Tariq Anwar
CS 6240- Sec1
Assign3

Programs Flow:

The preprocessing steps follows the instruction outlined in the instructions.I am using the sample parser provided with some additional function added to it to clean the data before creating the adjacency list. Preprocessing job reads the compressed file , create the adjacency list(outlinks) representation of the graph. It also removes the duplicate links in the outlinks representation. The total number of nodes is calculated using the global counter in the preprocessing step. It also modifies the adjacency list of the graph to accommodate the page rank of each node by appending the page rank column as the first column in the graph. The output of this job is used as input by PageRank Job

The PageRank job calculates the pagerank of each node. The first job assigns the initial page rank to all nodes . The sink mass is calculated in the Mapper function of the first iteration which is used subsequently in following iterations. The algorithm used in calculating the pagerank is same as defined in the module Graph Algorithm.

The only modification in the above pseudo code is the calculation of sink mass in mapper (through global counter) and using the sink mass to calculate the pagerank in subsequent iteration.

```
MAPPER

          Setup(){
           initialize SINK_MASS_COUNTER

          }

          Map(nid n, node N){

                    // Pass along the graph structure
                    emit(nid n ,node N)

                    //compute contributions to send along the adjacency list
                    p= N.pagerank/M.adjacencyList
                      for all nid m in N.adjacencyList
                        emit (m,p)

                    If (n.adjacencyList == null)  //sink node encountered
                        SINK_MASS_COUNTER += node.pagerank

          }
Reducer(){

        Setup(){
         sinkmass=  get SINK_MASS_COUNTER from context
         }

        Reduce(nid m , [p1,p2,p3….]){
        S=0;
        M=null;
```

```
            For(all p in [p1,p2,p3..])
                 if(isNode (p))
                       //Node found; retrieve graph
                       M=p;
                 else
                      //A pagerank contribution found
                    S += p
              M.pagerank = alpha/totalNodes + (1-alpha)( S + sinkmass/totalNodes)     // alpha =0.85

           Emit (nid n, M)

        }

}
```

The last job in the process calculates the top 100 pages with highest page rank. The algorithm used is same as mentioned in the online lecture module(Basic Algorithm) for TopKRecords.

The reason behind using the global counter approach to calculate the sinkmass was that it was fairly easy as compared to order Inversion / additional job.

AMOUNT OF DATA TRANSFER

The data below is no of records transferred between each phase

configuration 1: 5 workers

ITERATION1:

```
Mapper-> Reducer : 41451258
Reducer-> s3(HDFS): 2819197
```

ITERATION2:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

ITERATION3:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

ITERATION4:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

ITERATION5:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

ITERATION6:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

Tariq Anwar
CS 6240- Sec1
Assign3

ITERATION7:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

ITERATION8:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

ITERATION9:

```
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197

ITERATION10:
Mapper-> Reducer : 2819197
Reducer-> s3(HDFS): 2819197
```

As we can see from the above output values , the data remains constant, this is due to the fact that once the initial list is created we are just updating the values , there is no additional data being written. Has no of bytes been used in above phases, there would be slight change of data between each phase as pageranks are being updated but those updated bytes data are almost negligible.

PERFORMANCE COMPARISION:
Configuration 1: 5 Worker
PreProcessingTime: 27 minute 26 second

10 iteration pagerank: 21 minutes 37 second

Top-100 page time: 53 seconds

Configuration 1: 10 Worker
PreProcessingTime: 15 minute 23 second

10 iteration pagerank: 14 minutes 29 second

Top-100 page time: 39 seconds

As it is evident from the above performance comparison, Preprocessing time showed pretty good speedup, as more worker threads were involved in the second configuration.

Overall all the phases showed good speedup. Running time reduces drastically for all the three phases as we can see from the above running time comparison.

Are top 100 pages reasonable?

Most of the pages in top 100 are pages about a country, year, common words and major historical events. Based on my intuition it seems they are reasonable as they have high reference in any documents. Although, I believe that year should not come with high page rank , but I am not sure what exactly the year link was associated with(unless some historical event is associated with a particular year, in that case the rank seems fair).