IJCTA, 9(21), 2016, pp. 25-29

Parallel K-Means PageRank Algorithm Based on Map Reduce

Hema Dubey* Nilay Khare Alind Khare*** Laxmi Narayan******

Abstract : PageRank is a most popular link analysis algorithm for measuring the relative importance of web pages. A lot of research has been done on page rank to improve its efficiency and performance by using parallelizing techniques. In this paper we have proposed a novel parallel k-means PageRank Algorithm based on MapReduce. Previous implementation of PageRank on MapReduce framework generates lot of internal results which in turn produces huge shuffling of bytes for Hadoop task. The proposed algorithm lowered the amount of shuffled bytes by processing a sub graph in a mapper function rather than processing an edge in the mapper function. We have used K-means clustering technique so that there is a proper communication between different sub graphs. The proposed work performs faster PageRank calculation by employing map cache memory to load prior PageRank scores from HDFS, so that only one MapReduce job is required for a particular PageRank iteration. Experimental results show that the proposed K means PageRank algorithm provides better results in terms of execution time as compared to traditional PageRank and Local Iterative PageRank on MapReduce.

Keywords : PageRank, K-means, Local Iterative, Map Reduce.

1. INTRODUCTION

In the massive-scale data intensive operations for instance web data administration, remote sensing, social network analysis, sensor organizations, log investigation, etc., there is always a requisite for highly scalable parallel data processing platforms. Hadoop [1] is a vastly efficient platform for storing and processing huge graphs. Hadoop offers MapReduce [2, 3] framework that facilitate programming clusters of commodity hardware to perform extensive data processing in a solitary pass. A MapReduce [3] cluster is capable of scaling thousands of nodes in a reliable and fault-tolerant approach.

PageRank algorithm [4] is a well-known web structure mining technique, which is highly suitable for colossal data processing. Large volumes of data (that is petabytes) processed by many applications (like PageRank computation) also increases the computational requirements, so there is a need of efficient clustering algorithm. In this paper we have used k-means clustering [6] as a graph partitioning method in the computation of PageRank values of web pages on MapReduce. The foremost problem in implementing PageRank algorithm on MapReduce framework is to how to partition the immense amount of data [11], and the other challenge is how to reduce the communication cost. Previous implementations of PageRank on MapReduce [7, 8, 15, 16] are quite slow in terms of execution time because they have used two MapReduce jobs for a solo iteration of PageRank. Another drawback [7, 8] is that they process an edge in the mapper function that results in huge shuffling of bytes for the Hadoop job.

In the proposed work, a cache memory is used for the purpose of loading previous PageRank scores from HDFS; this also helps in reducing the number of MapReduce jobs required for an iteration of PageRank algorithm.

* Department of Computer Science and Engineering MANIT, Bhopal, Madhya Pradesh, India hema32150@gmail.com,

** Department of Computer Science and Engineering MANIT, Bhopal, Madhya Pradesh, India nilay.khare@rediffmail.com,

*** Department of Computer Science and Engineering IIIT, Delhi, India kharealind@gmail.com,

**** Department of Computer Science and Engineering MANIT, Bhopal, Madhya Pradesh, India laxmi.timansingh@gmail.com

Here we need merely one MapReduce job for a single iteration. In the proposed method, a mapper function takes sub graph as an input record instead of taking an edge of graph with an intention to minimize the shuffling of bytes. The reducer function synchronizes the results generated by the mapper function. Proposed method also employ well-known graph partitioning technique called k-means clustering for splitting the graph into sub graphs. Thus our method decreases the running time of PageRank algorithm in the distributed environment.

1.1. PageRank algorithm

PageRank [4, 5, 9] is the link-based analysis algorithm used by Google to rank web pages. PageRank is not merely based on the number of incoming links but also considers the importance of web pages linked to a page. This algorithm assumes a link from a web page to another web page as a recommendation. The basic notion behind the PageRank algorithm [4, 5] is the dissemination of prominence from one page to others, through its out links. The prominence that web pages *y* propagates to web page *x* can be defined as:

$$\text{rank}_{i+1}(x) = (1+\alpha) + \alpha \sum_{y \in Sx} \frac{\text{rank}_1(y)}{Ny} \quad (1)$$

In the above PageRank formula, let Sx symbolize the set of web pages which are incoming links of page *x* and Ny denotes the number of outgoing links from web page *y*. Rank(*x*) is the PageRank score of web page *x* at $i + 1^{th}$ iteration and rank(*y*) is the PageRank score of web page y at iteration *i*. Where á denotes damping factor which describes the probability of a random surfer following hyperlinks in the web graph. $1-\alpha$ is the probability when the random surfer gets bored and stop visiting links at page *x*.

