Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 1

Anton Romanov

Data Mining für Big Data

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Anton Romanov

Data Mining für Big Data

Ausarbeitung Anwendungen 1 eingereicht im Rahmen der Anwendungen 1

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Wolfgang Gerken

Eingereicht am: 12.02.2014

Inhaltsverzeichnis

1 Einleitung

„Each day Google has more than 1 billion queries per day, Twitter has more than 250 milion tweets per day, Facebook has more than 800 million updates per day, and YouTube has more than 4 billion views per day“ [1]

1.1 Themenbereich

Diese Arbeit beschäftigt sich mit Fragen des Informationsgewinns aus Big Data. Dabei wird zunächst geklärt was man unter den Begriffen „Data Mining“ und „Big Data“ in diesem Zusammenhang versteht. Die Gewinnung der Information wird am Beispiel von Cluster- und Assoziationsanalyse deutlich gemacht. Es werden Algorithmen besprochen, die diese Arten von Analyse ermöglichen. Des Weiteren werden die Fragen der Datenspeicherung und Datenverarbeitung geklärt. Dabei werden das MapReduce-Programmiermodell sowie dessen Arbeitsphasen vorgestellt. Es werden die zwei am meisten verbreiteten Implementierungen von MapReduce beschrieben – Apache Hadoop und Disco. Die mit Hilfe von diesen Technologien gewonnen Informationen können Entscheidungsträger in Unternehmen bei deren Entscheidungen unterstützen. Aus diesem Grund ist es wichtig sich mit Fragen des Informationsgewinns aus Big Data zu beschäftigen.

1.2 Gegenwärtige Situation

Die Menschheit hat seit Anfang ihrer Existenz versucht die Informationen über sich selbst zu hinterlassen. Als Beispiel dafür können die Petroglyphen betrachtet werden, die vor Tausenden von Jahren entstanden sind. Dieser Trend hat sich weiterentwickelt, so dass z.B. Google 25 Petabyte an neuen Daten alle 24 Stunden erzeugt [2]. Die aus diesen Daten gewonnenen Informationen werden von großen Konzernen wie Google selbst oder Amazon verwendet, um daraus Informationen zu gewinnen, die ihnen strategische Vorteile gegenüber der Konkurrenz verschaffen können [3, S. 114]. Einige von diesen Informationen sind nur eine bestimmte Zeit relevant und müssen deswegen innerhalb eines vorgegebenen Zeitraums gewonnen werden. Als Beispiel dafür können Aktienkurse oder Umsatzzahlen eines Unternehmens genannt werden. Aus diesem Grund versucht man die Daten mit Hilfe von modernen Technologien wie z.B. MapReduce parallel und verteilt zu verarbeiten. Die Wichtigkeit dieses Themenbereiches hat auch die wissenschaftliche Gesellschaft erkannt. Eine kleine Auswahl an Journals und Konferenzen ist aus der Tabelle 1.1 zu entnehmen.

Name	*Art*
ACM Transactions on Knowledge Discovery from Data (TKDD)	Journal
ACM Transactions on Storage (TOS)	Journal
ACM Transactions on Management Information Systems (TMIS)	Journal
Knowledge Discovery and Data Mining (KDD)	Konferenz
Web Search and Data Mining (WSDM)	Konferenz
IEEE BigData 2013 Main Conference	Konferenz

Tabelle 1.1: Journals und Konferenzen

1.3 Aufbau der Arbeit

Diese Arbeit besteht aus vier Kapiteln. Im ersten Kapitel wird das Themenbereich eingegrenzt und die gegenwärtige Lage dargestellt. Im zweiten Kapitel wird der Begriff „Data Mining“ geklärt. Des Weiteren werden Assoziations- und Clusteranalyse beschrieben und durch Beispiele erläutert. Im dritten Kapitel wird der Begriff „Big Data“ erläutert. Es wird das MapReduce-Programmiermodell mit seinen Arbeitsphasen beschrieben. Danach wird Apache Hadoop sowie Disco und deren Funktionsweise vorgestellt. Zum Schluss werden die Ziele für die Anwendung gesetzt, die im Laufe des Masterstudiums entwickelt werden soll.

2 Data Mining

In diesem Kapitel werden zwei Analyseverfahren vorgestellt, mit denen sich Data Mining betreiben lässt. Beim Data Mining geht es um die Extraktion der Informationen aus den vorhandenen Daten. Die extrahierten Informationen werden in diesem Zusammenhang nicht als ein Nebenprodukt, sondern als eine strategische Ressource verstanden, die einem Unternehmen Wettbewerbsvorteile gegenüber der Konkurrenz verschaffen kann. Das Ziel von diesem Prozess ist die vorher unbekannten Zusammenhänge, Muster und Trends zu finden [4, S. 103]. Dabei wird man häufig von der Software unterstützt, die diesen Vorgang automatisiert. In diesem Bereich beschäftigt sich z.B. Matteo Riondato et al. mit einem verteilten Algorithmus für die Assoziationsanalyse [5].

