Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung AW2 SS2012

Jan-Christoph Meier

Data Mining in der Cloud

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einleitung

Das Ziel von Data Mining (Han und Kamber (2006)) ist das Extrahieren von Wissen aus Daten. Dies erfolgt durch Anwendung verschiedener Algorithmen. Die zu verarbeitenden Datenmengen sind in den letzten Jahren stetig gewachsen, so ist zum Beispiel das durch die Benutzer von facebook generierte Datenaufkommen circa 500 Terabyte am Tag.
Um diese Mengen an Daten verarbeiten zu können, wird eine hohe Rechenleistung benötigt, die die Kapazitäten eines einzelnen Servers weit überschreitet. Daher wird hierfür auf die Rechenleistung mehrerer, zu Clustern zusammengefasster Server, die auch als “Cloud” bezeichnet werden, zurückgegriffen. Diese können kostengünstig angemietet werden, wobei Speicherplatz und Rechenleistung an den jeweiligen Bedarf angepasst werden können.
Damit die Cloud-Infrastruktur für Data Mining verwendet werden kann wurden verschiedene Ansätze entwickelt. Einer davon ist das 2004 von Google vorgestellte MapReduce-Framework (Dean und Ghemawat (2008)). MapReduce ist ein Programmierparadigma und eine Softwarearchitektur, die von Google für die Datenanalyse eingesetzt wird und sehr große Datenmengen verarbeiten kann. Mit MapReduce können Algorithmen entwickelt werden, die sehr einfach auf einer verteilten Rechnerinfrastruktur ausgeführt werden können. MapReduce ist zuständig für die Verteilung der Daten und koordiniert die Ausführung der Algorithmen.
Eine OpenSource Implementierung von MapReduce ist Hadoop. Sie stellt sowohl die Infrastruktur, als auch ein Framework zur Entwicklung von MapReduce-Algorithmen dar.
Verschiedene Data Mining Algorithmen, so zum Beispiel der Clustering-Algorithmus “K-Means” oder der aus dem Bereich des “Frequent pattern minings” stammende “frequent pattern growth”-Algorithmus, wurden bereits auf das MapReduce-Paradigma portiert. Vorteil hierbei ist, dass diese Data Mining Algorithmen auf einer MapReduce Infrastruktur und somit in einer Cloud ausgeführt werden können.
Das auf Hadoop basierende Framework Apache Mahout, enthält verschiedene Data Mining Algorithmen, die nach dem MapReduce-Paradigma implementiert sind.
Im Folgenden werden MapReduce und die Frameworks Hadoop und Mahout vorgestellt. Außerdem werden zwei wissenschaftliche Publikationen erörtert, die beschreiben, wie der K-means- und der “frequent pattern growth”-Algorithmus auf einer MapReduce-Infrastruktur ausgeführt werden können.

1.1 Motivation

Im Rahmen der Multiple-Sklerose Forschung sollen in einer Vielzahl von Proteinen Strukturen gefunden werden, die unter Umständen Aufschluss über das Auslösen einer Immunreaktion geben können. Bei Proteinen handelt es sich um eine zusammenhängende Kette von Aminosäuren. Einige dieser Aminosäure-Strukturen stehen im Verdacht, Mitauslöser einer Immunreaktion zu sein. Aminosäuren lassen sich generell nach Größe, Ladung und Hydrophobizität charakterisieren.
Mit Data Mining sollen eine Reihe von Proteinsequenzen, von denen bereits bekannt ist, dass sie eine Immunreaktion auslösen, untersucht werden. Ziel hierbei ist es, Strukturen in den Proteinsequenzen zu finden. Dies kann zum Beispiel durch eine Analyse mit Algorithmen für das Clustering erfolgen oder mit Algorithmen zum “Frequent pattern mining”, um häufig gemeinsam auftretende Aminosäuren zu finden.
Ziel ist die Entwicklung eines Systems, dass es ermöglicht, verschiedene Data Mining Algorithmen anzuwenden und die Ergebnisse der Analysen gegenüberzustellen und zu vergleichen.

