2019

MapReduce

ABGABE PVA2 | BDN

TOBIAS FELBER

Inhalt

Aufgabe 1

Aufgabenstellung

Schreiben Sie ein MapReduce-Programm in Python oder Java, welches den Throughput pro Minute bildet. Dabei wird das serverseitige Log (mw_trace50) verwendet. Darin sollen nur die res_snd Record beachtet werden. Das Ziel ist es zu zählen, wie viele res_snd pro Minute anfielen.

Diese Werte sollen als CSV-Datei ausgegeben werden und mit geeigneten Mittel graphisch dargestellt werden.

Vorgehen

Das MapReduce-Programm wurde in Python geschrieben. Dabei gibt es folgende drei Komponenten:

- Mapper:
  Der Mapper liest Zeile für Zeile und filtert als erstes die Logs „res_snd“, da nur diese benötigt werden. Danach gibt er den Timewert gerundet durch 6000 und gerundet als Key zurück. Als Value wird 1 zurückgegeben. Dieser wird für die Bildung der Summe benötigt.
- Combiner:
  Da der Combiner keine Daten vorverarbeiten muss, wird der Code des Reducers verwendet.
- Reducer:
  Der Reducer bildet für jeden Key (also für jeden Timewert) die Summe der Anzahl Datensätze.

Der Code wird mit folgendem Command ausgeführt: *python MapReduce1.py mw_trace50.csv | paste -sd ',' >> outputMapReduce1.csv*

Somit werden die Werte in ein CSV-File geschrieben und können danach mit dem separaten Visualization1.py Skript wieder eingelesen werden, um die Werte graphisch darstellen zu können.

Visualisierung

In der Abbildung 1: Throughput pro Minute sind die Werte graphisch dargestellt.

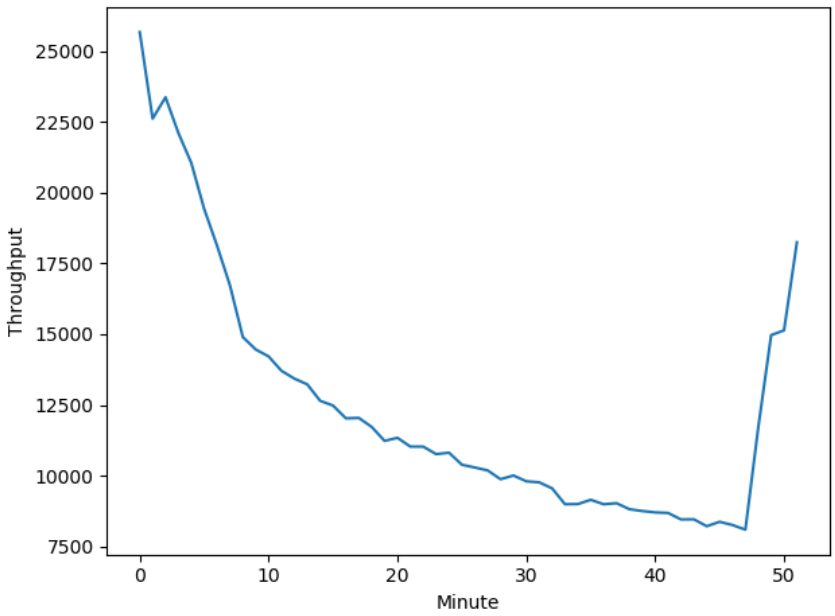

Abbildung 1: Throughput pro Minute

Aufgabe 2

Aufgabenstellung

Schreiben Sie ein MapReduce-Programm in Python oder Java, welches pro Minute den Mittelwert der Responsetime bildet. Dabei wird das clientseitige Log (client_trace50) verwendet.

Diese Werte sollen als CSV-Datei ausgegeben werden und mit geeigneten Mittel graphisch dargestellt werden.

Vorgehen

Das MapReduce Programm wurde in Python geschrieben. Dabei gib es folgende vier Komponenten:

- Mapper:
  Der Mapper liest Zeile für Zeile und filtert als erstes alles nach „msg_snd“ und „res_rcv“. Danach werden alle fehlerhaften Einträge (0 und -1) gefiltert. Als Key wird „client_id“:“loc_ts“ zurückgeben. Dies sind die eindeutige ID des Clients und der lokale Timestamp des Events. Als Value wird die Time zurückgegeben.
- Reducer1:
  Der erste Reducer gibt als Key die Minute zurück und als Value die Differenz des Zeitstempels. Diese Differenz entsteht aus der Subtraktion des tieferen Timewertes vom höheren Timewertes.
- Combiner:
  Der Combiner gibt den erhaltenen Key vom Reducer zurück (volle Minuten). Als Value wird eine Liste der Summe aller Responsetimes und die Anzahl der Responsetimes zurückgegeben.
- Reducer2:
  Der zweite Reducer nimmt diese Liste entgegen und enzippt sie. Dadurch wird die erhaltene Liste zu einer einzigen zusammengelegt. Mit der Division der Summe und der Anzahl Responsetimes entsteht der Mittelwert der Responsetime pro Minute.

Der Code wird mit folgendem Command ausgeführt: *python MapReduce2.py client_trace50 | paste -sd ',' >> outputMapReduce2.csv*

Somit werden die Werte in ein CSV-File geschrieben und können danach mit dem separaten Visualization2.py Skript wieder eingelesen werden, um die Werte graphisch darstellen zu können.

Visualisierung

In der Abbildung 2: Mittelwert der Responsetime pro Minute sind die Werte graphisch dargestellt.

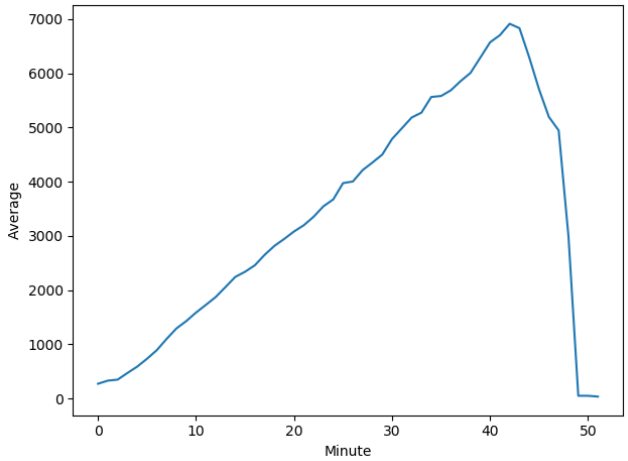

Abbildung 2: Mittelwert der Responsetime pro Minute

Interpretation

Die Daten bzw. die beiden Graphen zeigen das erwartete Resultat. Umso höher der Throughput ist, desto mehr Request kann der Server abarbeiten. Deshalb ist dementsprechend auch die Responsetime niedrig. Genau umgekehrt verhält es sich, wenn der Throughput tief ist. Der Server kann deutlich weniger Request abarbeiten, was zu einer höheren Responsetime führt.

Quellen

https://benjamincongdon.me/blog/2018/02/02/MapReduce-on-Python-is-better-with-MRJob-and-EMR/

https://www.edureka.co/blog/hadoop-streaming-mapreduce-program/

https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py

https://docs.python.org/2/library/math.html