Data Mining unterschiedet sich von klassischen Vorgehensweisen dadurch, dass man die Hypothese rückwärts aufbaut[6, S. 8]. In den klassischen Vorgehensweisen liegen die Daten erst am Ende eines Prozesses vor. Man weiß wie sie zu Stande gekommen sind und wie man sie auswerten soll. Bei Data Mining sind die Daten schon am Anfang des Prozesses in großen Mengen vorhanden. Dabei weiß man häufig nicht wie genau sie entstanden sind und wonach genau man sucht.

2.1 Assoziationsanalyse

Die Assoziationsanalyse stellt momentan einen der aktuellen Forschungsschwerpunkte im Bereich Data Mining dar. Die Ergebnisse einer solchen Analyse können sehr praktisch angewendet werden. So kann z.B. die Warenanordnung in einer Filiale an die Kundenwünsche angepasst werden. Eine weitere Möglichkeit für die Verwendung der Ergebnisse ist andere Artikel an Käufer zu empfehlen, die im Zusammenhang mit dem gekauften Artikel normalerweise gekauft werden. Dies wird häufig im Online-Handel gemacht. Im weiteren wird der Apriori-Algorithmus beschrieben, der eine solche Analyse ermöglicht.

2.1.1 Apriori-Algorithmus

Apriori-Algorithmus bildet die Regeln in „Wenn-Dann"-Form. Ein Beispiel dafür ist die Aussage „Wenn jemand Wein kauft, dann kauft er auch Käse". Die Datenmenge besteht dabei aus einer Menge von möglichen Items und einer Menge von Transaktionen, in denen diese Items vorkommen. Für die Bewertung der Regeln werden zwei Kenngrößen verwendet - Support und Konfidenz. Als weitere Eingabegrößen erwartet der Apriori-Algorithmus die minimale Support und minimale Konfidenz. Dadurch werden die Regeln eliminiert, die für die Auswertung nicht von Bedeutung sind. Die Support besagt wie häufig bestimmte Items in einer Transaktion vorkommen. Support muss für alle Produktpaare und für jeden einzelnen Item berechnet werden. Diese Berechnungen sind unabhängig voneinander und können deswegen parallel ausgeführt werden. Dafür kann z.B. ein MapReduce-Framework verwendet werden [5]. Für die Berechnung der Support wird folgende Formel verwendet [7, S. 103]:

$$Support(A \rightarrow B) = \frac{\text{Anzahl der Transaktionen mit A und B}}{\text{Gesamtanzahl der Transaktionen}}$$

Die Berechnung der Support soll am folgenden Beispiel erläutert werden. Die Datenmenge und die berechneten Support-Werte sind aus den Tabelle 2.1 und 2.2 zu entnehmen.

ID	*Enthaltene Items*
T1	Käse, Bier, Pizza
T2	Bier, Käse, Popcorn
T3	Bier, Olivenöl
T4	Wasser, Käse, Bier
T5	Käse, Bier, Pizza, Popcorn

Tabelle 2.1: Transaktionen und Items

Support	*Käse*	*Bier*	*Pizza*	*Popcorn*	*Olivenöl*
Käse		4/5	2/5	2/5	0/5
Bier	4/5		2/5	2/5	1/5
Pizza	2/5	2/5		1/5	0/5
Popcorn	2/5	2/5	1/5		0/5
Olivenöl	0/5	1/5	0/5	0/5	

Tabelle 2.2: Support-Werte

Eine weitere Kenngröße ist die Konfidenz. Sie zeigt die Stärke der Wechselbeziehung zwischen den einzelnen Items aus der Datenmenge. Für die Berechnung der Konfidenz wird folgende Formel verwendet [7, S. 104]:

$$Konfidenz(A \rightarrow B) = \frac{Support(A \rightarrow B)}{Support(A)}$$

Für die Berechnung der Konfidenz werden die berechneten Support-Werte aus dem obigen Beispiel verwendet. Als minimale Support wird 50% vorausgesetzt. Das Ergebnis der Berechnung ist aus der Tabelle 2.3 zu entnehmen. Nach der Berechnung der Konfidenz können nun

Regel	*Konfidenz*
Käse $\rightarrow$ Bier	$\frac{4/5}{4/5} = 1$
Bier $\rightarrow$ Käse	$\frac{4/5}{5/5} = 4/5$

Tabelle 2.3: Konfidenz

zwei Aussagen gemacht werden. Immer wenn jemand Käse kauft, kauft er auch Bier. Immer wenn jemand Bier kauft, kauft er in vier von fünf Fällen auch Käse. Durch die gewonnene Information kann z.B. die Warenanordnung in einem Supermarkt angepasst werden, so dass Käse und Bier nebeneinander liegen.