2 Verwandte Arbeiten

In den folgenden Abschnitten wird zuerst die von Google veröffentlichte Publikation zu MapReduce vorgestellt, darauffolgend wird das Framework Hadoop vorgestellt, das eine OpenSource-Implementierung von MapReduce ist. Im selben Abschnitt wird auf das Hadoop basierende Framework Mahout eingegangen, das verschiedene Data Mining Algorithmen implementiert.
Die beiden letzten Abschnitte stellen zwei Publikationen vor. Die eine Publikation zeigt einen Ansatz den K-Means-Algorithmus auf MapReduce zu portieren, die zweite beschäftigt sich damit wie der "frequent pattern growth"-Algorithmus auf das MapReduce-Paradigma angewendet werden kann.

2.1 MapReduce: Simplified Data Processing on Large Clusters

MapReduce (Dean und Ghemawat (2008)) ist ein Programmiermodell und eine Softwarearchitektur mit der es möglich ist Algorithmen parallel auf verschiedenen Servern auszuführen. Die in Abbildung 2.1 dargestellte MapReduce-Infrastruktur ist zuständig für die Verteilung der Programme, Zuweisung der Eingabedaten und koordiniert die Ausführung. Die Infrastruktur wird durch ein Framework bereitgestellt, auf dessen Bibliotheken die Entwicklung der MapReduce-Programme basiert.
Das Grundkonzept hierbei ist das Master-Worker-Pattern, wobei der Master die durch den Benutzer übermittelten Programme entgegennimmt und diese an die einzelnen Worker zur Ausführung weitergibt.
Der Ablauf teilt sich in zwei Phasen auf: "Map" und "Reduce". Die Phasen werden durch den Programmierer als Methoden vorgeben und durch das Framework auf den Workern aufgerufen. Die Eingabedaten liegen in einzelnen Blöcken vor und werden als Key/Value-Paar an die Methoden übergeben. Durch die eindeutige Partitionierung ist es möglich die Map- und Reduce-Methoden parallel auszuführen, wobei die Reduce-Methoden nach Beendigung aller Map-Methoden ausgeführt werden.
In Tabelle 2.1 sind die als Eingabe und Ausgabe verwendeten Datenstrukturen der Map- und Reduce-Methode dargestellt. Die Map-Methode nimmt als Parameter einen Key und

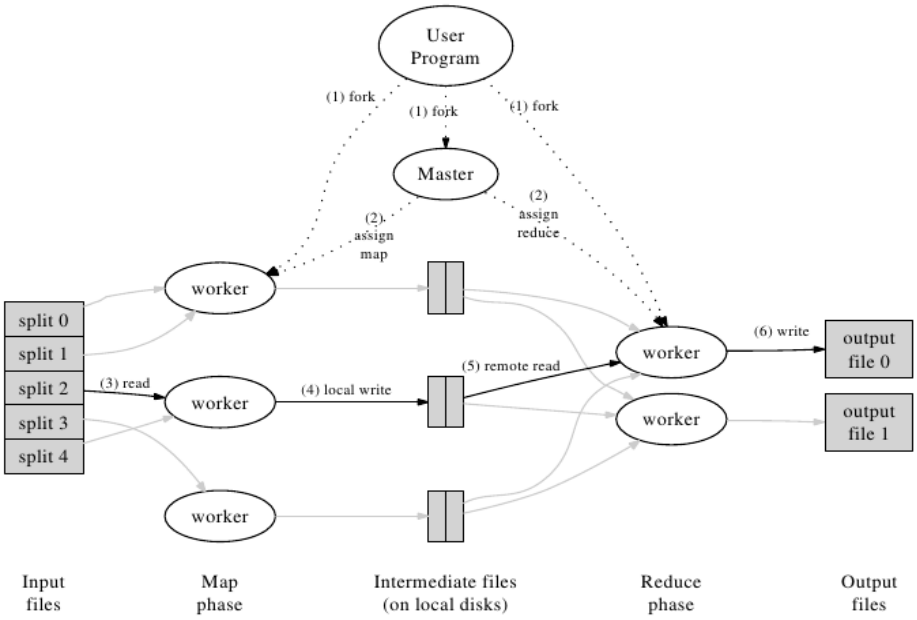

Abbildung 2.1: MapReduce Master/Worker-Architektur (aus Dean und Ghemawat (2008) Seite 3)

