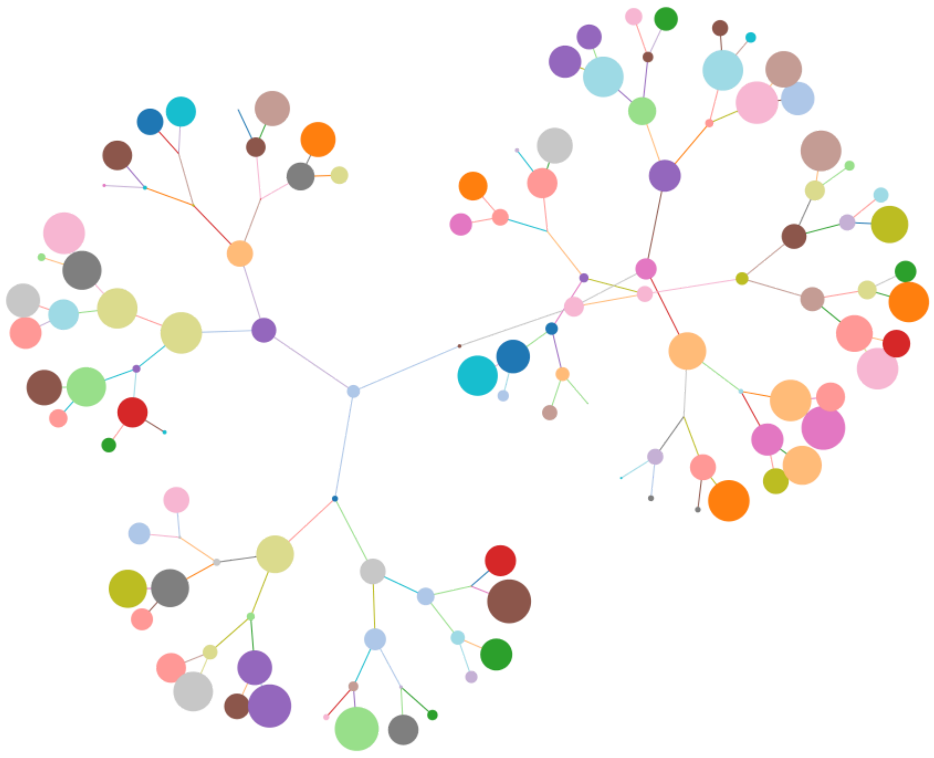
Social Media Analysis

Degrees of separation on social networks using Distributed File systems and Mapreduce



CS 546 : Design of Concurrent systems
Prof. Timothy A. Gonsalves

Project Report

Prateek Rajdev B13130
Sagar Ghai B13135

Contents

Introduction

In the modern world, social networks like facebook and twitter have really taken a toll. The massive amount of data that these websites are generating is a huge mine of information. With the rise in Big Data technologies like Hadoop and Mapreduce, analysing and visualisation of this data has become very easy and fast. Each network has some form of a relation between its nodes. Social network relations could be expressed using graphs. Graphs are one of the most used data structure in computer science. With the growth of social networks the graphs used to represent these relations have also become very complex. For the analysis of these graphs various algorithms have been developed,some are sequential and some are parallel. Our aim here is to find the distance of all the nodes of a social graph from a fixed source node, often this problem is referred to as single source shortest path. Sequentially this could be solved using Dijkstra's Algorithm but we will be using parallel Breadth First Search to solve this problem to achieve concurrency. The algorithm makes use of the Mapreduce paradigm of programming over Hadoop Distributed File system. Apart from the technical perspective this problem has a lot of social significance. We will use our results to check the validity of one very famous theory by Frigyes Karinthy ,called the Six Degrees of Separation.
We have discussed the results in the form of plots and figures and the algorithms and theories are explained in corresponding sections.

