http://web.mit.edu/Kerberos/. Version: Juni 2010

[59] Open Cloud Computing Interface Working Group: *Open Cloud Computing Interface*. http://www.occi-wg.org/doku.php. Version: Juli 2010

[60] Oracle: *Project OCFS*. http://oss.oracle.com/projects/ocfs/. Version: Juli 2010

[61] Oracle Inc.: *Java Swing*. http://java.sun.com/docs/books/tutorial/uiswing/start/about.html. Version: April 2010

[62] Redhat: *KVM Hypervisor*. www.linux-kvm.org. Version: Juli 2010

[63] Salesforce: *Salesforce Application Developement*. http://www.salesforce.com/platform/. Version: April 2010

[64] Sencha Inc.: *Ext GWT*. http://www.sencha.com/products/gwt/. Version: June 2010

[65] Yahoo Inc.: *Hadoop Summit 2010*. http://developer.yahoo.com/events/hadoopsummit2010/agenda.html. Version: Juni 2010