International Journal of Grid Distribution Computing
Vol.8, No.2 (2015), pp.273-284
http://dx.doi.org/10.14257/ijgdc.2015.8.2.24

A Parallel PageRank Algorithm with Power Iteration Acceleration

Chun Liu[1] and Yuqiang Li [2]

[1,2]School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China
[1]liuchun_0206@163.com, [2]liyuqiang@whut.edu.cn

Abstract

Based on the study about the basic idea of PageRank algorithm, combining with the MapReduce distributed programming concepts, the paper first proposed a parallel PageRank algorithm based on adjacency list which is suitable for massive data processing. Then, after examining the essential characteristics of iteration hidden behind the PageRank, it provided an iteration acceleration model based on vector computing. Following, using such acceleration model, the paper again brought forward a parallel PageRank algorithm with power iteration acceleration with MapReduce. Finally, after abundant experimental analyses, it has been proved that the both the two proposed algorithm can be suitable for massive data processing and the 2nd one can significantly reduce the numbers of iteration and improve the efficiency of PageRank algorithm.

Keywords: *PageRank, Power Iteration Acceleration, Parallel, MapReduce*

1. Introduction

The MapReduce [1], GFS [2] and BigTable [3] proposed by Google have made an indelible contribution to the development of cloud computing technology. In particular, the MapReduce distributed programming model provides a new solution to large data sets parallel computing because of its characteristics of simple, scalability, and easy to implement load balancing [4].

In the field of Web structure mining, the PageRank algorithm proposed by S, Brin and L, Page [5] is by far the most successful example. The algorithm quotes the web pages sorting ideas based on site topology, analyses the links between pages, calculates the importance degrees of pages and then get the ranking of each page. Such algorithm can get more accurate and objective results as the full account of the topology between pages. With the advent of Web2.0, the traditional serial PageRank algorithm cannot meet the growing demands for massive data calculations; the study of parallel PageRank algorithm is an important research direction in the era of big data.

2. PageRank Survey

2.1. PageRank Introduction

PageRank algorithm uses the ideas of citation analysis, studies the link structures between pages and analyses the importance degrees of web pages. The core idea is based on the following two assumptions [6]: (1) The hyperlink from one page to another is a kind of implicit recognition to the authority of the target page; (2) The web page pointing to other web pages, itself also has the authority values.

Based on the above two assumptions, the definition of PageRank value is given as following:

Definition 1: It's supposed that i and j are represented two web pages; Out_i is represented the link-out pages collection of web page i, In_i is represented the link-in pages collection of web page i; and Oi is represented the count of pages in $Out_{i.}$ So here comes the formula (1).

$$p(i) = \sum_{j \in In_i} \frac{p(j)}{Oj} \qquad (1)$$

Formula (1) shows that the PageRank value of web page **i** is determined by the PageRank values of all the link-in web pages of web page **i**. If web page **j** is one of the link-in web pages of web page **i**, and the count of link-out web pages of web page **j** is **Oj** ,then the contribution

PageRank value of web page j is $\frac{p(j)}{Oj}$ 。

In order to describe the calculating nature of the PageRank algorithm behind, formula (1) can be written in the form of Matrix-vector multiplication, and the PageRank algorithm can be defined in a new form.

Definition 2: P is represented an n-dimensional column vector of PageRank value, $P=(p(1),p(2),\ldots,p(n)^T)$;

$A=(a_{ij})_{n*n}$ is represented the Web Link Matrix, $a_{ij} = \begin{cases} 1, & \text{web page i has link to web page j} \\ 0, & \text{web page i hasn't link to web page j} \end{cases}$;

$M=(m_{ij})_{n*n}$ is represented the Probability transition matrix, $M=(m_{ij})_{n*n}=(\frac{a_{ij}}{\sum_{j=1}^{n} a_{ij}})_{n*n}$,

Then, the formula (1) can also be represented in the form of formula (2):

$$P_k = M^T P_{k-1} \qquad (2)$$

Formula (2) uses the Power iteration method to calculate the result: First, P_0 is given at will; then iteratively calculates $P_k= M^T P_{k-1}$, until $|P_k-P_{k-1}|<\varepsilon$; finally, when the iteration has finished, the P_k is the final result.

