Advances in Economics, Business and Management Research, volume 215
Proceedings of the 2022 7th International Conference on Social Sciences and Economic Development (ICSSED 2022)

PageRank: Graph Processing Using Dataflow to Rank Web Pages According to Importance

Ziyu Huang*

School of Engineering, University of Edinburgh, Edinburgh, EH9 3JU, United Kingdom
**Corresponding author Email: huangziyu2020@163.com*

ABSTRACT

When a huge number of web pages appear, it is difficult for users to find useful information they need. There is a method to solve this problem. It can sort web pages according to their importance. This method is called PageRank in Spark. In this work, we introduced how PageRank works to rank web pages by importance, as well as the purpose and calculation formula of each part of the programming models. What is more, we also discussed the efficiency of PageRank by tackling different sizes of the graph and further applications of PageRank. The conclusion is that PageRank takes less extra time to process larger-size images than it takes to process smaller-size images. In other words, it is efficient for PageRank to tackle large graphs. So, PageRank can be widely used in many aspects.

Keywords: *web graph, resilient distributed datasets, contribution*

1.INTRODUCTION

Nowadays, many companies are based on websites to operate, like search engines -- google, e-commerce operation--amazon. They have over 980 million active users, which generates over 600 TB of data per day.

There are lots of information on the Internet and the content of web pages is more diverse. To sum up and rank such many and diverse web pages is a big challenge for these companies. In this paper, we apply an algorithm called PageRank in Spark. An overall major ranking of every web page can be produced by taking advantage of the website structure.[1] Then, it effectively measures the points that humans are interested in or pay attention devoted to web pages.

2.BACKGROUND

There is a cluster computing framework called Spark, which includes the scalability and fault tolerance of MapReduce [2]. Spark can process data distributed in a cluster of thousands of collaborative physical or virtual servers at a time. There is an algorithm in Spark called PageRank. Let's explain it in specific.

We apply PageRank to measure the comparative significance of web pages. PageRank ranks web pages based on the graph of the web by measuring each vertex in a graph. A probability distribution result can be calculated by PageRank, and this result represents the likelihood that a person clicks the link to another related specific page.

Before the calculation process begins, we assume that all documents in the collection follow an evenly distributed model. PageRank calculation needs to calculate the approximate PageRank value through the set (called "iteration") many times so that the most real value can be obtained. In practice, a PageRank algorithm continues to run. It stops at the time that data converge. Aggregators are used to detecting whether the convergence condition is reached or not. If it reaches, aggregators send a signal to the PageRank algorithm to stop.

The probability range is between 0 and 1. For example, a document shows that the result of PageRank is 0.7. It means that people who click on random links have a 70% chance of being pointed to the document.

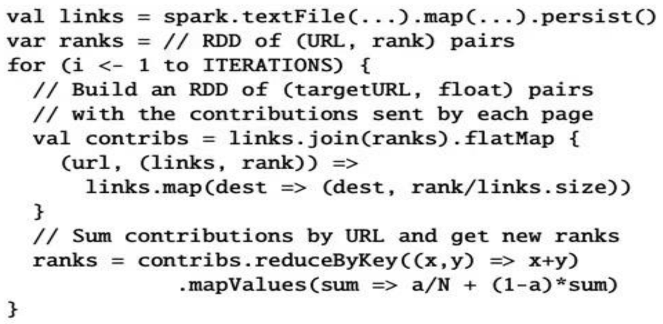
3.PROGRAMMING MODEL

To achieve the function of MapReduce, resilient distributed datasets (RDDs) are introduced. As the name suggests, RDD is a Fault-tolerant immutable Distributed collection of objects of any type and objects can locate in multiple nodes. An RDD only can read data from objects divided by a set of different machines.[3] The result can be computed by a handle that contains various information in reliable storage. In other words, it is easy for RDDs to reconstruct themselves no matter there are some failures or problems with nodes. In Spark, each Scala object represents each RDD.

The code includes three main parts. In terms of 1st part, PageRank reads the data file, and every URL is given a seed value in rank (). The 2nd part is to join the links and ranks together to form a contribution. Then, the 3rd part is to recalculate the ranks based on the previous contribution.

3.1 Reads the data file

Let us explain each part in specific. As shown in figure 1(reading the data file), line 1 sets iteration from the argument. Line 2 read text file into a dataset. It also produces a dataset of strings and transforms it into an RDD with each line in the file being one entire string within RDD. In the following three lines, we split a line into an array of 2 elements according to space and create the parts<ur1, ur1> for each line in the file.



```
val links = spark.textFile(...).map(...).persist()
var ranks = // RDD of (URL, rank) pairs
for (i <- 1 to ITERATIONS) {
  // Build an RDD of (targetURL, float) pairs
  // with the contributions sent by each page
  val contribs = links.join(ranks).flatMap {
    (url, (links, rank)) =>
      links.map(dest => (dest, rank/links.size))
  }
  // Sum contributions by URL and get new ranks
  ranks = contribs.reduceByKey((x,y) => x+y)
             .mapValues(sum => a/N + (1-a)*sum)
}
```