einen Value entgegen, der Rückgabewert ist eine Liste von Key/Value-Paaren. Diese werden anhand ihres Keys gruppiert, wobei jedem Key eine Liste mit Values zugeordnet wird. Der Reduce-Methode wird dann ein Key und eine Liste mit Values übergeben, sie berechnet das endgültige Ergebnis und gibt eine Liste mit Values zurück.
Bei der Entwicklung mit MapReduce muss der Programmierer sich nicht um die Verteilung und Parallelisierung der Algorithmen kümmern. Es können somit Entwickler ohne Erfahrung mit verteilten Systemen und paralleler Programmierung skalierbare Algorithmen entwickeln, die verteilt auf einem Cluster von Rechnern oder in einer Cloud ausgeführt werden können.

	Input	**Output**
Map	<k1, v1>	list(<k2, v2>)
Reduce	<k2, list(v2)>	list(<v3>)

Tabelle 2.1: Parameter und Rückgabewerte der Map- und Reduce-Methoden (aus Lam (2010))

2.2 Apache Hadoop und Mahout

Apache Hadoop (Lam (2010)) ist ein Open-Source MapReduce Implementierung, die unter anderem von Yahoo! und facebook entwickelt wird.
Hadoop besteht aus verschiedenen Komponenten. Kernbestandteil sind MapReduce und das verteilte Dateisystem “Hadoop distributed filesystem” (HDFS). Hadoop ist in der Programmiersprache Java programmiert, in der auch die MapReduce-Programme entwickelt werden. Die im vorherigen Abschnitt vorgestellte MapReduce-Architektur wird durch Hadoop bereitgestellt.
Das auf Hadoop basierende Open-Source Framework Apache Mahout (Owen u. a. (2011)) enthält verschiedene Data Mining Algorithmen. Hauptbestandteil von Mahout sind Algorithmen für Clustering, Pattern Mining und Machine Learning[1].
Die Algorithmen sind nach dem MapReduce-Paradigma implementiert und können in einem Hadoop Cluster ausgeführt werden. Sie sind für die Analyse sehr großer Datenmengen ausgelegt und so generisch implementiert, dass es prinzipiell möglich ist, beliebige Eingabedaten zu verarbeiten.

2.3 Parallel K-Means Clustering Based on MapReduce

Clustering Algorithmen partitionieren eine Menge von Daten in verschiedene Teilmengen. Hierbei sind die Objekte innerhalb der Teilmengen ähnlich und die Teilmengen als ganzes unterscheiden sich.
Das Schema des K-Means Clustering (MacQueen (1967)) Algorithmus wird im Folgenden erläutert:
Zu Beginn des Verfahrens werden k Objekte aus der Eingabemenge D zufällig ausgewählt, sie dienen als initiale Clustermittelpunkte. Jedes Objekt der Menge D wird im darauffolgenden Schritt einem Clustermittelpunkt zugewiesen. Die Zuweisung erfolgt mithilfe einer Distanzfunktion, die den “Abstand” bzw. die Ähnlichkeit von zwei Objekten bestimmt. Für die Abstandsberechnung gibt es verschiedene Verfahren, so zum Beispiel die euklidische Distanz oder die “Manhattan”-Distanz. Sie basieren darauf, dass die Objekte als Vektoren dargestellt werden. Bei einem zweidimensionalen Vektor kann der Vektor als Punkt im Koordinatensystem betrachtet werden, hierbei würde die Distanzfunktion entsprechend den Abstand der beiden Vektoren berechnen.
Nachdem jedes Objekt einem Clustermittelpunkt zugeordnet wurde, werden neue Clustermittelpunkte bestimmt, die den Objekten des temporären Clusters am nähesten sind. Diese Berechnung wird solange wiederholt, bis sich die Clustermittelpunkte nicht mehr verändern.

