PS Non-Standard Database Systems

Konrad Medicus
Jakob Schmollgruber
Fabian Oberreiter
2018-13-05

Checkpoint 2

Resources

Jeder Hadoop Job besteht aus einem Map-Reduce Code der entsprechend der Aufgabenstellung zu implementieren ist. Das bedeutet eine DB Abfrage benötigt einen Map und Reduce Code. Es stehen zu diesem Zweck einige *Design Patterns* zu Verfügung. Wir haben uns entschieden die Map-Reduce Scripts in python zu implementieren. Hadoop übergibt Daten mithilfe des Standard Input Streams **STDIN**. Informationen zu diesem Thema haben wir auf der Web-page Dive into Python gefunden. Um Bug-Fixing mit dem Hadoop Environment zu erleichtern war es uns ein Anliegen andere Fehlerquellen ausschließen zu können (wie z.B. Fehler im Map-Reduce Code). Der Map-Reduce Code lässt sich mithilfe von

```
1 python /Users/Jakob/map.py</Users/Jakob/forum_node.tsv>
    mappedData.tsv
2 python /Users/Jakob/red.py</Users/Jakob/mappedData.tsv>
    reducedData.tsv
```

ohne den Einsatz von Hadoop überprüfen. Wir empfehlen Redirection syntax zu besuchen - sehr hilfreich um die Consoleneingabe zu verstehen. Desweiteren finden sich auch dort Infos über Streams. Der Oben ausgeführte Code entspricht einer Hadoop Anfrage mit einem Data-Node. Es empfiehlt sich bei jedem Hadoop Job - auf diese Art (ggf. mit Teilmenge des Datensatzes) den Map-Reduce Code zu überprüfen. Des weiteren gibt die Seite praktischen Einblick wie man eine Hadoop Anfrage mithilfe von Map-Reduce Scripts ausführt. Für einen tiefer gehenden Einblick in das Framework möchte ich das Buch *Hadoop - Zuverlässige, verteilte und skalierbare Big-Data-Anwendungen* von Ramon Wartala[1] nahelegen. Es erklärt sehr einfach das manuelle Set-up eines Hadoop Clusters und gibt einen umfassenden Blick über das gesamte Hadoop Environment. Dieses Buch sollte in unseren Augen dennoch nicht als Einstieg in das Thema Hadoop genutzt werden. Dazu empfehlen wir den **Udacity Kurs Hadoop**. Besonders der Video basierte Unterricht ermöglichen es schnell die Anwendungsmöglichkeiten von Hadoop zu verstehen.

Setup

Da **Cloudera** eine Virtul Machine mit vorinstallierter Hadoop Distribution zu Verfügung stellt ist das Set-up relativ einfach. Cloudera lässt die Wahl zwischen den Plattformen VirtualBox, VMVare, KVM und DockerImage. Die VM lässt sich hier downloaden. Wir haben uns für die Virtual Box entschieden. Die VM heißt QuickStart for CDH 5.13 und steht für persönliche Verwendung und Bildungseinrichtungen frei zu Verfügung. Ein didaktisch hochwertige Beschreibung zur Inbetriebnahme einer VM findet sich hier. Wie bereits erwähnt haben wir uns für das Hadoop Python Streaming (werden wir später genauer erklären) entschieden. Dazu benötigt man einen Python Interpreter. Wir haben uns entschieden Anaconda3 5.1.0. - Linux- x86_64 zu installieren. Es waren zwar keine weiteren Änderungen am Set-up nötig - dennoch möchten wir darauf hinweisen, dass die VM mindestens 8 GB RAM verlangt und wir unseren Computer mit zusätzlichem Arbeitsspeicher ausstatten mussten.

