

# MapReduce

## CS 236 Project Report

Umesh Moghariya  
umogh001@ucr.edu, 861083538

Xin Li  
xli020@ucr.edu, 861062776

### 1 Problem Statement

To implement Fagin's Algorithm for TopK using MapReduce Framework.

### 2 Algorithm

**Mapper**, `mapper.py`

The mapper would read the data from standard input and pass the result to reducer via standard output. This mapper is based on Fagin's algorithm to implement topk. In the mapper, the input is relation database record, the output is the topk records within the partial records. The output would be <record name, function value>, where the function value=attr.1+attr.2+attr.7+attr.8+attr.9. The mapper first read the records, then different attributions dispatched to different groups. Because we just need attr 1,2,7,8,9, we just store these attribution groups. Each element in the attribution group is like <record name, attr value>. In python, I just tuple to store the element, and use list as the group. After that, use `sort()` method to sort each group, the key=attr value. After this step, classic Fagin's algorithm is used. In the last, we again use `sort()` method to sort the seen group. And pick out the topk and write to standard output. The reducer would receive the standard output later.

**Reducer**, `reducer.py`

The reducer receives the standard input, and writes its result to standard output. The standard input is actually the output from the mapper. The formation of the input is <record name, function value>, as I have described above. The reducer just collects all results from all mappers, then sorts the result, picking up the topk.

### 3 Results

The program produces results for top5. The value of K can be varied in the program itself.

ID	Score
27004	2.230281
27424	2.2268658
28011	2.2199928
29042	2.2053035
22883	2.176018

Table 1: Results for Top5

## 4 Time Taken for different configurations

All the values are average of 3 runs.

**dataset1.txt**

**2 mappers**

*reducer 1*

mapper:

mapper 1: 1mins, 33sec

mapper 2: 2mins, 18sec

reducer:

reducer 1: 51sec

total: 2mins, 34sec

*reducer 2*

mapper:

mapper 1: 1mins, 48sec

mapper 2: 2mins, 42sec

reducer:

reducer 1: 1mins, 3sec

reducer 2: 1mins, 3sec

total: 3mins, 1sec

*reducer 4*

mapper:

mapper 1: 1mins, 48sec

mapper 2: 2mins, 48sec

reducer:

reducer 1: 1mins, 12sec

reducer 2: 1mins, 6sec

reducer 3: 12sec

reducer 4: 12sec

total: 3mins, 17sec

**4 mappers**

*reducer 1*

mapper:

mapper 1: 1mins, 1sec

mapper 2: 51sec

mapper 3: 33sec

mapper 4: 21sec

reducer:

reducer 1: 42sec

total: 1mins, 41sec

*reducer 2*

mapper:

mapper 1: 1mins, 3sec

mapper 2: 48sec

mapper 3: 33sec

mapper 4: 21sec

reducer:

reducer 1: 45sec

reducer 2: 42sec

total: 1mins, 40sec

*reducer 4*

mapper:

mapper 1: 1mins, 0sec

mapper 2: 48sec

mapper 3: 30sec

mapper 4: 18sec

reducer:

reducer 1: 39sec

reducer 2: 39sec

reducer 3: 12sec

reducer 4: 12sec

total: 1mins, 48sec

**dataset2.txt****2 mappers***reducer 1*

mapper:

mapper 1: 5mins, 39sec

mapper 2: 5mins, 58sec

reducer:

reducer 1: 30sec

total: 6mins, 17sec

*reducer 2*

mapper:

mapper 1: 5mins, 23sec

mapper 2: 5mins, 42sec

reducer:

reducer 1: 27sec

reducer 2: 24sec

total: 6mins, 2sec

*reducer 4*

mapper:

mapper 1: 5mins, 31sec

mapper 2: 5mins, 50sec

reducer:

reducer 1: 27sec

reducer 2: 24sec

reducer 3: 12sec

reducer 4: 12sec

total: 6mins, 22sec

**mappers: 4**

*reducer: 1*

mapper:

mapper 1: 2mins, 12sec

mapper 2: 1mins, 52sec

mapper 3: 2mins, 11sec

mapper 4: 1mins, 48sec

reducer:

reducer 1: 2mins, 23sec

total: 4mins, 27sec

*reducer: 2*

mapper:

mapper 1: 2mins, 12sec

mapper 2: 1mins, 48sec

mapper 3: 2mins, 11sec

mapper 4: 1mins, 37sec

reducer:

reducer 1: 2mins, 42sec

reducer 2: 2mins, 39sec

total: 4mins, 39sec

```
reducer 4
map:
mapper 1: 2mins, 21sec
mapper 2: 1mins, 57sec
mapper 3: 2mins, 3sec
mapper 4: 1mins, 21sec
reduce:
reducer 1: 2mins, 16sec
reducer 2: 2mins, 13sec
reducer 3: 12sec
reducer 4: 12sec
total: 4mins, 35sec
```

## 5 Workload between Members

Xin Li

- Hadoop setup in Ubuntu
- Python implementation of TopK Map/Reduce program
- Algorithm design
- Tried to implement on HDFS on multiple machines

Umesh Moghariya

- Hadoop Setup in OSX
- Algorithm Design
- Java Implementation of Map/Reduce programs
- Report Writing
- Tried to implement on HDFS on multiple machines

## 6 Conclusion

- TopK implemented successfully in hadoop framework without any bugs/ errors.