[1] Übersicht über die in Mahout implementierten Algorithmen: http://cwiki.apache.org/confluence/display/MAHOUT/Algorithms, abgerufen am 25.08.2012

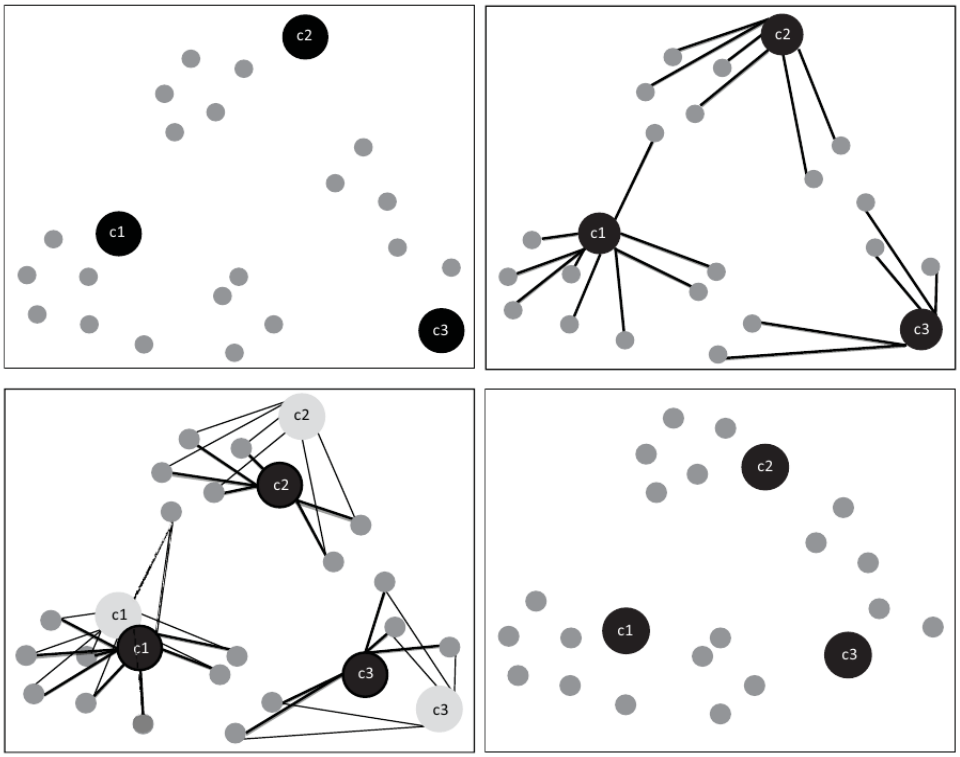

Abbildung 2.2: Schematik des K-Means Algorithmus (Bild aus Owen u. a. (2011) Seite 147)