2.1.2 FP-growth

Einer der Kritikpunkte vom Apriori-Algorithmus ist die Laufzeit, die mit der steigender Anzahl der Items explodiert. Der FP-growth-Algorithmus dient genauso wie der Apriori-Algorithmus zur Bestimmung der Zusammenhängen zwischen den einzelnen Transaktionen. Der Unterschied zum Apriori-Algorithmus ist, dass FP-growth auf Prefix-Bäumen arbeitet, wodurch die Laufzeit reduziert wird. Als Eingabe bekommt er die Menge der Transaktionen und den minimalen Support-Wert. Aus allen Transaktionen werden die Items ausgeblendet, die kleineren Support-Wert besitzen. Danach werden die Items nach ihrem Support-Wert sortiert. In der Tabelle 2.4 ist eine solche Ausblendung und Sortierung dargestellt. Als minimale Support wurde zwei ausgewählt.

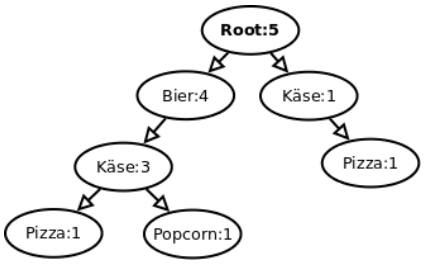

Abbildung 2.1: FP-growth Präfixbaum

ID	*Enthaltene Items*	*Gefiltert und sortiert*
T1	Käse, Bier, Pizza	Bier, Käse, Pizza
T2	Bier, Käse, Popcorn	Bier, Käse, Popcorn
T3	Bier, Olivenöl	Bier
T4	Wasser, Käse, Bier	Bier, Käse
T5	Käse, Pizza, Popcorn	Käse, Pizza

Tabelle 2.4: FP-growth Items

Anschließend werden die Transaktionen in den Baum hinzugefügt (s. Abb. 2.1). Für die Bestimmung der Support-Werte für Tupelpaare wird die Zahl im jeweiligen Knoten durch die Zahl in der Wurzel geteilt. So ist z.B. die Support für (Bier $\rightarrow$ Käse):

$$Support(\text{Bier} \rightarrow \text{Käse}) = \frac{3}{5}$$

2.2 Clusteranalyse

Clusteranalyse wird verwendet, um die Objekte bestimmten Gruppen zuzuordnen, die man als Cluster bezeichnet. Die Voraussetzung dafür, ist dass die Objekte bestimmte Merkmale aufweisen. Nach diesen Merkmalen werden sie einem oder anderen Cluster zugeordnet. Die Anzahl der Cluster, die am Ende rauskommen müssen, kann vorgegeben werden. In der Realität ist sie jedoch häufig am Anfang unbekannt. Clusteranalyse kann verwendet werden, um z.B. zu bestimmen wer die Zielgruppe für ein bestimmtes Produkt ist. Eine andere Anwendungsform ist z.B. die Web-Suche, bei der der Benutzer zunächst die Oberkategorien (Clusternamen) anklickt und später bei dem gewünschtem Artikel landet [8].

Einer der am meisten verbreiteten Algorithmen in diesem Bereich ist der k-means-Algorithmus. Er gehört zu den partitionierenden Clusterverfahren. Bei solchen Verfahren wird die Anzahl der Cluster, die am Ende rauskommen müssen, am Anfang festgelegt[7, S. 97]. Der k-means-Algorithmus wurde im Jahre 1967 von MacQueen vorgeschlagen[9]. Folgende Schritte werden vom Algorithmus ausgeführt:

1. Anfangsposition vorgeben
2. Clusterzentren berechnen
3. Objekte auf Basis der Merkmale verschieben
4. Abrechnen, wenn X mal hintereinander kein Objekt verschoben wurde. Sonst gehe zu 2.

Bei allen von diesen Punkten steigt der Aufwand mit der Menge der Daten. In den letzten Jahren wurden viele Verbesserungen vorgeschlagen, um dieses Problem zu lösen. S. Poomagal und Prof. Dr. T. Hamsapriya schlagen eine Möglichkeit zur initialen Bestimmung der Klassenzentren vor. Die Klassenzentren werden dabei nicht zufällig bestimmt, sondern aus Matrixeinträgen, die problemspezifischen Kennzahlen enthalten [8, S. 2]. Fahim A. et al. schlagen eine optimierte Methode für die Zuweisung der Objekte an die Cluster vor. Dabei wird die Distanz zwischen dem Objekt und allen anderen Klassenzentren nicht bei jeder Iteration berechnet, sondern nur wenn das Objekt größere Distanz zur neuen Klasse als zu seiner Klasse hat [10].

