Analyse von Log-Files eines Web-Servers mithilfe eines Rechnerverbunds

Steffen Hafner und Daniel Landler-Gärtner

Hochschule
Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

PROJEKTARBEIT

Systemadministration

in der Hochschule Ravensburg-Weingarten

im April 2017

Erklärung

Wir erklären eidesstattlich, dass wir die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

der Hochschule Ravensburg-Weingarten, am 5. April 2017

Steffen Hafner und Daniel Landler-Gärtner

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Verfügbarkeit von Daten hat sich in den vergangenen zehn Jahren drastisch verändert. Die Anzahl verschiedener Datenquellen steigt stetig durch die zunehmende Verbreitung mobiler und internetfähiger Geräte. Dadurch sehen sich Unternehmen heute mit sehr viel größeren Datenmengen konfrontiert. Diese gilt es zu erfassen, zu speichern und auszuwerten. Dabei ist es nicht nur die Datenmenge selbst, die den Unternehmen Probleme bereitet, sondern darüber hinaus auch die Struktur und die Art der Daten, sowie die Geschwindigkeit, mit der sie anfallen.

Der Begriff Big-Data ist in den letzten Jahren vom reinen Buzz-Word hin zu einem greifbaren technischen Begriff gereift. Big-Data sind Datenmengen, die zu groß für traditionelle Datenbanksysteme sind sowie eine hohe Schnelllebigkeit besitzen. Diese Datenmengen sind entweder unstrukturiert oder semi-strukturiert und enstsrpechen somit nicht den Richtlinien herkömmlicher Datenbanksysteme. Die Herausforderung liegt darin die Daten dennoch zu speichern und zu verarbeiten, damit neue Informationen gewonnen werden können. Mögliche neue Informationen sind z.B. empfohlene Kontakte in sozialen Netzwerken, passende Produktempfehlungen in E-Commerce Lösungen oder Artikelvorschläge auf Nachrichtenseiten.

Eine treffende Definition von Big-Data lässt sich am Besten durch die drei V veranschaulichen. Volume (Speichergröße und Umfang), velocity (die Geschwindigkeit mit der Datenmengen generiert und transferiert werden) und variety (Bandbreite der Datenquellen)[1]. Diese Definition kann durch value und validity ergänzt werden, welche für einen unternehmerischen Mehrwert und die Sicherstellung der Datenqualität stehen[2].

Auf Grund des hohen Aufwands von Echtzeit-Analysen großer Datenmengen wird Big-Data häufig in Verbindung zu Cluster Computing gebracht. Um Cluster Computing zu realisieren ist ein Framework nötig um die Verarbeitung der Daten auf eine große Anzahl an Computern zu verteilen.

Die Visualisierung der gewonnen Informationen erfordert die Bildung von Korrelationen zwischen den einzelnen Datensätzen, um diese in Abhängigkeit voneinander präsentieren zu können. Dies erfordert, im Gegensatz zu normalisierten Daten von relationalen Datenbanken, bei Plain-Text-Analysen einen erheblichen Mehraufwand.

[1] Gartner IT Glossary: http://www.gartner.com/it-glossary/big-data

[2] Big Data – Fluch oder Segen? – Unternehmen im Spiegel gesellschaftlichen Wandels

Kapitel 2

Problem

Bereits vor dem Aufkommen von Big-Data haben Unternehmen Log-Files zur Gewinnung von Einblicken genutzt. Jedoch ist das Problem, dass durch das exponentielle Wachstum aller Datenquellen die Verwaltung und Analyse der Log-Files zu einer immer größeren Herausforderung wird. Log-Files enthalten eine große Menge an Informationen, wobei nicht alle für den jeweiligen Betreiber von gleicher Bedeutung sind. Zudem liegen die Informationen innerhalb der Log-Files in einem schlecht leserlichen Format vor, weshalb eine Analyse von relevanten Informationen und Korrelationen sehr aufwendig ist.

Serverlogs können abhängig von dem Log-Level der jeweiligen Architektur sehr groß ausfallen, wodurch die manuelle Verwaltung und Analyse nahezu unmöglich wird. Andererseits ist die Analyse von System-Logs notwendig um beispielsweise potenzielle Sicherheitsrisiken und Netzwerkfehler erkennen zu können.

Es gibt prinzipiell zwei Arten von Log-Files:

1. **Ereignis-Logs** – ermöglichen einen umfassenden Überblick über die Funktionalität des Systems sowie allen Komponenten zu einem bestimmten Zeitpunkt.
2. **Benutzer-Logs** – ermöglichen einen detailierten Einblick in das Nutzerverhalten wie z.B. auf Webseiten. Durch die Analyse der Benutzer-Logs können genauere Informationen über das Verhalten der Benutzer gewonnen werden als mit gewöhnlichen Webanalyse-Diensten wie Google Analytics oder Omniture.

Ein effizienter und automatisierter Prozess wird benötigt damit schnell und akkurat Muster erkannt werden können sowie die großen Datenmengen der Serverlogs erfolgreich bewältigt werden können. Andererseits laufen Unternehmen Gefahr wertvolle Informationen in der riesen Datenflut zu verlieren und dadurch einen datengestützen Wettbewerbsvorteil zu verlieren.

2.1 Webserver Log-File Analyse

Dig Logfile-Analyse bezeichnet den Prozess der gezielten Überprüfung und Auswertung eines Logfiles. Durch die Auswertung von Webserver Logs können allgemeine Informationen über das Verhalten und die Aktivitäten der Seitenbesucher gewonnen werden.