In der Abbildung 2.2 wird die Schematik des Algorithmus verdeutlicht. Hier stellen die schwarz hinterlegten Punkte C1, C2 und C3 die initialen Clustermittelpunkte dar. Die Linien in der Abbildung repräsentieren die Abstandsberechnung. Die grau hinterlegten Punkte C1, C2 und C3 sind die neu berechneten Clustermittelpunkte.
Die Operation mit dem höchsten Rechenaufwand ist die Abstandsberechnung der Objekte der Menge $\mathbf{D}$ zu den $\mathbf{k}$ Clustermittelpunkten. Diese Berechnung erfolgt insgesamt »n * k« häufig, was bei vielen Objekten und Clustermittelpunkten zu vielen Iterationen führt.
Der in der Publikation “Parallel K-Means Clustering Based on MapReduce” (Zhao u. a. (2009)) vorgestellte Ansatz zeigt, wie der K-Means Clustering Algorithmus in einem MapReduce Framework ausgeführt werden kann. Hierfür wird die Schematik von K-Means in einen “Map”-, eine “Combine”- und in einen “Reduce”-Schritt aufgeteilt.
Die MapReduce Implementierung basiert darauf, dass die Cluster-Mittelpunkte global gespeichert werden und jede Map-Methode Zugriff auf diese Daten hat. Die Verteilung erfolgt über ein verteiltes Dateisystem. Der Map-Methode wird bei jedem Aufruf ein Objekt übergeben, für das sie den Abstand zu allen Cluster-Mittelpunkten bestimmt. Der Clustermittelpunkt mit dem geringsten Abstand wird als Ergebnis zurückgegeben.
In der “Combine”-Methode werden die Ergebnisse mehrerer Aufrufe der “Map”-Methode zu-

sammengefasst, hierbei werden mehrere Vektoren aufaddiert, um die Mittelwertsberechnung zur Bestimmung des neuen Cluster-Mittelpunkts vorzubereiten.
Die “Reduce”-Methode berechnet sobald alle “Map”-Methoden ihre Ergebnisse zurückgeliefert haben die neuen Clustermittelpunkte. Hierfür summiert sie alle Vektoren auf und bildet den Mittelwert, der den neuen Clustermittelpunkt darstellt.
Sofern sich die Clustermittelpunkte nicht mehr verändern ist die Berechnung abgeschlossen, andernfalls wird eine neue Iteration gestartet, bei der “Map”, “Combine” und ”Reduce” erneut ausgeführt werden.

2.4 Pfp: parallel fp-growth for query recommendation

Das “Frequent Pattern Mining” ist ein Teilgebiet des Data Minings, das verschiedene Algorithmen zusammenfasst, mit denen häufig gemeinsam auftretende Objekte in Daten gefunden werden können. Die Daten können zum Beispiel eine Reihe von Verkaufstransaktionen sein, wobei in diesem Fall mit dem “Frequent Pattern Mining” Waren gefunden werden können, die häufig gemeinsam gekauft werden.
Ein Algorithmus hierfür ist der “FP-Growth”-Algorithmus (Han u. a. (2000)). Dieser basiert auf der Konstruktion eines “Frequent pattern trees” (kurz FP-Tree) auf den die Transaktionen abgebildet werden und mit dem häufig gemeinsam auftretende Elemente gefunden werden können.
Bevor der FP-Tree erzeugt werden kann, muss über alle Transaktionen iteriert und bestimmt werden, wie häufig jedes Element in den Transaktionen vorhanden ist. Diese Häufigkeit wird als “Support” bezeichnet. Im darauffolgendem Schritt werden die Elemente aus den Transaktionen entfernt, deren Support-Wert einen bestimmten Schwellenwert unterschreitet. In Tabelle 2.2 sind mehrere Transaktionen aufgeführt, deren Elemente einen minimum Support-Wert von 3 haben. Die Elemente sind nach der Größe des Support-Wertes sortiert. Der

TID	Elemente mit Support >= 3
1	f, c, a, m, p
2	f, c, a, b, m
3	f, b
4	c, b, p
5	f, c, a, m, p

Tabelle 2.2: Transaktionen, deren Elemente einen Support-Wert größer/gleich drei haben