Dataset

Wir verwenden das bereits im Checkpoint 1 beschriebene Dataset. Der von uns gewählte Datensatz stammt aus einem Udacity Forum. Diese Foren laufen auf Basis einer Freeware namens OSQA, welche dem bekannten StackOverflow Forum ähnelt. Die grundlegende Struktur ist die folgende: Formeinträge sind Nodes. Alle Nodes bestehen aus einem Rumpf und der author_id. Top Level Nodes sind Fragen und haben einen Titel sowie Tags. Fragen können zugehörige Antworten haben. Sowohl Fragen als auch Antworten können mit Kommentaren versehen sein. Dieser Datensatz wird von Udacity zur Verfügung gestellt.

- id: id des Nodes
- title: Der Titel des Knoten. Im Fall wenn node_type answer oder comment ist, dann ist dieses Feld leer
- tagnames: Leerzeichen trennen die Liste der Tags
- author_id: id des Autors
- body: Inhalt des Posts
- node_type: Typ des Nodes, entweder Frage- question, Antwort - answer oder Kommentar- comment

Import

Der Import eines Datasets in das HDFS (Hadoop Distributed File System) lässt sich wie folgt bewältigen: Zuerst erstellen wir (wenn gewünscht) mithilfe der Consoleneingabe

```
hadoop fs -mkdir /mynewdir
```

einen neuen Ordner. Mithilfe von von

```
hadoop fs -ls /
```

erfahren wir welche Ordner im root Verzeichnis zu finden sind und können den gewünschten Ordner suchen. Mit

```
hadoop fs -put /home/cloudera/Desktop/forum_node.tsv /mynewdir
```

können wir nun das File 'forum_data.tsv' in das HDFS in den Ordner mynewdir importieren. Der Datensatz wird nun automatisch auf die verschiedenen Data-Nodes aufgeteilt. Wir können mithilfe von

```
hadoop fs -ls /mynewdir
```

sehen, dass unser neu angelegter Order das File 'forum_data.tsv' enthält. Der Import lässt sich unseres Wissens nach nicht verbessern - ggf. bessere Hardware wäre eine Lösung. Da unser File nur 120 MB groß ist, wird es in wenigen Sekunden geladen. Da wir aber nur einen Data-Node zu Verfügung haben, Hadoop daher im stand-alone Betrieb läuft, lässt sich an diesem Punkt nicht viel über das Thema Effizienz sagen. Wir wollen aber im Kapitel *Key System Features* genauer darauf eingehen. Es waren keine Änderungen am Datensatz notwendig. Hadoop ist sehr flexibel in der Datenverarbeitung- die Handhabung von Daten unabhängig ob strukturiert oder nicht ist durch die individuellen Map-Reduce Codes möglich. Unser Datensatz liegt in Tabellenform vor und wird durch Tabs getrennt (genauer: durch eine variierende Anzahl von Leerzeichen- dies dürfte für Standarddatenbankensysteme problematisch sein).
Viele Attribute liegen in einer nicht atomaren Form vor - dies dürfte ebenso den Einsatz von traditionellen System erschweren. Unter der Annahme, dass alle Elemente des Datensatzen in atomarer Form vorliegen könnte man den Datensatz (unter Berücksichtigung der oben genannten Probleme) im Schema Forum_data[id, title,tagnames,author_id,body, ...] speichern. Die zu unseren Aufgabe äquivalente SQL abfrage würde lauten:

```
SELECT COUNT (tagnames)
FROM Forum_data
GROUP BY tagnames
ORDER BY COUNT (tagnames) DESC
```

Implementation

Hadoop erlaubt das Verwenden von map- reduce Code in beliebiger Programmiersprache. Dieses Feature wird vom **Hadoop Streaming Interface** zur Verfügung gestellt - somit ist es für den Einsatz von Hadoop nicht notwendig Java zu erlernen (Hadoop wurde in Java geschrieben). Dies hat natürlich den Vorteil, dass man die Programmiersprache wählen kann, die man am besten beherrscht oder Problemabhängig die Programmiersprache wählen in der sich der Code am einfachsten oder effizientesten schreiben lässt. Wir haben uns entschieden dieses Interface zu nutzen und unsere Map-Reduce Codes in Python zu schreiben. Im Fachgebiet Machine Learning wird Python aufgrund der umfassenden Bibliotheken immer mehr zum gängigen Standard. Da häufig Machine Learning und Big - Data Anwendungen wie Hadoop in einem Atemzug genannt werden, schien es uns sinnvoll einen einheitlichen Weg einzuschlagen.