Webserver Log-Files enthalten folgende Informationen:

- IP-Adresse und Hostname
- Zugriffszeitpunkt
- Vom User verwendeter Browser
- Vom User verwendetes OS
- Herkunftslink bzw. -URL
- Verwendete Suchmaschine inklusive genutzter Keywords
- Verweildauer
- Anzahl aufgerufener Seite
- Zuletzt geöffnete Seite vor dem Verlassen der Webseite

Eines der größten Probleme der Webserver-Logfile-Analyse wird durch das zustandslose HTTP Protokoll verursacht. Durch die separate Behandlung der Anfragen, behandelt der Webserver zwei verschiedene Seitenaufrufe eines Clients als zwei unterschiedliche Instanzen. Wodurch eine Analyse des Nutzerverhaltens deutlich erschwert wird.

Um diesen Problemen entgegen zu wirken gibt es zwei gängige Lösungsmöglichkeiten:

1. **Vergabe einer Session-ID:** Die Session-ID ist eine serverseitig generierte ID, die im Browser des Nutzers gespeichert wird. Alle folgenden Anfragen eines Nutzers werden durch die vergebene ID kenntlich gemacht.
2. **Nutzeridentifikation via IP-Adresse:** Nutzer werden über ihre eindeutige IP-Adresse erkannt und bei allen folgenden Anfragen durch diese identifiziert. Voraussetzung dafür ist die Zustimmung des Nutzers zur Erhebung seiner vollständigen IP-Adresse zu Analysezwecken. Ein weiteres Problem ergibt sich aus der dynamischen Vergabe von IP-Adressen oder durch die mehrfache Nutzung der gleichen IP-Adresse.

2.2 Log-Files des HRW Webserver

Für die Veranschaulichung des genannten Problems verwenden wir die Log-Files des Webservers der Hochschule Ravensburg-Weingarten. Um die Log-Files des HRW-Webservers nutzen zu können, mussten diese aus Datenschutzgründen zunächst anonymisiert werden.

Ein anonymisierter Log-File Eintrag sieht wie folgt aus:

123.234.12.34 - - [06/Apr/2017:06:24:35 +0200] "GET /c/document_library/get_file ?uuid=0b9ef55d-812a-4915-9de1-fe5e7f3a0021&groupId=65432 HTTP/1.1"200 6021

"http://www.hs-weingarten.de/web/willkommen/startseiteMozilla/5.0 (iPad; CPU OS 10/2/1 like Mac OS X) AppleWebKit/602.4.6 (KHTML, like Gecko) Version/10.0 Mobile/14D27 Safari/602.1"

123.234.12.34 repräsentiert die IP-Adresse des Clients der eine Anfrage an den Webserver gestellt hat. Die IP-Adresse ist durch einen Zufallswert so ersetzt, dass ein eindeutiger Zufallswert für jede vorkommende IP-Adresse verwendet wird. Dabei ist ein Zusammenhang noch erkennbar, allerdings eine Rücküberführung unmöglich.

"GET /c/document_library/get_file?uuid=0b9ef55d-812a-4915-9de1-fe5e7f3a0021&groupId=65432 HTTP/1.1" Der Request Eintrag enthält die verwendete Zugriffsmethode, die angefragte Ressource sowie die HTTP-Version.

200 6021 repräsentiert den Statuscode der Anfrage sowie die Größe des Antwortpakets für den Client.

"http://www.hs-weingarten.de/web/willkommen/startseite" enthälft Informationen über welche Seite der Client auf die angefragte Seite zugegriffen hat.

"Mozilla/5.0 (iPad; CPU OS 10/2/1 like Mac OS X) AppleWebKit/602.4.6 (KHTML, like Gecko) Version/10.0 Mobile/14D27 Safari/602.1" zeigt Informationen die der Browser des Clients über sich selbst berichtet.

2.3 Geplante Umsetzung

Aufgrund der hohen Komplexität eines realen Computer-Clusters soll in diesem Projekt ein Prototyp entstehen, der ein virtuelles Computer-Cluster simuliert und dadurch die möglichen Funktionen eines realen Computer-Clusters veranschaulicht. Zudem soll der Prototyp einfach portierbar, skalierbar und nutzbar sein.

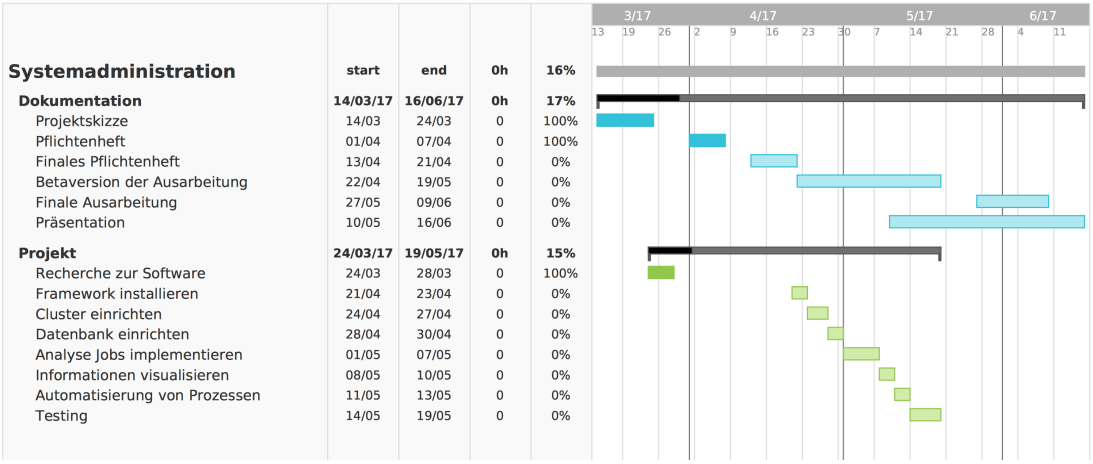

Abbildung 2.1: Gantt-Diagramm

Kapitel 3

Anforderungsanalyse

Die geplanten Anforderungen sind aufgeteilt in *Muss* und *Kann* Kriterien.