Actually, Matrix M is a Markov state transition matrix, so the matrix iterative convergence must satisfy two conditions : (1) the web graph is strongly connected ;(2) there are no dead ends in web graph, which makes the final results of PageRank value tend toward zero. Therefore, in actual application process, considering both the random surfer model and the existence of dead ends, Google gives the improved model of PageRank algorithm, which is illustrated as Formula (3).

$$P_k = \beta M^T P_{k-1} + \frac{(1-\beta)}{n} E \qquad (3)$$

In Formula(3), β is coefficient usually 0.85.E is an n-dimensional unit column vector, n is the total count of web pages in Web graph, $\beta M^T P$ is represented that the random surfer chooses one link-out page of the current page with the probability of β , $(1-\beta)E/n$ is represented the random surfer choose random page to visit with the probability of $(1-\beta)$. Formula (3) is adopted to calculate the PageRank value in this paper.

2.2. PageRank Research Statuses

Though PageRank algorithm is successful, but is not perfect. PageRank algorithm currently has two major problems:

(1) With the rapid development of the Web, the total number of Web pages grows exponentially; traditional serial PageRank algorithm has been unable to meet the computing needs of massive data on performance.

(2) PageRank algorithm use power iteration method to calculate the PageRank value, with the natures of numerous iterative times, large time consumption, together with slow execution speed and convergence rate.

So a lot of domestic and international scholars have done a lot of researches and improvements on PageRank algorithm according to its shortcomings.

According to the facts that in web links graph the vast majority of links are inner-links within the server, and the page rank calculated by local PageRank algorithm is almost the same with one calculated by global PageRank algorithm, WANG [7] in university of Wisconsin – Madison put forward a distributed PageRank algorithm that in distributed environment after different sites independently calculating its “local PageRank” , all of the “local PageRank” are combined to form the “server PageRank” which is the unique rank of hyperlinks. The experiment results show that the performance of distributed PageRank algorithm is better than centralized. Using distributed parallel ideas to reform the PageRank, Wang’s algorithm breaks through the bottleneck of serial PageRank algorithm, and provides a solution for big data processing. But the distributed algorithm proposed by WANG is not based on any kind of distributed platform. So, in the process of algorithm design the communication between sites should be taken into account, which is the core issue in distributed system, and is difficult to implement.

Sepandar [8] of Stanford University found that pages have different converging speeds in PageRank algorithm. Most pages converge fast, while there is a small part of the pages taking a relatively long time to converge. And those pages converging slow usually have higher PageRank value. So Sepandar proposes an adaptive acceleration PageRank algorithm that the converged pages no longer are recalculated in each iteration. Experimental data show that the adaptive PageRank algorithm can improve the efficiency of traditional PageRank algorithm by 30%. But, such adaptive algorithm although can improve the efficiency by reducing calculation, but it cannot reduce the number of iterations. In a distributed parallel system, the IO time consumption of iteration is bigger than that of calculation.

Using the power method, Wu Jiaqi [9] in Fudan University, proposes a modified linear extrapolation method to calculate the PageRank. It uses the characters of the 2^{nd} eigenvalue of Google matrix to get acceleration model which makes the power method achieve fast convergence。 But Wu’s algorithm accelerates the vector P as a whole, so it essentially is a serial algorithm which is not suitable for parallel computing.

