Impact Factor Value 4.046 e-ISSN: 2456-3463
National Conference on Quality Up-gradation in Engineering, Science and Technology (NC-QUEST018)
Organized by College of Engineering and Technology, Dhanamgaon Rly-444709
International Journal of Innovations in Engineering and Science, Vol. 3, No.6, 2018
www.ijies.net

Page Rank Algorithm in Hadoop By MapReduce Framework

Ankita S. Sachdev[1], Ankita N. Rawale[2], Palak D. Pande[3], Kajal V. Sawadh[4]

Department of Computer Science & Engineering
DES's College of Engineering &Technology
Dhamangaon (Rly), Amravati, India, 444709

Abstract –One of the most popular algorithm in processing internet data i.e WebPages is page rank algorithm which is intended to decide the importance of a webpages by assigning a weighting value based on any incoming link to those pages. The large amount of internet data may lead into computational burden in processing the page rank algorithm. To take into account those burden, in this paper we present a page rank processing algorithm over distributed system using Hadoop MapReduce framework called MR PageRank. Our paper intended to first parse the raw webpages input to produce title of page and its outgoing links as key and value pair, respectively, as well as total dangling nodes weight and total amount of pages. We next calculate the probability of each page and distribute this probability to each of outgoing link evenly. Each of the outgoing weight is shuffled and aggregated based on similarity of page the to update anew weighting value of each page.

Keywords- pagerank, hadoop, data.

1. INTRODUCTION

In recent days, we are experiencing the rapid growth of internet data i.e (Webpages). On the other side, there is a need for analyzing those large amount of data to obtain any important information. One of the famous analysis in recent large data processing is page rank algorithm which is intended for ranking a webpage by assigning a weighting value and distributed evenly to each of outgoing link. Each of the outgoing weight is shuffled and aggregated based on similarity of page title to update the weighting value of each page. In this calculation, we also consider the dangling node and random jumping factor. Finally all the page are sorted based on their weighting value.MapReduce is distributed programming. Technique proposed by Google for large scale data processing in computing environment.

Basically, the page rank algorithm is relatively easy to be implemented in a single machine since this algorithm only needs page information (link, id or title), outgoing link information and total amount of the pages. However, processing those large amount of internet data in single machine may leads into scalability problems such computational time. Therefore, the goal of this paper is to process page rank algorithm over multiple machines in distributed system. PageRank calculation is based on power method which need iterative implementation. This method is very slow to reach convergence. The problem with computing PageRank for very big graph in traditional way is we need machine with high processing power and huge memory. Now we can use Hadoop MapReduce framework to calculate the PageRank in parallel computation and distribute the processing and memory usage accross cluster with cheaper machine. Technique proposed by Google for large scale data processing in computing environment. Its implified the implementation of many data parallel applications by removing the burden from programmer such as tasks scheduling, fault tolerance, and messaging.

2. SYSTEM MODEL AND NOTATIONS

Fig. 1 illustrates a graph representing a case of some pages with total amount |G|, in those case |G|=5. Let us consider a page interest denoted by n and set of pages that link to n denoted by L(n), where L(n)={k, l, m, n, o} .We can also figure out a webpages which link to n denoted by m where m=L(n). The m has some set of outgoing link to another pages denoted by C(m), where C{o, l, n}. Fig.1 shows outgoing nodes called dangling nodes denoted by D with total weight of dandling node is d.

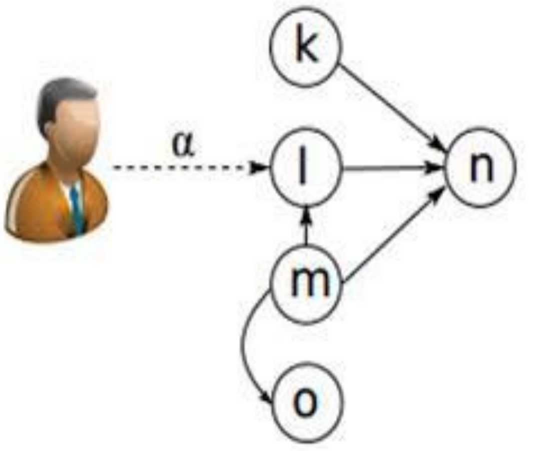

Fig.1 System Model

As figured in 1, while accessing a webpage, a user has two possibility representing his/her behavior in accessing a webpage: by random access and by following a link listed in one webpage. Random access means that a user is randomly select the webpage address he/she want to visit without considering previous visited webpage. On the other side, a user may access a webpage by following a link listed in webpage he/she previously visited. In this paper, the random access probability is denoted by a .