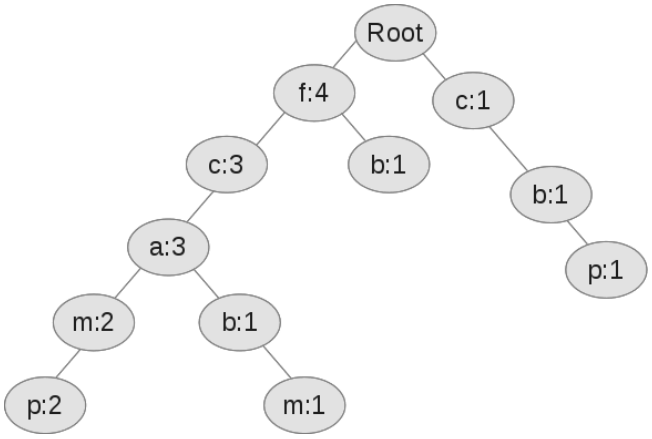

Abbildung 2.3: Baumstruktur des “Frequent pattern growth”-Algorithmus

FP-Tree wird gebildet indem aus den Transaktionen ein Baum erzeugt wird. Die Elemente der Transaktionen stellen hierbei die Blätter dar und werden anhand der Höhe des Support-Wertes in den Baum einsortiert. Abbildung 2.3 zeigt den FP-Tree der aus den Transaktionen in Tabelle 2.2 erzeugt wurde. Die Blätter enthalten den Namen des Elementes und den Support-Wert (Name:Supportwert). Anhand des Baumes wird dann für bestimmte Elemente bestimmt, wie häufig sie gemeinsam in den Transaktionen enthalten sind. Die Elemente “f” und ”p” zum Beispiel sind in zwei Transaktionen gemeinsam vertreten.
In der Publikation “Pfp: parallel fp-growth for query recommendation” (Li u. a. (2008)) wird ein Ansatz vorgestellt, der es ermöglicht den “FP-Growth”-Algorithmus in einem MapReduce-Framework auszuführen. Der Algorithmus wird hierfür in insgesamt 5 Schritten ausgeführt:

Schritt 1: Sharding

Die Transaktionen werden partitioniert und auf P verschiedene Rechner verteilt, was als “Sharding” bezeichnet wird. Die einzelnen Partitionen werden hierbei als “Shard” bezeichnet.

Schritt 2: Paralleles Zählen

Mit einem MapReduce-Programm werden die Support-Werte der einzelnen Elemente der Transaktionen bestimmt. Jedem Aufruf der Map-Funktion wird eine Partition (Shard) der Transaktionen übergeben, diese berechnet dann die Support-Werte der Elemente der Transaktionen. Das Ergebnis der Berechnung wird in der “Frequency-List” (F-List) gespeichert. Sie enthält die Elemente mit ihren Häufigkeiten.

Schritt 3: Elemente gruppieren

Die Elemente der F-List werden in Q Gruppen aufgeteilt und in einer Liste mit Gruppen, der “Group-List” (G-List) gespeichert. Jede Gruppe erhält eine eindeutige Gruppen-ID (group-id).

Da diese Listen nicht groß sind, wird dieser Schritt nicht als MapReduce-Programm, sondern auf einem einzelnem Rechner ausgeführt.

Schritt 4: Parallel FP-Growth

Schritt 4 ist der wichtigste Schritt des PFP-Algorithmus. Er teilt sich in einen Map- und in eine Reduce-Phase auf.
Der Map-Methode wird der in Schritt 1 erzeugte Teil (Shard) der Datenbank übergeben. Die Map-Methode gibt Key/Value-Paare zurück, wobei der Key eine Gruppen-ID und der Value eine Liste mit der Gruppe zugehörigen Transaktionen ist. Die Gruppen-ID wird anhand der G-List erzeugt.
Der Reduce-Methode wird dann eine Gruppen-ID und eine Liste mit Transaktionen übergeben. Für diese Transaktionen erzeugt sie einen FP-Tree und analysiert diesen. Die Ergebnisse werden dann über den Rückgabewert der Reduce-Methode zurückgegeben.

Schritt 5: Aggregieren der Ergebnisse

Im fünften Schritt werden die in Schritt 4 erzeugten Teilergebnisse zu einem Endergebnis zusammengefasst.

In diesem Ansatz wird der PFP-Algorithmus durch geschicktes Aufteilen der Eingangsdaten parallelisiert. Die Erzeugung des FP-Tree erfolgt auf einer Partition der Eingabedaten, für die ein Ergebnis bestimmt wird. Eine Parallelisierung der Erzeugung des FP-Trees ist in diesem Ansatz nicht vorgesehen.

