Report

Yang Xia

2017/11/13

Specifications of host execution Environment

Attribute	Value
Java Version	1.8.0_102
Model Identifier	MacBookPro13,2
Processor Name	Intel Core i5
Processor Speed	2.2 GHz
Number of Processors	1
Total Number of Cores	4
Memory	16 GB
Driver Memory	2 GB
Executor Memory	2 GB

Summary of the design

The implementation start with reading the data. I had a class called "SongInfo" which store the related information as fileds. This makes the code cleaner. I didn't do the so-called clean-up test before hand due to the fact that this might cause inaccuracy in the result. The milionSong object is the entry point to the program that taken in song_info.csv and artist_terms.csv locations in the input file and calls the methods in the SongInfo class to get the desired output.

Result Analysis

Assumptions

Even if one of the cell is empty or doesn't have the correct value, the other part of the record is invalid.

For the whole dataset:

Number of distinct songs: 999056

Number of distinct artists: 44745

Number of distinct albums: 149275

Top 5 loudest songs:

(Assumption: a larger loudness implies a louder song)

Song Id	Song Name	Loudness Score
SOCMYZF12AB0186FF4	Modifications	4.318
SONHDXQ12AB018C1F1	Track 02	4.3
SOZCIHV12AC46894E3	Hey You Fuxxx!	4.231
SOEJMJF12AC90715D1	War Memorial Exit	4.166
SOBEMPS12A8C13BD46	Meta Abuse	4.15

Top 5 longest songs:

(Assumption: a larger duration implies a longer song)

Song Id	Song Name	Duration
SOOUBST12AC90977B6	Grounation	3034.90567
SOXUCQN12A6D4FC451	Raag - Shuddha KalyaN	3033.5995
SOOMVZJ12AB01878EB	Discussion 2	3033.44281
SOTNVEE12A8C13F470	Chapitre Un (a): Toutes Les Histoires	3032.76363
SOGFXNB12A8C137BE5	Der Geist des Llano Estacado Ein Spion	3032.58077

Top 5 fastest songs:

(Assumption: a larger tempo implies a faster song)

Song Id	Song Name	Tempo Score
SOVVTEZ12AB0184AAB	Beep Beep	302.3
SOMSJWX12AB017DB99	Late Nite Lounge: WVIP	296.469
SOUTBKH12A8C136286	A Place Called Hope	285.157
SOEVQJB12AC960DA2C	Bellas Lullaby - Perrier Citron	284.208
SOTUXOB12AB0188C3A	Troubled Times	282.573

Top 5 most familiar artists:

Artist Name	Artist Id	Familiarity Score
Akon	ARCGJ6U1187FB4D01F	1.0
Akon_ San Quinn_ JT the Bigga Figga	ARCGJ6U1187FB4D01F	1.0
Akon / Eminem	ARCGJ6U1187FB4D01F	1.0
Akon / Styles P	ARCGJ6U1187FB4D01F	1.0
Akon / Wyclef Jean	ARCGJ6U1187FB4D01F	1.0

Top 5 hottest artists:

Artist Name	Artist Id	Hotness Score
Daft Punk	ARF8HTQ1187B9AE693	0.997066533839045
Daft Punk	ARF8HTQ1187B9AE693	0.997004803235357
Black Eyed Peas	ARTDQRC1187FB4EFD4	0.982623202516712
Kanye West	ARRH63Y1187FB47783	0.972399563931911
Kanye West / Jamie Foxx	ARRH63Y1187FB47783	0.972399563931911

Top 5 hottest songs:

Song Id	Song Name	Hotness Score
SONMVZB12AB01829BA	Der Maggot Tango	0.521321041187445
SOAGFNE12A8C134D82	Donde Caigo	0.454192988218022
SOIUPEW12A8C13BCB2	Willow Weep For Me (Live)	0.266955186275539
SOAYGMO12A6D4F69E8	I’d Have Never Found Somebody New	0.249065794853703
SOFJZNE12A8AE45C1F	The??blind Walk Over The Edge	0.492398352817721

Top 5 hottest genres (mean artists hotness in artist_term)