Algorithm

As we stated earlier, our aim is to find single source shortest paths using Dijkstra's algorithm.There are sequential as well as parallel algorithms to this. The sequential algorithm for this is called the Dijkstra’s algorithm. Dijkstra’s algorithm makes use of priority queue implemented via heap data structure to find the shortest path between any two nodes or all nodes from a single source. It is a greedy algorithm that uses minimum cost path to propagate further. Now we are not actually interested in the path of traversal but the cost of traversal which makes our problem easier to handle. Dijkstra’s algorithm is fairly similar to Breadth first search in terms of propagation and traversal, meaning that if the cost of each edge is 1 then there is absolutely no difference between their returned values of cost of travel. Hence for our concurrent implementation of the algorithm we make use of Parallel Breadth First search only. The main difference between sequential and parallel algorithm is that we don’t maintain a global data structure in parallel algorithm, since the algorithm works in distributed environment.

Map Reduce Algorithm

As stated above we can’t maintain a global priority queue in map reduce framework, so we need to use brute force.But now we have many nodes instead of just one node. The similarity in both the cases is that we assume the graphs in both the cases to be represented by adjacency list and the distance to that node initialized by ∞ except for the source node. In our algorithm we have assumed the distance between every node to be 1.This assumption has social significance in the analysis that we will do later.

“The intuition behind the algorithm is this: the distance of all nodes connected directly to the source node is one; the distance of all nodes directly connected to those is two; and so on. Imagine water rippling away from a rock dropped into a pond— that’s a good image of how parallel breadth-first search works.”[2]

The pseudo code is mentioned below. Here n represents the node id of the node (integer) and N represents the node’s data structure (adjacency list and current distance). The input to the mapper is the node id n and corresponding data structure N. The mapper works by emitting a key value pair for every neighbour of node n with key as the id of node n and value as the current distance of node n plus1. After the shuffle and sort phase a reducer will receive many values corresponding to a particular node id,it needs to pick the lowest and emit that value along with the mode id m as key. It is clear that after running this algorithm once we will only be able to find the shortest path for neighbours of the source. We will give the output generated by reducer to to the same map reduce algorithm again and again till all the nodes don’t have values less than infinity.The type of algorithms in which output of one iteration serves as input for another are called iterative algorithms.

Pseudo Code

```
class Mapper
    method Map(nid n, node N )
        d ← N.Distance
        Emit(nid n, N ) Pass along graph structure
        for all nodeid m ∈ N.AdjacencyList do
            Emit(nid m, d + 1)      Emit distances to reachable nodes
class Reducer
    method Reduce(nid m, [d 1 , d 2 , . . .])
        d min ← ∞
        M ←∅
        for all d ∈ counts [d 1 , d 2 , . . .] do
            If IsNode(d) then
                M ← d
            else if d < d min then
                    d min ← d
    M.Distance ← d min
    Emit(nid m, node M )
```

Source : pseudo code Data Intensive Text Processing with Map Reduce

Step by Step Description

1. Mapper - It takes in key values pairs and emits the key value pairs of the form <adjacent node, distance from source + 1>. All these pairs are then passed onto the Reducer. Before the output goes to the reducer , the following things take place.
    a. Shuffle : The incoming key value pairs from the mapper and shuffled by key to maintain randomness and load balancing on all the reduce tasks.
    b. Group by : The key value pairs are grouped together to form new pairs that contain single and list of values from all mappers of the form : <Node , [distance 1, distance 2, distance 3 ….] >
    c. Sort : The pairs are then sorted and passed onto each reducer in chunks.
2. Reducer : The reducer gets key value pairs from the intermediate shuf-sort and reduces the list of values for each key into single value of minimum distance from source node.