3. PAGE RANK ALGORITHM OVER DISTRIBUTED SYSTEM BY USING HADOOP MAPREDUCE(MRPAGERANK)

In this section, we present MRPageRank which is intended to process page rank algorithm over distributed system by using Hadoop MapReduce framework. MRPageRank can be decomposed into three jobs. First, data processing job which is intended to extract page identifier and its outgoing link as key and value pair, respectively as well as total weight of dangling nodes and total amount of pages. Next page ranking job which is intended to calculate page rank weight of each page and distribute it evenly to each of outgoing link. Each of the outgoing weight is shuffled and aggregated based on similarity of outgoing page title to update a weighting value. Finally, all of the pages are sorted based on their weighting values.

Fig. 2 shows architecture of MapReduce, PageRank. The web Graph is spitted into some partitions and each partition is sorted as an adjacency matrix file. Each Map task processes one partition and calculates the partial rank values for some pages. The Reduce task merge all the partial values and generate the global Rank values for all the web page. Hadoop is also a MapReduce framework that support data intensive distributed applications. The Hadoop PageRank need reload web graph partition file form disk to memory each iteration during the computations.

3.1 Data parsing job

First job in *MRPageRank* is to extract page identifier and its outgoing link as key and value pair, respectively.This job also emit total weight of dangling nodes and total amount of pages. Notice that, for total weight of dangling nodes *D* and total amount of pages |*G*|. we reserve special key **d* and **a* respectively. It is important to consider this job because it makes next job (page ranking) become easier to be implemented.

Procedure 1 describes the mapper function for data parsing job. First, it receives document as page input, in this case webpages or xml data. The page is parsed using a regular expression method to extract the page title and its outgoing link. For each of outgoing link n, the mapper job store n into an array *C*(*m*). Then, it checks whether this current page has outgoing link or not. If it has, the job will emit page identifier m as key and *C*(*m*) as value pair. Otherwise, we consider this page as dangling node and emit special key **d* and its weight as value.

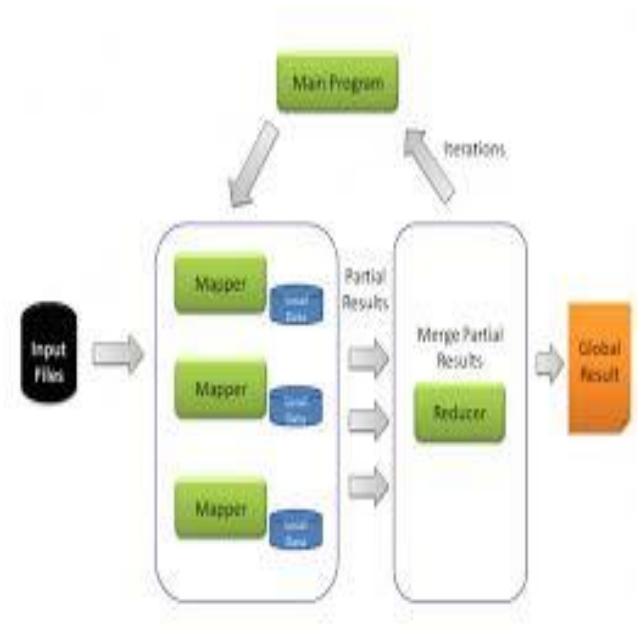

Fig.2 Architecture of Page Rank MapReduce

3.2 Page ranking jobs

Hadoop Map/Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. On the other side, there is a requirement for mining the internet data for obtaining any valuable information. One of the popular analysis in internet data is page rank algorithm which is intended for ranking a webpage based on its weighting value. Implementing page rank algorithm in single machine is relatively easy due to it only requires basic information of webpage such as page identifier (title, link) and its outgoing links. However, the large amount of recent internet data may leads into computational burden i.e. running time and scalability issue. To take into account this problem, we consider a problem of processing page rank algorithm over multiple machines in distributed system.