Genres	Mean Hotness
christmas song	0.6084021
kotekote	0.6022205
female artist	0.5753481
girl rockers	0.5737886
alternative latin	0.5737588

Top 5 Prolific artists:

Artist Id	Artist Name	Number of songs by the artist
AR6681Y1187FB39B02	Ike & Tina Turner	208
ARXPPEY1187FB51DF4	Michael Jackson	204
ARH861H1187B9B799E	Johnny Cash	201
AR8L6W21187B9AD317	Diana Ross & The Supremes	196
ARLHO5Z1187FB4C861	Beastie Boys	194

Top 5 most popular keys (must have confidence > 0.7):

Key	Ocurrances
7	53144
9	46855
2	46431
4	35099
11	35078

Top 5 most common words in song titles (excluding articles, prepositions, conjunctions):

(Assumption: (ablum version) doesn't count as part of the title)

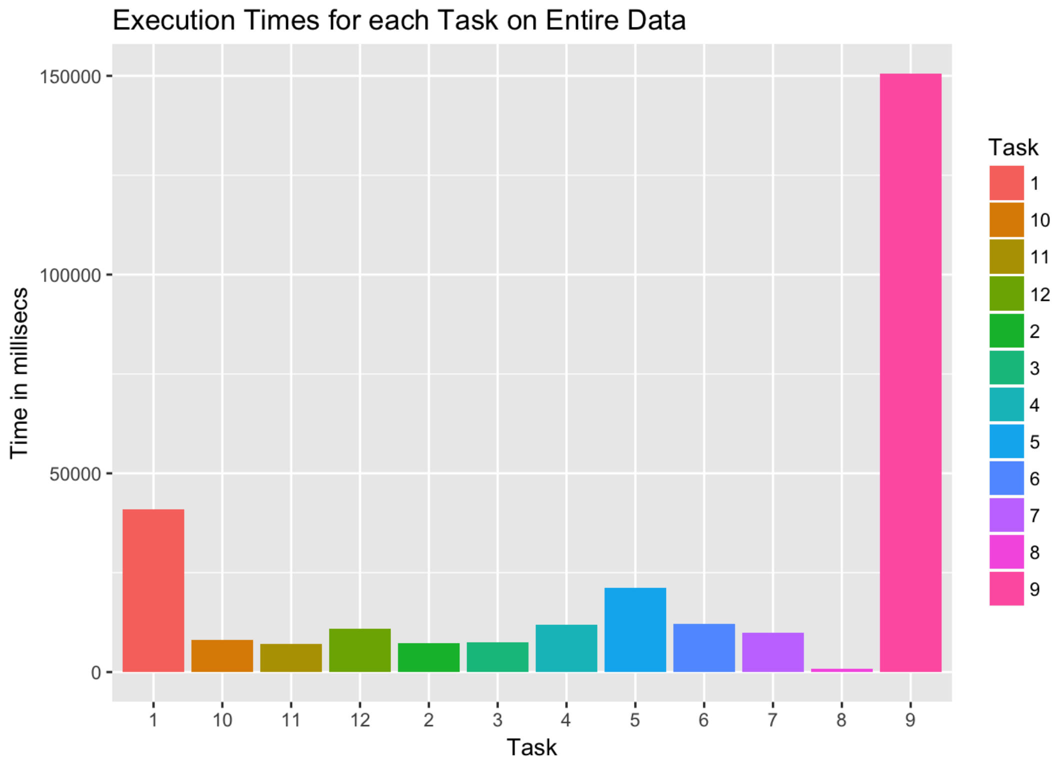
Word	Ocurrances
LOVE	9882
LIVE	6192
YOUR	5333
NO	4822
REMIX	4462

