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 zu implementieren, die eine Lieder-Datenbank darstellt. Dabei soll die Datenbank so installiert werden, dass sie ihre Stärken mit sehr großen Datensätzen, in diesem Falle dem Million-Song-Datensatz, ausspielen kann.

Das Team wird sich zuerst mit den Grundlagen von NoSQL-Datenbanken befassen, vor allem explizit mit den Grundlagen von Hadoop/Hbase. Darauf aufbauend lässt sich eine Argumentation formulieren, warum sich die Behandlung von derart großen Datensätzen wie dem Million-Song-Datensatz mit Hadoop/Hbase effektiver gestalten lässt als mit klassischen relationalen Datenbanken. Dieser Teil beinhaltet also im wesentlichen Grundlagen zu den Wide-Column-Datenbanken sowie Grundlagen zu Hadoop und Hbase.

Im praktischen Teil wird zunächst die benötigte Software installiert, i.e. Java, Hadoop und HBase. Um die Installation auf einer Windows-Maschine auszuführen wird zusätzlich Cygwin installiert. Um auf die HBase-Datenbank zugreifen zu können, muss HBase zunächst konfiguriert werden.

Nach erfolgreicher Installation wird HBASE zunächst im Standalone-Modus auf einer Maschine betrieben. Zur Anschauung wird eine Tabelle angelegt und es werden CRUD-Operationen geschildert. Im Anschluss daran wird gezeigt, wie man programmatisch mit der Datenbank arbeiten kann mit Hilfe von JRuby.

Nach den ersten Tests wird der Import von Daten (One-Million-Song-Datensatz) beschrieben und durchgeführt. Für die Verwaltung der Daten wird eine Client-Anwendung mit Ruby geschrieben.

Im letzten Schritt wird HBase auf dem Hochschul-Cluster installiert und für den Cluster-Modus konfiguriert.

Die geplante Gliederung sieht wie folgt aus:

- Grundlagen von Wide-Column-Datenbanken (only MapReduce, 6C)
- Grundlagen Hadoop/Hbase (6D1/2)
- Anforderungen an die Anwendung
- Aufbau der Datenbank (6F)
- Installation
- Cluster-Betrieb
- Benchmarks

Folgende Meilensteine werden definiert:

28.10 - Expose

25.11 - HBase im Standalone-Modus auf einer Windows-Maschine installieren und CRUD-Operationen mit Testdaten durchführen.

25.11 - One-Million-Dataset importieren

02.12 - Zwischenbericht

02.12 - MapReduce-Kapitel 6C

02.12 - Hadoop-Kapitel 6D1

16.12 - HBase-Kapitel 6D2

16.12 - Client-Anwendung fertig stellen

23.12 - Cluster-Installation abgeschlossen

06.01 - Ausarbeitung Projektarbeit

Inhaltsverzeichnis

1 Das MapReduce-Framework

MapReduce wurde 2004 von zwei Mitarbeitern von Google vorgestellt [DG08]. Es ist ein Framework für die Programmierung von massiver, parallele 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 Zusammenarbeit 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 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 Günde 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 Datenmodel

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

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 28. Oktober 2016
Ort, Datum

Moritz Kemp, Johann Martens