Im Folgenden beschreiben wir detailliert, wie man einen Hadoop Job umsetzt. Wir werden zu diesem Zweck wie in der Application Description angekündigt die Anfrage -**Was sind die meistgenutzten Tags über alle Postings im Forum?**- bearbeiten.

Wie bereits beschrieben haben wir mit folgendem Code den Ordner *mynewdir* im HDFS erstellt

```
hadoop fs -mkdir /mynewdir
hadoop fs -put /home/cloudera/Desktop/forum_node.tsv /mynewdir
```

und das File *forum_ node.tsv* darin gespeichert.

Um einen Hadoop über das Streaming Interface laufen zu lassen benötigen wir das File *hadoop-streaming-2.6.0-cdh5.13.0.jar*. In Listing 1 können wir sehen wie ein Hadoop Job aussieht.

Listing 1: Hadoop Job

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.13.0.jar
-input /mynewdir/forum_node.tsv \
-output /mynewdir \
-mapper "python /home/cloudera/Desktop/map.py" \
-reducer "python /home/cloudera/Desktop/red.py"
```

Dem Hadoop Streaming Programm müssen Mapper, Reducer, Inputfile und Ausgabeort als Parameter übergeben werden.

Das Python Skript erhält von STDIN Zeilenweise den Datensatz. Im Map Code 'selektieren' wir somit Zeilenweise alle *tagnames* - diese liegen nicht notwendigerweise in atomarer Form vor. Für jeden auf diese Weise gesichteten Tag wird ein Tupple (tag, 1) ausgegeben.

Listing 2: Map Code

```
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
        words = line.split('"')
        if len(words)>5:
                words=words[5]
                words=words.split(' ')
        for word in words:
                print('%s\t%s' % (word, 1))
```

Im Reduce Codes wird nun für alle Tags die Anzahl der Sichtungen adiert. Dies realisieren wir in Python mithilfe eines Dictionaries - man kann diesen Datentyp mit einer Hashtabelle vergleichen. Dann ordnen wir die Tags absteigend entsprechend der Häufigkeit.

Listing 3: Reduce Code

