Fakultät Informatik, Mathematik und
Naturwissenschaften
Studiengang Medieninformatik Master

Abstrakt zum Oberseminar Datenbanksysteme: Aktuelle Trends

Apache Hadoop

Autor: Oliver Blum
Matrikelnr.: 67921
Abgabedatum: 28.06.2017

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Den ersten großen Schritt machten Jeffrey Dean und Sanjay Ghemawat in ihrer Arbeit *MapReduce: Simplified Data Processing on Large Clusters* [JD04]. In diesem wissenschaftlichen Paper aus dem Jahre 2004 wird das Programmiermodell MapReduce zur parallelen Abarbeitung von Aufgaben in einem Cluster vorgestellt: Die Basis für das Hadoop Framework ist geschaffen. Das Hadoop Framework bildet das Grundgerüst bei der verteilten Abarbeitung von strukturierten sowie unstrukturierten Datenmengen im Petabyte-Bereich in einem Cluster. In einer Zeit in der digitale Daten einer der wichtigsten Rohstoffe der Wissensgesellschaft sind, versucht Hadoop den technischen Rahmen für die Analyse dieser Daten zu bilden. Der Vortrag *Apache Hadoop* im Oberseminar *Datenbanksysteme: Aktuelle Trends* möchte die Kernkomponenten des Frameworks vorstellen. Das Ziel ist es einen Einblick in die grundlegende Arbeitsweise von Hadoop zu geben, um ein fundamentales Verständnis der verteilten Datenspeicherung und -analyse auf Basis dieser Technologie zu übermitteln. In den nächsten Abschnitten des Abstrakts werden diese Kernkomponenten vorgestellt. Des weiteren gibt es unterschiedliche Distributionen von Datenanalyseplattformen basierend auf Hadoop. Da es sich nicht um das zentrale Thema der Präsentation handelt, verzichtet diese Arbeit auf eine nähere Beschreibung. Es wird an dieser Stelle auf die offizielle Dokumentation der Cloudera Inc. verwiesen. [ove17]

2 Die Hadoop Kernkomponenten

Das Lesen einer herkömmlichen HDD-1-Terrabyte Festplatte kann bei einer Lesegeschwindigkeit von 100MB/s bereits einiges an Zeit beanspruchen. Um die Analyse einer Datenmenge im Petabyte-Bereich zu ermöglichen, arbeitet Hadoop die Rechenprozesse parallel in einem Computercluster ab. Dieses Cluster kann wenige Maschinen bis hin zu tausende vernetzte Server umfassen. Um so mehr Maschinen im Einsatz sind desto höher ist auch die Wahrscheinlichkeit von Datenverlust. Fällt eine Maschine während eines Prozess aus, dürfen die darauf gespeicherten Daten im Analyseverfahren nicht verloren gehen. Ein konventionelles Dateisystem stößt hierbei an seine Grenzen. Hadoop besitzt ein für diesen Anwendungsfall eigens konzipiertes Dateisystem zur verteilten Speicherung großer Datenmengen, das Hadoop Distributed Filesystem. Neben der verteilten Speicherung ist das zentrale Ressourcenmanagement der vielen Maschinen ein wesentlicher Bestandteil des Frameworks. Es muss erkannt werden welche Rechner im Cluster ausgelastet sind und welche Maschinen den laufenden Prozessen Einsatzmittel (z.B. CPU und Arbeitsspeicher) zur Verfügung stellen können. Der Vortrag möchte die Kernkomponenten des Hadoop Frameworks vorstellen, um einen generellen Überblick über die Arbeitsweise von Hadoop zu erlangen. Das

Dateisystem HDFS befindet sich in der untersten Schicht die für die Speicherung der Daten zuständig ist. Darüber befindet sich die Schicht zur Verwaltung der Ressourcen mit Hilfe des Ressourcenmanagementsystems YARN. Die eigentlichen Anwendungen setzen dann auf die Verwaltungsschicht auf und werden mit Hilfe eines Frameworks für MapReduce umgesetzt. MapReduce ist ein Programmiermodell zur parallelen Abarbeitung von Aufgaben. Der Artikel [Fis17] von *heise.de* geht auf den Unterschied zwischen dem Programmiermodell und einem Framework von MapReduce ein. Darüber hinaus gibt es Erweiterungen die im praktischen Einsatz Verwendung finden wie etwa Tez und Spark, um die Arbeit für speziellen Anwendungsfällen zu erleichtern. Bevor die einzelnen Komponenten in diesem Kapitel vorgestellt werden, soll an dieser Stelle das Buch *Hadoop: The Defintive Guide* [Whi15d] von Tom White erwähnt sein. Es ist ein umfangreiches Werk zu Hadoop mit vielen veranschaulichten Anwendungsbeispielen das oft Antworten während der Bearbeitung des Themas geben konnte. Desweiteren wird auf die offizielle Seite des Hadoop Projekts [apa17a] der Apache Software Foundation verwiesen.