Muss-Kriterien

- Verarbeitung erfolgt in virtuellem Cluster
- Open-Source Framework für skalierbare und verteilt arbeitende Software
- Persistente Speicherung der analysierten Daten in einer Datenbank
- Open-Source Datenbank, die mit gewähltem Framework kompatibel ist
- Gewonnene Informationen sollen Nutzer in einer Textdatei zur Verfügung stehen
- Prototyp soll plattformunabhängig und maximal in einer Arbeitsstunde verwendbar sein
- Prototyp soll in isolierter Umgebung arbeiten, sodass keine Manipulation an Hostsystem stattfindet

Kann-Kriterien

- Gewonnene Informationen werden auf einer Website dargestellt
- Automatisierte Ausführung der Analyse-Jobs innerhalb des Clusters
- Automatisierte Instanziierung/Installation des Prototypen auf anderen Rechnern

Kapitel 4

Lösungsvorschläge

4.1 Framework

Gesucht ist ein Framework mit dem alle *Muss-Kriterien* bestmöglich umgesetzt werden können. Des Weiteren sollen alle Tasks im Gantt-Diagramm 2.1 fristgerecht erledigt werden können. Um das Projekt im genannten Zeitraum durchführen zu können, beschränken wir uns bei der Suche auf die bekanntesten Frameworklösungen auf dem derzeitigen Markt.

Die zwei bekanntesten Framworks für skalierbare, verteilt arbeitende Software im Zusammenhang mit großen Datenmengen sind *Hadoop* und *Spark*. Sowohl Hadoop als auch Spark werden unter Linux entwickelt und verwenden native Linux Libraries. Daher begrenzt sich unsere Auswahl des zu verwendenden Betriebssystems auf Linux Distributionen. Dies erleichtert zum einen das Einrichten und zum anderen die Wartung des Frameworks.

4.1.1 Hadoop

Hadoop ist ein Java-Framwork der Apache Software Foundation zum verteilten Speichern von Daten und zu deren parallelen Verarbeitung. Hadoop wird dabei in einem horizontal skalierbaren Cluster betrieben, das auf einfachstem Weg wie gewünscht skaliert werden kann. Große Unternehmen wie Yahoo betreiben so Cluster mit über 4000 Knoten[1]. Statt der Anschaffung neuer, schnellerer Hardware (Scale Up) wird beim Betrieb von Hadoop vielmehr die Erweiterung des Clusters (Scale Out) um weitere Knoten empfohlen.

Zu den Basiskomponenten von Hadoop, die bei der Installation mitgeliefert werden gehören:

- **Hadoop Distributed File System (HDFS):** Ein über das gesamte Cluster verteiltes Dateisystem zur Speicherung der zu verarbeitenden Daten.
- **Map-Reduce:** Ein Programmierframework zur verteilten Verarbeitung von Daten gemäß der zweiphasigen Verarbeitung durch Mapper- und Reducer-Klassen.
- **YARN:** Verwaltet die Resourcen eines Clusters dynamisch für verschiedene Jobs.

[1] Referenzzahlen für Unternehmen, die Hadoop einsetzen: http://wiki.apache.org/hadoop/PoweredBy

Zudem verfügt Hadoop über ein großes Ökosystem, das zahlreiche Technologien enthält, die ergänzend zu den genannten Technologien, installiert werden können.

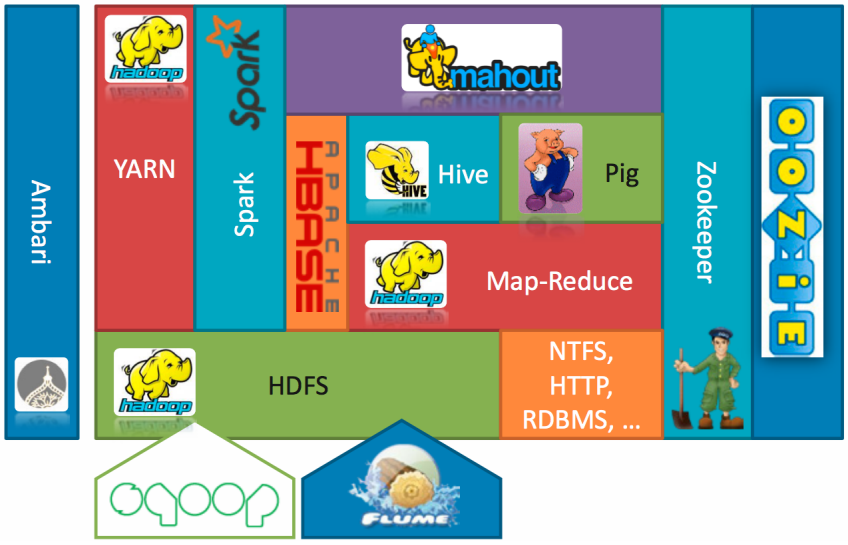

Abbildung 4.1: Hadoop-Ökosystem

4.1.2 Spark

Die Daten-Analyse Plattform Spark für clustergestützte Berechnungen wird hauptsächlich für die schnelle Ausführung von Jobs genutzt. Mit Apache Spark können Daten transformiert, fusioniert sowie mathematischen Analysen unterzogen werden. Spark ist darauf ausgelegt die Daten dynamisch im RAM des Server-Clusters zu halten und dort zu verarbeiten. Die sogenannte In-Memory-Technologie gewährleistet eine extrem schnelle Auswertung riesiger Datenmengen.