3 Big Data

Der Wunsch nach einer skalierbaren und verteilten Datenverwaltung ist in den letzten Jahren größer geworden [11, S. 1]. Dieser Wunsch wird durch die „3V“ der Big Data (Volume, Variety, Velocity) begründet. Es müssen immer mehr Daten innerhalb eines bestimmten Zeitraumes analysiert werden. Diese Daten liegen in verschiedensten Formaten wie z.B. Bilder, Videos,

Text oder Zahlen vor. Unterschiedliche Formate, die Menge dieser Daten und die Geschwindigkeit mit der sie erzeugt werden, führt dazu, dass diese Daten nicht mit konventioneller Software analysiert werden können. Für deren Analyse werden mächtige Supercomputer benötigt. Als Reaktion auf diese Herausforderung hat Google im Jahre 2004 die Verwendung von MapReduce vorgeschlagen [12, S. 1]. MapReduce bietet ein sehr einfaches Programmiermodell und ermöglicht eine parallele und verteilte Datenverarbeitung. Dabei setzt es keine leistungsfähige und meistens teure Hardware voraus, sondern vergleichsweise billige und stark verbreitete Hardware [13, S. 1]. MapReduce besteht aus mindestens zwei Phasen, die im Weiteren beschrieben werden. Das Konzept von Jeffrey Dean und Sanjay Ghemawat wird von der wissenschaftlichen Gesellschaft weiterentwickelt. Michael Cardosa et al. schlagen eine Architektur vor, die auf MapReduce aufbaut und von geographisch verteilten Daten ausgeht. Shrinivas B. Joshi beschäftigt sich mit der Optimierung von Apache Hadoop und Guohui Wang arbeitet an einer Netzwerkarchitektur für Big Data-Anwendungen, die auf Apache Hadoop aufbaut.

3.1 MapReduce

MapReduce verwendet das „teile und herrsche“-Prinzip. Es besteht im Wesentlichen aus zwei Arbeitsphasen – Map und Reduce (s. Abb. 3.1). Die Knoten, auf denen diese Prozesse stattfinden, werden als Mapper bzw. Reducer bezeichnet. Die Map-Phase bekommt als Input eine Liste von Key/Value-Paaren. Auf jeden Wert aus der Liste wird eine Map-Funktion angewendet. Als Ausgabe kommt eine andere Liste mit Key/Value-Paaren, die als Zwischenergebnis bezeichnet wird [12, S. 2]. Die Map-Phase kann kaskadiert betrieben werden. In diesem Fall bekommt jeder Mapper nicht eine Liste, sondern eine Menge von Listen, die er zwischen seinen Sub-Mapper und Sub-Reducer verteilt. Die Sub-Mapper und Sub-Reducer sind nur für den Mapper sichtbar. Das hat zur Folge, dass die gesamte Hierarchie nach außen als einziger Mapper agiert. Diese Vorgehensweise kann ihre Vorteile zeigen, wenn die Eingabedaten oder die Rechenkapazitäten geographisch verteilt sind [14, S. 2].

Die Reduce-Phase bekommt das Ergebnis der Map-Phase als Eingabe. Die Aufgabe der Reduce-Phase besteht darin, die Anzahl der Werte zu reduzieren. Dies kann z.B. durch Aggregation der Werte erreicht werden [12, S. 2]. Die Reduce-Phase kann genauso wie die Map-Phase kaskadiert betrieben werden. Da die Anzahl der Reducer deutlich weniger als die Anzahl der Mapper ist, wird die Kaskadierung meistens in der Map-Phase angewendet.

Der Vorteil dieser Vorgehensweise entsteht durch die parallele bzw. verteilte Ausführung. Die Daten werden in Form von Listen zwischen den Mappern verteilt. Die Mapper laufen auf verschiedenen Maschinen und können unabhängig voneinander lokal mit ihren Daten arbeiten, ohne dass sie durch Aufrufe der anderen Mapper gestört werden. Es ist auch möglich mehrere Mapper auf einer CPU mit mehreren Kernen zu betreiben. In diesem Fall muss sichergestellt werden, dass jeder Mapper auf seinem eigenen Kern läuft, um die parallele Ausführung nicht zu behindern [15, S. 163].

Als Beispiel kann man sich eine Situation vorstellen, in der Kassenbondaten für eine bestimmte Filiale vorliegen. Während des Extract-Transform-Load-Prozesses(ETL) möchte man

