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

Text

2. Hadoop / Hive

Text

2.1. Hadoop Framework

Text

2.2. Hadoop Distributed File System

Text

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:

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

- 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

// Resümee
// Ausblick

A. Anhang

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