Given *m*, *C*(*m*), |*G*| and *D*, in this section we will calculate page rank weight for each of the page by considering dangling node and jumping factors. where *a* is random jumping factor, |*G*| is the total number of pages, *L*(*n*) is the set of pages that link to *n*, and |*C*(*m*)| is the out-degree of node *m*. If we consider page m as element of set.

Procedure 1 Data Parser Mapper Class

```
class Mapper
    procedure Map(pageid m, page d)
        $C(m)$ new $AssociativeArray$
        for all outgoingLink $n \in d$ do
            $C(m) \leftarrow C(m) + n$
        end for
        if $C(m).size = 0$ then
            Emit($*d$,1)
        else
            Emit($m$,$C(m)$)
        end if
        Emit($*a$,1)
    end procedure
end class
```

4. FINAL SORTING JOB

The last job in MRPageRank is the final sorting job which is intended for sorting all of pages based on its final page rank weight. The sorting will be done in descending mode. However, by default, the sorting method for Hadoop is in ascending mode. Therefore, for this purposes, we need to extend the comparator class in Hadoop MapReduce to accept sorting in descending mode.

Procedure 2 Data Parser Reducer Class

```
class Reducer
    procedure Reduce(pageid $m$, outgoingLinks $C(m)$)
        $P(m) \leftarrow InitialPageWeight$
        if $m = *a$ then
            for all $n \in C(m)$ do
                $counter \leftarrow counter + 1$
            end for
            $m.amount = counter$
            Emit($m$)
        else if $m = *d$ then
            for all $n \in C(m)$ do
                $dangling \leftarrow dangling + InitialPageWeight$
            end for
            $m.weight = dangling$
            Emit($m$)
        else
            Emit($m$,$\{P(m), C(m)\}$)
        end if
    end procedure
end class
```

5. NUMERICAL RESULT

In this section, we will discuss about our experiment and implementation of MRPageRank in Hadoop distributed system. The MRPageRank is implemented using Hadoop 2.7.2 with Java-based code. Each of the job is executed as map and reduce job. The job is executed over virtual machine based worker node with following hardware specification: single core Intel i5 processor, 1GB memory, 8GB storage with Ubuntu 14.04 as operating system.

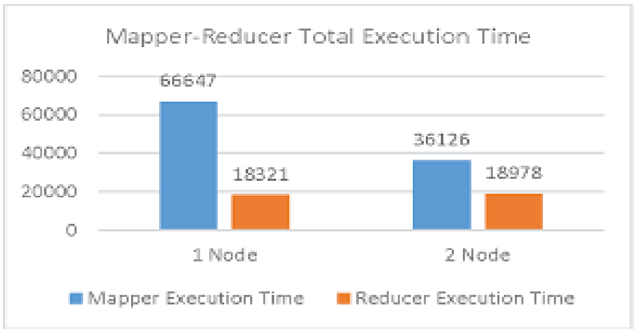

Fig.3 Mapper Reducer Total Execution Time

Figure 3 shows the total execution time in milliseconds of both mapper and reducer job at two different worker node configurations i.e. 1 and 2 worker node. From the figure, we observe that the execution time of mapper significantly reduced as the number of worker node increases.

6. CONCLUSION

We proposed MRPageRank which implement page rank processing algorithm over distributed system using Hadoop MapReduce framework. Our algorithm can be decomposed into three processes, each of which is implemented in one Map and Reduce job data parsing job,page rank calculating job and final sorting job . We proposed MRPageRank which implementation output with reasonable result.

REFERENCES

[1] Lawrence Page, Sergey Brin, Rajeev Motwani and Terry Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," StanfordInfoLab, No. 1999-66, Nov. 1999.

[2] Fabrizio Lamberti, Andrea Sanna, and Claudio Demartini, "A RelationBased Page Rank Algorithm for Semantic Web Search Engines," ieee transactions on knowledge and data engineering, Vol. 21, No. 1, Jan 2009.

[3] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI 04: 6th Symposium on Operating Systems Design and Implementation, 2004.

[4] Tom White, Hadoop : Definitive Guide 4th Edition. Sebastopol : O'Reilly, 2015, pp. 30-36.

[5] S. Landset, T.M. Khoshgoftaar, A.N. Richter, T. Hasanin, "A survey of open source tools for machine learning with big data in the Hadoop ecosystem," Journal of Big Data, Vol. 2, No. 24, Nov. 2015.