2.1 HDFS Dateisystem

Sobald der Datenumfang die Speicherkapazität einer physikalischen Maschine übertrifft, wird es nötig die Daten auf mehrere Maschinen zu verteilen. Die größte Herausforderung dabei ist Datenverluste während der Berechnung zu vermeiden falls eine Maschine ausfällt. Hadoop besitzt ein verteiltes Dateisystem, das Hadoop Distributed Filesystem (HDFS), das sich dieser Problematik annimmt.
HDFS besteht aus drei Komponententypen: dem *namenode* dem *datanode* und dem Client des Dateisystems. Der *namenode* verwaltet den Namespace und die Metadaten der gespeicherten Dateien wie etwa Zugriffsrechte, Änderungsdatum und Zugriffsdatum. Es gibt einen zentralen *namenode* für jedes Cluster auf dem die Speicherorte einer Datei abgelegt sind. Die Dateien wiederum sind in Blöcke von standardgemäß 128MB aufgeteilt. An der Blockgröße ist bereits zu erkennen, dass das veteilte Dateisystem von Hadoop für sehr große Dateien ausgelegt ist. Die Rede ist von Dateigrößen die hunderte Gigabyte umfassen können. Die Blöcke einer Datei können im gesamten Cluster verteilt abgelegt sein. Die Speicherung erfolgt redundant auf den *datanodes*. Hierbei wird jeder Block reproduziert und auf unterschiedlichen *datanodes* abgelegt. Den Überblick darüber behält der *namenode*. Die Replikationen beugen dem Datenverlust vor falls ein *datanode* während der Bearbeitung ausfällt. Möchte eine Anwendung einen Block einer Datei lesen oder schreiben geschieht dies über den Client des Dateisystems. Beim Lesen wird der *namenode* über den Client kontaktiert und gibt eine Liste von *datanodes* zurück die die Replikationen des gewünschten Blocks enthalten. Diese Liste ist entsprechend der Entfernung der *datanodes* im Netzwerk

geordnet. Der Client kontaktiert anschließend den am nächsten gelegenen freien *datanode* direkt, um die Übertragung des gewünschten Blocks zu veranlassen. Eine ausführlichere Einführung in die Architektur des verteilten Dateisystems von Hadoop beschreibt Robert Chansler et al. auf der Webseite [ea17]. Zur Vertiefung sei ebenso das Kapitel 3 [Whi15b] über das Hadoop Distributed Filesystem im bereits erwähnten Buch von Tom White erwähnt. Beide Quellen dienten auch der Recherche für den Vortrag. Zum Abschluss dieses Abschnitts sei noch die spaltenorientierte Datenbank HBase erwähnt. Sie baut auf das verteilte Dateisystem auf, um die Verwaltungsaufgaben an dieses auszulagern. Darüber hinaus ermöglicht HBase schnelle Lese- und Schreibzugriffe durch Indizierung der HDFS-Blöcke. Diese und viele weitere Informationen befinden sich auf der offiziellen Webseite [hba17] der Apache Foundation.