Because of the web link data showing massive features, combining with the MapReduce distributed programming concepts the paper first proposes a parallel PageRank algorithm based on Power Method. Then, after examining the essential characteristics of iteration hidden behind the PageRank algorithm, it provides a distributed iteration acceleration model. The algorithm can accelerate the PageRank computing through applying the distributed acceleration model to each component pi of vector P. The experimental results show, the proposed algorithm can significantly reduce the number of iterations without increasing the time and space consumption, so it can improve the execution efficiency of PageRank algorithm. At the same time, it is suitable for parallel computing because the acceleration model is applied to each component pi of P which computes individually。 This algorithm uses MapReduce distributed design ideas without much thought to the details of distributed communication, so it is easy to implement and provides an improved ideas for other vector iteration algorithms such as HITS.

3. PageRank Parallel Algorithm based Power Method

As previously mentioned, For Web graph containing n nodes, typically a matrix M of n*n is designed to save the relationships between nodes. As for each element in M_{n*n}, m_{ij}=1, shows that the web page i has link to web page j. while m_{ij}=0, shows that the web page i has no link to web page j. But in actually web structure graph, n is huge, so the expression and preservation of M is difficult. And M is a huge sparse matrix, a lot of zero elements which are unnecessary in PageRank algorithm calculation should be saved in the form of adjacency matrix. In this paper, the sparse matrix M is expressed and preserved with the ideas of adjacency list in data structure. The data are saved with text format file, every row in adjacency matrix is saved as one line in the text format file, with the format of sURL: tURL1, tURL2,…,tURLm. The format expresses that the sURL has m link-out tURLs, those are (sURL, tURL1), (sURL, tURL2),…, (sURL, tURLm); Here m is the count of link-out links， statistics show the average value of m is 7. The storage space of m can greatly reduces through this improvement.

When calculating the PageRank value of one page, according to fomular (2) and formular (3), it is found that the core of calculation is to complete the multiplication between matrix and vector. According to the principle of matrix-vector multiplication, each row of the matrix and the vector needs to be multiplied. As for one page, first the contribution PageRank value of every link-in page to the page should be calculated, then all of the contribution values should be accumulated, finally, amendment is done to the sum to get the new PageRank value. According to such thinking, it is need to load the PageRank values of all pages, that is the whole vector P, into memory. But when the number of components of vector P is too enough to load into the memory, the calculation will fail.

Here uses a reverse thinking. As for each page, the PageRank value is determined by the contribution PageRank values provided by its link-in pages, in other words, it also has contribution to every link-out page of its. So as for one page, only the contribution PageRank value to its link-out pages is needed to calculate. For all pages has completed its contribution value calculation to its link-out pages, the calculation values that the link-in page give to the page has already been calculated. Then all the calculation values that different link-in pages give to the page are collected and accumulated to get the new PageRank value of the page. Under the guidance of this idea, the whole vector P is not needed to load into memory, only one component pi is processed in each process.

Matrix M and P are saved together in file F. The format of F is like this: sURL, PR: tURL1, tURL2,…, tURLm。PR represents the PageRank value of sURL。Through the above analysis，here is the parallel PageRank algorithm based on power method - algorithm 1.

```
Algorithm 1- parallel PageRank algorithm based on power method
Step1: Algorithm begins. Input coefficient β, the total number of nodes in graph n, and iteration threshold ε.
Step2: Parallel computing starts.
    Step2.1: (Map)
        (1) Read one line in F, "sURL", "$PR_k$" and "tURL1,tURL2,…,tURLm " are separated;
        (2) Use the "Out" to represent the "tURL" statistics count of "sURL";
        (3) as for one "sURL", calculate the contrubition PageRank value to each target URL of the source URL: sURLpart= $PR_k$/Out;

    (4) as for every "tURLi", output the result as the format like<tURLi, $PR_k$ /Out>, at the same time, output the source URL and its PageRank value which will be used as template later, the format is <sURL, $PR_k$ >.
    Step2.2: (Reduce)
    (1) The output data are aggregated by "tURLi", and all the part PR value of the same tURLi are added up together here. Then there are the <tURLi, sum($PR_k$ /Out)> and <sURL, $PR_k$ > passed over from Map Job in this processing node. It is worth mentioning
```

that the "tURLi" and "sURL" are the same in value;