Die besondere Stärke ist das beinhaltete maschinelle Lernen (Machine Learning) mit den Zusätzen MLib (Machine Learning Bibliothek) sowie SparkR(Direkte Verwendung von R-Bibliotheken unter Spark). Dadurch lassen sich iterative Schleifen sehr gut verarbeiten was eine wichtige Vorrausetzung für Machine Learning Algorithmen darstellt.

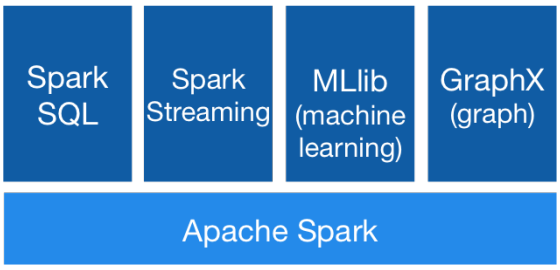

Abbildung 4.2: Apache Spark Framework

4.2 Datenbank

4.2.1 HBase

Apache HBase ist eine quelloffene, spaltenorientierte NoSQL-Datenbank, die sich in ihrer Architektur und ihrem Aufbau an Google's *BigTable* orientiert. Eine Besonderheit von HBase ist, dass ein Datensatz beliebig viele Spalten haben kann, auch mehr oder weniger als der vorige oder folgende Datensatz. Diese Eigenschaft hilft bei der schnellen persistenten Speicherung von Daten, da diese zuvor nicht normalisiert werden müssen. HBase ist Teil des Hadoop-Ökosystems 4.1 und kann daher im *verteilten Modus* ausgeführt werden, in dem sie auf Hadoop aufsetzt und das HDFS nutzt, um ihre Daten darin zu speichern. Der Vorteil beim verteilten Speichern von Daten liegt wie auch beim HDFS darin, besonders große Datenmengen unterzubringen und auf Wunsch zu skalieren, indem man weitere Knoten dem Cluster hinzufügt, wenn Performance oder Speicherkapazitäten knapp werden. Bekannte Unternehmen, die HBase verwenden sind: Adobe, Facebook, Netflix, Spotify, Yahoo! uvm.

4.2.2 Cassandra

Apache Cassandra ist ebenfalls eine spaltenorientierte NoSQL-Datenbank, die als skalierbares, ausfallsicheres System für den Umgang mit großen Datenmengen in verteilten Systemen (Clustern) konzipiert wurde. Sie entstand ebenfalls nach dem Vorbild von Google's *BigTable*. Cassandra wird häufig zusammen mit dem Framework Spark verwendet und bildet mit zusätzlichen Technologien den SMACK-Stack, bestehend aus **S**park, **M**esos, **A**kka, **C**assandra und **K**afka. Daher wird Cassandra eher mit den genannten Technologien verwendet und harmoniert weniger gut mit Hadoop. Bekannte Unternehmen, die Cassandra verwenden sind: Twitter, Digg und Reddit. Auch Facebook nutzte bis 2011 Cassandra, bis diese durch eine Kombination von HBase und HDFS ersetzt wurde.

4.3 Virtualisierungssoftware

4.3.1 Docker

Die Open-Source-Technologie Docker ermöglicht es Anwendungen mithilfe von Betriebssystemvirtualisierung in Containern zu isolieren. Durch die Verwendung von Containern können die Applikationen erstellt, ausgeführt, getestet und verteilt werden.

Durch das Verpacken von Anwendungen in standardisierte Einheiten erfolgt eine Trennung sowie Verwaltung der auf dem Rechner verwendeten Ressourcen. Diese Einheiten enthalten alle Komponenten die eine Software für die Ausführung benötigt wie z.B. Code, Laufzeitmodul, System-Tools und Systembibliotheken. Dadurch ermöglicht Docker eine schnelle, zuverlässige und einheitliche Bereitstellung von Anwendungen - unabhängig von der Umgebung.

Die Isolation der Anwendungen erfolgt bei Docker auf Software Ebene.

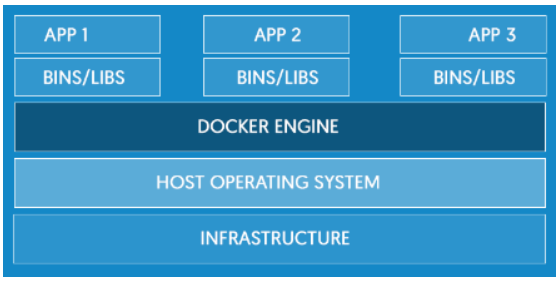

Abbildung 4.3: Aufbau Docker

4.3.2 VirtualBox

Die Virtualisierungssoftware von Oracle ermöglicht die Installation von virtuellen Maschinen auf einem Host System. Dabei greift VirtualBox auf die Hardware-Ressourcen des Hostsystems zurück und stellt einen Teil dem Gastsystem zur Verfügung. Die Isolation der Virtuellen Maschinen erfolgt auf Hardware Ebene. Jede Virtuelle Maschine besitzt somit ein eigenes Betriebssystem.

