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

**Total communication cost**: Communication between mappers and reducers

**Replication rate**: Average number of key-value pairs that the mappers create from each input

**Reducer size**: Maximum size of value list of one key. In case of skew, there may be huge discrepancies between the lengths of the value lists of different keys

**Skew**: Different length of value lists (value lists of different keys)

**Top-K Query**: Retrieve a relatively small number of top K records according to some ranking scheme in the data set, no matter how large the data is. The map function reads in all tuples from the input split of relation R; retains the tuples with the K highest values in a local buffer; finally output these K tuples with NULL-key. Reduce function of a single reducer will for NULL-key, receive a list of tuples; retain the tuples with the K highest values in a local buffer; finally output these K tuples with NULL-key

a) *Suppose we do not pre-filter the local k largest elements at each Map task, and simply have each Map task send the output of all their mapper functions on to the reduce phase.*

When we don’t pre-filter the k-largest elements for each of the Map tasks and we just send the output to the reducer, we will significantly increase the communication cost (for each of the mappers to the reducer, because it’s just one) and replication rate. We can also do an identity function (just simply write all key-value pair into the output), then the reducer should do all the sorting.

The computation power will be greater in the reducer rather than the mapper because it will have to sort all the data and then to compute top K.

b) *What if we consider a multi-round version of Top-k? Assume the first round to have some number of Reduce tasks, say 10, and the second round then collects the output of these Reduce tasks from the first round at a single Reduce task in the second round.*

We still have higher communication cost as data passes to the second round. The replication rate will increase as the intermediate data form the first round.

The reducer size will be larger in the second round as the first one because it will need to process all the output from the first one.

c) *Suppose that we increase the number of k from 10 to 1000*

Communication cost will increase as we transfer more amounts of data from mapper to reducer. As the number of unique keys increases replication time also increases.

Reducer size will also increase because we need to store more intermediate data as well the skew.

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

A)

*We are given three relations T(A, B, C), U(B, C). We define the operation F2, giving us a new output relation F2(T, U), as follows: F2(T, U) := T ÷ U*

Mapper:

1. If the record belongs to T, emit (B, C, (A, B, C, T)) for each record where (B, C) is the join key.

2. If the record belongs to U, emit (B, C, (U,)) for each record where (B, C) is the join key.

Reducer:

1. For each key (B, C), collect all the values associated with that key.

2. For each value that belongs to T, check if there is a corresponding value in U. If there is, emit (A, B, C).

3. The output of the reducer will be the relation F2(T, U) where each record contains the columns (A, B, C).

B)

*We are given three relations X(A, B, C) and Y (B, C, D) and Z(C, D, E). We define the operation F1, giving us a new output relation F1(X, Y, Z), as follows: F1(X, Y, Z) := πd,e((σE>1(Z) ⋉ (σB>1¸(Y ) ⋉ σC>B+A(X))))*

σE>1(Z), σB>1(Y), σC>B+A(X) are being done with the mapper, since we can do the comparison with the tuples.

While making the selection on C>B+A for X it also omits the key-values B and C which will be needed for the semi-join. The same procedure is being done with Y relation. First the selection process B > 1 and then B and C for keys. The selection process for Z as well E > 1 and C and D for keys.

In the reducer we create list of tuples from relation X and Y which have the same values of keys B and C. Then we are joining the lists on and extracting the values of A, B, C and D.

Then we are creating list of tuples from relation D which have the same value of C as the current B and C. Then we are joining the combined list on attribute D.

For each tuple in the new list, extract the values of D and E. Emit a tuple with attributes D and E as the final output of the reducer.