Apache Hadoop
Abstract

Tim Delle
HTWK Leipzig
Karl-Liebknecht-Str. 132 04277 Leipzig
tim.delle@stud.htwk-leipzig.de

Oberseminar
Datenbanksysteme
aktuelle Trends

In diesem Abstract wird der Vortrag, des Oberseminars: Datenbanksysteme aktuelle Trends, zum Thema *Apache Hadoop* zusammengefasst. Die Beschreibung beschränkt sich auf die Kernkomponenten von *Apache Hadoop*. Das Apache Hadoop Projekt wird von Tom White wie folgt beschrieben. [1]

> "In a nutshell, this is what Hadoop provides: a reliable, scalable platform for storage and analysis"

Aus dem Wort 'platform' lässt sich ableiten, dass es sich bei Hadoop nicht um eine fertige Komplettsoftwarelösung handelt. Es ist ein komplexes Konstrukt aus mehreren Komponenten und Erweiterungen. Hadoop entstand aus einer Open Source Web Search Engine[2] mit Doug Cutting als Hauptentwickler. Auf Basis von *Google's Distributed Filesystem*[3] entstandt 2004 das *Nutch Distributed Filesystem* um Skalierungsproblemen vorzubeugen. Yahoo übernahm das Projekt 2006 und trieb die Entwicklung kontiunierlich voran. Seit 2008 ist Hadoop unter dem Namen Apache Hadoop ein Top-Level Projekt der *Apache Software Foundation*. Apache Hadoop besteht aus den Kernkomponenten *Hadoop Distributed Filesystem* und aus einem *MapReduce* Framework. Die Komponenten werden in folgendem Abschnitt zusammenfasst.

[1] Autor des Hadoop Referenzwerkes: *Hadoop - The Definitive Guide*
[2] *Apache Nutch - 2002*
[3] Google Paper - 2003

Hadoop Distributed Filesystem(HDFS)

HDFS wurde unter besonderen Design Richtlinien entworfen. Eine Instanz des Dateisystems soll verteilt auf tausenden Knoten laufen. Dabei findet handelsübliche Hardware Anwendung [1]. Die Ausfallwahrscheinlichkeit ist dementsprechend hoch. Bei einer hohen Wahrscheinlichkeit eines Ausfalles, muss die Ausfallerkennung und Wiederherstellung schnell reagieren können. Das Dateisystem muss mit sehr großen Dateien umgehen[2]. Daten werden einmalig geschrieben und häufig gelesen. die folgenden Konzepte machen HDFS zu einem skalierbaren, performanten und verteilten Dateisystem. Die Design Richtlinien werden durch diese Konzepte realisiert.

Moving Computation ist ein Konzept, welches es ermöglicht datenintensive Berechnung auf dem Knoten auszuführen, welcher die Daten bereitstellt. *Apache* bezeichnet dieses Konzept auch als ***Data Locality Optimization*** und ermöglicht damit eine erhebliche Reduzierung der Netzwerklast, da große Datenmengen nicht über das Netzwerk zwischen Knoten transportiert werden müssen.

Blöcke erlauben die Zerlegung der goßen Dateien. Der Standardwert für die Blockgröße beträgt 128 Megabyte. Das Blockkonzept bietet einen einfachen Ansatz zur Replikation (auf Blockbasis) und erlaubt die Speicherung von sehr großen Dateien. Eine einzelne Datei kann viel größer sein als die Festplatte eines einzelnen Knotens.

NameNode ist ein Master Knoten im HDFS, welcher die Zugriffe von Clients auf die Daten verwaltet. Dabei verwaltet das NameNode den *HDFS Namespace* [3]. Änderungen an Dateien werden in einem Edit Log und der Filesystem Baum in einem Namespace Image gespeichert. Ab der Version 2.x sorgt ***HDFS High Availability*** für stark erhöhte Ausfallsicherheit des NameNodes. Bei diesem Konzept wird ein aktives NameNode im Standby bereitgehalten. Das Namespace Image und der Editlog werden in einem hochverfügbaren geteilten Speicher vorgehalten.