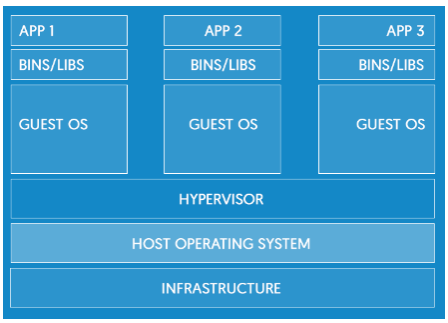

Abbildung 4.4: Aufbau VirtualBox

Kapitel 5

Lösungsauswahl anhand der Anforderungen

5.1 Framework

Als Framework für die gewünschte Umsetzung wird Hadoop genutzt. Die Kriterien die für Hadoop sprechen sind: gute Skalierbarkeit, Bekanntheit, sowie eine solide Grundausstattung, die bei der Installation des Frameworks bereits enthalten ist. Dazu gehören die bereits genannten Bausteine HDFS, Map-Reduce und YARN. Mit diesen zugehörigen Komponenten ist eine Umsetzung des gewünschten Use-Case bereits ohne zusätzliche Installation von Komponenten umsetzbar. Des Weiteren besitzt Hadoop aufgrund seiner Bekanntheit ein großes Ökosystem mit zahlreichen Technologien, die sich bei Bedarf ohne großen Aufwand hinzufügen lassen. Hadoop ist durch die Verwendung des Map-Reduce Verfahrens auf textbasierte Eingabequellen in Form von Text-Files spezialisiert, was für unseren Use-Case von großer Bedeutung ist, da die Webserver Log-Files in Form von Text-Files vorliegen.

Spark im Vergleich dazu, hat seine Vorteile in der Verarbeitung von Datenströmen. Zudem übernimmt Spark nur ein kleiner Bestandteil der Funktionen die Hadoop als Framwork übernimmt. Spark kann nicht alleine betrieben werden, sondern braucht als Basis ein Hadoop Cluster. Spark wäre somit eine Alternative zur Software, die auf dem Cluster verteilt arbeitet und das Cluster organisiert. Damit müssten bei der Verwendung von Spark noch zusätzliche Komponenten installiert werden, die bei Hadoop bereits enthalten sind. Zusätzlich werden bei der Installation von Spark Technologien mitgeliefert, die bei unserem Use-Case keinen Gebrauch finden würden, wie zum Beispiel machine learning.

5.2 Datenbank

Aufgrund der vorigen Wahl des Frameworks wird als Datenbank HBase verwendet. HBase zeigt bereits durch ihren ausgeschriebenen Namen "Hadoop Database"die enge Verknüpfung mit Hadoop. Wie bereits im Schaubild Hadoop-Ökosystem 4.1 gezeigt wurde, gehört HBase zur großen Sammlung an Technologien, die mit Hadoop bequem verknüpfbar sind. Dadurch lässt sich HBase von Haus aus mit Hadoop kombinieren und auf dem bereits vorhandenen Hadoop Clusters verteilt installieren. Zudem bietet der Aufbau der spaltenorientierten NoSQL-Datenbank mit der Besonderheit, dass Datensätze unabhängig voneinander unterschiedliche Spaltenzahlen haben können, einen Vorteil bei der Speicherung der extrahierten Daten aus den Webserver-Log Files.

Cassandra unterscheidet sich von ihrer Funktionalität her sehr wenig bis gar nicht von HBase. Der einzige ausschlaggebende Unterschied liegt darin, dass Cassandra zusammen mit Spark verwendet wird und weniger gut mit Hadoop harmoniert.

5.3 Virtualisierungssoftware

Für den Prototyp zur Analyse von Log-Files wird Docker als Virtualisierungssoftware eingesetzt, aufgrund des äußerst sparsamen Umgang mit Ressourcen sowie den kurzen Startzeiten. Docker-Container weisen eine kurze Startzeit auf, da sie nicht erst das Betriebssystem, Ressourcen und Bibliotheken laden müssen, sondern direkt auf die Komponenten und Daten der Betriebsumgebung zugreifen können.

Somit erfüllt Docker alle Anforderungen des Projekts.

Kapitel 6

Umsetzung

6.1 Hadoop

6.1.1 HDFS

Das HDFS ist Bestandteil des Hadoop Frameworks und erfüllt folgende Anforderungen:

- Betrieb auf Commodity-Hardware
- Ausfallsicherheit einzelner Knoten
- Speicherung und Verarbeitung großer Datenmengen
- Einfache Skalierbarkeit

Ein einziger Masterknoten, genannt Name-Node, verwaltet alle Metadaten des Dateisystems, darunter Verzeichnisstrukturen, Dateien und Dateizugriffe der Clients (Apache Software Foundation, 2013). Parallel dazu existieren mehrere Data-Nodes, die den Speicher verwalten, der den entsprechenden Knoten im Cluster zugeordnet ist. Das HDFS bietet ein Set an Funktionen an, das es erlaubt, Daten in das Dateisystem zu schreiben und daraus zu lesen. Es ist nicht nötig, eine eigene Partition für ein HDFS anzulegen, denn es setzt auf einem existierenden Dateisystem, z.B. dem gängigen ext4 (Fourth Extended Filesystem), auf.

