



**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

Semester-Projekt

Aufbau einer NoSQL-Datenbank mit Apache Hadoop und Apache HBase für das Million Song Dataset

- Schemalose Datenbanken -

Fachbereich Informatik
Referent: Prof. Dr. Harm Knolle

eingereicht von:
Moritz Kemp, Johann Martens
Matr.-Nr.

Sankt Augustin, den 12.10.2016

Zusammenfassung

Das Ziel der Projektarbeit zum Thema *Hadoop/Hbase* ist es, eine Datenbank mittels Hadoop/Hbase auf einem bereitgestelltem Cluster so zu installieren, dass der Anwender in dem One-Million-Datensatz nach Liedern suchen kann. Im ersten Teil der Projektarbeit werden die theoretischen Grundlagen von Hadoop/Hbase behandelt. Dazu gehört ein Kapitel zu Map/Reduce, ein Kapitel zu Hadoop und ein Kapitel zu Hbase. Die Theorie soll etwa ein Drittel der Projektarbeit ausmachen.

Der zweite Teil der Projektarbeit, zwei Drittel der Projektarbeit, beschreibt die praktische Umsetzung, die aus der Installation und Konfiguration der Hadoop/Hbase-Datenbank und der Implementierung der Client-Software besteht. Eingangs werden Anforderungen, Anwendungsfälle und technische Rahmenbedingungen kurz beschrieben, um einen Überblick zu geben. Anschließend folgen das Kapitel der Installation bzw. Implementierung von Hadoop/Hbase auf den Clustern. Hier wird das Vorgehen beschrieben, wie die Installation und Konfiguration abläuft. Eingesetzte Programmiersprachen sind hier Java und JRuby. Fragestellungen werden sein, wie sich die Datenbank auf einen Cluster installieren lässt und welche Konfigurationsparameter hinsichtlich des Anwendungsfalls eines One-Million-Datensatzes berücksichtigt und angepasst werden müssen. Auch die konkrete Implementierung der map und reduce Funktionen wird in diesem Kapitel dargelegt. Auch Filter- und Optimierungsparameter, die besonders Hbase betreffen, werden evaluiert und gegebenenfalls angewendet. Zum Abschluss wird gezeigt, ob und wie sich die Datenbank auch ohne eigens programmiertem Client Ad-Hoc ansprechen lässt, beispielsweise über eine laufende Shell. Das nächste Kapitel beschreibt die Implementierung der Client-Software, die mittels Java als Serveranwendung und Javascript als Client-Anwendung dem Anwender die Möglichkeit bieten soll, über eine grafische Oberfläche mit dem One-Million-Datensatz zu arbeiten. Der Client hat dabei aber wenige, nur funktionale Anforderungen, eine aufwendig grafische Oberfläche ist nicht das Ziel. Ein weiteres Kapitel im praktischen Teil behandelt die Installation des One-Million-Datensatzes in das Hadoop-Dateisystem. Wichtige Fragestellungen spielt dabei, inwieweit die Daten bereits für die Speicherung im Hadoop-Dateisystem geeignet sind sowie die Frage nach der Replikation auf dem Cluster-System.

Zum Abschluss des praktischen Teils folgt eine kurze Leistungsdarstellung des Systems, indem die Antwortzeiten der Datenbank zu bestimmten Abfragen gemessen werde. Optional ist eine Leistungsbewertung im Vergleich mit den anderen Teams denkbar, die ebenfalls mit dem selben Datensatz arbeiten.

Inhaltsverzeichnis

Zusammenfassung	II
1 Das MapReduce-Framework	1
1.1 Arbeitsweise von MapReduce	1
1.2 Anwendung für Datenbanken	2
1.3 Optimierung	2
1.4 Beispiel	2
2 HBase	3
2.1 Allgemeiner Überblick	3
2.2 Gründe für HBase	3
Abbildungsverzeichnis	5
Tabellenverzeichnis	6
Listings	7
Abkürzungsverzeichnis	7
Literaturverzeichnis	9
Eidesstattliche Erklärung	10

1 Das MapReduce-Framework

MapReduce wurde 2004 von zwei Mitarbeitern von Google vorgestellt [DG08]. Es ist ein Framework für die Programmierung von massiver, paralleler Datenverarbeitung sehr großen Datenmengen. Der Programmierer kann sich dabei komplett auf die Verarbeitung der Daten konzentrieren, während das MapReduce-Framework sich um die Verteilung der Daten und der parallelen Ausführung der Anwendung kümmert. Im Zusammenarbeits mit speziellen Filesystemen für große Cluster-Systeme, die zum Beispiel automatisch redundante Kopien auf mehreren Maschinen anlegen, ermöglicht eine mit MapReduce programmierte Anwendung eine hohe Toleranz gegenüber dem Ausfall von einzelnen Maschinen.