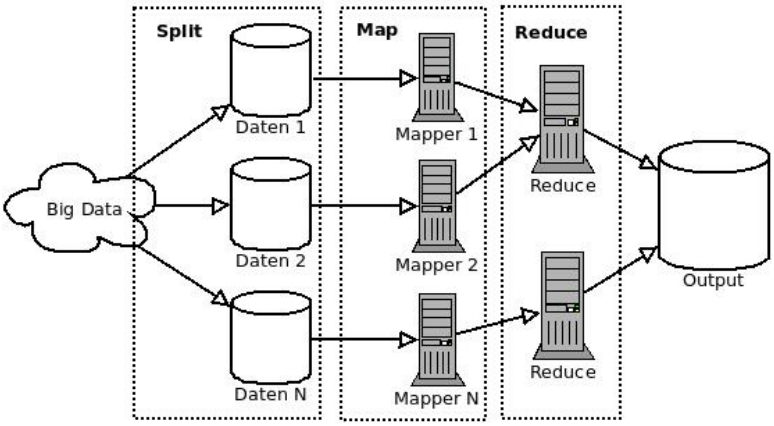

Abbildung 3.1: Arbeitsphasen von MapReduce

die Daten extrahieren, den Gesamtumsatz pro Produkt ermitteln und diesen Wert im Data Warehouse speichern. Die Map-Funktion bekommt als Eingabe die Daten von einem Kassenbon mit Produkten und Preisen. Der Kassenbon repräsentiert somit die Liste, Produkte sind Keys und Preise sind Values. Die Map-Funktion ermittelt den Umsatz für jedes Produkt aus dem Kassenbon. Im Zwischenergebnis steht das Produkt als Key und der Umsatz als Value. Die Reduce-Funktion bekommt die Zwischenergebnisse von vielen Mappern und kann den Gesamtumsatz pro Produkt für die Filiale berechnen, indem sie die Ergebnisse der einzelnen Mapper aufsummiert. Das Ergebnis wird dann im Data Warehouse gespeichert.

Die Berechnung der Umsätze pro Kassenbon kann durch die Verwendung von vielen Mappern parallelisiert und verteilt werden. Somit werden die Umsätze pro Kassenbon parallel und nicht sequentiell berechnet.

3.2 Apache Hadoop

Apache Hadoop ist ein Java-basiertes verteiltes Framework für Anwendungen, die MapReduce-Programmiermodel nutzen wollen. Es besteht aus einer effizienten Implementierung von MapReduce und einem verteilten Dateisystem – Hadoop Destributed File System (HDFS). Dieses Framework hat sich in den letzten Jahren sehr stark verbreitet [16]. Es wurden viele Anwendungsarten dafür gefunden, die sowohl auf einem einzelnen Knoten als auch auf einem Cluster mit tausenden von Knoten arbeiten können [17, S. 1]. Zu solchen Anwendungen gehört auch ein Data Warehouse System, welches von einer verteilten bzw. parallelen Ausführung und Speicherung stark profitieren kann.

In einem Hadoop-System unterscheidet man zwischen zwei Arten von Knoten – Master und Slave (s. Abb. 3.2). Deren Aufgaben können auf das Programmiermodell von MapReduce zurückgeführt werden. Ein Slave-Knoten besteht aus einem Task-Traker und einem Data Node. Die Aufgabe des Task-Trakers besteht darin, die vom Master-Knoten zugewiesenen Tasks zu verwalten. Task-Traker kümmert sich nicht um die persistente Speicherung der Ergebnisse

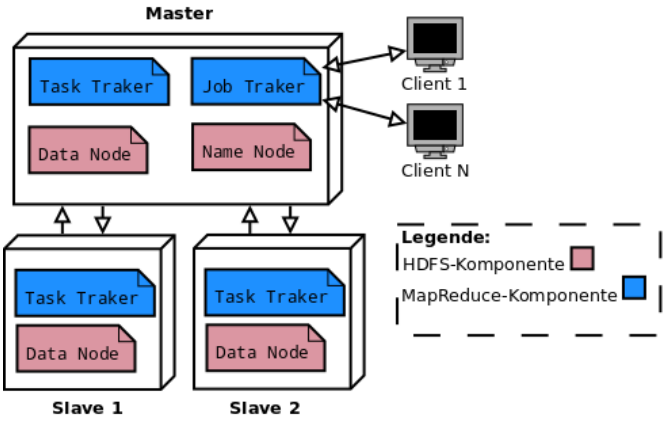

Abbildung 3.2: Knotenarten von Apache Hadoop

seiner Arbeit, sondern verrichtet sie nur. Es entspricht dem Mapper aus dem MapReduce-Programmiermodell. Für die Speicherung der Ergebnisse ist der Data-Node zuständig. Er sorgt dafür, dass die vom Task-Traker berechneten Ergebnisse in HDFS persistent gespeichert werden [18, S. 4]. Ein Master-Knoten besitzt neben dem Task-Traker und Data-Node noch zwei weitere Komponenten. Das sind Job-Traker und Name-Node. Die Aufgabe des Job-Trakers besteht darin, die einzelnen Jobs in Tasks zu zerlegen und sie zwischen den Slave-Knoten zu verteilen [19, S. 3]. Diese Aufgabe entspricht der Split-Phase im MapReduce-Programmiermodell. Wenn ein Slave-Knoten sein Task ausgeführt hat, liefert er das Ergebnis an Job-Traker. Job-Traker sammelt die Teilergebnisse der einzelnen Slave-Knoten und erzeugt ein Gesamtergebnis. Somit hat der Job-Tracker auch die Aufgabe des Reducers aus dem MapReduce-Programmiermodell. Die Name-Node-Komponente ist die Verwaltungskomponente des HDFS. Im Name-Node wird ein Index auf alle Dateien im HDFS gespeichert [18, S. 5]. Er weiß somit, welche Dateien auf welchem Knoten im System vorhanden sind.