6.1.2 Einrichtung und Konfiguration

Bei der Installation von Hadoop unterscheidet man hauptsächlich zwischen drei möglichen Konfigurationsarten:

Bezeichnung	Beschreibung	Verwendung
Standalone	Hadoop läuft als einzelner Java-Prozess auf einer einzigen Maschine	Apache empfiehlt, diese Konfiguration lediglich für Entwicklungs- und Debugging-Zwecke zu verwenden. Hadoop ist nach dem Entpacken bereits fertig für den Stand-alone Modus konfiguriert
Pseudo distributed	Hadoop wird auf einer einzelnen Maschine eingerichtet und jeder Hadoop-Daemon läuft in einem eigenen Java Prozess	Dieser Modus erfordert eine Konfiguration der einzelnen Komponenten. Durch das Aufsetzen eines Pseudo-Distributed-Clusters können Konfigurationen erprobt und das gesamte Setup der Hadoop-Instanz getestet werden
Fully distributed	Hadoop läuft auf mehreren Maschinen im Cluster	Ein Fully-Distributed-Setup wird in Produktionsumgebungen eingesetzt. Nur durch die Verwendung mehrerer Maschinen kann Hadoop seine Stärken in der verteilten Verarbeitung ausspielen

Abbildung 6.1: Hadoop Konfigurationsarten

Für unseren Use-Case ist die pseudo distributed Variante die Lösung, die unseren Anforderungen am besten entspricht. Mit dieser Konfiguration lassen sich die Funktionalitäten von Hadoop in einem virtuellen Cluster vollständig umsetzen. Hadoop wird offiziell von der Apache Software Foundation[1] für Unix-Systeme zur Verfügung gestellt.

Um das Hadoop Framework auf einem Unix-System zu installieren, muss Java, sowie SSH installiert sein.

Anschließend müssen die Konfigurationsdateien von Hadoop angepasst werden, um die gewünschte Konfiguration umzusetzen. Dabei ist eine Anpassung der folgenden Konfigurationsdateien erforderlich:

core-site.xml

```
1 <configuration>
2       <property>
3           <name>fs.defaultFS</name>
4           <value>hdfs://hadoop:9000</value>
5       </property>
6  </configuration>
```

fs.defaultFS in Zeile 3 bestimmt, wo der *Name-Node* zu finden ist. Der *Name-Node* wird mithilfe eines Pfads angegeben. Dabei gibt der Pfad den Rechner-Knoten an, auf dem das HDFS betrieben wird. In diesem Beispiel läuft das HDFS auf dem Rechner mit dem Hostname *hadoop* und der Service ist über den Port *9000* ansprechbar.

[1] Apache Software Foundation: http://hadoop.apache.org/releases.html

hdfs-site.xml

```
1 <configuration>
2     <property>
3         <name>dfs.replication</name>
4         <value>1</value>
5     </property>
6 </configuration>
```

Durch die Eigenschaft *dfs.replication* wird festgelegt, wie viele Kopien der zu verarbeitenden Daten später auf dem Cluster verteilt werden sollen. Da in diesem Fall nur ein Knoten verwendet wird, kann man lediglich eine Kopie speichern.

mapred-site.xml

```
1 <configuration>
2     <property>
3         <name>mapreduce.framework.name</name>
4         <value>yarn</value>
5     </property>
6 </configuration>
```

In der obigen Datei wird bestimmt, dass das neue Map-Reduce-Framework *YARN* benutzt wird.

yarn-site.xml

```
1  <configuration>
2    <property>
3      <name>yarn.nodemananger.aux-services</name>
4      <value>mapreduce_shuffle</value>
5    </property>
6    <property>
7      <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
8      <value>org.apache.hadoop.mapred.ShuffleHandler</value>
9    </property>
10   <propery>
11     <name>yarn.nodemanager.vmen-pmem-ratio</name>
12     <value>3</value>
13   </property>
14   <property>
15     <name>yarn.nodemanager.delete.debug-delay-sec</name>
16     <value>600</value>
17   </property>
18 </configuration>
```

Die Eigenschaften in den Zeilen 3 und 7, legen fest, wie Hadoop später die Verteilung (Mischung) der Knoten im Cluster handhaben wird. Der *yarn.nodemanager.vmem-pmem-ratio* stellt das Verhältnis von physikalischem zu virtuellem Speicher dar. Verwenden wir beispielsweise 4 GB RAM, stehen im Cluster 4 * 3 = 12 GB virtueller Speicher für die Ausführung der Map-Reduce-Jobs oder YARN-Anwendungen zur Verfügung. Die Eigenschaft in Zeile 15 gibt an, wie viel Sekunden die Anwendungsdaten für auf *YARN* laufende Anwendungen bestehen bleiben, bevor sie automatisch gelöscht werden.

6.1.3 Web-Interface von Hadoop

Sobald die Services von Hadoop konfiguriert und erfolgreich gestartet wurden, kann man über die zur Verfügung stehenden Web-Interfaces den aktuellen Status des Clusters und seiner einzelnen Knoten abfragen. Mithilfe der IP-Adresse und dem zugehörigen Port lassen sich folgende Seiten abrufen:

Übersicht von Hadoop und dessen Name-Node über den Port *50070* Die Seite gibt Informationen zur installierten Hadoop-Version, zum verfügbaren Festplattenspeicher, zur Startzeit des Clusters, sowie den Status der vorhandenen Knoten.

Übersicht der laufenden und abgeschlossenen Jobs auf dem Cluster, über den Port *8088* Die Übersicht zeigt zum Einen den Status einzelner Knoten im Cluster und zum Anderen Anwendungen, die im Cluster ausgeführt werden/wurden. Dabei lassen sich Log Files der absolvierten Jobs anschauen, um eventuelle Fehler ausfindig zu machen.

6.1.4 Map-Reduce

Im Jahr 2004 veröffentlichten zwei Google-Mitarbeiter ein Paper (Dean et al., 2004) zu einem neuen Ansatz, um große, unstrukturierte Daten anzuzeigen und darin suchen zu können. Aus dem Problem heraus, dass die im Unternehmen gespeicherten Datenmengen zu schnell und zu stark wuchsen, um sie mit herkömmlichen Mitteln verarbeiten zu können, entstand das in dem Paper vorgestellte Programmiermodell *Map-Reduce*. Es beschreibt nicht nur, wie man große Datenmengen durchsucht, auswertet und in Schlüssel-Wert-Paare zusammenfasst, sondern auch, wie man diese sogenannten Map-Reduce-Jobs effizient über ein Cluster auf *Commodity-Hardware* ausführt. Die Arbeitsweise des Algorithmus lässt sich in drei Prozessschritte unterteilen.

