VERY LARGE
BUSINESS APPLICATIONS
Carl von Ossietzky Universität Oldenburg

Hadoop / Hive

Referat
im Rahmen des Modul Business Intelligence 2

Themensteller:	Prof. Dr.-Ing. Jorge Marx Gómez
Betreuer:	M.Sc. Stefan Wunderlich

Vorgelegt von:	Nils Lutz Semesteranschrift PLZ Wohnort: Telefonnummer: mustermann@uni-oldenburg.de

Abgabetermin:	25. August 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Apache Hadoop ist ein Framework für skalierbare, verteilt arbeitende Software. Es ermöglicht u.A. effiziente Big-Data-Analysen. Das Framework ist allerdings eine relativ neue Technologie (vgl.:[anG15]). Im Rahmen des Moduls „Business Intelligence II. “wird dieses Thema bearbeitet. Das Ziel dieser Ausarbeitung ist das Erarbeiten des aktuellen Stand der Technik. Außerdem soll bestimmt werden, aus welchen Komponenten das Hadoop Framework besteht. Die Ausarbeitung ist so strukturiert, das als Erstes, das Hadoop Framework bearbeitet wird. Dabei wird analysiert aus welchen Komponenten Hadoop besteht. Hadoop verwendet ein bestimmtes File System zum Arbeiten. Das Hadoop Distributed File System wird im zweiten Unterkapitel beschrieben. Im dritten Unterkapitel wird das MapReduce-Verfahren behandelt. Dabei soll geklärt werden, wie das Verfahren funktioniert. Im abschließenden Fazit werden die Ergebnisse der Ausarbeitung betrachtet.

2. Hadoop / Hive

Text

2.1. Hadoop Framework

Text

2.2. Hadoop Distributed File System

Das Hadoop Distributed File System ist ein Bestandteil von Hadoop. In diesem Unterkapitel werden die Grundlagen dazu erläutert. Das File System wird definiert und kategorisiert. Außerdem werden die Ziele vom HDFS genannt. Abschließend wird die Architekture erläutert.

Definition Das Hadoop Distributed File System wird definiert als ein „hochverfügbares Dateisystem zur Speicherung sehr großer Datenmengen auf den Dateisystemen mehrerer Rechner (Knoten). Dateien werden in Datenblöcke mit fester Länge zerlegt und redundant auf die teilnehmenden Knoten verteilt. Dabei gibt es Master- und Slave-Knoten. Ein Masterknoten, der so genannte NameNode, bearbeitet eingehende Datenanfragen, organisiert die Ablage von Dateien in den Slaveknoten und speichert anfallende Metadaten. HDFS unterstützt dabei Dateisysteme mit mehreren 100 Mio. Dateien.“([Inc15a])

Distributed File Systems Das Hadoop Distributed File System ist ein verteiltes Dateisystem. Ein verteiltes Dateisystem ist ein spezielles Dateisystem, mit dem der Zugriff auf Dateien über ein Rechnernetz erfolgt und das Zugriff und Dateienspeicherung auf mehreren als Server eingesetzten Rechnern erlaubt. Das Gegenstück zu solch einem Netzwerk-Dateisystem ist ein klassisches lokales Dateisystem, welches unmittelbar an den Computer angeschlossene Massenspeicher verwaltet.(vgl.:[Inc15b])

Aufgaben von HDFS Normale Distributed File Systems können in bestimmten Situationen unpraktisch sein. Die Dateien lagern auf einer Maschine. Es können nur so viele Dateien auf einer Maschine gespeichert werden, wie die Maschine Speicher frei hat. Ist die Maschine nicht erreichbar, weil es z.B. Netzwerkprobleme gibt, können die Clients keine Dateien aufrufen. Außerdem kann der Zugriff von vielen Clients den Server überfordern.