2.2 Ressourcenmanagement mit YARN

Der Vortrag geht im nächsten Schritt auf das Ressourcenmanagementsystem von Hadoop Apache YARN (Yet Another Resource Negotiator) ein. Es ist dafür verantwortlich die Systemressourcen der einzelnen Maschinen im Cluster optimal für die Abarbeitung der Prozesse zu verteilen. Diese Systemressourcen werden hierbei in virtuellen Umgebungen, den sogenannten Containern, gebündelt und Prozesse einer YARN Applikation darin ausgeführt. Weniger abstrakt ausgedrückt handelt es sich bei Containern um Linux *cgroups* mit der Möglichkeit Prozesse zu gruppieren und ressourcenbeschränkten Subsystemen zuzuweisen. Weitere Informationen zu *cgroups* gibt es in der ausführlichen Dokumentation [Men17] von Paul Menage.
Eine Applikation im Kontext von YARN ist entweder ein einzelner *job* oder eine Bündelung mehrere aufeinanderfolgender *jobs*. Ein *job* repräsentiert eine Aufgabe bzw. Teilaufgabe einer Applikation. Es wird wieder auf den Abschnitt [Whi15c] des Buchs von Tom White hingewiesen. MapReduce setzt auf YARN auf und erleichtert das Schreiben einer Applikation deren Abarbeitung über YARN gesteuert wird. YARN teilt das Ressourcenmanagement und Job Scheduling in zwei separierte Dienste auf. Es gibt einen zentralen *Resource Manager* im Cluster und jeweils einen *Application Master* für jede Applikation. Der *Resource Manager* vermittelt Ressourcen an alle Applikationen bzw. *jobs* im System. Der *Application Master* einer Applikation ist dafür zuständig Ressourcen zur Bearbeitung mit dem *Resource Manager* auszuhandeln. Außerdem arbeitet der *Application Master* zusammen mit dem *Node Manager* an der Ausführung und Überwachung der Aufgaben einer Applikation. Der *Node Manager* übernimmt hierbei die Rolle eines Agenten der auf jedem Host installiert ist. Er ist verantwortlich für die Verwaltung der Container des Knotens, überwacht den Ressourcenverbrauch und informiert den *Resource Manager* bzw. den Scheduler darüber. Nähere Informationen bietet des weiteren die Dokumentation zu YARN [YAR16]

auf der offiziellen Hadoop Webseite der Apache Software Foundation.
Was geschieht wenn eine Applikation über YARN gestartet wird, ist ausführlich in [Whi15c] geschildert. Zusammengefasst kontaktiert der Klient der Applikation den *Resource Manager*, um einen anwendungsspezifischen *Application Master* zu starten. Daraufhin sucht der *Resource Manager* im Cluster einen Knoten der über seinen *Node Manager* den *Application Master* in einem *Container* bereitstellt. Der *Application Master* arbeitet anschließend die Berechnungen ab und gibt das Ergebnis an den Klienten zurück. Er kann aber auch weitere *Container* vom *Resource Manager* anfragen, um eine verteilte Bearbeitung der Berechnungen anzustoßen.

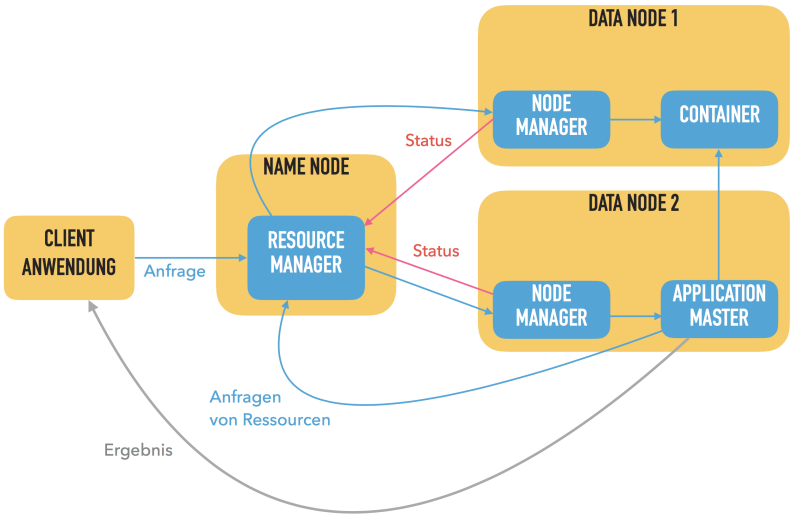

Abbildung 1: Grundlegende Arbeitsweise von YARN

Die Abbildung 1 zeigt außerdem einen Überblick der verschiedenen Komponenten während der verteilten Bearbeitung einer Applikation die von einem Klienten über den zentralen *Resource Manager* gestartet wurde. Dieses Beispiel besteht aus einem zentralen *namenode* der den *Resource Manager* beherbergt und zwei *datanodes* mit jeweils einem *Node Manager*. Für die Applikation wurde bereits ein *Application Master* über den *Node Manager* des *Data Node 2* bereitgestellt. Dieser *Application Master* hat beim *Resource Manager* weitere *Container* zur Berechnung angefragt, woraufhin der *Node Manager* des *Data Node 1* Ressourcen über einen *Container* zur Verfügung gestellt hat. Am Ende der Berechnung kann der *Application Master* das Ergebnis an den Klienten zurückgeben.