Figure 1 read data

3.2 Calculate contributions and recalculating ranks

For part2 and part3, it calculates contributions and recalculates Ranks. Every page has several forward links and backlinks. (The definition of a backlink is a link from one website to another. Backlinks help search engines find and index your content for search results. If backlinks pointing to your website is high-quality, your target keywords rank higher than others) Generally, we regard highly linked pages as a more important one. A contribution of r/n is sent to its neighbors by each document (r stands for rank and n stands for the total number of neighbors). Then, rank should be recalculated by using the formula α/N+(1-α) Σci.[4] Ci represents that all contributions add together. We define α as 0.15. So, the total rank formula is 0.15 + 8.5*contribution. For the next step, it iterates to step 2 with a new rank.

In the flow chart below (Figure 2), there is a whole process. The link RDD and the rank RDD are joined together to form RDD1.RDD2 is only the value part of RDD1 and when RDD2 is flat mapped, it generates RDD contrib1. In the next step, it integrates the same key, and we get RDD3. At this time, we use the formula (new rank = 0.15 + 0.85 * contribution) to get a new round of RDD rank (RDD rank 1). Next, RDD rank 1 and RDD links are combined to start a new round of calculations.

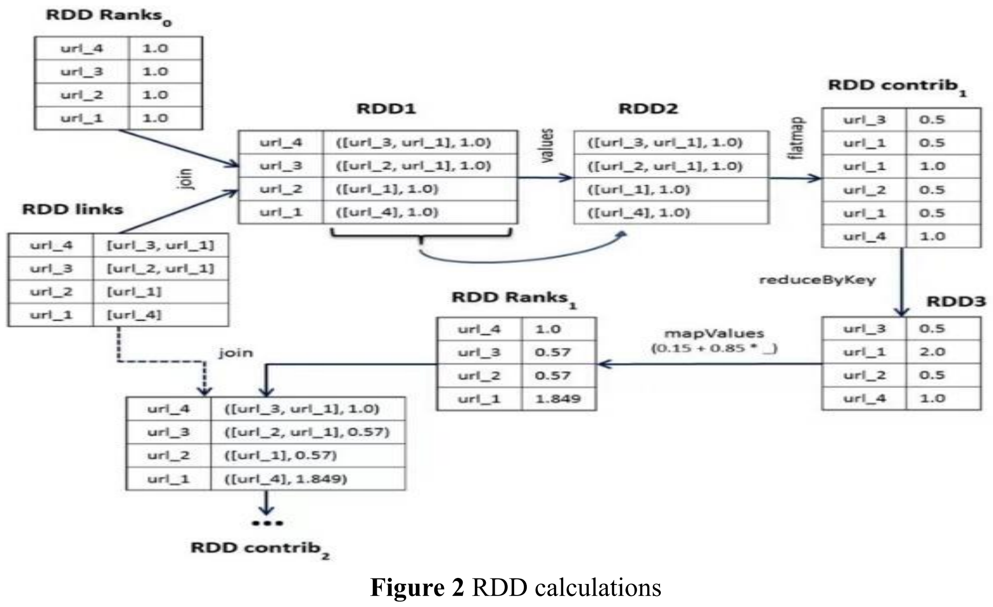


Figure 2 RDD calculations

3.3 Example

We take the first two iterations as examples (as shown in Figure 3). In terms of Iter#1 in Figure 3, four URLs have the same value (1.0).URL_1 receives three forward links from URL_2, URL_3 and URL_4. At the same time, it gives one forward link to URL_4. URL_3 and URL_4 split their contributions in half. The contributions of URL_3 are given to URL_1 and URL_2, while URL_4 gives its contributions to URL_1 and URL_3. Since the exchange of contributions for each URL, there exit some new values of contributions. URL_3 has 2 forward links to URL_2 and URL_1. Then, we can calculate the contribution of URL_3 is 0.5(= 1/2). According to the rank formula, we can get rank as 0.57(= 0.15 + 0.85 * 0.5). This result passes to the next contribution cycle. In this case, the value of URL_2 is 0.57 for second interaction. The contribution of URL_3 splits in half, which is 0.285(= 0.57 /2). So, the value of URL_1 is 1.85(=1 + 0.57 + 0.285). Because URL_2 passes the whole 0.57 rank to URL_1 and URL_1 also receives 0.285 contribution from URL_3. By adding these values together, the final result of URL_1 is 1.85. This process is called the contribution cycle. For each cycle, the URL passes its contribution to its neighbors, and the values of each URL change.

