CS 6240- Large-scale Parallel Data Processing

Map-Reduce Implementation of Twitter-follower:

The pseudo-code for Twitter-follower count program is as follows:

Pseudo-code:

```
class MAPPER
    method Map (key, value)
        words ← Tokenize ‘value’ delimited by “,”
        EMIT (words [2], count 1)          # Emit second word in tokenized list “words”
class REDUCER
    method Reduce (term t, <list of count values>)
        sum ← 0
        for each c ∈ <list of count values>:
            sum ← sum + c
        EMIT (term t, count sum)
```

Brief Idea of solution:

This implementation of Twitter-follower count program reads the input line by line. It constitutes of two stages, namely ‘Map’ and ‘Reduce’. The ‘Map’ function parses a line to extract the words separated by “,”. For each sets of words, it outputs the “second” word along with the count of “1”. These words emitted from ‘Map’ phase are grouped by ‘word’ as the key and then the reduce function computes overall count for each word by aggregating the sum of counts received from ‘Map’ function.

Running Time Measurements:

Run 1 measurements:

Job execution time: 9 minutes

GC time elapsed (ms)=21804
CPU time spent (ms)=471380

Total time spent by all map tasks (ms)=1182578
Total time spent by all reduce tasks (ms)=150206

Amount of data transferred:

Number of bytes to mappers = 1319497722
Mapper to Reducer bytes = 961483442
Reducer to output = 67641452

Run 2 measurements:

Job execution time: 8 minutes

GC time elapsed (ms)=25846
CPU time spent (ms)=464860

Total time spent by all map tasks (ms)=1186303
Total time spent by all reduce tasks (ms)=148577

Amount of data transferred:

Number of bytes to mappers = 1319474986
Mapper to Reducer bytes = 961483442
Reducer to output = 67641452

Argument about speedup:

Compared to sequential processing, here in the MapReduce design, the data was processed in parallel by the 20 Mappers and 10 Reducers. This parallel computation resulted in the better speedy execution of program on entire dataset.
Furthermore, we can't improve the speed more because there is a sequential part in the 'Map' stage where we iterate over list of words to get the second word. This sequential part limits the speedup even with the more mappers and reducers.

Log files path:
HW1/logs/syslog-#.txt ; where '#' represents the run number

Output files path:
HW1/output/output_# ; where '#' represents the run number

Report path:
HW1/Srikanth_Mandru_HW1.pdf