MapReduce

MapReduce ist ein Framework für die Verarbeitung von enormen Datenvolumen durch das Aufteilen der Daten und Berechnungen in hochskalierbaren Systemen, das von Google 2004 vorgestellt wurde. Eine der bekanntesten MapReduce-Implementierungen ist Hadoop, das auch mit seinem Hadoop Distributed File System auch gleich mit einem verteilten Dateisystem daherkommt.

Hintergrund

Die Idee hinter MapReduce ist, eine riesige Datenmenge auf viele Server in einem Cluster aufzuteilen. Da diese einzelnen Rechner nicht nur über eigenen Speicher für die Datenhaltung, sondern auch einen Prozessor für die Datenverarbeitung verfügen, ist es möglich, jeden Rechner parallel Aufgaben auf seinem Teil der Daten ausführen zu lassen, anstatt alles auf einer einzigen Maschine zentralisiert zu berechnen. Die Teilergebnisse werden zum Schluss zu einem Gesamtergebnis zusammengeführt. Dabei besteht ein Cluster, auf dem MapReduce ausgeführt wird, aus einem Master-Knoten, der die Arbeiter koordiniert, über ihren Status Buch führt und bei etwaigen Knotenausfällen Arbeit umverteilt, und einer Reihe von Arbeiter-Knoten, auf denen dann die Map- und Reduce-Funktionen zum Einsatz kommen.

Die Komplexität der Parallelisierung, Kommunikation zwischen den Maschinen oder dem Vorgehen bei Hardwareausfällen bleibt dem Programmierer dabei verborgen. Ein MapReduce-Job besteht aus zwei Phasen, die von Map- und Reduce-Funktionen anderer Sprachen wie LISP inspiriert wurden. (Dean/Ghemwat 2004: S. 1) Beide Phasen kommen dabei nacheinander in einem Cluster parallel zur Ausführung. Der Vorteil bei der Verteilung auf einem Cluster ist, dass auf diese Weise auch Datenmengen verarbeitet werden können, die eine einzelne Maschine überfordern würden. Je mehr Maschinen hinzugefügt werden, desto effizienter kann die Last verteilt werden.

Das MapReduce-Framework sorgt dafür, dass jeder der Map-Aufträge auf einem Knoten arbeitet, der nur seine Daten speichert, dass er also unabhängig von den anderen arbeiten kann. Auf die Weise kommt es zu keinen Dead Locks oder Race Conditions, da keine Ressourcen geteilt werden müssen (Edlich et al. 2010: S. 13). Der Anwender muss also nur die beiden Map- und Reduce-Funktionen schreiben.

Ablauf

Im ersten Schritt eines MapReduce-Jobs werden die Input-Daten in Segmente unterteilt, die auf verschiedene Maschinen verteilt werden. Jede dieser Maschinen verarbeitet ihr Datensegment zu Key-Value-Paaren, die sie dann an die vom Programmierer geschriebene **Map**-Funktion als Input übergeben. Aufgabe der Map-Funktion ist es, eine Liste von Zwischen-Aggregaten (wiederum Key-Value-Paare) zu erstellen, die im Hauptspeicher der jeweiligen Maschine zwischengespeichert, und regelmäßig auf Festplatte persistiert werden. Ist der Map-Prozess abgeschlossen, benachrichtigt der Arbeiter-Knoten den Master-Knoten und übermittelt ihm die Speicherstelle (auf Disk), damit er sie an den Reducer leiten kann.

