Assignment 2 Report

Name: Xinyu Jiang, Guichi Zhao
Student ID: 440042680, 420178745

Hive

The hive implementation is a mixture of HiveSQL and Mapreduce streaming in Python

```
from
(map owner,date_taken,place_url
using 'mapper.py'
as owner,country,duration
from
        (
select photo_table.owner,photo_table.date_taken,place_table.place_url
        from
                (select place_id,place_url
                from share.place)place_table
                join
                (select owner,date_taken,place_id
                from share.photo)photo_table
                on place_table.place_id=photo_table.place_id
        cluster by photo_table.owner,photo_table.date_taken
)user_data_place
)owner_date_duration
reduce owner,country,duration
using 'reducer.py'
as owner,record
;
```

HiveSQL :

It got [place_id place_url] from place and [owner date_taken place_id] from photo and join them on place_id ,after that distribute and sort the result based on owner and date_taken.As a result,same user will go to the same partation and be grouped together,in addtion date_taken for the same owner group will be sorted chronologically.

The intermediate result is something as follow:

```
10530649@N05 2007-04-19 12:31:34     /Australia/QLD/Bundaberg/Kensington
10530649@N05 2007-05-23 08:08:27     /Australia/QLD/Cairns/Brinsmead
10530649@N05 2008-03-24 09:04:52     /Brazil/Maranhao/Primeira+Cruz
10530649@N05 2008-04-02 11:23:03     /Brazil/Espirito+Santo/Anchieta
10530649@N05 2008-04-10 16:33:25     /Brazil/Espirito+Santo/Guarapari
10530649@N05 2008-04-20 15:17:43     /United+Kingdom/Scotland/Spean+Bridge
10530649@N05 2008-05-06 13:24:03     /United+Kingdom/Scotland/Eshaness
10530649@N05 2009-10-23 11:00:16     /United+States/California/Cayucos
10530649@N05 2009-10-25 08:21:44     /United+States/California/Mill+Valley
10530649@N05 2009-12-10 20:53:56     /Australia/SA/Adelaide/Eden+Hills
10530649@N05 2009-12-14 10:11:31     /Australia/SA/Whyalla/Whyalla+Norrie
12037949631@N01 2006-12-31 23:35:30  /United+States/Gold+Mine+Hill
```

Python Streaming:

Map

Firstly ,the above result will be mapped by mapper.py

the mapper will hold a variable “last_user” and walk through each record,when the user in current record not equal to the “last_user”,it is regards a new user.In this way,we can traverse information for each user.Simolarly,we hold variable last_country,last_start_time,last_end time.last_country is used to identify new country while last_start_time and last_end_time is used to calculate duration for each country.Note that the first record and last record in a group need some special consideration,refer to the code for details.

pipeline result from last stage to mapper (testInput is result produced by hiveSQL as displayed above):

```
$ cat testInput | python mapper/mapper.py
```

will produce:

```
10530649@N05      Australia       339.9
10530649@N05      Brazil  27.3
10530649@N05      United+Kingdom  550.9
10530649@N05      United+States   48.5
10530649@N05      Australia       3.6
12037949631@N01   United+States   0.0
```

Reduce:

The above result will them feed into reducer to get the desired format

in reducer.py .We hold a dictioary whose key is country and value is list of duration for the country .It is easy to produce the desired result with the help of the dictionary

```
$ cat testInput | python mapper/mapper.py | python reducer/reducer.py
```

produce:

```
51035784065@N01 United+States(1,9.9,9.9,9.9,9.9)
10530649@N05    Brazil(1,27.3,27.3,27.3,27.3),Australia(2,339.9,3.6,171.8,343.5)
,United+Kingdom(1,550.9,550.9,550.9,550.9),United+States(1,48.5,48.5,48.5,48.5)
12037949631@N01 United+States(1,0.0,0.0,0.0,0.0)
```

Originally,we try to code everything in HiveSQL

The problem is there is no obvious way to compute value between different records,if possible,more stages may be introduced compared to python streaming.So we believe python streaming is a better option.

Spark Design

In our Spark application, there are 4 stages and 9 tasks in total.

- The first stage is to extract the useful information from the place file.

  1. The **task: photoExtraction** uses mapToPair transformation and return a key/value pair: PlaceId -> userId, date-taken.