Theoretical Analysis

The map reduce algorithm can be basically divided into three main phases namely :

1. Map
2. Shuffle and Sort
3. Reduce

The map phase and reduce phases are the parallel phase, the shuffle and sort phase can't start before all the mappers have finished their tasks and reduce phase won't start before the shuffle and sort phase finishes. The parallelism in the map and reduce phase can be increased by increasing the number of mappers and reducers. In general , mappers in the map phase reduce the input data to be processed by the reducers so generally we require less numbers of reducers. But in our case this is not the case.In our case the data to be processed by the reducers is more than the data to be processed by the mappers. So we need to have at least as much reducers as mappers. The following criterias are important in determining the running time of the algorithm:

1. **Network Overhead** : As the number of mappers and reducers increase network overhead is expected to increase in general.But again in our case we have a very small file size and very high speed connection between the nodes. Consider a scenario where we have m mappers running ,each mapper will have 80mb/M data to be processed.(M is the number of mappers).It will only emit as much data it will process.

   a. Data processed by mapper = (file_size) / (number of mappers)
   b. Max value of this term = 80mb , file size = 80mb and M =1,
   c. Time = 80mb / (128mb) = 0.625s
   d. Note this data will not be emitted sequentially so we can expect the time to be much less.

   And after the shuffle and sort phase the data transferred to reducers will have similar latency.So whether or not network overhead plays a major role in depends on the actual running time.If this is a significant fraction of running time.(Since we know the results while writing this,this doesn't not play a major role).
2. **Number of mappers(M) and and number of reducers (R)** : In our case more the number of mappers and reducers we expect less time but the cluster should be able to run the specified number of tasks. Also provided that the overhead of starting these jobs and network overhead should not bypass this factor.
3. **Shuffle and sort phase** : This phase seems to be the most expensive phase in our algorithm but we expect this to take almost constant time for every number of mappers and reducers.
4. **File Input and Output** : Since our algorithm is iterative algorithm we need to take input and write output to hdfs which is very expensive.To check this clain we could run our code in spark instead on hadoop.But this analysis is out of the scope of this book.

Comparison with Sequential : Due to above factors and the fact that our algorithm uses brute force and also due to framework overhead we may compensate speedup obtained distributed computing.

Technical Experiments

Data Description

We have a social network graph from Yahoo Messenger[4] which contains approx 1.8 million nodes and nearly 4 million edges. The graph is from real world data and contains all edges in a csv file. The file size for the data is ~80 MB.

Using Hadoop

Cluster Specifications

PC Lab
DFS : Hadoop Distributed File System v1.2.1
Configured Capacity : 713.69 GB
Namenode : PC-Lab-01
Datanodes : 10
Default Block size : 64 MB

Replication Factor : 3
Network Bandwidth : 1 Gb/s

Machines Used

PC Lab machines
OS - Ubuntu 15.04
CPU - Intel(R) Core(TM)2 DUO CPU : 3.06GHz
RAM - 2 GB
(HARD DISK SPACE : 79.3 GB (for the cluster))