PageRank is an iterative algorithm which computes rank value of a web page based on the rank values from the previous iteration. The algorithm halts when the rank scores of web pages converge.

1.2. Local iterative (LI) PageRank algorithm

LI-PageRank algorithm [10] considers that the map function performs rank calculation typically in the main memory so as to shrink the communication cost between the Hadoop clusters. This algorithm takes sub graph as an input record to map function and requisite only one MapReduce job for a single iteration. LI also incorporates cache memory for loading previous rank scores from HDFS. LI algorithm [10] includes four key steps: (1) partition of graph into sub graphs, (2) mapping phase along with cache update, (3) reducing phase along with message grouping, (4) and finally checks the convergence of PageRank values.

LI PageRank algorithm uses the original PageRank formula :

$$\text{rank}_{i+1}(u) = (1+\alpha) + \alpha \sum_{j=1}^{k} \sum_{v \in Bj(u)} \frac{\text{rank}_i(v)}{Nv} \quad (2)$$

In Equation 2, rank (*u*) is the rank score of page *u* at $i + 1^{th}$ iteration. Bj(*u*) is the suite of web pages pointing towards *u* at j^{th} partition. PageRank calculation is performed locally for a sub graph in a map function and the intermediary values of every iteration are summed up by reducer function. As mention in Equation 2, the input graph is splitted into k sub graphs and the later part of equation 2 is calculated by k map functions. For every partition, map function yields PageRank score for web page v. The output of Mapper function is treated as an input to the reducer function. The reducer function performs summation of all the partial results (in the form of < page, rank > pairs) produced by the mapper function. This algorithm uses map cache memory to obtain the rank scores of earlier iteration with the help of setup API available in Hadoop. The mapper function computes PageRank scores of web pages connected to one another within the subgraph. The reducer function performs summation of all the partial results to obtain the final PageRank value of each web page.

1.3. K-means algorithm

K -means algorithm [6] is an eminent clustering technique used to split a set of *n* items into *k* clusters with an intention to elevate the resulting intra-cluster resemblance and lower down the inter-cluster resemblance. The resemblance within the cluster is calculated based on the mean value of the items within the cluster.

This algorithm works in the following manner: Initially, it arbitrarily selects *k* items from an entire suite of items that denotes initial cluster centers. Depending on the distance in the midst of object and the cluster center, each left over item is assigned to the cluster to which it is the most similar. After that the new mean value of each cluster is computed. This procedure repeats until the standard criteria meets.

The rest of the manuscript is structured as follows. Section II discusses Proposed Parallel k-means PageRank algorithm. Section III confers experiments and results. Section IV describes concluding observations.

2. PROPOSED ALGORITHM

2.1. Parallel *k*-means PageRank algorithm

The proposed algorithm integrates *k*-means clustering with LI PageRank algorithm [10] with an aim to speed up running time of PageRank algorithm. The proposed algorithm requires only single MapReduce job for computing rank values. The task of mapper function is to assign each sample to the nearby center whereas the task of reducer function is to update the new centers. A combiner function is introduced to cope up with the partial aggregation of intermediate values with the identical key within the same map task. This helps in abating the network communication cost.

The proposed algorithm is a blend of horizontal partitioning and vertical partitioning [12] to segregate a huge graph into number of sub graphs. Horizontal partitioning is performed with an aim to lessen the amount of PageRank scores accumulated in map cache. And vertical partitioning is done with the purpose of reducing the amount of output records of reducer task. The motive of incorporating horizontal and vertical partitioning together assists in storing the sub graphs with approximately same number of edges. Considering huge graph that contains additional web pages as compared to threshold value, horizontal partitioning is required to segregate the huge graph into groups. Every group consists of web pages along with their neighbors.