- The second stage is to extract the useful information from the photo file.

  2. The **task: placeExtraction** uses mapToPair transformation and return a key/value pair: placeId -> country.

- The third stage is to process the key/value pairs generate in the first and second stage.

  3. The **task: joinResults**, uses join transformation to join the key/value pairs in stage one and stage two. And the result key/value pair is placeId -> userId dateTaken, url.

  Next, the operation**: tempSampleData,** including two tasks:

  4.first **task: joinResults.value** which gets the value from the key/value pair generated from joinResults.

  5.Second **task: joinResults.value.mapToPair** generates a key/value pair as userId date-taken-> url.

  6. Then, **task sampleData** transforms the key/value pair to userId-> date-Taken country.

  7. Final **task: userValueList** in this stage, performs groupByKey transformation, and the key/value pair is userId -> Iterable(date-taken country).

- The fourth stage, is to compute how many times a user has visited a particular country, as well as the maximum, minimum, average and total time that he has spent there.

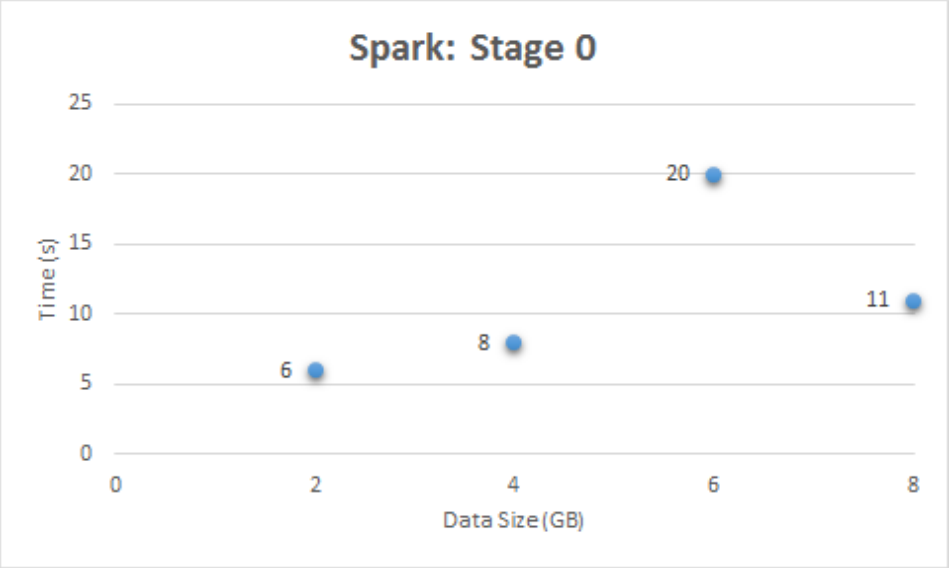
  8. **task computeTimeSpent** uses mapToPair to transform the key/value pair as: userId-> country (timeVisist, maxDuration, minDuration, average, total),

  9. **task computeTimeSpent.saveAsTextFile** materialize the result in previous data by saving it to the disk.