Experimental Procedure

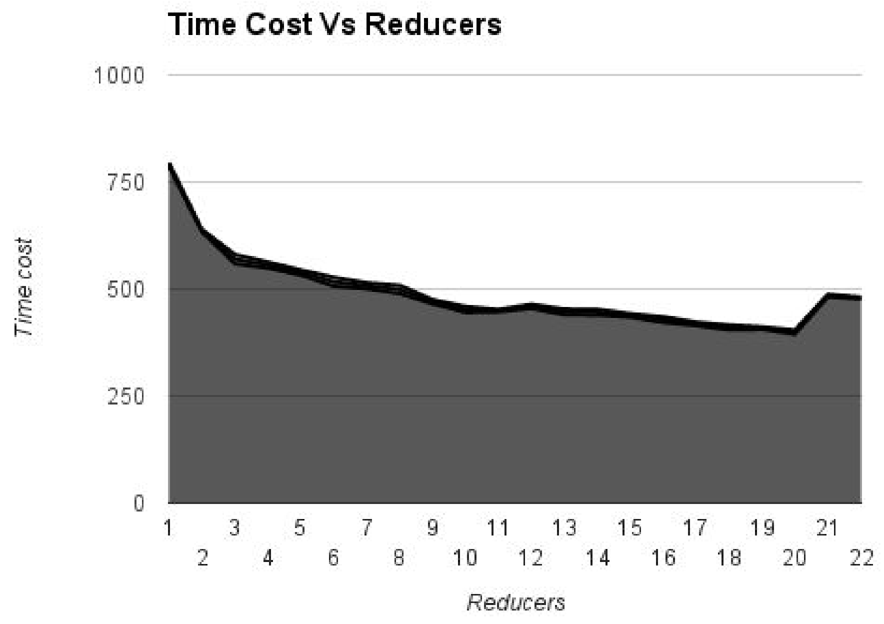
In hadoop, each mapper is given an input which is called an input split. This is equal to the size of a file block on hdfs.Since we have small file size we have kept the block size small and varied it.The number of mappers is equal to the input splits generated for input file.As described above the number of mappers and reducers play a vital role in determining the time for running a map reduce job. Basically,these are the only things that we can change as a programmer apart from a few optimizations.
For the first experiment,we keep the no of mappers constant and change the no of reducers.

   a. Our File size was 80 MB so by default the no of mappers that run is 2. In order to change that we change the hdfs default block size to 4 MB , 8 mb and 16 MB and repeat our experiments for the same. We take 3 value iterations for each observation recorded.
   b. We plot our results with a 95% confidence interval to generate a minima that we will be using for our second experiment.

2. For the second experiment we change the no of mappers keeping the no of reducers the same which is from the results we obtained above.
   a. Again it would not have mattered if we had changed the no of nodes in the cluster as the file splits itself into two parts with default size of 64MB. But instead we changed the block size in order to change the no of mappers being run.
   b. We change the block size at intervals of 4 and generate our results with a 95% confidence interval.

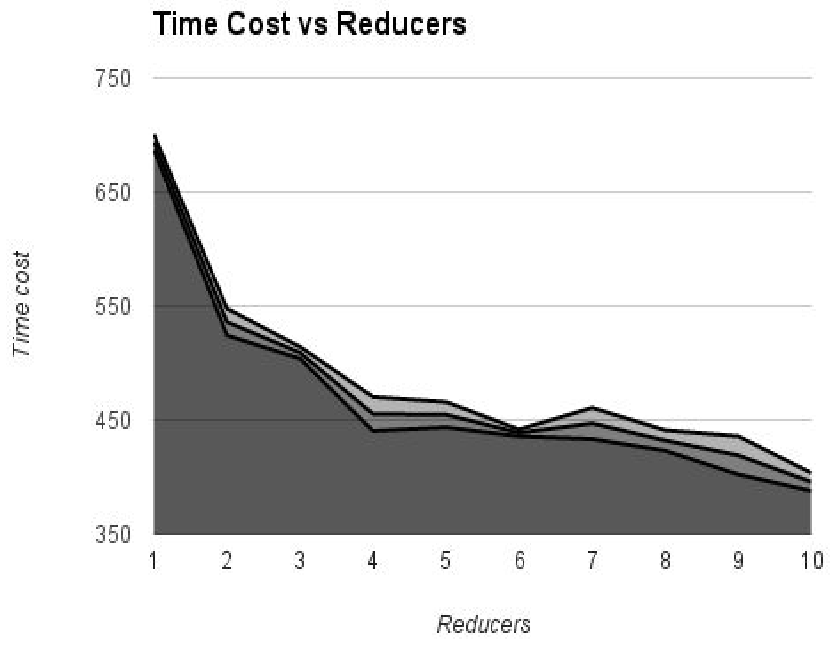
Results plotted with 95% confidence interval

