Exercise Sheet 2, 2021

6.0 VU Advanced Database Systems

Mal Kurteshi (11924480)

25. April 2021

Exercise 1 (Costs of MapReduce) *[1 point]*

(a) Based on the fact how map reduce jobs produce their results and the logic behind the mapper and the reducer, if we don't use a combiner we can expect that there would be a skew in the times taken by different reducers to process their value lists. As we know from the lecture, every map reduce job function produces a pair of type (k,1) which after some sorting or shuffling may end with keys that have huge lists which would likely cause skew when taken from the reducer to produce to output.

(b) In the word frequency example we see that more frequent words take longer to be processed by the reducer, this is happening due to fact that the principle of Map reduce is to separate tasks based on keys, so there are cases as mentined in a) that there can be keys that take more time to be processed in our case more frequent words, so even if you combine reducers the same situation will occur when some reducers need more time than others.

(c) Yes it makes sense because as we know when we deal with combiner function the idea is to use some reduce in the map task. In this matter we are reducing the communication costs since we are reducing the number of key and value pairs which are given from mapper to the reducers.

(d) The communication cost is the total number of input output actions of map and reduce tasks. In this case we do not expect to have a increase of communication cost because we have the same amount of input just partitioned into a number of pieces. As for the replication rate, it is the average number of key value pairs that the mapper create from each input. Starting from the fact that the replication rate is not dependent on the number of reducer, we can expect the communication cost not to change. Even with increase of number of reducers, the case of the reduces size which is based on individual keys and it max size of value list of one key, it will not have any difference because taking in consideration the fact that the reducer can receive a key with big value list size. .

Exercise 2 (Relational Operations) *[2 points]*

(a) Mapper: For each input in F1 we produce a tuple as of (R,S), which is taken from the right outer join first performed in the relations R and S.

Reducer: In the reducer we take the output from mapper and then we make the conditions that difference in C and B should not be equal to 32 and B should be grater and equal than D in case there fulfill we emit tuple (R,S) and take the output which will be also a tuple lets say (R',S').

Communication cost: The mapper emits from every F1 a tuple (R,S), therefore the communication cost would be the $|R| + |S| - |r \in R, |C - D| < 32| - |B \in S, B \neq 0|$

(b) Mapper: In this case we have three variables consisting on the three relations T, U and V. In the mapper we are receiving as an input three elements (T',U',V') taken from the relations, which is done by full join in between T relation and U relation and the left semi join with V relation.

Reducer: From the mapper we receive the output in the reducer since there is not any condition to reduce the output of the mapper in the reducer we would just print the result obtained from the mapper output.

Communication cost: The mapper emits from every F2 a tuple (T,U,V), therefore the communication cost would be the $|T| + |U| + |V|$

Exercise 3 (MapReduce in Practice) *[4 points]*

(a) Solutions done under *Exercise*3*_MapReduceInPractice.ipynb* file

(b) Unable to solve the count in the reducer!

(c) Replication Rate for Task 1. : 13518772/17
Communication Cost: 13518773 + 2*13518772 +258967
Input Size for Task 1. : 2278886238
Output Size for Task 1. : 258967

Exercise 4 (Hive) *[4 points]*

(a) Creation and population of all the tables for the dataset is found under *Exercise*4*_Queries.sql*, bellow i am just giving one example for one of my tables in order to see my approach used.

```
--Create database
create database e11924480;
USE e11924480;

--Create table
CREATE TABLE e11924480.badges
(‘id‘ INT, ‘class‘ INT,
```

```
‘date‘ TIMESTAMP, ‘name‘ STRING, ‘tagbased‘ BOOLEAN, ‘userid‘ INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ’,’
STORED AS TEXTFILE
LOAD DATA LOCAL INPATH
’/home/adbs21/shared/hive/badges.csv’ INTO TABLE e11924480.badges;
```

(b) Could not do it

(c)
```
SELECT p.id
FROM comments c, badges b, posts p, postlinks l, users u, votes v, votes v2
(SELECT COUNT(b)as nrupvotes FROM users) k
WHERE c.postid=p.id
AND b.id = k.nrupvotes
AND v.userid = p.owneruserid
AND v2.creationdate < p.creationdate
AND l.relatedpostid > p.id
AND v.usedid = b.userid
AND (b.name LIKE ’Autobiographer’);
```

The Query above is the fix tentative for the symmetrical output in hive, important to note is the fact that hive does not support more than one sub query, so my tentative was to use joins and left joins in order to avoid sub query issue.

Exercise 5 (Spark in Scala) *[4 points]*

(a) All the solutions done under *Exercise*5_*SparkInScala.ipynb* file

(b) All the solutions done under *Exercise*5_*SparkInScala.ipynb* file

(c) Narrow dependencies are the case where each parent partition is used by at most one child partition, in our case are the cases as map, filter, union and join with input-copartitioned. In the other hand the wide dependency are the case in which the parent partition can be used by many children partitions cases like that are mostly the group by key and join with inputs not co partitioned. In our case of query plans all the dependencies are narrow dependencies.