Die Anwendung wendet sich immer an die Job-Tracker-Komponente im Master-Knoten und bekommt eine Antwort auch nur vom Master-Knoten. Die einzelnen Slave-Knoten sind für die Anwendung nicht sichtbar. Im oben erwähnten Beispiel entspricht der ETL-Prozess der Anwendung. Der ETL-Prozess liefert dem Job-Traker im Master-Knoten alle Kassenbondaten und bekommt als Ergebnis die Gesamtumsätze pro Produkt. Somit muss die Anwendung nichts von der parallelen und verteilten Ausführung wissen.

3.3 Disco

Disco ist ein Akronym und steht für Distributed Computing. Dabei handelt es sich um ein weiteres Framework, welches das MapReduce-Programmiermodell umsetzt [20]. Der Kern von Disco, welcher für die Verwaltung der Jobs und Tasks verantwortlich ist, ist in Erlang implementiert. Die Bibliotheken, die von Anwender benutzt werden sind im Gegensatz dazu in Python implementiert. Genauso wie Hadoop besteht Disco aus Master- und Slave-Knoten. Die

Architektur von Disco ist in Abbildung 3.3 dargestellt. Die Schnittstelle des Disco-Systems zu der Außenwelt ist der Masterknoten. Seine primäre Aufgabe ist die Kommunikation mit externen Anwendungen. Des Weiteren ist er für die Verwaltung und Allokation der Ressourcen sowie für die Zuweisung von Tasks und Jobs zuständig. Die auf den Slave-Knoten laufenden Python-Worker sind für die Ausführung der MapReduce-Aufgaben zuständig, die sie in Form von Tasks zugewiesen bekommen. Auf jedem Slave-Knoten wird ein Worker Supervisor vom Master-Knoten gestartet. Der Supervisor hat die Aufgabe die einfachen Worker zu überwachen. Die im

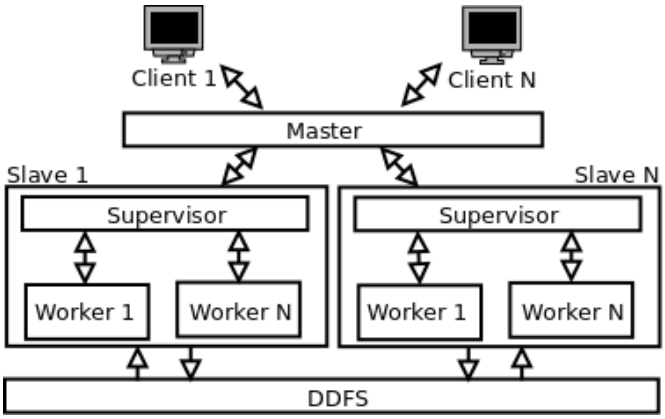

Abbildung 3.3: Architektur von Disko (vereinfacht)

System vorhandenen Daten werden genau so wie in Hadoop in einem verteilten Dateisystem gespeichert. Hier wird es Disco Distributed File System (DDFS) genannt. Während HDFS POSIX-konform ist, und somit für viele relativ kleine Dateien mit Veränderungen geeignet ist, werden bei DDFS große Dateien mit möglichst wenig Veränderungen vorausgesetzt. Der zweite wesentliche Unterschied zu Hadoop ist, dass Disco einen direkten Zugriff auf die Daten für externe Anwendungen ermöglicht. Solche Anwendungen können genau so wie bei Hadoop den Jobs mitgegeben werden. Bei Hadoop müssen die Daten in diesem Fall zwei Mal eingelesen werden [21, S. 6].

4 Anwendung

Die zu entwickelnde Anwendung soll die wertvollen Informationen aus einer großen Datenmenge extrahieren. Um dieses Ziel zu erreichen, wird ein Data-Warehouse aufgebaut, dessen Speicherplatz sich in Apache Hadoop bzw. Disco befindet. Es werden mehrere Knoten gestartet um die parallele Berechnung zu ermöglichen. Die im Data-Warehouse gespeicherten Daten sollen die Einkaufstransaktionen darstellen. Zweck der Anwendung ist die Durchführung der

Warenkorbanalyse. So kann z.B. mit Hilfe von Assoziationsanalyse berechnet werden welche Waren zusammen gekauft werden und welche nicht. Später können noch weitere Algorithmen dazukommen, damit man deren Effizienz miteinander vergleichen kann. In Abhängigkeit davon kann z.B. die Platzierung dieser Waren in einer Filiale gewählt werden. Eine mögliche Architektur der Anwendung ist in Abbildung 4.1 dargestellt.