1. Experiment 1
   a. 4MB block size 　　Number of mappers = 80 mb / 4 = 20

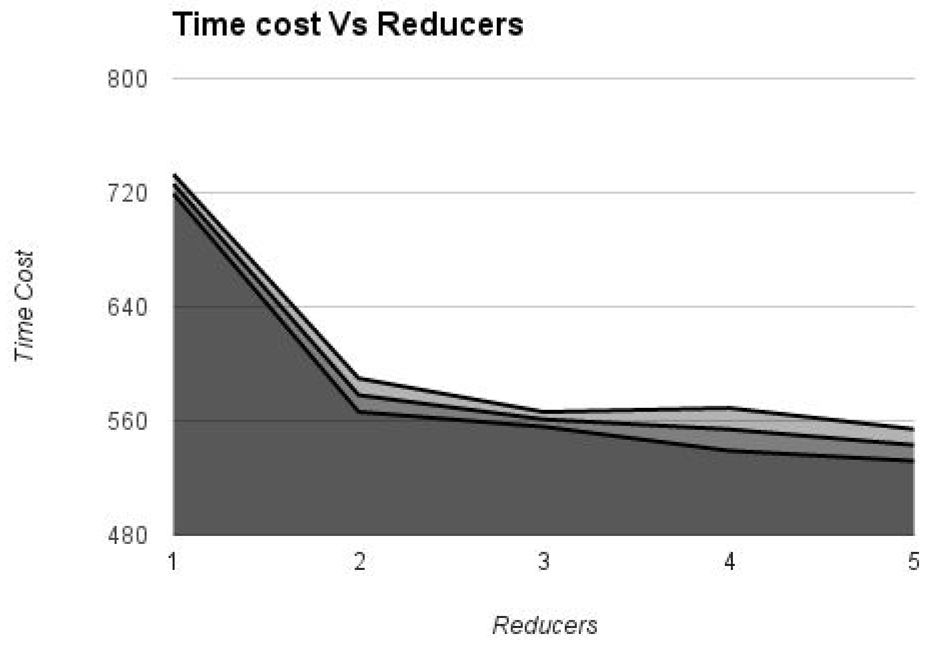


Graph 1

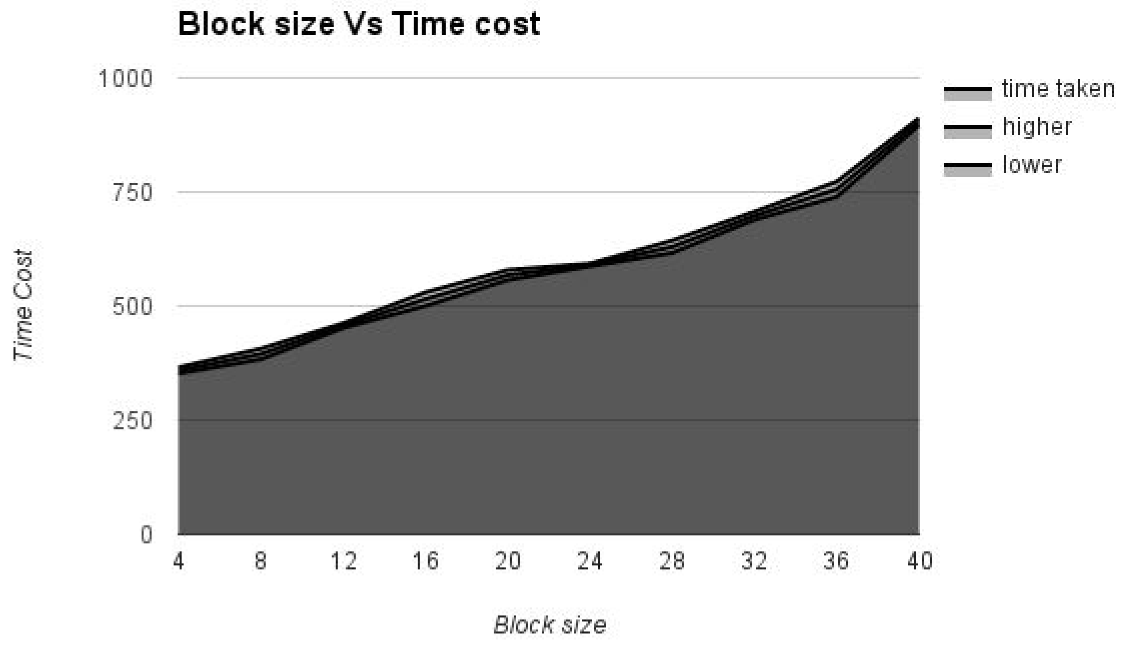
b. 8MB Block size Number of mappers = 10



