**PROJECT 2: Hadoop PageRank February 15, 2015**

**INTRODUCTION:**

PageRank is a well-known web graph ranking algorithm developed by Larry Page, which is designed to search for information on the World Wide Web. It helps Internet users to sort page hits by their importance.

In this project, we implemented the PageRank algorithm on a Hadoop distributed cluster.

**IMPLEMENTATION:**

The implementation involves three major steps:

1. **CreateGraph:**

Let us first understand what the PageRank algorithm does.

* 1. **PageRank algorithm:**

PageRank algorithm calculates a numerical value for each element of a hyperlinked set of webpages, reflecting the probability that a random Internet surfer will access that page.

The formula is:

PageRank formula

Or, in a simplified version:

**PR(P) = (1-d)/N + d (PR(T1)/Count(T1) +…+ PR(Tn)/Count(Tn))**

Where,

*PR(P)* is the PageRank of a given page P,

*d* is a damping factor generally set between 0 and 1 (generally assumed as 0.85,

*N* is the total number of unique URLs,

*PR(Ti)* is the PageRank of pages Ti that link to page P, and

*Count(Ti)* is the number of outgoing links from page Ti.

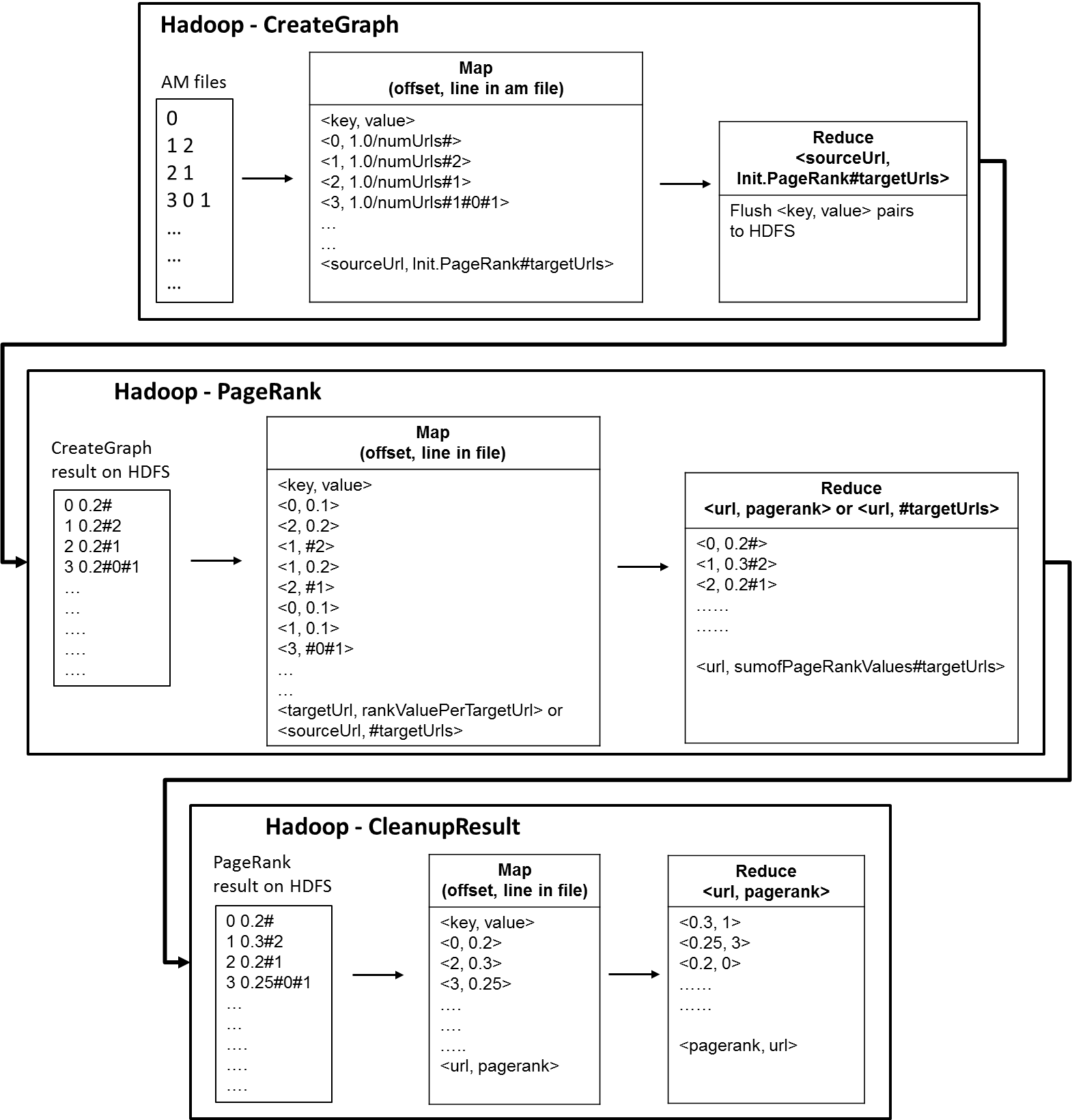
As per the PageRank theory, a surfer who is randomly clicking on the links will eventually stop clicking them. The damping factor *d* signifies the probability that, at any step, the surfer will continue clicking on the links.

