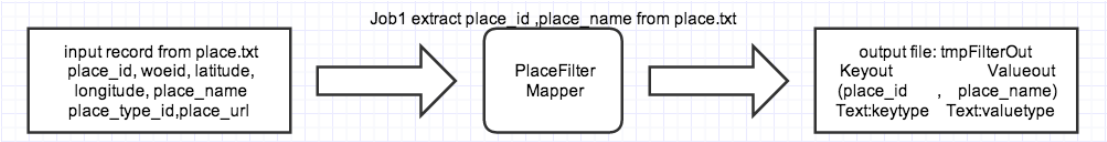
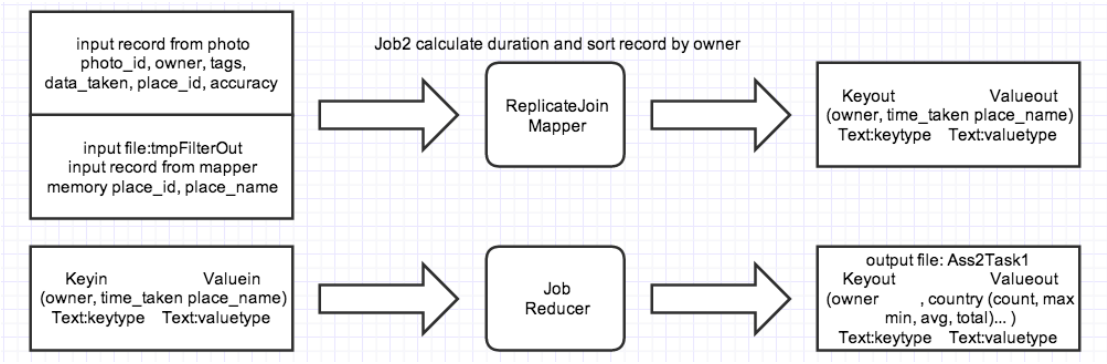
MapReduce Job Design

In MapReduce implementation, two mappers (PlaceFilterMapper and ReplicateJoinMapper) and one reducer (JobReducer) are used to generate the final result. In job1, we used PlaceFilterMapper to extract place_id and palce_name from input record place.txt.

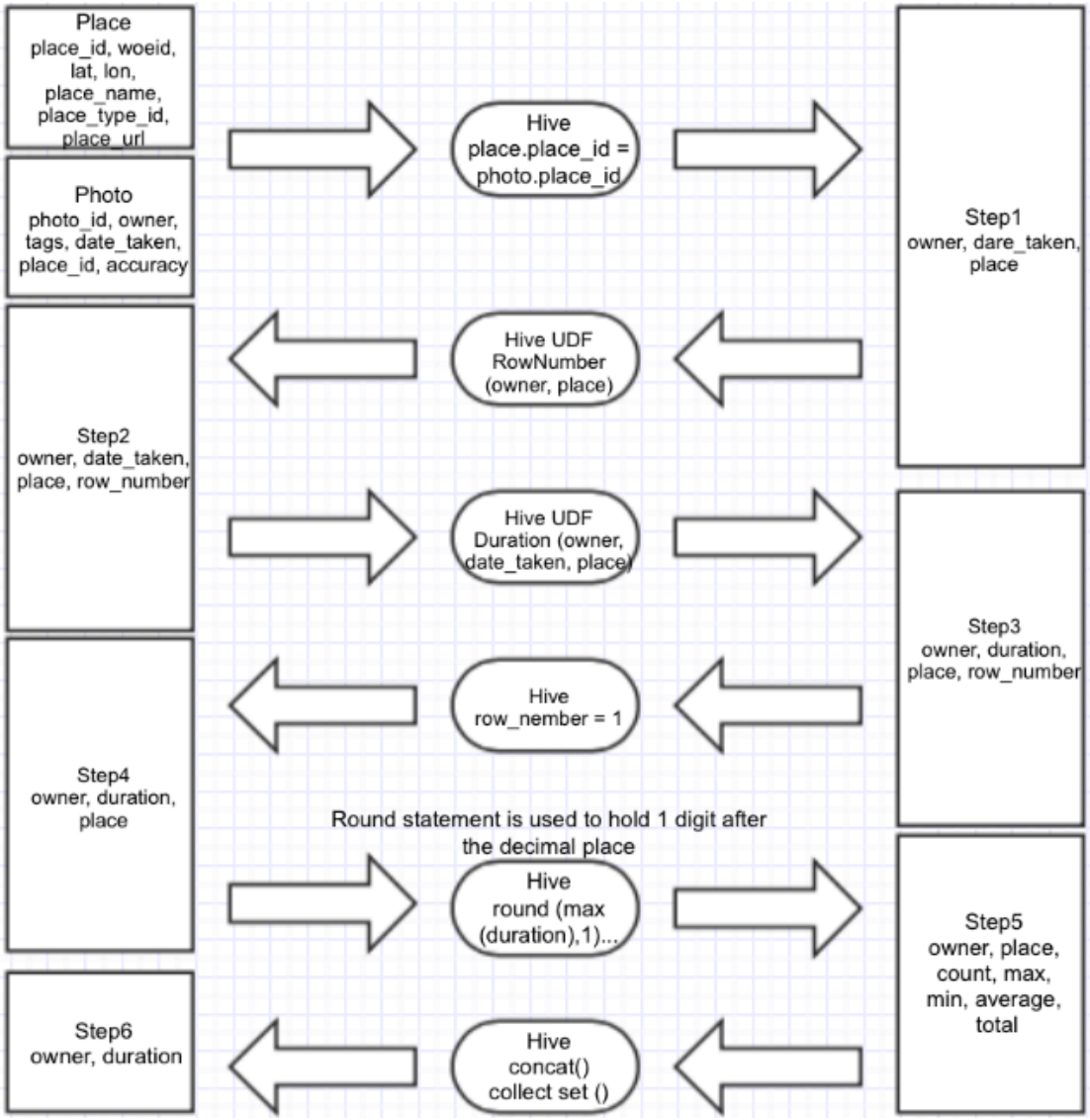


In the job2, first used the ReplicateJoinMapper to extract the owner,date_taken and place_name from both input record of tmpFilterOut created by first job and photo. Then used JobReducer to generate final result. There are two classes (duration and DateComparator) used in this step. Class DateComparator used to sort the owner, date and country in ascending order based on the date_taken. Then used duration class to count how many times a user has visited a particular country as well as the maximum, minimal, average and total time user spend in the corresponding countries. Finally, output the result in the format: "owner country (times, max, min, average, total)….".