(2) Use the formular $PR_{k+1} = \beta * \mathrm{sum}(PR_k/Out) + (1-\beta)/n$ to calculate the new PageRank value of the sURL, as PR_{k+1};

(3) Abstract the old PageRank value of the "sURL"from the template <sURL, PR_k > as PR_k ; Compare PR_k and PR_{k+1}, if the difference between PR_k and PR_{k+1} is less than ε, then it illustrates that the convergence value of the currently being calculated page has been reached, so the value of Counter increases1。

（4）Construct the format < sURL, (PR_{k+1}, tURL1, tURL2,…,tURLm)> as output data and write them into file F for next iteration.

Parallel computing ends.

Step3:Read the value from the Counter, if it is less than n, thus it illustrates that there are still some nodes have not reached its convergence value, then the algorithm jumps to step 2 for the next round parallel iterative operation. If it is equal to n, thus it illustrates all nodes have already achieved the condition of convergence, the algorithm ends. The final results are saved in file F.

4. Parallel PageRank Algorithm Based on Power Iteration Acceleration

The essence of PageRank algorithm is using the power iteration method to solve the eigenvectors, which is corresponding with the eigenvalues of the state matrix. The power iteration method merit is that the algorithm is simple and easy computer realization, but its shortcomings are slow convergence rate and low execution speed. The acceleration methods commonly used by the power method are origin displacement method, Rayleigh quotient acceleration method and so on. But it is more difficult to select the amount of displacement P in origin displacement method, and the matrix M is required to be a symmetric matrix in Rayleigh quotient acceleration method. So the two methods both are not suitable for the Power computing of PageRank. During solving nonlinear equations in real domain (or complex field), Aitken promoted an acceleration method [10, 11]: Assume the iterative function is x = φ (x), given x_k, and use the iterative function twice to get $x_{k+1} = \varphi(x_k)$, $x_{k+2} = \varphi(x_{k+1})$, using the accleration model $x_k^* = x_{k+2} - \frac{(x_{k+2}-x_{k+1})^2}{x_{k+2}-2x_{k+1}+x_k}$, then the sequence $\{x_k^*\}$ has a higher convergence rate than the sequence $\{x_k\}$. So the paper wants to find a new method to solve the approximate solution of linear equations by using the Aitken acceleration technology to the vector field.

4.1. Power Iteration Acceleration

In PageRank algorithm, the eigenvalues of state transition matrix M satisfies the inequality: $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$, and the corresponding eigenvectors u_1, u2, … , un are linearly independent. Then random non-zero vector $v^{(0)}$ can be linearly expressed with $u_1, u_2, \ldots, u_n$, that is $v^{(0)} = a_1u_1 + a_2u_2 + \cdots + a_nu_n$, not all $a_1, a_2, \ldots a_n$ are zero。 Also because the maximum eigenvalues of matrix M satisfies the equation: $\lambda_1 = 1$, then the iterative sequence $\{v^{(k)}\}$ satisfies the following equations.

$$v^{(1)} = M * v^{(0)} = u_1 + a_2\lambda_2u_2 + \cdots + a_n\lambda_nu_n$$

$$\cdots\cdots\cdot$$

$$v^{(k)} = M^k * v^{(0)} = u_1 + a_2\lambda_2^ku_2 + \cdots + a_n\lambda_n^ku_n$$

Because there is the inequality: $1 > |\lambda_2| > \cdots . > |\lambda_k|$, when the k becomes more and more big, $\lim v^{(k)} = \lim M^k * v^{(0)} = u_1$。 Therefore, in other words, when k is big enough, u_1 can be regarded as the approximate estimate of $v^{(k)}$.