Das HDFS wurde so konzipiert, dass die Probleme anderer Distributed File Systems, beim Einsatz von HDFS irrelevant sind. HDFS wurde für diesen Einsatzzweck kreeirt. Es ist optimiert für die Speicherung von sehr großen Datenmengen von Informationen. Auch Datenmengen im Bereich von Terabytes oder Petabytes können verarbeitet werden. Die Daten werden über mehrere Maschinen verteilt. Es werden größere Dateiformate als bei normalen Distributed File Systems unterstützt. Die Daten sind auch bei Ausfällen zuverlässig geschützt. Fällt ein Server aus, übernimmt der nächste Sever die Funktion. Das File System skaliert mit der Datenmenge. Werden höhere Datenmengen von vielen Clients angefordert, können weitere Rechner zum Rechnernetz hinzugefügt werden.(vgl.:[Inc15c])

Hardware-Defekte passieren regelmäßig. Ein HDFS kann aus mehr als 1000 Maschinen bestehen, wovon jede ein Teil des File System speichert. Da bei der Anzahl von Maschinen anzunehmenen ist, dass nicht jede Maschine problemlos funktionieren wird, verfügt HDFS dafür, über eine automatische Wiederherstellungsfunktion. Das Hadoop Distributed File System ist ausgerichtet für Batchjobs statt für interaktiven Benutzerzugriff. Der Fokus liegt auf einem hohen Datendurchsatz statt auf einer niedrigen Latenz.(vgl.:[Bor15])

Architektur Im Folgenden wird die Architektur von HDFS beschrieben. Die Architektur ist ausgelegt, für Anwendungsfälle wo Daten einmal ins File System übergeben werden und von dort vielfach ausgelesen werden. Damit die Daten verteilt gespeichert werden können, werden die Daten in Blöcke aufgespalten und auf den Knoten des Hadoop-Clusters verteilt. Für jeder dieser Blöcke werden Kopien auf verschiedenen Servern im Cluster angelegt, um Fehler und Ausfallsicherheit zu garantieren. Das zugehörige Management wird von HDFS übernommen. Der Zugriff auf das HDFS ähnelt Unix. Sowohl Konsolen-Befehle, als auch die Rechteverwaltung der Dateien funktionieren wie bei Unix.

Die Architektur ist eine Master-Slave-Architektur (Siehe Abbildung:1). Die Architektur umfasst dabei verschiedene Komponente, wie Master, Client, NameNode und DataNode. Der Client fordert ein File an vom Server. Der Server ist der Master. Der Master beinhaltet ein NameNode. Das NameNode ist die zentrale Masterkomponente und verwaltet die Metadaten für alle gespeicherten auf allen DataNodes. Im Block Map speichert der Master alle aktuellen Speicherorte der Blöcke von Verzeichnissen und Dateien. Der Master beinhaltet außerdem noch das Journal Log, in dem aktuelle Dateioperationen gespeichert werden. Auf der Grundlage von Block Map und Journal Log, kann der Master Anfragen nach Dateien beantworten. Die DataNodes sind die Slave-Komponenten. Diese sind für die Datenspeicherung zuständig. Soll der HDFS-Cluster vergrößert werden, werden DataNodes hinzugefügt. Der NameNode wird vom Secondary NameNode unterstützt. Der Secondary NameNode entlastet den NameNode beim Zusammenführen und Verarbeiten der Block Map und Journal Log.(vgl.:[Sel15b])