The PageRank algorithm is iterative in calculating the new access probability for each webpage based on values calculated in the previous iteration.

* 1. **PageRank using the MapReduce framework of Hadoop:**

We have implemented a parallel version of the PageRank algorithm using MapReduce. The need for using MapReduce for calculating the page rank arises from the fact that there are millions of webpages scattered all over the Internet which need to be ranked efficiently in order to improve the web search results. By processing such large input data in parallel we can efficiently leverage the computing power of multiple compute nodes, thereby optimizing the processing.

The following diagram shows an overview of how the PageRank algorithm is executed using MapReduce in Hadoop:



*Figure 1: Hadoop PageRank dataflow*

*Input:* The PageRank input data is stored as an adjacency matrix on a file(s) on the local file system. Later, it will be uploaded to the Hadoop Distributed File System (HDFS) and distributed across the compute nodes.

*Map:* Hadoop framework reads the application records from HDFS with the *InputFormat* interface and generates <*key, value*> pair input streams. Each *Map* function produces zero or more intermediate <*key, value*> pairs whose value is the partial rank of every webpage, by consuming one input <*key, value*> pair. The output is stored on the local file system at this stage.

Referring to figure 1, the Mapper outputs <*3, 1.0/numUrls#0#1*> for the given input ‘*3 0 1*’ of the AM input file.

The output corresponds to <*sourceUrl, Init.PageRank#targetUrls*> format.

*Reduce:* The Reducer just pushes the <*key, value>* pairs from the local file systemto the HDFS.

1. **PageRank:**

At this stage, the new page rank is generated from the previous partial page rank values. The PageRank becomes more accurate after multiple iterations.

*Input:* The <*key, value*> pairs from the *CreateGraph* job.

*Map:* For each webpage, the Mapper outputs the webpage, its rank and the outgoing links.

<*key, value*> pairs of format <*targetUrl, rankValuePerTargetUrl*> or <*sourceUrl, #targetUrls*> are generated. The job is executed within the HDFS itself. At this stage, the Mapper job is distributed amongst various nodes of the HDFS for processing.

*Reduce:* The Reducer calculates the new page rank and outputs the webpage, its rank and the outgoing links in the <*url, sumofPageRankValues#targetUrls*>format. The job is executed within the HDFS itself.

1. **CleanupResult:**

*Input:* The <*url, sumofPageRankValues#targetUrls*> pairs from the *PageRank* job.…………………………………  
*Mapper:* The Mapper will map the input pairs as <*url, PageRank>* pairs and sort the output in an ascending order based on the *url* key., so now the bottom most value will depict the highest PageRank. The job is executed within the HDFS.  
*Reducer:* The Reducer does not perform any major task, it just converts the <*url, PageRank*> pairs to the   
<*PageRank, url*> pairs output format. The job is executed within the HDFS itself.

**FILES:***PageRankMap.java:* Mapper file

*PageRankReduce.java:* Reducer file

*pracshah\_abhivija\_ppkapoor\_HadoopPageRank\_output.txt:* Mentions ten Urls with their PageRank.

*pracshah\_abhivija\_ppkapoor\_HadoopPageRank\_report.docx: Serves as the* README and Report file for the project.

**TOOLS:**Ubuntu OS installed on a Virtual Box with pre-installed Hadoop setup, vim editor.

**COMPILE AND RUN:………………………………………………………………………………………………**$ cd /root/MoocHomeworks/HadoopPageRank/

$ ./compileAndExecHadoopPageRank.sh

The one-click script file *compileAndExecHadoopPageRank.sh* compiles the two .java files and outputs compilation errors, if any.

**OUTPUT:|**The result is generated as /root/hbaseMoocAntProject/output/project1.txt

$ cd /root/MoocHomeworks/HadoopPageRank/

$ cat output/\*

**REFERENCES:**http://blog.xebia.com/2011/09/27/wiki-pagerank-with-hadoop/  
http://michaelnielsen.org/blog/using-mapreduce-to-compute-pagerank/