```
import sys
import operator
dictionary={}
# input comes from STDIN (standard input)
for line in sys.stdin:
        word, count = line.split('\t', 1)
        if word in dictionary:
                dictionary[word]=dictionary[word]+1
        else:
                dictionary[word]=1
items = [(v, k) for k, v in dictionary.items()]
items.sort()
items.reverse()              # largest is first
items = [(k, v) for v, k in items]
for keys,values in items:
        print(keys +' , ' + str(values))
```

Das Ergebnis des Hadoop Jops findet sich dann - wenn erfolgreich beendet- im gewählten output Verzeichnis.

Der aufmerksame Leser fragt sich nun warum wir in Map das Tupple (tag,1) ausgeben und nicht sofort die Häufigkeit der tags zählen. Die Antwort ist eine einfache: Dieser Hadoop Jop ist ein rein akademisches Beispiel. In der Praxis würde man schon im Map Code die Häufigkeit der Tags zählen und würde dann im Reduce Code Tupple der Form (tag, Anzahl) von vielen verschiedenen Data-Nodes verarbeiten. Oder in einfachen Worten: Man würde auf jedem Data-Node mit dem Map Code eine Häufigkeitstabelle anlegen und dann mithilfe des Reducer Codes diese Häufigkeitstabellen zu einer allumfassenden Häufigkeitstabelle zusammenfügen. Da wird Hadoop im stand-alone Betrieb verwenden und somit nur ein Data-Node existiert hätte dies hier nicht viel Sinn.

Kernmerkmale von Hadoop

Hadoop, ebenso wie andere verteilte Systeme, besitzt den großen Vorteil, dass die Berechnungen verteilt durchgeführt werden können. Dabei spielen mehrere Aspekte eine große Rolle:

- Verteilter Prozess: Wie bereits angemerkt wird die Berechnung an mehreren Knoten gleichzeitig durchgeführt und somit wird weniger Rechenleistung in jedem einzelnen Knoten benötigt um die gleiche Leistung zu erreichen. Zusätzlich hat es den Vorteil, dass nicht (unbedingt) alle Daten an jedem Knoten vorhanden sein müssen.

- Verteilte Datenspeicherung: Mit dem HDTS (Hadoop Distributed File System) kann man alle Vorteile einer verteilten Datenspeicherung genießen. Dazu gehören etwa die hohe Erreichbarkeit, Zuverlässigkeit und Ausfalltoleranz.

- Neben der verteilten Datenspeicherung trägt dazu auch die mehrfache Speicherung des gleichen Datenblocks bei. Standardmäßig sind dies in einem Hadoop-System 3 Kopien. Damit ist sichergestellt, dass zumindest eine Kopie (ziemlich) sicher erreichbar ist und somit jede Applikation mit den vollständigen Daten durchgeführt werden kann, selbst wenn zu einigen Knoten keine Verbindung hergestellt werden kann.

- Zusätzlich von Vorteil ist, dass Hadoop ein open-source Projekt ist und somit, falls benötigt, auch speziell Anwendungsbezogen angepasst werden kann.

Alternative Implementierung - MongoDB

Eine Alternative zu Hadoop ist die in C++ geschriebene NoSQL-Datenbank MongoDB. Diese ist dokumentenorientiert aufgebaut und kann damit Daten auch in komplexeren Strukturen als typischen Tabellen behandeln. Wenn die gewünschte Information in nicht nur einem Feld liegt, können über Javascript benutzerdefinierte Funktionen für Abfragen erstellt werden. Um die Anfrage aus dem vorherigen Kapitel auszuführen wäre dies vermutlich der beste Weg.

MongoDB	Hadoop
Designed um RDBMS zu ersetzen	Soll RDBMS nicht ersetzten - nur zusätzliche features
Ist wirklich eine Datenbank	Sammlung verschiedener Softwareprodukte- dient als Datenbearbeitungsframework
Ersetzt RDBMS	Größte Stärke: Big Data Verarbeitung
Format: CSV oder JASON	Format: Jedes Verfügbare Format - kann mit strukturierten und unstrukturierten Daten arbeiten

1 Einleitung

Der Sehsinn hilft uns maßgeblich unsere Umgebung wahrzunehmen. Dazu wird visuelle Information auf die Retina, eine mit Lichtsinneszellen bedeckte Region des Auges, projiziert. Elektromagnetische Strahlung wird von Gengenständen, die man betrachtet, reflektiert und mithilfe dieser Zellen in Nervensignale umgewandelt. Das Bild der Umgebung wird in mehrere Teile zerlegt und vom Sehnerv in den visuellen Kortex, einer Gehirnregion, übertragen und dort rekonstruiert, die einzelne Teile werden somit wieder zusammengefügt. Dieser Prozess erfordert eine sehr effizient Umsetzung, daher trifft das Gehirn auf Grundlage von Erfahrungen bestimmte Annahmen, um den Anforderungen gerecht zu werden. So können bei unzureichender Informationslage auch Fehlschlüsse gezogen werden. Das Bild der Umgebung kann aufgrund von Wahrnehmungsfehlern oder invaliden Annahmen falsch rekonstruiert werden. Dies resultiert in einer optischen Täuschung - das von uns wahrgenommene Bild der Umgebung entspricht nicht der Realität. Dieses Phänomen wird in der Wahrnehmungspsychologie untersucht und beschrieben. Ein Bild, das technisch Konstruiert wurde um gezielt eine Sinnestäu-

schung auszulösen wird häufig auch Illusion genannt.[1]

[1] www.simplyscience.ch