For each partition, mapper function produces PageRank value for each web page. The mapper generates partial results in the form of < page, rank > pairs which is treated as an input record to reducer function. The reducer performs summation operation. We have incorporated map cache to store the PageRank scores of previous iteration with the assist of setup API available in Hadoop framework. Here we are processing a sub graph; so the map function computes PageRank scores of all the web pages linked together within the sub graph. On the other hand, reduce function performs summation of all the partial results to attain the final PageRank score of each web page. Proposed method takes lesser execution time as compared to previous research [14, 15, 16, 17] done on PageRank algorithm.

Thus the proposed algorithm performs (*a*) graph partitioning using *k*-means clustering technique, (*b*) mapping phase with cache update, (*c*) reducing phase along with message grouping, and (4) then finally check the converged PageRank scores.

The proposed work provides better performance in terms of execution time as compared to traditional PageRank and Local Iterative (LI) PageRank algorithm, with all three implemented on Hadoop framework.

3. EXPERIMENT AND RESULT

The proposed work is experimented on two standard real-world social network datasets. The first dataset is Soc-Live Journal which contains more than ten million nodes and 100 million edges and the second dataset is Twitter which includes nearly 40 million nodes and 1468365182 edges. Both these datasets are taken from "Laboratory for Web Algorithms" [13]. We have performed experiments on a cluster configured with Hadoop version 2.7.1 and the number of machines in this cluster is 5. The configurations of individual machines in a cluster are given below:

Master node-CPU : Intel Xeon processor, E3-1245 @3.30GHz, Core: 4, Processor: 4.

Slave 1/4-CPU : Intel Xeon E3-1245 @3.30GHz Core: 4, Processor: 4.

Slave 2/3-CPU : Intel Xeon E5-2630 @2.30GHz Core: 6, Processor: 12.

Table 1. Description of Data sets [13]

S. No.	*Data sets*	*Description*	*Types*	*Nodes*	*Edges*
1	Soc-Live Journal	Virtual Community Social Site	Directed Graph	11264052	386915963
2	Twitter	Social Networking Site	Directed Graph	41652230	1468365182

We run proposed K-means PageRank algorithm, traditional Page Rank and Local Iterative (LI) Page Rank on Hadoop framework by using the two standard datasets Soc-Journal and Twitter as described in table 1. Running time of all the three algorithms is compared and the results are shown in Table 2, 3 for 10-iteration jobs.

Table 2. Running times of three different PageRank Algorithms and speed up of proposed algorithm for Soc-live Journal dataset

Iteration Number	*Running Time of Traditional PageRank (ms)*	*Running Time of LI-PageRank (ms)*	*Running Time of Proposed K-means PageRank (ms)*	*Speed up b/w Traditional PR & Proposed K-means*	*Speed up b/w LI-PR & Proposed K-meansPageRank*
2	105	68	62	1.69	1.09
4	316	188	178	1.77	1.05
6	534	318	302	1.76	1.05
8	789	468	445	1.77	1.06
10	1008	593	565	1.78	1.04

As it is seen from the table 2, for data set Soc-live Journal, the speed up between Proposed K-means algorithm and Traditional Page Rank comes out to be nearly 1.7 and the speed up between Proposed K-means algorithm and LI-Page Rank comes out to be about 1.05. Thus the Proposed K-means Page Rank algorithm reduces the running time to 44% as compared to Traditional Page Rank and by 8% as compared with LI-Page Rank on Hadoop.

Table 3. Running time of three different PageRank Algorithms and speed up of proposed algorithm for Twitter dataset

Iteration Number	*Running Time of Traditional PageRank (ms)*	*Running Time of LI-PageRank (ms)*	*Running Time of Proposed K-means PageRank (ms)*	*Speed up b/w Traditional PR & Proposed K-means*	*Speed up b/w LI-PR & Proposed K-meansPageRank*
2	202	107	97	2.08	1.13
4	662	338	315	2.10	1.07
6	1025	571	548	1.87	1.04
8	1468	795	762	1.92	1.04
10	1894	1008	987	1.91	1.02

For Twitter data set, as seen from the table 3, the speed up between Proposed K-means algorithm and Traditional Page Rank comes out to be nearly 1.95 and the speed up between Proposed K-means algorithm and LI-Page Rank is about 1.06. Hence, Proposed K-means Page Rank algorithm reduces the running time to 36% compared with Traditional Page Rank and by 10% as compared to LI-Page Rank on Hadoop.