3 Fazit und Ausblick

Mit MapReduce können Algorithmen entwickelt werden, die sehr einfach parallelisiert und in einer verteilten Rechnerarchitektur ausgeführt werden könnnen. Der Entwickler muss sich hierbei nicht um die Parallelisierung oder Verteilung kümmern.
Hadoop ist eine OpenSource Implementierung von MapReduce, mit der MapReduce Algorithmen in der Programmiersprache Java entwickelt werden können. Beim Data Mining müssen große Datenmengen analysiert werden, was unter Umständen sehr rechenaufwändig ist. Aus diesem Grund wurden verschiedene Ansätze entwickelt, um Data Mining Algorithmen auf das MapReduce-Paradigma anzuwenden. So zum Beispiel für den “K-Means”- und “FP-Growth”-Algorithmus. Das Data Mining Framework Mahout, das auf Hadoop basiert, implementiert verschiedene Data Mining Algorithmen, die in einer MapReduce Infrastruktur ausgeführt werden können.
Ziel der Masterarbeit ist es, ein System auf Basis von Hadoop und Mahout zu entwickeln, mit dem verschiedene Data Mining Algorithmen evaluiert und verglichen werden können. Das System soll es ermöglichen, nahezu beliebige Daten mit Data Mining Algorithmen zu analysieren.
Es ist denkbar, dass bisher nicht in Mahout vorhandene Algorithmen implementiert oder neue Ansätze entwickelt werden, Data Mining Algorithmen auf das MapReduce-Paradigma anzuwenden.

Literaturverzeichnis

[Dean und Ghemawat 2008] DEAN, Jeffrey ; GHEMAWAT, Sanjay: MapReduce: simplified data processing on large clusters. In: *Commun. ACM* 51 (2008), Januar, S. 107–113. – ISSN 0001-0782

[Han und Kamber 2006] HAN, Jiawei ; KAMBER, Micheline: *Data Mining. Concepts and Techniques*. Morgan Kaufmann, 2006. – ISBN 978-1558609013

[Han u. a. 2000] HAN, Jiawei ; PEI, Jian ; YIN, Yiwen: Mining frequent patterns without candidate generation. In: *SIGMOD Rec.* 29 (2000), Mai, Nr. 2, S. 1–12. – ISSN 0163-5808

[Lam 2010] LAM, Chuck: *Hadoop in Action*. Manning Publications, 12 2010. – ISBN 9781935182191

[Li u. a. 2008] LI, Haoyuan ; WANG, Yi ; ZHANG, Dong ; ZHANG, Ming ; CHANG, Edward Y.: Pfp: parallel fp-growth for query recommendation. In: *Proceedings of the 2008 ACM conference on Recommender systems*. New York, NY, USA : ACM, 2008 (RecSys '08), S. 107–114. – ISBN 978-1-60558-093-7

[MacQueen 1967] MACQUEEN, J. B.: Some Methods for Classification and Analysis of MultiVariate Observations. In: CAM, L. M. L. (Hrsg.) ; NEYMAN, J. (Hrsg.): *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1, University of California Press, 1967, S. 281–297

[Owen u. a. 2011] OWEN, Sean ; ANIL, Robin ; DUNNING, Ted ; FRIEDMAN, Ellen: *Mahout in Action*. 1. Manning Publications, Januar 2011. – ISBN 1935182684

[Zhao u. a. 2009] ZHAO, Weizhong ; MA, Huifang ; HE, Qing: Parallel K-Means Clustering Based on MapReduce. In: JAATUN, Martin (Hrsg.) ; ZHAO, Gansen (Hrsg.) ; RONG, Chunming (Hrsg.): *Cloud Computing* Bd. 5931. Springer Berlin / Heidelberg, 2009, S. 674–679. – 10.1007/978-3-642-10665-1$_7$1