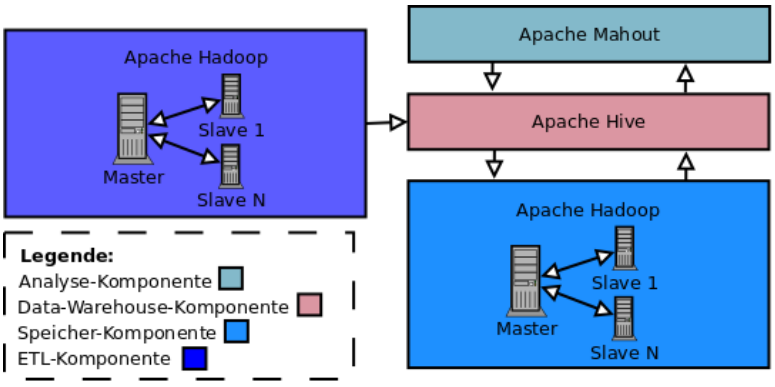

Abbildung 4.1: Architektur der Anwendung

Für die Durchführung der Assoziationsanalyse kann z.B. Apache Mahoot verwendet werden [22, S. 3]. In diesem Fall muss die Zusammenarbeit zwischen der Data-Warehouse-Komponente und Mahout hergestellt werden. Die Vorteile von MapReduce sollen schon während des ETL-Prozesses gezeigt werden. Somit kann z.B. die Transformationsphase ebenfalls parallel und verteilt durchgeführt werden. Die Daten für das Data-Warehouse können aus drei Quellen importiert werden. Als erstes können die Daten aus der Bachelorarbeit verwendet werden. Des Weiteren können neue Daten generiert werden. Die dritte Möglichkeit ist die Beschaffung der Daten von einem externen Unternehmen.

5 Zusamenfassung und Ausblick

In dieser Arbeit wurde gezeigt wie man die Informationen aus Big Data gewinnen kann. Es wurden zwei große Analyseverfahren aus dem Bereich der Data Mining vorgestellt - Assoziationsanalyse und Clusteranalyse. Zu jedem von diesen Verfahren wurden Algorithmen genannt und erklärt. Des Weiteren wurde das MapReduce-Programmiermodell ausführlich beschreiben. Es wurden zwei Implementierungen von diesem Programmiermodel vorgestellt - Apache Hadoop und Disco. Zum Schluss wurde die Zielsetzung für die Anwendung beschrieben, die auf Basis der in dieser Arbeit beschriebenen theoretischen Grundlagen, im Laufe der PJ-Veranstaltungen in kommenden Semestern implementiert werden soll.

Literatur

[1] Wei Fan und Albert Bifet. "Mining Big Data: Current Status, and Forecast to the Future". In: *SIGKDD Explor. Newsl.* 14.2 (04/2013), S. 1–5. ISSN: 1931-0145. URL: http://doi.acm.org/10.1145/2481244.2481246.

[2] Ranga Raju Vatsavai, Auroop Ganguly, Varun Chandola, Anthony Stefanidis, Scott Klasky und Shashi Shekhar. "Spatiotemporal Data Mining in the Era of Big Spatial Data: Algorithms and Applications". In: *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*. BigSpatial '12. Redondo Beach, California: ACM, 2012, S. 1–10. ISBN: 978-1-4503-1692-7. URL: http://doi.acm.org/10.1145/2447481.2447482.

[3] Kenneth C. Laudon, Jane Price Laudon und Detlef Schoder. *Wirtschaftsinformatik: Eine Einführung*. 2. Aufl. München: Pearson Studium, 2010. ISBN: 3827373484.

[4] K. Farkisch. *Data-Warehouse-Systeme kompakt: Aufbau, Architektur, Grundfunktionen*. Springer Berlin Heidelberg, 2011. ISBN: 9783642215322. URL: http://books.google.at/books?id=kCClF-bQfGgC.

[5] Matteo Riondato, Justin A. DeBrabant, Rodrigo Fonseca und Eli Upfal. "PARMA: A Parallel Randomized Algorithm for Approximate Association Rules Mining in MapReduce". In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM '12. Maui, Hawaii, USA: ACM, 2012, S. 85–94. ISBN: 978-1-4503-1156-4. URL: http://doi.acm.org/10.1145/2396761.2396776.

[6] Bernd Knobloch. "Der Data-Mining-Ansatz zur Analyse betriebswirtschaftlicher Daten". In: *Bamberger Beiträge zur Wirtschaftsinformatik №58*. Bamberg, Germany: Otto-Friedrich-Universität Bamberg, 2000. URL: http://www.ceushb.de/forschung/downloads/%5BKnob00%5D.pdf.

