Report for the F-Secure Homework

Narges Yousefnezhad

August 10, 2016

Justification for language/environment selection

To select a language among several possible candidates to do a project, we should first evaluate the data set and the requirements to solve the problem e.g. computational limits, hardware resources, etc. The input data set used in this homework is quite large and classified as big data which makes it almost impossible (at least for my laptop!!) to analyze it incrementally. So, another alternatives are: (I) to analyze the data in a streaming mode, or (II) to use parallelization. Since the second alternative is not the case of this homework (according to the instructions), the first choice is selected. I decided to use Apache Spark as the environment for doing the homework. It is an open source framework built around speed, ease of use and analytics mainly for big data analysis. Spark can handle programs in different languages including Scala, Java, Python and R, where among them, Scala is the best choice for big data analysis. Although, I was not fluent in Scala and processing the data with Python was much more easier for me (I'm fluent in Python programing), I chose Scala for event processing since according to my search it is a powerful tool in analyzing big data/projects. To sum up, I used Spark as the framework for programing and Scala as the language and I did the programing in a Linux OS.

In the Spark environment, first I converted the data with json format to a table using SQLContext, and then I use some Spark SQL queries in Scala for answering each question. Here is the code:

```
    val sqlContext = new org.apache.spark.sql.SQLContext(sc)
    val events = sqlContext.jsonFile("obfuscated_data")
    events.registerTempTable("events")

```

Listing 1: Code_1

Question 1: Detecting the number of times a particular product has been launched during the time period

```
    val query1= sqlContext.sql("SELECT COUNT(source) FROM events WHERE source='"+
    product_name+"' and type = 'launch' ")

```

Listing 2: Code_2

I checked the number of launches for all four products (a,b,c,d). Table 1 shows the results.

product-a	11932
product-b	168859
product-c	16690
product-d	223

Table 1: The number of times each product launched

Question 1: The number of first time launches

In this question user can specific the name of the product.

```
val query12= sqlContext.sql("SELECT COUNT(DISTINCT device.device_id) from events
  WHERE type='launch'")
```

Listing 3: Code_3

Here, I checked the number of devices which create lunch events. Based on this query, there are 36532 firs-time launches in this data set.

Question 2: Detecting duplicate events

If there be two or more records with same event_id, they are duplicate.

```
 val query2= sqlContext.sql("SELECT event_id , COUNT(event_id) FROM events GROUP BY
  event_id HAVING count(event_id)>1")
```

Listing 4: Code_3

Question 3: Observing timestamp

```
   val query3= sqlContext.sql("SELECT type,source FROM events WHERE time.
  create_timestamp>sender_info.received_timestamp" )
```

Listing 5: Code_3

According to timestamps, some events have been received before they were created (time.create_timestamp > sender_info.received_timestamp).

Question 4: Interesting statistics

There could be several interesting statistics among which I beleive the number of times a specific product has been processed in each geographical location would be one of the most important ones since it can help managers to detect the best products to advertise in each location.

```
   val query4= sqlContext.sql("SELECT sender_info.geo.country,COUNT(*) FROM events
  WHERE source='product-a' GROUP BY sender_info.geo.country ")
```

Listing 6: Code_3

Here is the output of the code: Array([LK,25], [LT,46], [LV,301], [SE,2556], [FI,47069], [SG,604], [SI,41], [SK,75], [FO,42], [FR,1154], [MA,16], [ZA,5], [SR,34], [ME,6], [SV,1], [A1,3], [GB,2560], [MT,10], [GG,1], [MV,8], [MX,220], [MY,4015], [TH,29], [GR,437], [AE,5], [GT,183], [TR,19], [AO,23], [NL,3257],...

Also the number of network carriers used by each device could be important.

```
  val query42= sqlContext.sql("SELECT device.operating_system.kind,COUNT(*) FROM
  events GROUP BY device.operating_system.kind")
```

Listing 7: Code_3

Here is the output: [ios,97599] , [android,368174] , [osx,13362] ,[windows,1036719] ,[web,5]
The number of times each product is purchased/launched could also be considered as an important statistics since using that the managers could decide to invest on which of the products more. This is coded in question 1.

Question 5: Storing data for further easier processing

In order to make the processing of the data easier, it would be nice to store them based on one of the followings:

- Source: since we have 4 product, one of the best cases to store data would be to have 4 files each containing the events related to one of the products. In this setting, we can easily process each product and find which of them are launched/purchased more. With this setting, for each product we can easily decide managerial decision
- Geographical location: with this setting we could have one file for each geographical location which helps us in detecting the most favorite product for the corresponding place, e.g. in each country which product has more requests.
- Event type: With this type of storing, it is possible to find the future market trend by evaluating the statistics of purchased products and the products which are launched the most.

Question 6: Best time for maintenance

We should find in which time of this period, we have the lowest number of events happening. Also among the events, those which has the type of purchase should have higher priority. Unfortunately, because I was very busy during the period I was working on the project, I did not code this question.

Question 7: Detecting device with longest activity time

First, I calculate the activity time for each device. Then I sorted the device ids based on the activity time. Finally I select the device with longest activity time.

```
    val query7= sqlContext.sql("SELECT r.dev_id ,r.activity_time FROM (SELECT device.device_id as dev_id ,MAX(time.create_timestamp)−MIN(time.create_timestamp) as activity_time FROM events GROUP BY device.device_id) r ORDER BY r.activity_time DESC limit 1")

```

Listing 8: Code_3

In this dataset, device with id= 43138fb09eb72a68108ab255b27b49fb9174d70d has the longest activity time (1436565681121).