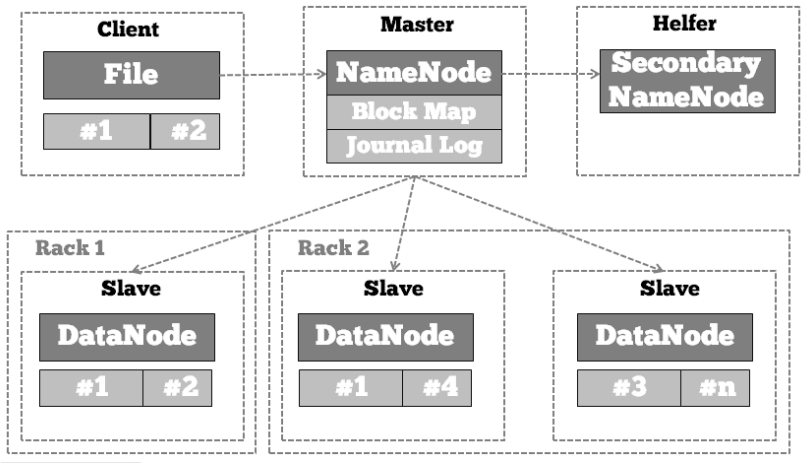

Abbildung 1: Architektur[Sel15a]

2.3. MapReduce Algorithmus

MapReduce ist ein von Google Inc. entwickeltes und eingeführtes Programmiermodell. Es wurde speziell zur nebenläufigen Berechnung über große Datenmengen auf Computerclustern entworfen. MapReduce nutzt dabei mehrere Phasen - Map, Shuffle, Reduce - um die Daten zu verarbeiten. Dabei können zwei der Phasen durch den Anwender parametrisiert werden. So lassen sich die Berechnungen parallelisieren und über mehrere Computer verteilen. In einigen Fällen ist eine Parallelisierung alleine durch die großen Datenmengen unumgänglich, da ein einzelner Prozess mit der Verarbeitung überfordert wäre.
MapReduce nutzt dafür eine verteilte Speicherung der Daten in Blöcken. Dies sorgt für Datenqualität bei fehlerhaften Schreib- oder Lesevorgängen. Des Weiteren lässt sich MapReduce mit handelsüblichen Computer verwenden. Es ist keine spezielle Server Hardware nötig.

Definition der *MapReduce* Funktion

Die *MapReduce* Funktion baut sich aus zwei separaten Abläufen, *Map* und *Reduce*, auf. Der Input wird aus einer Menge unstrukturierten oder semi-strukturierten Daten gebildet. Die Basis der *Map* und *Reduce* Abläufe ist die Bildung von Schlüssel-Wert-Paaren. Es wird aus einer Liste von Schlüssel-Wert-Paraen eine neue Liste von Schlüssel-Wert-Paaren. Die sich daraus ergebenden Formeln lauten wie folgt:

$$MapReduce : (K \times V)^* \rightarrow (L \times W)^*$$
$$[(k_1, v_1) \ldots, (k_n, v_n)] \rightarrow [(l_1, w_1) \ldots (l_n, w_n)]$$

- Die Mengen K und L enthalten Schlüssel, die Mengen V und W enthalten Werte.
- Alle Schlüssel $k \in K$ sind vom gleichen Typ, z. B. Strings.
- Alle Schlüssel $l \in L$ sind vom gleichen Typ, z. B. ganze Zahlen.
- Alle Werte $v \in V$ sind vom gleichen Typ, z. B. Atome.
- Alle Werte $w \in W$ sind vom gleichen Typ, z. B. Gleitkommazahlen.
- Wenn A und B Mengen sind, so ist mit $A \times B$ die Menge aller Paare (a, b) gemeint, wobei $a \in A$ und $b \in B$ (kartesisches Produkt).
- Wenn M eine Menge ist, so ist mit M^* die Menge aller endlichen Listen mit Elementen aus M gemeint (angelehnt an den Kleene-Stern) - die Liste kann auch leer sein.

