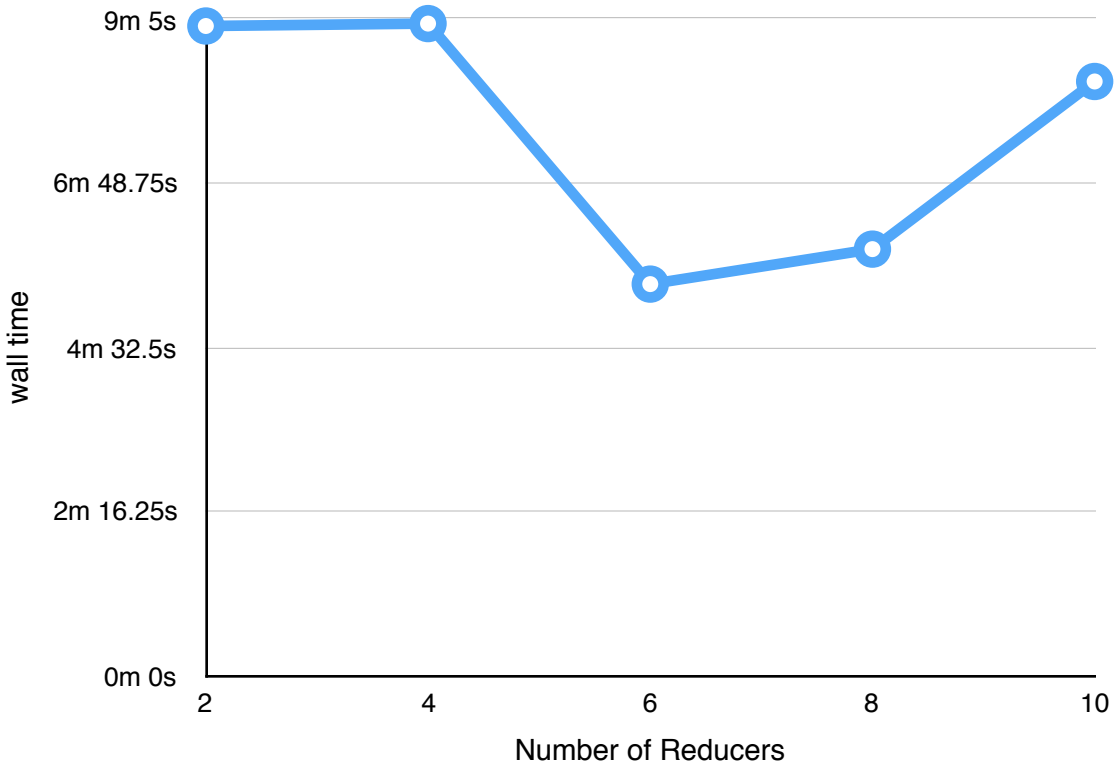
# Report for HW1-B

Yao Li
Andrew ID: yaol3

- Did you receive any help whatsoever from anyone in solving this assignment? **No**.
- Did you give any help whatsoever to anyone in solving this assignment? **No**.

2. For parallel computing, the optimal speedup gained through parallelization is linear with respect to the number of jobs running in parallel. For example, with 5 reducers, ideally we would expect parallel computing to take 1/5 wall clock time of single ma- chine run. However, this optimal speedup is usually not achievable. In this question, set the number of reducers to 2, 4, 6, 8, 10, and record the wall clock time. (The wall clock time of reducers can be inferred from syslogs - use the time between first log line with “map 100%” and “map 100% reduce 100%”). Plot a curve, where the horizontal axis is the number of reducers, and the vertical axis is the wall time. Is the wall time linear with respect to the number of reducers? Explain what you observed. (10 points)



| Number of Reducers | Time Mapper ends | Time Reducer ends | Wall Time |
|---|---|---|---|
| 2 | 23:47:53 | 23:56:52 | 8m 59s |
| 4 | 00:04:42 | 00:13:43 | 9m 1s |
| 6 | 23:34:08 | 23:39:33 | 5m 25s |
| 8 | 22:57:45 | 23:03:39 | 5m 54s |
| 10 | 23:12:53 | 23:21:06 | 8m 13s |

**Answer:**

As we can see from the plot above, the wall time does not linearly decrease as we expect, and even increase after the number of reducers is larger than 6.

The increase of time can be from the following reason:

1.The of the NameNode takes time, the increasing amount of the reducers will lead to the initialization time to increase.

2.The data must be splited after shuffle through hashing, the increase of reducers will therefore increase the time to split the shuffled file, shuffled data will be sent to different reducers, therefore takes more time for network transfer and parsing.

3. In information retrieval, sometimes we need to know the most relevant documents in a corpus given a search query. This can be done by calculating the relevance score of each document to that query, and ranking the documents according to their scores.

**Solution:**

Since k1 and b is known, what we really need to get to calculate score(D,Q) is the IDF,TF,|D| and avgdl according to the equation. The IDF score will require the N and n(qi).
In the first MapReduce, what we can get from the input data will be TF(w,d), |D|, avgdl, N, n
1st MapReduce:

Map input:

data: d1 \t russian police raid rights group memorial and other...

Map output:

1. d \t 1 —count the number of documents
2. d,|D| \t m —get the length of each document
3.q, d \t 1 —generate the keyword, doc pair (q, d)

sort and output to reducer

Reduce output:

1.N — number of documents, aggregating the count of documents
2.q, DF — document frequency for each keyword in each document, by aggregating the unique keyword, doc pair
3.q, d, TF(q, d) —unnormalized TF(qi, D), by aggregating the (q, d) pair
4.avgdl —aggregating and then divided by the number of documents(N)

2nd MapReduce:

Length of documents is now given, we can therefore update TF to normalized TF.
Map input: output of last reduce step
Map output:
1. q \t DF
2. q \t d, TF
3. q \t
sort and output to reducer
Reduce output:
1.q, d \t TF, DF, N, |D|, avgdl

3rd MapReduce:

Calculate the score using the given b,k1 and other parameters.
Map input: output of last reduce step, and the query Q(q1, q2, q3……)
Map output:
d, q, score(d, q)
Sort and output to reducer
Reduce output:
d, score(d,Q)