c. 16MB Block size Number of mappers = 5



2. Increasing mappers (Block size)



Observations and Inference

Observation
We can see that the time taken to perform the job reduces as we increase the number of reducer.This is in par with the results we expected. Also we see that there is a slight increase in the time in experiment **1 a** as the number of reducers increases from 20 to 21.
In general we observed that on increasing the number of mappers and reducers the time decreased.

Inference
In **1a** this is because of the fact that more reducers means more parallel processes. And also we can see that the total time for a job in considerably more than the expected network overhead.So we can ignore the effect of network overhead.We see a sligh increase in the time when the number changed from 20 to 21 because the machines in the cluster are dual core so at a time only 20 reducers can run at a time.
Also the general increase in performance was obtained due to the reasons stated in the theoretical analysis.

Social Experiment

Six degrees of separation

A node in a social graph represents a person, an edge represents relationship among the two people represented by the nodes. If two nodes are directly connected we define separation between them as 1. If the individuals are not directly connected we define separation as minimum distance between them. The theory of Six Degrees of separation states that every person on earth is separated by a maximum of six persons and hence the six degrees of separation.

We have tried to check the validity of this theory by running our algorithm on various social network graphs. We chose the following 3 datasets :

1. Facebook social graph[5]
2. Pokec social graph[6]
3. Yahoo messenger[7]

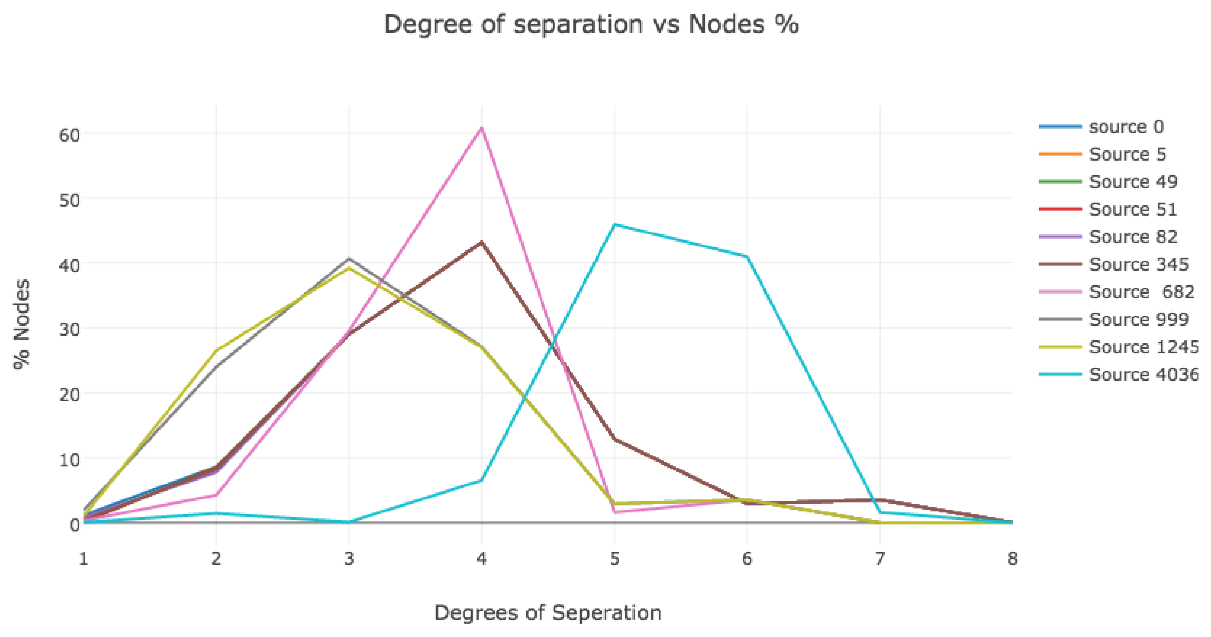
We seek to establish that the theory of Six Degrees of Separation is true.