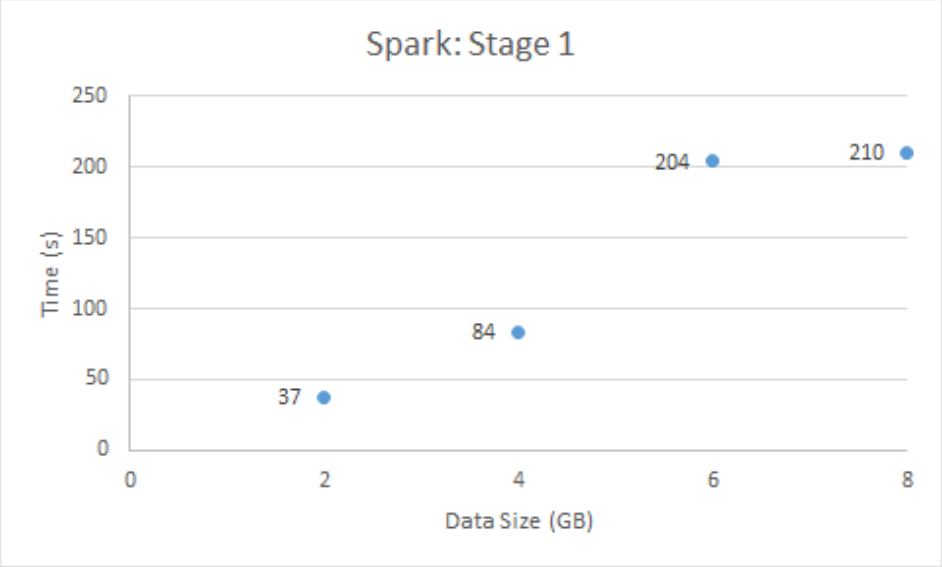
Performance Analysis

Spark:

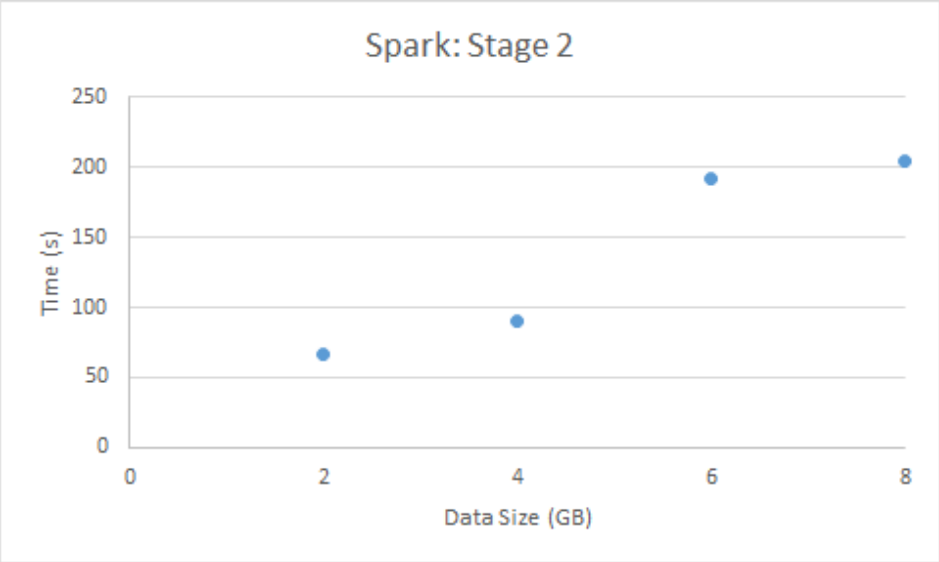
Stage 0- Data Size vs. Time Comparison



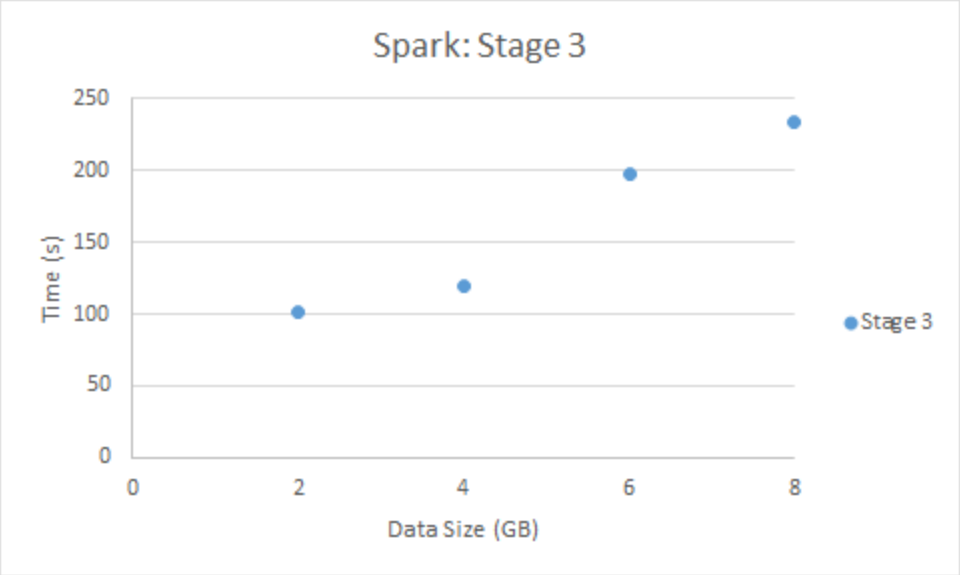
Stage 1- Data Size vs. Time Comparison



Stage 2- Data Size vs. Time Comparison

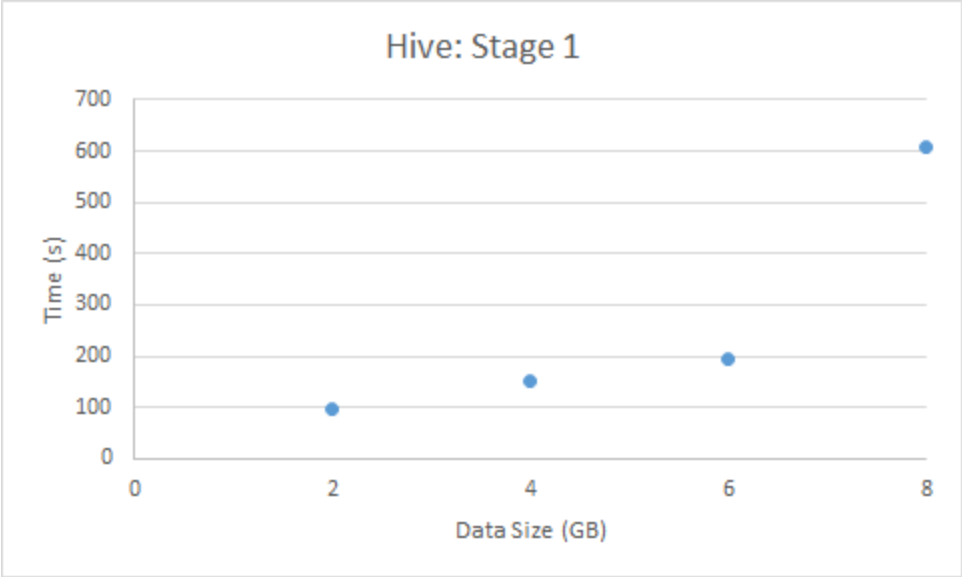


Stage 3- Data Size vs. Time Comparison

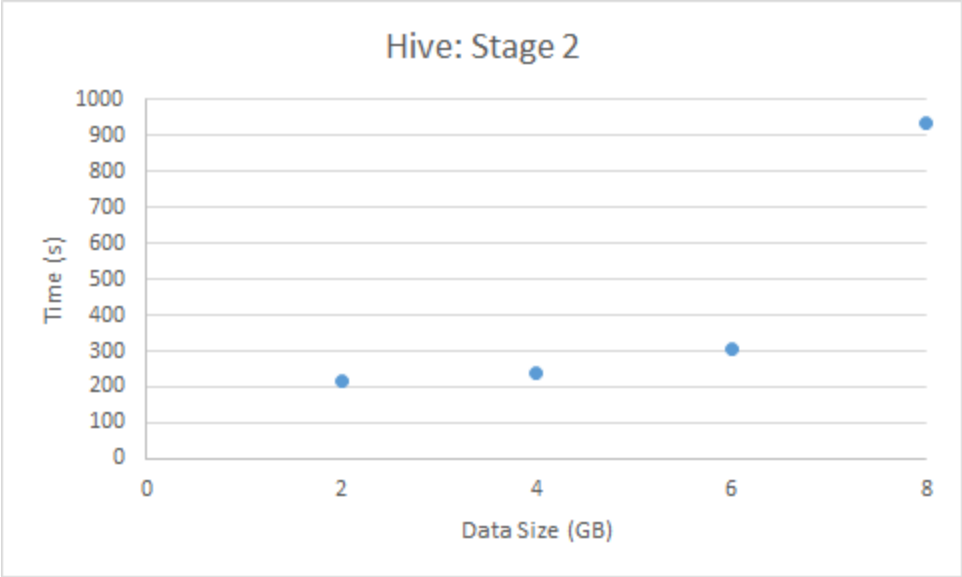


Hive:

Stage 1- Data Size vs. Time Comparison



Stage 2- Data Size vs. Time Comparison



Hive vs Spark Performance Comparison