2.3 Map Reduce Jobs

Zur Vervollständigung wird nun das MapReduce-Programmiermodell vorgestellt, um die Zusammenarbeit von Knoten eines Clusters anhand eines Anwendungsbeispiels zu verdeutlichen. MapReduce setzt auf das Ressourcenmanagement auf und ermöglicht YARN-Applikationen zur verteilten Verarbeitung im Cluster zu schreiben. Dieses Programmiermodell ist in verschiedenen Programmiersprachen mit unterschiedlichen Implementierungsfeinheiten in MapReduce-Frameworks umgesetzt. Hier sei nochmal auf den Artikel [Fis17] von Oliver Fischer hingewiesen. Das Modell unterteilt das Verfahren in zwei Phasen: einer *map*-Phase und einer *reduce*-Phase bestehend aus einer *map*-Funktion und einer r*educe*-Funktion. Die konkrete Implementierung beider Funktionen legt der Programmierer anwendungsspezifisch fest. Als kleiner Exkurs wird an dieser Stelle auf die Data Warehouse Software Hive [apa17b] der Apache Foundation verwiesen. Es ermöglicht MapReduce-Anfragen in einer SQL-artigen Form stellen zu können. In einer kurzen Demonstration am Ende des Vortrags wird eine solche Anweisung präsentiert. Es soll die Arbeitserleichterung gegenüber einer Java-Implementierung von MapReduce verdeutlichen. Mit einem Beispiel aus [Whi15a] wird die Vorgehensweise von MapReduce in der Abbildung 2 veranschaulicht. Dieses Beispiel befindet sich nicht in den Folien. Es wird an dieser Stelle gewählt, da es während der Recherche für den Vortrag hilfreich war und kann als eine sinnvolle Ergänzung betrachtet werden.

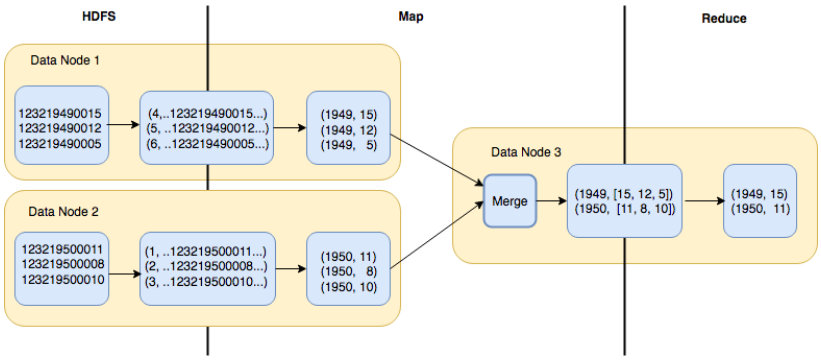

Abbildung 2: Das MapReduce Programmiermodell anhand eines Beispiels

In diesem Beispiel handelt es sich um Wetterdaten die im HDFS Dateisystem zweier *datanodes* als Text abgelegt sind. Aufgabe ist es die Höchsttemperatur eines jeden Jahres zu ermitteln. Nachfolgend sind Ausschnitte von drei Zeilen der Wetterdaten illustiert. Es sind Jahreszahl und Temperatur darin enthalten. Die Informationen werden zeilenweise eingelesen, um Tupel aus Zeilennummer und Zeileninhalt zu bilden die der *map*-Funktion als Eingabe übergeben werden.

```
(1, ...1232_1949_00_15...)
(2, ...1232_1949_00_12...)
(3, ...1232_1949_00_05...)
```

Die Implementierung der *map*-Funktion sorgt in diesem Fall für das Herausfiltern der benötigten Informationen um daraus Tupel aus Jahreszahl und Temperatur zu bilden. Das Mappen der Daten geschieht im besten Fall auf dem *datanode* der die entsprechenden Daten enthält. Mehrere *datanodes* können somit parallel die Informationen für die *reduce*-Phase vorbereiten.

```
(1949, 15)
(1949, 12)
(1949, 5)
```