Experimental Procedure

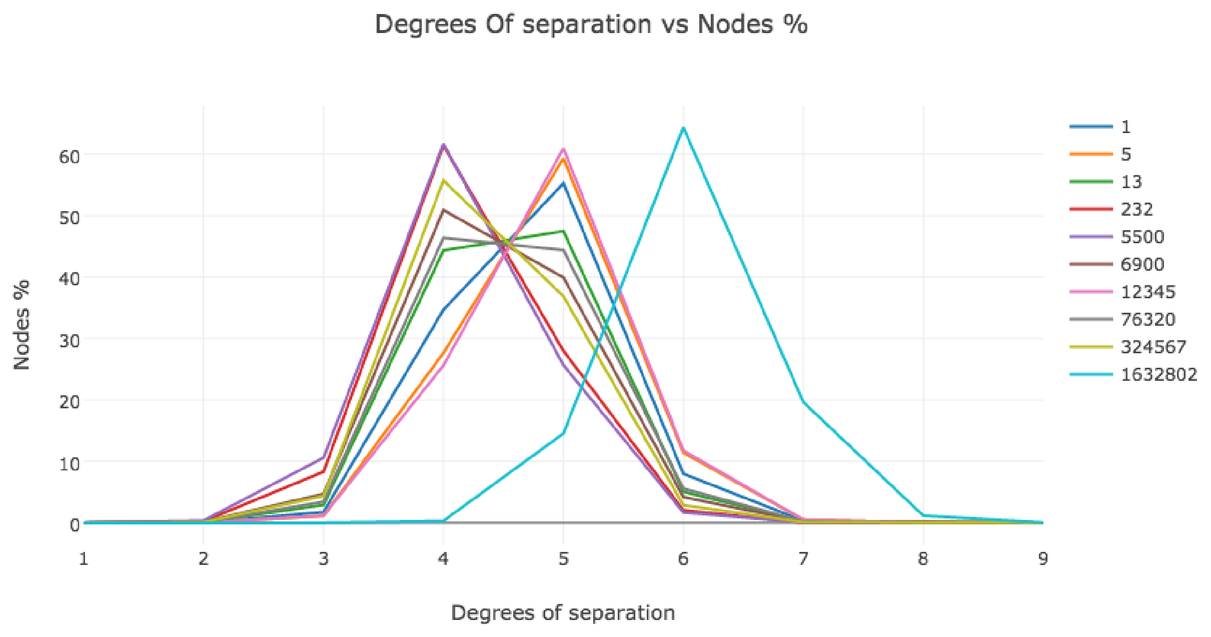
1. Using the results from the above experiments and generating more results we find the maximum degree of separation with randomly chosen sources.
2. We also look for the percent of people at every degree of separation i.e how much percent of people are separated by 4 degrees or 5 .