HIVE Design

In HIVE implementation, two UDF's temporary functions (RowNumber and Duration) and a number of HIVE queries are used to generate the final results. The function Rownumber is used to sort (Descending) and attach label (such as 1, 2, 3…) to every records according to date_taken and country. The function Duration is used to calculate the duration that a user stays at different countries. Also there are 6 step involved to generate the final results. Each step uses HIVE queries to pick up data, which is used for next step.



By using the HIVE queries, we are able to easily extract, transform and load large datasets stored in hadoop without the need to write map-reduce programs. For some special need, we can implement it by using UDF.

Evaluation

Number of MapReduce jobs

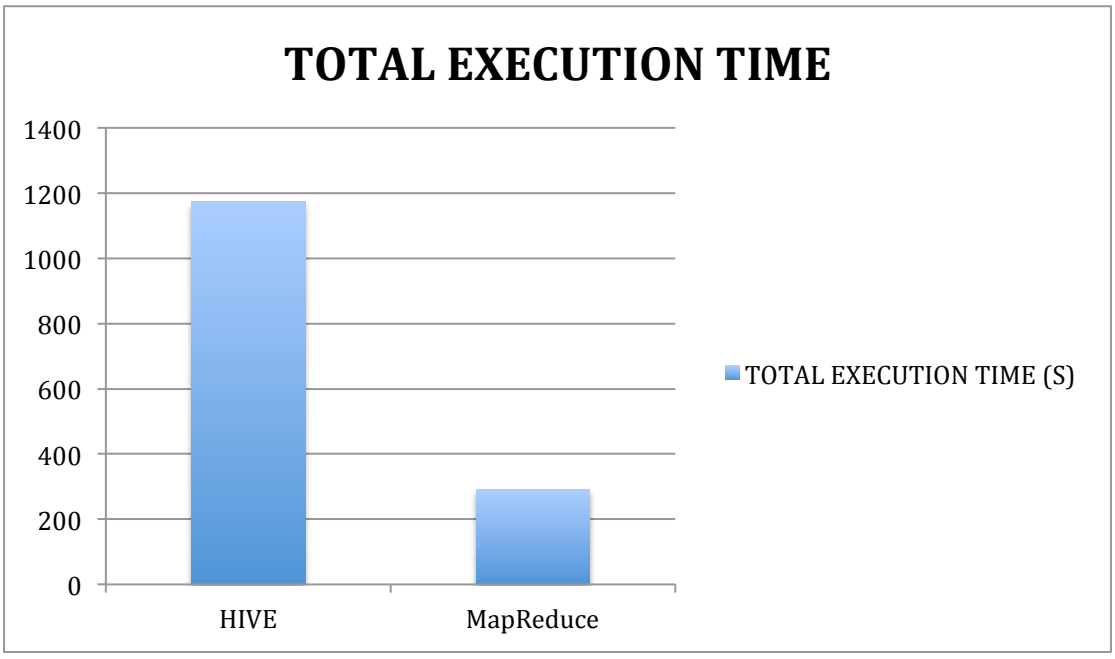
In MapReduce implementation, there are two MapReduce jobs are involved in order to generate the final results, which include two mappers and one reducer.

In HIVE implementation, there are 11 MapReduce jobs are involved to generate the final results.

Specifically, two mappers are used to extract the owner, date_taken and country from photo record and place record in MapReduce implementation, then one reducer is used to calculate the duration and sort every record in order to generate the final results. On the other hand, HIVE implementation has 6 steps and each step use one or two queries to implement it. Also, each query runs one to three MapReduce jobs to extract data from data resource. There are two HIVE UDF's temporary functions involved in step2 and step 3, which include 7 mappers and 3 reducers.

After comparing to the number of MapReduce job in two implementations, we can easily find the number of job in HIVE implementation (11) is much more than the number of job in MapReduce implementation (2). The reason is every HIVE query would deploy a number of mappers and reducers to finish data extraction, and then those mappers and reducers would constitute a job, which means the more queries we used, the more jobs we get. In another word, we can determine how many MR jobs are used to generate the result, but for the HIVE implementation, we cannot determine how many jobs would be used, because the number of MR jobs is generated automatically. The only thing we can do is to determine how many queries are used.

Execution time



According to the graph, which compare to the execution time in two implementations, we can easily find that the HIVE implementation spend more

time to extract data user need. The reason is HIVE implementation have more jobs than MapReduce implementation. What's more, HIVE is easy for joining multiple data source and is suitable for adhoc analysis. But HIVE only deal with structured data and not easy to accomplish complex business logic. On the other hand, MapReduce is good at working on both structured and unstructured data. Also it is good for writing complex business logic. But it is hard to achieve join functionality.

Appendix

Implementation 1: MapReducer
/user/lzha4956/Ass2Task1--small
/user/lzha4956/Ass2Task1-v4

Implementation 2: HIVE
/user/lzha4956/Ass2Task2
/user/lzha4956/Ass2Task2--small