[7] Helge Petersohn. *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur*. München: Oldenbourg, 2005. ISBN: 978-3-486-57715-0.

[8] S. Poomagal und T. Hamsapriya. "Optimized K-means Clustering with Intelligent Initial Centroid Selection for Web Search Using URL and Tag Contents". In: *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. WIMS '11. Sogndal, Norway: ACM, 2011, 65:1–65:8. ISBN: 978-1-4503-0148-0. URL: http://doi.acm.org/10.1145/1988688.1988764.

[9] J. B. MacQueen. "Some Methods for Classification and Analysis of MultiVariate Observations". In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Hrsg. von L. M. Le Cam und J. Neyman. Bd. 1. University of California Press, 1967, S. 281–297.

[10] Torkey F.A. Fahim A.M. Salem A.M. "An efficient enhanced k-means clustering algorithm". In: *Journal of Zhejiang University Science* 10.7 (2006). Published by Foundation of Computer Science, New York, USA, S. 1626–1633.

[11] Divyakant Agrawal, Sudipto Das und Amr El Abbadi. "Big Data and Cloud Computing: Current State and Future Opportunities". In: *Proceedings of the 14th International Conference on Extending Database Technology*. EDBT/ICDT '11. Uppsala, Sweden: ACM, 2011, S. 530–533. ISBN: 978-1-4503-0528-0. URL: http://doi.acm.org/10.1145/1951365.1951432.

[12] Jeffrey Dean und Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *OSDI'04: PROCEEDINGS OF THE 6TH CONFERENCE ON SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION*. USENIX Association, 2004.

[13] Tim Kaldewey, Eugene J. Shekita und Sandeep Tata. "Clydesdale: Structured Data Processing on MapReduce". In: *Proceedings of the 15th International Conference on Extending Database Technology*. EDBT '12. Berlin, Germany: ACM, 2012, S. 15–25. ISBN: 978-1-4503-0790-1. URL: http://doi.acm.org/10.1145/2247596.2247600.

[14] Michael Cardosa, Chenyu Wang, Anshuman Nangia, Abhishek Chandra und Jon Weissman. "Exploring MapReduce Efficiency with Highly-distributed Data". In: *Proceedings of the Second International Workshop on MapReduce and Its Applications*. MapReduce '11. San Jose, California, USA: ACM, 2011, S. 27–34. ISBN: 978-1-4503-0700-0. URL: http://doi.acm.org/10.1145/1996092.1996100.

[15] Pavlo Baron. *Big Data für IT-Entscheider : Riesige Datenmengen und moderne Technologien gewinnbringend nutzen*. Hanser, 2013. ISBN: 3-446-43339-2.

[16] Wikipedia. *Apache Hadoop : Commercially supported Hadoop-related products – Wikipedia, The Free Encyclopedia*. Abruf 2013-11-30. 2013. URL: http://en.wikipedia.org/wiki/Apache_Hadoop#Commercially_supported_Hadoop-related_products.

[17] Shrinivas B. Joshi. "Apache Hadoop Performance-tuning Methodologies and Best Practices". In: *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. ICPE '12. Boston, Massachusetts, USA: ACM, 2012, S. 241–242. ISBN: 978-1-4503-1202-8. URL: http://doi.acm.org/10.1145/2188286.2188323.

[18] Dhruba Borthakur. *Big Data für IT-Entscheider : Riesige Datenmengen und moderne Technologien gewinnbringend nutzen.* 2007. URL: https://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf.

[19] Guohui Wang, T.S. Eugene Ng und Anees Shaikh. "Programming Your Network at Run-time for Big Data Applications". In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks.* HotSDN '12. Helsinki, Finland: ACM, 2012, S. 103–108. ISBN: 978-1-4503-1477-0. URL: http://doi.acm.org/10.1145/2342441.2342462.

[20] Nokia Corporation und the Disco Project. *What is Disco? - Disco documentation.* Abruf 2013-12-24. 2013. URL: http://disco.readthedocs.org/en/latest/intro.html.

[21] Prashanth Mundkur, Ville Tuulos und Jared Flatow. "Disco: A Computing Platform for Large-scale Data Analytics". In: *Proceedings of the 10th ACM SIGPLAN Workshop on Erlang.* Erlang '11. Tokyo, Japan: ACM, 2011, S. 84–89. ISBN: 978-1-4503-0859-5. URL: http://doi.acm.org/10.1145/2034654.2034670.

[22] Sachin Gulabrao Walunj und Kishor Sadafale. "An Online Recommendation System for e-Commerce Based on Apache Mahout Framework". In: *Proceedings of the 2013 Annual Conference on Computers and People Research.* SIGMIS-CPR '13. Cincinnati, Ohio, USA: ACM, 2013, S. 153–158. ISBN: 978-1-4503-1975-1. URL: http://doi.acm.org/10.1145/2487294.2487328.