Therefore, it can be concluded from the table 2 and 3 that the proposed K means PageRank algorithm implemented on Hadoop is more efficient in terms of execution time.

4. CONCLUSION

The proposed algorithm improves the running time of PageRank algorithm using MapReduce framework. We have used K-means clustering technique to guarantee the efficiency of communication among different sub graphs. This paper focuses on reducing the cost of communication within the cluster during computation of PageRank values. We run experiments on Hadoop framework using standard real world datasets and concluded that the proposed K means PageRank algorithm provides better results in terms of execution time as compared to traditional PageRank and Local Iterative PageRank.

5. REFERENCES

1. http://hadoop.apache.org/
2. Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified data processing on large clusters", in OSDI, pages 137–150, 2004.
3. Condie T., Conway N., Alvaro P., Hellerstein J.M., Elmeleegy K., Sears R, "Mapreduce online", in: NSDI, pp. 313–328, 2010.
4. S. Brin and L. Page, "The anatomy of a large-scale hyper textual web search engine", in the Proceedings of the 7th WWW Conference.1998
5. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. "The PageRank citation ranking: Bringing order to the web," Technical report, Stanford Digital Library Technologies Project, 1999.
6. W. Zhao, H. Ma, Q. He, "Parallel k-means clustering based on MapReduce", in cloud computing, Springer, Berlin, Germany, pp. 674–679, 2009.
7. Bu Y., Howe B., Balazinska M., Ernst M., "Haloop: Efficient iterative data processing on large clusters", PVLDB 3(1), 285–296, 2010.
8. Kang U., Tsourakakis C.E., Faloutsos C., "Pegasus: A peta-scale graph mining system", in: ICDM, pp. 229–238, 2009.
9. Hema Dubey and B. N. Roy, "An Improved Page Rank Algorithm based on Optimized Normalization Technique", International Journal of Computer Science and Information Technologies, vol. 2, issue 5, pp. 2183-2188, Sep. 2011.
10. Qiuhong L., Wei W., Peng W., "Combination of In-Memory Graph Computation with MapReduce: A Subgraph-Centric Method of PageRank", Springer, Berlin, Heidelberg, pp. 173-178, 2013.
11. B. Perozzi, C. McCubbin, and J. T. Halbert, "Scalable graph clustering with parallel approximate PageRank", Springer-Verlag Wien March 2014, DOI:10.1007/s13278-014-0179-3.
12. A.Cevahir, C.Aykanat, A.Turk, and B. Barla Cambazoglu, "Site-Based Partitioning and Repartitioning Techniques for Parallel PageRank Computation", IEEE Transactions on Parallel and Distributed Systems, vol. 22, issue 5, pp. 786-802, May 2011. DOI: 10.1109/TPDS.2010.119.
13. http://law.di.unimi.it/datasets.php.
14. Malewicz G., Austern M.H., Bik A.J.C., Dehnert J.C., Horn I., Leiser N., Czajkowski G., "Pregel: a system for large-scale graph processing". In: SIGMOD Conference, pp. 135–146, 2010.
15. Arne K., Felix H., Irina A., Fred K., Thomas R., Sebastian S., "Efficiency Experiments on Hadoop and Giraph with PageRank", 24th IEEE Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), Heraklion, pp. 328 – 331, Feb. 2016, DOI: 10.1109/PDP.2016.49
16. Hao Z., Wei Z., Xiaoyu W., Ge F., Zhiyong X., Jizhong H., "A Novel Clustering Algorithm on Large-Scale Graph Data", IEEE International Conference on Cloud Computing and Big Data (CCBD), Wuhan, pp. 47-54, Nov. 2014, DOI: 10.1109/CCBD.2014.23.
17. Ashish R., Suman S., S. P. Ghrera, "Time efficient ranking system on map reduce framework", IEEE Third International Conference on Image Information Processing (ICIIP), Waknaghat, pp. 496 – 501,Dec. 2015, DOI: 10.1109/ICIIP.2015.7414823.