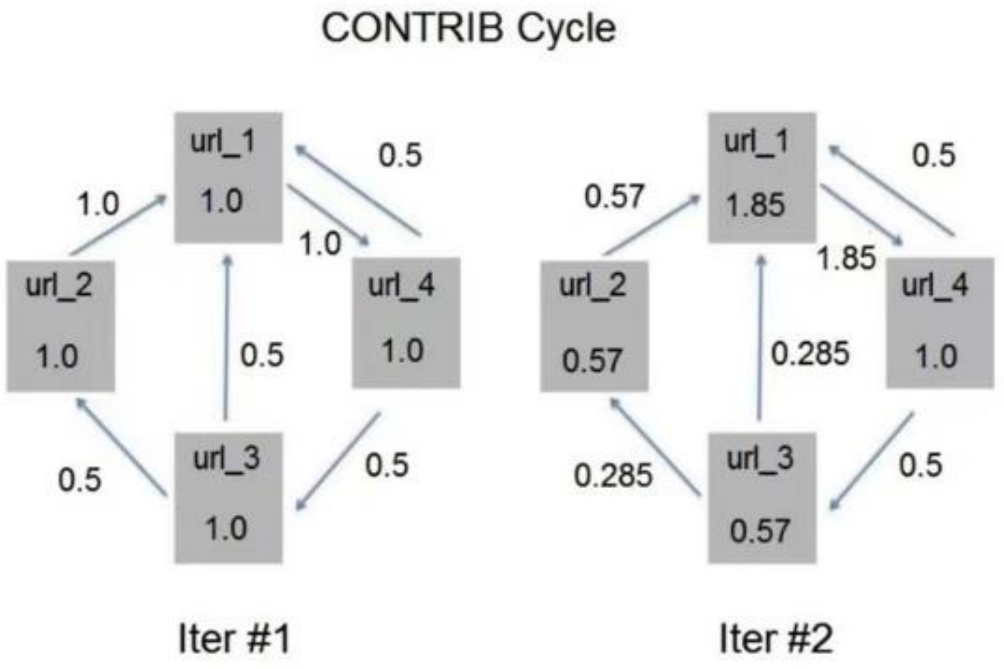


Figure 3 contribution cycle

4.EVALUATION

In order to detect the efficiency of PageRank. We designed several groups of experiments. See how long PageRank takes to process different sizes of graphs. For example, if we double the size of the graph, we need to observe whether the processing time of PageRank has also been doubled.

In the original graph, there are only 4 nodes. The PageRank only takes 0.65324s to complete a calculation. As we expand the graph size to 8 nodes, it spends 0.665832s to finish. By repeating the above steps, we get serious data and I plot a table below (Table 1).

Table 1 processing time

Size of graph	time
4	0.65324s
8	0.665832s
16	0.702659s
32	0.732467s

According to the table 1, we can see that period changes nonlinearly with the size of the graph. As the graph increase in size, there is a slight difference between the period. When the graph increases from 4 to 8, it only takes 0.0126s longer than before. As the graph grows to 8, period increases by 0.0368s. So, it is efficient for PageRank to tackle large graphs.

5. APPLICATION

PageRank can act as a Backlink predictor. In other words, PageRank performs better at predicting than citation counting, even if the prediction object is the citation number itself.[5] Because it is more likely that citation counting lurches in local collections than PageRank. Thus, it needs to spend more time to extend the range and find more frequently excerpted pages in other areas. But PageRank has some links and it can quickly map the citation structure of the web. We can say that it is the best choice to use PageRank to count the citation approximation since it is high efficiency.

PageRank provides a good way for users to judge the authority and authenticity of website content. For instance, a user may be favorably disposed toward to trust information that is directly quoted from official Wikipedia than other webs. The links in the Wikipedia age are trustworthy for the user.

6. CONCLUSION

In this paper, Spark is used for graph processing, and PageRank is one of the algorithms in Spark. The main function of PageRank is to rank all web globally only based on the location of the Web's graph structure. So, more important and valuable web pages can be placed in the front seat by using PageRank. The whole working process of the programming model includes 3 steps. Firstly, data of graphs can be read from the file and then join the links and ranks together to form a contribution. Finally, recalculate the rank to get the right answer. We also discuss the effectiveness of PageRank when the size of the graph becomes larger. The larger the graph is, the smaller the extra time is needed.

There are lots of applications that PageRank can achieve, such as Backlink predictor or authority judger. Personalized PageRank can also be designed to sort web pages for a specific perspective.

REFERENCES

[1] X.Y. Huang, X.B. Zhu, K.L. Xu, and J.H. Wu. (2014) Research Paper Influence Measurement and Applications: A Machine-Learning-Based Approach. Advanced Materials Research, 1049-1050.

[2] Srirama, S.N., Jakovits, P., Vainikko.E(2011) Adapting scientific computing problems to clouds using MapReduce. Future Generation Computer Systems,184-192

[3] Sargolzaei,P. , Soleymani,F., International Mathematical Forum(2010), no. 19, 937 – 956.

[4] Zaharia, M. (2016) An Architecture for Fast and General Data Processing on Large Clusters. Association for Computing Machinery and Morgan & Claypool

[5] (1998) The PageRank Citation Ranking: Bringing Order to the Web. http://www.m-hikari.com/imf-2010/17-20-2010/soleymaniIMF17-20-2010.pdf