Results and Inference

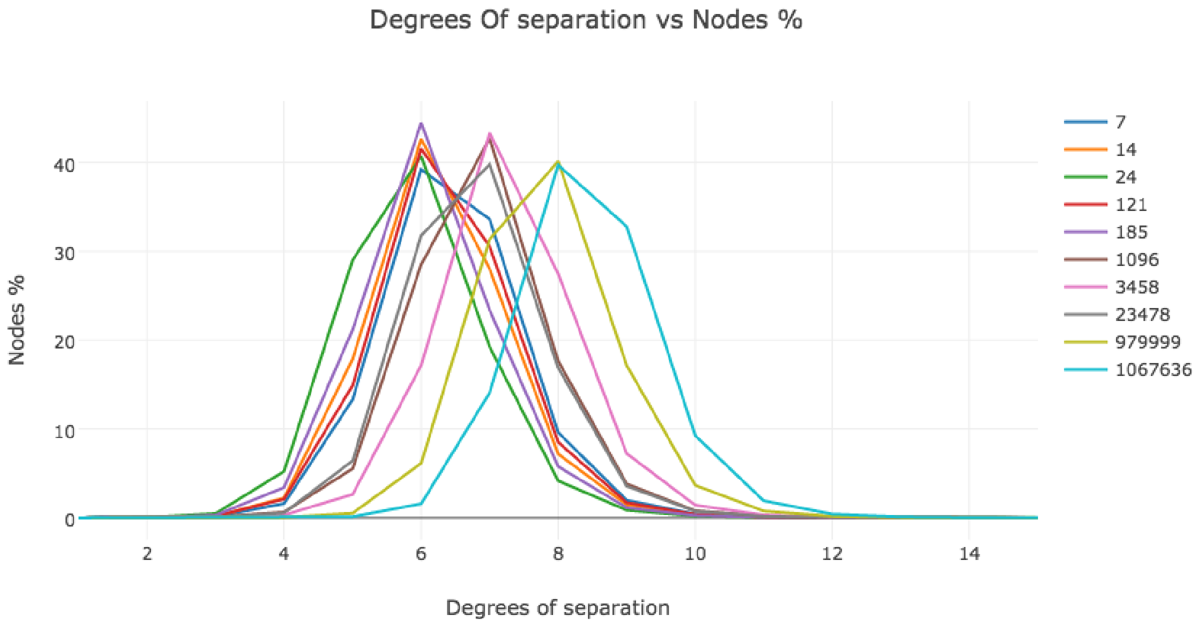
1. Facebook social graph - 97.5% below 6 degree of separation



2. Pokec graph - 92.3% below 6 degrees of separation



3. Yahoo messenger graph - 54% below 6 degrees of separation (94.5% below 8 degree of separation)



We see that for all the three graphs, there is a cluster around less than 6 which clearly indicates that the theory is true to quite some level.

Conclusion

Technical

There is not general rule for analysis and optimization of map reduce program. To get better performance we need to understand the file system and the programming paradigm in a good way. If we would have approached our problem in the traditional way we would have been stuck with two mappers with a cluster of 10. Also network delay does not always play an important role.

Social

We can see from the above pattern that the Theory of Six Degrees of separation seems to be true for social networks. Though for the Yahoo dataset see obtain only 54% of people who are separated by six degrees,but this graph is very sparse and is very old.As the density of graph increases i.e we get more close to actual social relations the theory seems to be true. The facebook data set is the most dense data that we analysed and we got 97 % people are separated by only six degrees.So it seems that this world is indeed a small place.

References

[1] https://en.wikipedia.org/wiki/Six_degrees_of_separation , latest on 17:00 june 7,2016.
[2] https://lintool.github.io/MapReduceAlgorithms/MapReduce-book-final.pdf ,latest on 12:00 june 7,2016
[3] https://github.com/anvaka/ngraph/tree/master/examples/fabric.js/Node%20and%20Browser , latest on 17:15 june 7, 2016
[4]https://en.wikipedia.org/wiki/Dense_graph ,latest on 14:08 june 8,2016
[5]https://snap.stanford.edu/data/egonets-Facebook.html on May 25,2016
[6]https://snap.stanford.edu/data/soc-pokec.html on May 25,2016
[7] Yahoo webscope data on request from Yahoo Labs.