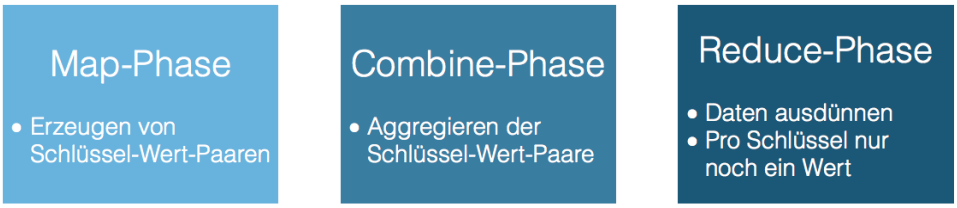

Abbildung 6.2: Die drei Prozessschritte des Map-Reduce Algorithmus

Der Prozess beginnt mit der *Map-Phase*, in der die Rohdaten entgegengenommen und ausgewertet werden. Dabei wird ein gewünschter Index festgelegt (numerisch, inkrementeller Wert, Datum, Zeichenfolge etc.). Diesen definierten Indizes werden einzelne Teile der Rohdaten zugeordnet, sodass am Ende eine Liste aus Schlüssel-Wert-Paaren vorliegt, bei denen der Index den Schlüssel und die zugehörigen Daten die Werte darstellen. Dabei können einzelne Indizes in der ersten Phase noch mehrmals vorkommen. In der *Combine-Phase* werden nun alle Werte eines Schlüssels zu einem einzelnen Schlüssel-Wert-Paar aggregiert, sodass keine doppelten Indizes mehr vorkommen. Abschließend werden in der *Reduce-Phase* die Werte dieser neuen Liste, falls möglich, zusammengefasst (z.B. aufsummiert) oder ausgedünnt.

Zur Verdeutlichung der Funktionsweise des *Map-Reduce Algorithmus* wird ein fiktives Beispiel herangezogen. Es soll der Notendurchschnitt aller Studenten einer Hochschule in den letzten zehn Jahre berechnet werden. Ziel ist es, für jedes Jahr genau eine Durchschnittsnote zu erhalten. Als Datenbasis wird eine Log-Datei herangezogen, die folgende Struktur aufweist:

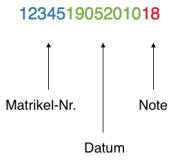

Abbildung 6.3: Noten Log-Datei

Die Log-Datei besteht aus einer Folge numerischer Werte. Die ersten fünf Ziffern stehen für die Matrikelnummer, die folgenden acht Stellen für das Datum und die letzten beiden Ziffern stehen für die Note (mit dem Wert 10 multipliziert). Nun wird die Log-Datei zeilenweise von den zuvor beschriebenen Schritten verarbeitet.

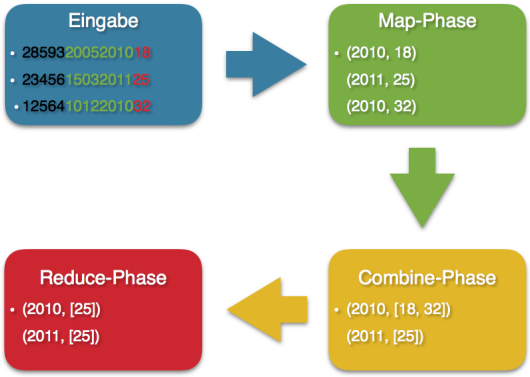

Abbildung 6.4: Map-Reduce Phasen mit Notenbeispiel

Die Eingabedatei wird zeilenweise an den ersten Prozessschritt weitergeleitet. In der *Map-Phase* wird die Datei ausgelesen und eine erste Hash-Map erstellt, die das Jahr der eingetragenen Note als Schlüssel und die Note selbst als Wert extrahiert. In der *Combine-Phase* werden die Noten eines Jahres gesammelt und zusammengefasst. In der *Reduce-Phase* wird nun die Durchschnittsnote jedes Jahres berechnet und als Wert für das jeweilige Jahr eingetragen.

6.1.5 Implementierung der Map-Reduce Jobs

Map-Reduce Jobs sind in Java geschrieben. Sie werden üblicherweise in drei Klassen aufgeteilt, die jeweils eine andere Aufgabe übernehmen. Typischerweise wird zwischen *Driver*, *Mapper* und *Reducer* unterschieden.

- **Driver:** Hier werden Eigenschaften, wie das Input-Format, Output-Format, sowie den Input/Output-Pfad definiert und die Klassennamen der Mapper- und Reducer-Klassen bekanntgegeben.
- **Mapper:** Übernimmt die Aufgabe der *Map-Phase* und bildet nach gewünschtem Vorgehen, Schlüssel-Wert-Paare. Dabei wird die Klasse für jede Zeile innerhalb der Input-Datei aufgerufen. Die gebildeten Schlüssel-Wert-Paare werden in ein Context geschrieben, der anschließend an den *Reducer* weitergereicht wird.
- **Reducer:** Nimmt den im *Mapper* erzeugten Context entgegen und übernimmt die Aufgabe der *Reduce-Phase*. Verarbeitet die Schlüssel-Wert-Paare nach dem beliebig definierten Code und schreibt das Ergebnis in einen weiteren Context, der in das gewünschte Output-Format geschrieben wird.

Die fertig implementierten Jobs werden als ausführbare JAR-Dateien exportiert und können anschließend von dem Hadoop-Framwork verwendet und ausgeführt werden.