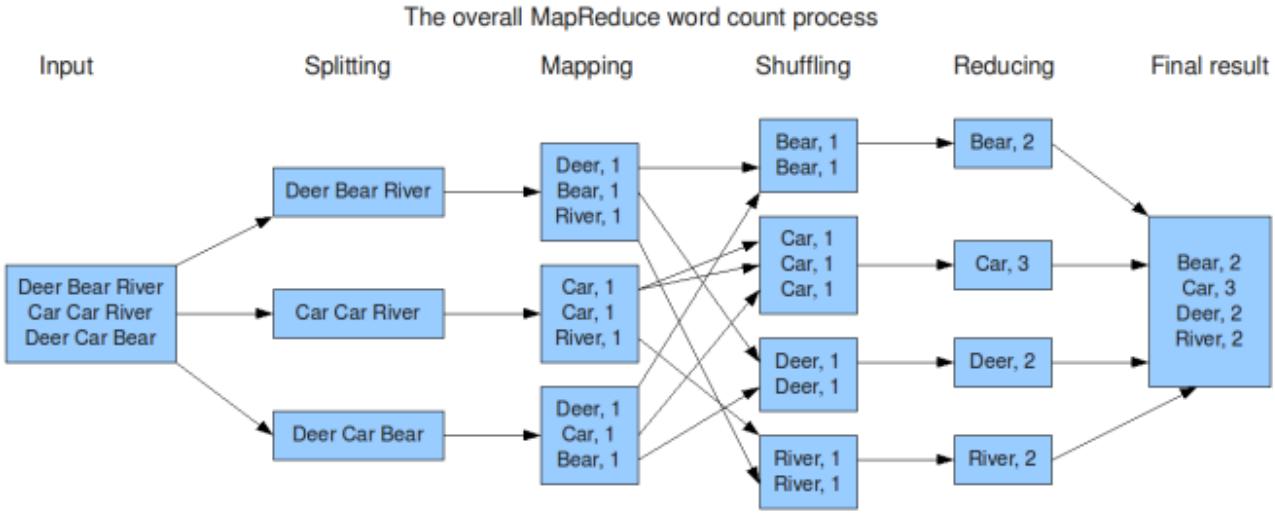
Um möglichst wenige Daten über das Netzwerk senden zu müssen, kann die Ergebnismenge einer Map-Funktion mit einer optionalen **Combiner**-Funktion reduziert werden. Sie nutzt dabei den im

Hauptspeicher befindlichen Mapper-Output als Input (noch bevor dieser als Zwischenergebnis auf Disk gespeichert wird) und rechnet alle Werte desselben Schlüssels zusammen, sodass zu jedem Schlüssel nur noch ein Wert existiert. Der Unterschied zur Reduce-Funktion ist nur der, dass der Combiner-Output als Zwischenergebnis gespeichert wird, der dann an den Reducer gesendet wird, wohingegen der Reducer-Output als Datei im Dateisystem abgelegt wird. (Vgl. Dean/Ghemwat 2004: S. 6) Der Combiner wird manchmal auch als „lokaler Reducer“ oder „mini Reducer“ bezeichnet.

Im sogenannten „**Shuffling**“ werden die Ergebnisdaten der Mapper von den Reducer eingelesen und anschließend nach ihrem Schlüssel sortiert und gruppiert. Der **Reduce**-Funktion werden dann nacheinander ein Schlüssel mit dem Satz seiner zugehörigen Werte zur Verarbeitung übergeben. Die Ausgabe wird dann an ein finales Output-File angehängt. Sind alle Map- und Reduce-Funktionen abgeschlossen, benachrichtigt der Master das Benutzerprogramm. Das Ergebnis liegt dann in den Output-Files der einzelnen Reducer vor.

Beispiel

Ein klassisches Beispiel für einen MapReduce-Einsatz ist das Zählen von Wörtern in einem Großen Text. In diesem Fall würde der Text in Abschnitte unterteilt und auf Maschinen verteilt werden, wo die Abschnitte dann als Input der Map-Funktion dienen würden. Diese würde Schlüssel-Wert-Paare der Art `<Wort, 1>` (die 1 ist hart codiert) erstellen, die anschließend nach ihrem Key, also dem Wort, sortiert würden. Die Reduce-Funktion bekäme die gruppierten Schlüssel-Wert-Paare als Input und würde sie in diesem Falle einfach zählen und in der Art `<Wort, Anzahl>` zusammenfassen und schließlich alle Wort-Anzahl-Paare speichern und ausgeben.



(Bild-Quelle: Groningen 2009)

MapReduce eignet sich also für Probleme, die sich in Unteraufgaben teilen lassen, die durch die Map-Funktion individuell und unabhängig voneinander bearbeitet werden können. Die Ergebnisse der Unteraufgaben müssen sich wiederum unabhängig durch die Reduce-Funktionen zusammentragen lassen. Nur so lassen sich die Aufgaben auf ein großes Rechner-Cluster verteilen und effizient massiv parallel verarbeiten.

From:
https://gpm.wi-wiki.de/ - **Wirtschaftsinformatik Wiki - Kewee**

Permanent link:
https://gpm.wi-wiki.de/doku.php?id=bigdata:mapreduce

Last update: **2015/10/05 20:40**