1.1 Arbeitsweise von MapReduce

Auf der obersten Abstraktionsebene besteht das MapReduce-Framework nur aus zwei Funktionen: *map* und *reduce*. Beide Funktionen müssen vom Anwender des MapReduce-Frameworks spezifiziert werden. *map* nimmt eine (ungeordnete) Liste von Daten entgegen, die Schlüssel-Wert-Paare (key/value) enthält. Die Funktion verarbeitet nun diese Liste nach dem vom Programmierer spezifizierten Funktion und liefert als Ergebnis wiederum eine Liste von Schlüssel-Wert-Paaren. Die Logik der *map*-Funktion ist vom Anwendungsfall abhängig, bildet aber in der Regel die ungeordneten Daten in eine Liste ab, die für den Anwendungsfall interessanten Daten enthält. Ein gern genommenes Beispiel ist das Zählen von Wörtern in tausenden von Dokumenten. Die Liste für die Eingabe in die *map*-Funktion enthält für jeden Schlüssel als Wert ein ganzes Dokument. Diese Dokumente werden durchsucht und gleichzeitig eine neue, deutlich größere Liste mit Schlüssel-Wert-Paaren erstellt, die für jedes einzelne Wort den String selbst als Schlüssel speichert, und als Wert eine 1 angibt. Somit ist jedes Wort in dieser Liste repräsentiert. Diese neue Liste enthält sehr viele gleiche Schlüssel, wobei alle den Wert 1 tragen.

Das Ergebnis von der *map*-Funktion ist aber immer nur ein Zwischen-Ergebnis und wird direkt als Eingabe für die *reduce*-Funktion verwendet, die am Ende das tatsächliche Ergebnis der Verarbeitung ausgibt. Die *reduce*-Funktion hat in der Regel die Aufgabe, das Ergebnis der *map*-Funktion zusammenzufassen. Bezogen auf das Beispiel mit der Zählung von Wörtern bekommt die *reduce*-Funktion nun eine (sehr lange) Liste von allen Wortvorkommnissen. Die Funktion sortiert nun diese Liste und erzeugt eine neue Liste, die alle Schlüssel-Werte enthält, aber nur ein einziges mal in der Liste. Mehrfache Einträge in der ursprünglichen Liste mit gleichen

Schlüsselwerten werden also zu einem Eintrag zusammengefasst, wobei die Werte der einzelne, mehrfachen Einträge aufaddiert werden. Das Ergebnis ist eine Liste mit jedem Wort, das in den Dokumenten vorkommt, als einmaliger Schlüssel und als Wert die jeweilige Häufigkeit.

1.2 Anwendung für Datenbanken

1.3 Optimierung

1.4 Beispiel

2 HBase

Im folgenden wird HBase als Datenbank vorgestellt und anschließend werden Gründe aufgezeigt, warum HBase im Rahmen von BigData eingesetzt werden sollte.

2.1 Allgemeiner Überblick

Die spaltenorientierte Datenbank HBase basiert auf der Datenbank BigTable von Google, weshalb diese Datenbanken Gemeinsamkeiten bezüglich ihrer Funktionalität aufweisen. Beispielsweise unterstützen beide die Komprimierung und Versionierung der Daten.

Des weiteren erinnert HBase an eine relationale Datenbank, da sie ihre Daten in Tabellen speichert, die Zellen enthalten. Jedoch verhalten sich die Tabellen nicht wie Relationen und Zeilen nicht wie Datensätze in relationalen Datenbanken. Auch sind die Spalten nicht durch ein Schema definiert.

Da HDFS ein Filesystem ist, fehlt ihm die zufällige Lese- und Schreibfähigkeit. Eine mögliche Lösung dafür ist HBase. Es wird innerhalb des Hadoop-Clusters betrieben und stellt in Echtzeit Lese- und Schreibzugriff zu den Daten her. HBase ist besonders wertvoll, wenn es um die Verarbeitung von sehr großen Datenmengen geht. Aus diesem Grund wird es auch oftmals als Logging- und Suchsystem in großen Unternehmen eingesetzt.

2.2 Gründe für HBase

Für den Einsatz von HBase gibt es gleich mehrere Gründe:

- Skalierbarkeit
- Versionierung
- Komprimierung
- Garbage Collection

- speicherbasierte Tabellen
- Durch write-ahead-Logging und eine verteilte Konfiguration kann sich Hbase schnell von Serverausfällen erholen
- Bietet geringe Latenz bei Zugriff auf kleine Teil-Datenmengen
- Bietet ein flexibles Datenmodell

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

API	Application Programming Interface
BASE	Basically Available, Soft State, Eventual Consistency
BG	Barahmand Ghandeharizadeh
CAP	Consistency Availibiltiy Partition Tolerance
CLI	Command-Line Interface
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
DBA	Datenbankadministrator
DBS	Datenbanksystem
DNS	Domain Name System
DTD	Document Type Definition
GUI	Graphical User Interface
HDD	Hard Disk Drive
IP	Internet Protocol
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
NoSQL	Not only SQL
OLTP	Online Transaction Processing
RDBMS	Relational Database Management System
RFC	Request For Comments
SLA	Service Level Agreement
SPEC	Standard Performance Evaluation Corporation
SQL	Structured Query Language
SSD	Solid State Drive
TPC	Transaction Processing Performance Council
UML	Unified Markup Language
XML	Extensible Markup Language
YCSB	Yahoo! Cloud Serving Benchmark

Literaturverzeichnis

- [DG08] DEAN, Jeffrey ; GHEMAWAT, Sanjay: MapReduce: simplified data processing on large clusters. In: *Communications of the ACM* 51 (2008), Nr. 1, S. 107–113

3 Eidesstattliche Erklärung

Ich versichere an Eides Statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe. Alle Stellen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Ich versichere außerdem, dass ich keine andere als die angegebene Literatur verwendet habe. Diese Versicherung bezieht sich auch auf alle in der Arbeit enthaltenen Zeichnungen, Skizzen, bildlichen Darstellungen und dergleichen.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

den 1. November 2016
Ort, Datum

Moritz Kemp, Johann
Martens