Für den Use-Case der Webserver-Log Files Analyse sind zwei Map-Reduce Jobs vorhanden. Der erste Job extrahiert die relevanten Daten aus dem Log-File und speichert die gewünschten Daten in einer HBase-Tabelle. Anschließend greift der zweite Map-Reduce Job auf die eben beschrieben Tabelle zu und extrahiert gewünschte Daten und berechnet definierte Zusammenhänge zwischen ausgewählten Parametern der Daten, wie z.B. die Anzahl einer IP-Adresse, die Anzahl der benutzten Browser. Diese Werte werden im zweiten Map-Reduce Job abschließend in eine Textdatei geschrieben.

6.2 Hbase

6.2.1 Einrichtung

Nach der Installation von Hbase, muss die Datei *hbase-site.xml* konfiguriert werden. Die Konfigurationsdateien sind im gleichen Schema aufgebaut als die Konfigurationsdateien von Hadoop.

hbase-site.xml

```
1 <?xml version="1.0"?>
2 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3 <configuration>
4   <property>
5     <name>hbase.cluster.distributed</name>
6     <value>true</value>
7   </property>
8   <property>
9     <name>hbase.rootdir</name>
10    <value>hdfs://hadoop:9000/hbase</value>
11  </property>
12 </configuration>
```

Durch die Eigenschaft *hbase.cluster.distributed* kann der verteilte Modus von Hbase aktiviert werden. Die zweite Eigenschaft in Zeile 9 teilt Hbase mit, die Daten im HDFS und nicht lokal auf der Festplatte zu speichern.

6.2.2 Zugriff auf HBase

Hbase bietet dem Anwender zwei Möglichkeiten um Daten, die in Form von Dateien oder in Form von Tabellen einer Datenbank vorliegen, in eine HBase-Tabelle zu laden.

Zugriff über die Shell

Eine Möglichkeit auf Hbase zuzugreifen ist durch die Verwendung der Shell. Dazu müssen die einzuspielenden Daten in einer CSV-Datei vorliegen, bei denen der Separator frei gewählt werden kann. Dabei muss die Eingabedatei sowie das Ausgabeverzeichnis im HDFS liegen.

Zugriff über die Java-API

Die Hbase Java-API ermöglicht es, aus Java-Anwendungen heraus Tabellen und Daten zu betrachten und zu manipulieren. Diese API umfasst unter anderem die folgenden Funktionalitäten:

- Das Anzeigen von Daten in Tabellen
- Anlegen, (de)aktivieren, leeren und löschen von Tabellen
- Suche nach Datensätzen in einer definierten Spalte
- Daten in Form von CSV-Texten in Tabellen zu importieren

6.3 Docker

6.3.1 Aufbau

Docker-Container sind abgeschlossene und einfach zu konfigurierbare Einheiten, in denen Anwendungen ausgeführt werden können. Ein sogenanntes *Dockerfile* beschreibt

einen Container, um diesen auf jedem beliebigen Rechner ausführen zu können. Das Dockerfile ist ein einfach aufgebaute Textdatei, die eine Bauanleitung für ein Docker-Image enthält. In unserem Projekt werden für Hadoop und Hbase jeweils ein Dockerfile verwendet.

Das Verwalten und Verlinken von mehreren Docker-Containern ist aufwendig, unübersichtlich und schlecht dokumentierbar. Um dieses Problem zu lösen wurde Docker Compose eingeführt. Mit Docker Compose lassen sich innerhalb einer Datei mehrere Container definieren und ihre Beziehungen untereinander einstellen. Somit können mehrere Container mit lediglich einem Befehl gestartet werden. Das Herzstück von Docker Compose ist eine YAML Datei, die ebenfalls eine einfach aufgebaute Textdatei darstellt.

Die untenstehende YAML Datei ist die in unserem Projekt verwendete Docker Compose Konfigurationsdatei.

docker-compose.yml

```
1  version: '2'
2  services:
3    hbase:
4      build: ./hbase
5      container_name: hbase
6      hostname: hbase
7      ports:
8       - "16010:16010"
9      depends_on:
10      - hadoop
11   hadoop:
12     build: ./hadoop
13     container_name: hadoop
14     hostname: hadoop
15     ports:
16      - "50070:50070"
17      - "8088:8088"
18     volumes:
19      - ./hadoop/mr-jobs-volume:/usr/local/hadoop/mr-jobs
20      - ./hadoop/mr-input-volume:/usr/local/hadoop/mr-input
```

Die Eigenschaft *version* in Zeile 1 definiert die verwendete Version der Konfigurationssprache. Innerhalb von *services* werden alle verwendeten Container aufgelistet.

In den Zeilen 4 und 12 werden Pfade, relativ zu unserer YAML Datei, zu den jeweiligen Dockerfiles angegeben. Diese sollen beim Starten der Container ausgeführt werden.

Durch die Verwendung von Docker Compose sind die gemeinsam gestarteten Container über den angegebenen *hostname* in Zeile 6 und 14 erreichbar.

Ein Docker-Container ist eine in sich abgeschlossene Einheit, dadurch können standardmäßig keine Verbindungen weder nach innen noch nach außen erfolgen. Mit Hilfe der Eigenschaft *ports* ist es möglich angegebene Ports im Container zu öffnen. Mit Doppelpunkten getrennte Zahlen mappen Ports vom Host in den Container.

Die Eigenschaft in Zeile 18 ermöglicht es, ein Verzeichnis vom Host im Container verfügbar zu machen. Dafür muss der Pfad auf dem Host sowie im Container angegeben werden.

Kapitel 7

Fazit