Die eigentliche Parametrisierung durch den Nutzer wird durch die beiden Funktionen Map und $Reduce$ ermöglicht.

$$Map : K \times V \to (L \times W)^*$$
$$(k, v) \to [(l_1, x_1) \ldots (l_r k, x_{rk})]$$

$$Reduce : L \times W^* \to X^*$$
$$(l, [y_1, \ldots, y_{sl}]) \to [w_1, \ldots, w_{ml}]$$

Map Phase

Die Daten sind am Anfang partitioniert und repliziert im Hadoop Cluster. Jeder Rechner liest die lokalen Daten zeilenweise als Schlüssel-Wert-Paare und übergibt sie dem Map-Task (Implementation der Map Funktion auf einer Maschine). Der Schlüssel dieser Paare ist meistens der Byte-Offset einer Zeile und der Wert ist die Zeile selbst.Die Ausführung der Map Funktion auf jedes gelesene Paar erzeugt ein anderes temporäres Schlüssel-Wert-Paar. Außerdem sortiert der Map-Task seine generierten Paare nach Schlüssel und speichert sie temprär lokal. Es folgt eine Umverteilung der kompletten Ausgaben der Map Phase, indem die temporären Paare mit demselben Key aus allen Map-Tasks zum selben $Reduce$-Task gesendet werden.

Shuffle Phase

Bevor die Reduce Phase starten kann, müssen die Ergebnisse der Map Phase gruppiert werden anhand ihrer Schlüssel. Dazu ist ein koordinierter Datenaustausch nötig. Die Performance der gesamten *MapReduce* Funktion hängt maßgeblich von der Geschwindigkeit des Datenaustausch während der Shuffle Phase ab.

Reduce Phase

Wurden alle Ergebnisse der Map Phase gruppiert werden die Paare des Zwischenergebnisses an den *Reduce*-Task übergeben. Die *Reduce* Funktion erzeugt schließlich ein aggregiertes Paar. Wie auch in der Map Phase können in der Reduce Phase die einzelnen *Reduce*-Tasks über mehrere Rechner innerhalb des Clusters verteilt werden.

3. Fazit

Zusammenfassung Hadoop ist ein Framework, das konzipiert wurde um große Datenmengen schnell verarbeiten zu können. Das Framework besteht aus vielen Komponenten, die bestimmte Funktionen übernehmen. So können intensive Rechenprozesse mit großen Datenmengen von großen Computerclustern. Ein besonderer Bestandteil von Hadoop sind MapReduce und das Hadoop Distributed File System. Diese Unterstützen Hadoop bei seinen Aufgaben.

Ausblick Das Thema ist aktuell. Große Firmen wie Facebook, AOL oder IBM verwenden Hadoop. Da es plattformunabhängig ist, kann es in vielen Szenarien verwendet werden. Die Stärken spielt Hadoop aus, wenn große Datenmengen verarbeitet werden müssen. Hadoop wird besonders interessant im Kontext von Big Data. Unternehmen die große Datenmengen analysieren wollen, können Hadoop verwenden. Ein Unternehmen mit relativ wenig Datenmengen, wird nicht durch einen Einsatz von Hadoop profitieren.

A. Anhang

Literatur

[anG15] av-news GmbH. Wie hadoop einsetzen, 2015.

[Bor15] Dhruba Borthakur. Hdfs architecture guide, 2015. Aufgerufen am: 17.08.2015.

[Inc15a] Wikimedia Foundation Inc. Apache hadoop, 2015. Aufgerufen am: 17.08.2015.

[Inc15b] Wikimedia Foundation Inc. Verteiltes dateisystem, 2015. Aufgerufen am: 17.08.2015.

[Inc15c] Yahoo! Inc. Module 2: The hadoop distributed file system, 2015. Aufgerufen am: 17.08.2015.

[Sel15a] Uwe Seller. Einfuehrung in hadoop - die wichtigsten komponenten von hadoop, 2015. Aufgerufen am: 17.08.2015.

[Sel15b] Uwe Seller. Einfuehrung in hadoop - die wichtigsten komponenten von hadoop, 2015. Aufgerufen am: 17.08.2015.

Abschließende Erklärung

Ich versichere hiermit, dass ich meine Diplomarbeit / Individuelle Projekt ...(Titel der Arbeit)... selbständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlegenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

Oldenburg, den 22. August 2015 Nils Lutz; Janick Asmuß