DataNode ist ein SlaveKnoten, welcher für die Erstellung, das Löschen und die Replikation von Blöcken zuständig ist. Durch ***Blockreports*** (Informationen darüber, welche Blöcke sich auf welchem DataNode befinden) an das NameNode, werden alle Blöcke im HDFS verwaltet. Zusätzlich werden hochfrequene *Heatbeats* an das NameNode geschickt. Somit ist eine schnelle Ausfallerkennung gewährleistet. Die Wiederherstellung ist durch die Replikation der Blöcke gegeben.

Block Caching erhöht die Performanz des HDFS beim Zugriff auf Daten druch Clients. DataNodes halten Blöcke in einem schnellen Cache Speicher vor. In den *Blockreports* an das *NameNode* ist vermerkt, welcher Block gecached ist. Somit kann das *NameNode* bevorzugt Adressen von *DataNodes* ausliefern, welche die Angefragten Daten gecached haben.

[1] Keine ausfallsichere Serverhardware
[2] Gigabyte/Terrabyte
[3] Zuordnung von Blöcken zu DataNodes

Hadoop MapReduce

Hadoop MapReduce ist allgemein bezeichnet ein Software Framework für die Verarbeitung von riesigen Datenmengen auf vielen verteilten Knoten. Die Verarbeitung der Daten erfolgt in zwei Phasen, welche vom Anwender selbst definiert werden müssen.

Die ***Map Phase*** erhält *Key-Value* Paare als Eingabeparameter, bereitet die erhaltenen Daten durch die Map-Funktion auf und liefert Key-Value Paare als Ergebnis zurück. Das Ergebnis wird durch das MapReduce Software Framework nach Key sortiert. Genau betrachtet laufen also drei Phasen ab (Map - Shuffle - Reduce) wobei der Anwender nur auf die Map und Reduce Phasen einfluss hat.

Die **Reduce Phase** erhält die nach Key sortierten Key-Value Paare der Shuffle Phase als Eingabeparameter und berechnet aus diesen Zwischenwerten die Ergebniswerte.

Folgendes Beispiel von Wetterdaten illustriert die Funktion der Phasen. Die textbasierten Daten stammen von einer Wetterstation und es ist nur ein Auschnitt des Jahres 1901 dargestellt. Die Temperaturdaten ergeben durch 100 geteilt die Temperatur in Grad Celius[1]

Eingabe-Key := Offset vom Dateianfang
Eingabe-Value := Text der Zeile

$(\quad 0, 002902 \ldots 1901010106004 \ldots 00781 \ldots)$
$(134, 002902 \ldots 1901010113004 \ldots 00721 \ldots)$
$(268, 002902 \ldots 1901010120004 \ldots 00941 \ldots)$

Die Map Funktion extrahiert aus diesen Rohdaten nun die Temperatur und das Jahr, da diese Werte für eine Einschätzung zur Temperaturerhöhung verwendet werden sollen. Die Ausgabe der **Map Phase** wird nach Key gruppiert und dient als Eingabe Key-Value Paare für die **Reduce Phase** .

$(1901, 782)(1901, 721)(1901, 941) \longrightarrow (1901, [782, 721, 941])$

Am Ende wird die Reduce Funktion auf die verbleibenden Datensätze angewendet um die maximale Temperatur des Jahres zu berechnen.

$(1901, 941)$

Der MapReduce Ansatz ist nicht neu, die Effiziente Abarbeitung im Hadoop System liegt an der Ausgliederung der **Map Phase** auf eine Vielzahl von Knoten. Im gezeigten Beispiel kann jedes Jahr von einem Knoten berechnet werden. Die Zwischenwerte werden zum Zielknoten transpotiert und dort wird die **Reduce Phase** ausgeführt um die Ergebniswerte zu berechnen.

Das Zusammenspiel aus dem *Hadoop Distributed Filesystem*, mit der effizienten Abarbeitung von *MapReduce* Jobs macht *Apache Hadoop* genau zu dieser verlässlichen und skalierbaren Plattform, welche am Anfang beschrieben wurde.

[1] Temperaturen stammen aus einer kälteren Jahreszeit

References

[1] White, Tom, Hadoop - The Definitive Guide, O'Reilly, Fourth Edition.

[2] Apache Software Foundation, MapReduce Tutorial
http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

[3] Apache Software Foundation, HDFS User Guide
http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html

[4] Rathbone, Matthew, A Beginners Guide to Hadoop,
http://blog.matthewrathbone.com/2013/04/17/what-is-hadoop.html