Performance Analysis

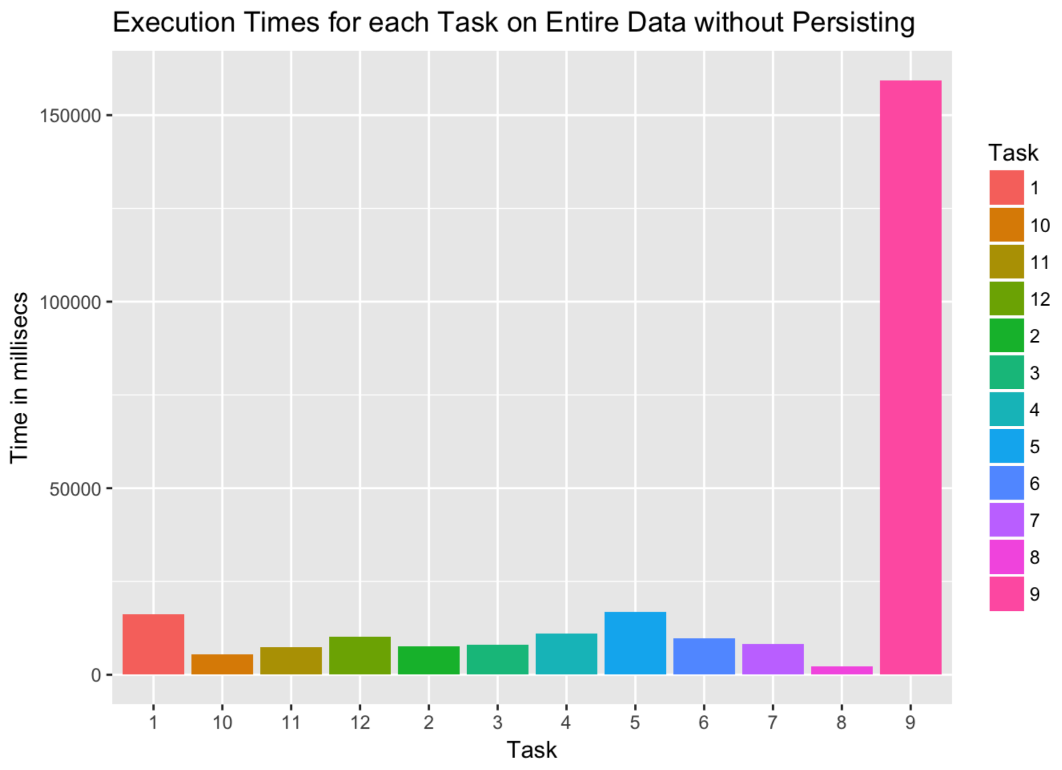
I run the programm on the whole data set a couple of times, and in average it takes around 3 minutes to run on the whole dataset. Instead of ploting the volin graph for the total runing time, I think it's more intersting to look at the time spent on every query, and compare the performance with and withour persist.

Times taken for different operations on **entire data with persist**:

```
##    Task              Query performed
## 1     1          Find distinct songs
## 2     2        Find distinct artists
## 3     3         Find distinct albums
## 4     4     Find top 5 loudest songs
## 5     5     Find top 5 longest songs
## 6     6     Find top 5 fastest songs
## 7     7     Find top 5 hottest songs
## 8     8  Find top 5 familiar artists
## 9     9    Find top 5 hottest artists
## 10   10     Find top 5 hottest Genres
## 11   11 Find top 5 most popular keys
## 12   12  Find top 5 prolific artists
## 13   13       Find top 5 common words
```



####Times taken for different operations on the **entire data without persist**:



I ran the code on the subset as well as the entire dataset. Two things that are interesting. First, We can see that for majority of the tasks, the execution time is faster when we do not persist the RDD and keep generating it on the fly. This is because we are running our code in a standalone mode rather than in a distributed mode. Second, the “hottest genres” query takes up most of the time. I think it’s because a joining two is very expensive. Since it’s a bottleneck in this program, we can try to optimize it in future work.

Conclusion

This program takes million songs data as input and do some simple analysis on the whole dataset. Using Spark instead of Hadoop saves a lot of effort. Though I’ve never implemented in hadoop, I can imagine doing the same things in Hadoop require many map job and reduce job, which in most of the cases are not as efficient as I did here with spark. The bottleneck here is the join operation; some optimization could be done to improve the performance. In a stand-alone mod, persist does not give me the speed-up as it should be while in a distributed mode.