Die Resultate werden anschließend an den *datanode*, der vom Ressourcenmanagement für die *reduce*-Phase ausgewählt wurde, übertragen und zusammengefasst.

```
(1949, [15, 12, 5])
```

Die *reduce*-Funktion erhält somit ein aufbereitetes Tupel mit eindeutigen Key-Value-Paaren die schnell auf die gesuchte Höchsttemperatur reduziert werden kann.

```
(1949, 15)
```

Die *reduce*-Phase muss nicht auf einen *datanode* begrenzt sein. Auch hier können mehrere Knoten daran beteiligt sein die Daten aus der *map*-Phase entgegenzunehmen und auf das Wesentliche zu reduzieren.

3 Fazit und Ausblick

Im Laufe des Vortrags wird das Kernkonzept von Hadoop erläutert. Die Hauptkomponenten des Frameworks werden vorgestellt und deren Zusammenspiel erläutert. Anschließend widmet sich der Vortrag kurz der Datenanalyseplattform Cloudera CDH Plattform der Cloudera Inc. Neben den Hauptbestandteilen von Hadoop integriert diese Distribution des Frameworks wichtige Nebenprojekte der Apache Software Foundation zur Erweiterung der Hadoop-Umgebung. Des weiteren bietet die Cloudera Plattform ein nützliches Verwaltungswerkzeug für die eingesetzte Infrastruktur. Als Ausblick wird an dieser Stelle die Integration des Projekts Apache Spark in die Hadoop-Umgebung erwähnt. Es bietet eine In-Memory Datenanalyse die den Hauptspeicher eines verteilten Systems zusammenfasst. Darüber hinaus erweitert das Framework die Verarbeitungsprimitiven von MapReduce um weitere Phasen. So bietet es einen größeren Funktionsumfang bei schnellerer Verarbeitung die auch Echtzeitanalysen ermöglicht. Diese und weitere Informationen befinden sich auf der offiziellen Webseite des Apache Spark Projekts [apa17c].

Literatur

[apa17a] Welcome to apache™ hadoop®! In *http://hadoop.apache.org*. The Apache Software Foundation, den 22.06.2017.

[apa17b] Apache hive tm. In *https://hive.apache.org*. The Apache Software Foundation, den 28.06.2017.

[apa17c] Apache spark: Lightning-fast cluster computing. In *https://spark.apache.org*. The Apache Software Foundation, den 28.06.2017.

[ea17] Robert Chansler et al. The hadoop distributed file system. In *http://www.aosabook.org/en/hdfs.html*, den 22.06.2017.

[Fis17] Oliver Fischer. Programmiermodell und framework. In *https://www.heise.de/developer/artikel/Programmiermodell-und-Framework-964823.html*, den 22.06.2017.

[hba17] Welcome to apache hbase™. In *https://hbase.apache.org*. The Apache Software Foundation, den 28.06.2017.

[JD04] Sanjay Ghemawat Jeffrey Dean. Mapreduce: Simplied data processing on large clusters. Google, Inc., 2004.

[Men17] Paul Menage. Cgroups. In *https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt*, den 22.06.2017.

[ove17] Overview of cloudera and the cloudera documentation set. In *https://www.cloudera.com/documentation/enterprise/5-4-x/topics/introduction.html*. Cloudera Inc., den 22.06.2017.

[Whi15a] Tom White. Chapter 2: Mapreduce. In *Hadoop: The Definitive Guide, Fourth Edition, S. 19-42*. O'Reilly Media, Inc., 2015.

[Whi15b] Tom White. Chapter 3: The hadoop distributed filesystem. In *Hadoop: The Definitive Guide, Fourth Edition, S. 43-78*. O'Reilly Media, Inc., 2015.

[Whi15c] Tom White. Chapter 4: Yarn. In *Hadoop: The Definitive Guide, Fourth Edition, S. 79-96*. O'Reilly Media, Inc., 2015.

[Whi15d] Tom White. *Hadoop: The Definitive Guide, Fourth Edition*. O'Reilly Media, Inc., 2015.

[YAR16] Apache Hadoop YARN. Apache hadoop yarn. In *https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html*. The Apache Software Foundation, den 22.06.2016.