Here only uses the first two eigenvectors u_1, u_2 to estimate $v^{(k)}$. When k is more and more big, this estimate will be more and more precise. Then has the following formula:

$$v^{(k)} = u_1 + a_2u_2 \qquad (4)$$

$$v^{(k+1)} = Av^{(k)} = u_1 + a_2\lambda_2 u_2 \quad (5)$$
$$v^{(k+2)} = A^2v^{(k)} = u_1u_1 + a_2\lambda_2{}^2u_2 \quad (6)$$

Define the two variables: $T1i = v_i{}^{(k+2)} - v_i{}^{(k+1)}$, $T2_i = v_i{}^{(k)} + v_i{}^{(k+2)} - 2v_i{}^{(k+1)}$, $v_i{}^{(k)}$ represents the ith component of vector $v^{(k)}$。

$$T1_i^2 = a_2{}^2\lambda_2{}^2(\lambda_2 - 1)^2\,(u_2)_i{}^2 \quad (7)$$
$$T2i = a_2(\lambda_2 - 1)^2(u_2)_i \quad (8)$$

Formula (7) divides formula (8) and gets the formula (9) ：

$$\frac{T1i2}{T2i} = a_2\lambda_2{}^2(u_2)_i \quad (9)$$

Associate formula (6) with formula (9) , and get formula (10).

$$(u_1)_i = v^{(k+2)}{}_i - a_2(u_2)_i = v^{(k+2)}{}_i - \frac{T1_i{}^2}{T2_i}$$
$$= v^{(k+2)}{}_i - \frac{\left(v_i{}^{(k+2)} - v_i{}^{(k+1)}\right)^2}{v_i{}^{(k)} + v_i{}^{(k+2)} - 2v_i{}^{(k+1)}} \quad (10)$$

Here we get the vector acceleration model which is similar to the real number field. This acceleration model is applied to each component $(u_1)_i$ of vector u_1 for accelerating computation separately. When the value of each component has reached the iteration condition, vector u_1 tends to steadily. One constraint condition of this model is that the computed result of this formula $v_i{}^{(k)} + v_i{}^{(k+2)} - 2v_i{}^{(k+1)}$ cannot be zero. If the computed result of this formula $v_i{}^{(k)} + v_i{}^{(k+2)} - 2v_i{}^{(k+1)}$ is zero, $v_i{}^{(k)}$ stops accelerating in this round. Also, the accelerating computation of each $(u_1)_i$ is unrelated to each other, so that the accelerating method can be implemented in parallel.

4.2. Parallel PageRank Algorithm Based On Power Iteration Acceleration

Based on the front discussions, through the acceleration model applied to each component pi of the P vector, namely that the new $p_k i$ is calculated by $p_k i$, $p_{k+1}i$ 和 $p_{k+2}i$ according to the formular $p^{(k)}{}_i{}' = p^{(k+2)}{}_i - \frac{(p^{(k+2)}{}_i - p^{(k+1)}{}_i)^2}{p^{(k)}{}_i + p^{(k+2)}{}_i - 2p^{(k+1)}{}_i}$, and combined with algorithm 1, the paper gets the parallel PageRank algorithm based on Power Iteration Acceleration - algorithm 2.

Algorithm 2- parallel PageRank algorithm based on Power Iteration Acceleration

```
Step 1: Algorithm begins. Input coefficient β, the total number of nodes in graph n, and iteration threshold ε.
Step 2: Parallel computing starts.
    Step 2.1:The first iterative computation $P_{k+1}i$
    Step2.1.1（Map1）:
    (1) Read one line in F, “sURL”, “$PR_k$” and “tURL1, tURL2,…,tURLm ” are separated;
    (2) Use the “Out” to represent the “tURL” statistics count of “sURL”;
    (3) as for one “sURL”, calculate the contrubition PageRank value to each target URL of the source URL: sURLpart= $PR_k$/Out;
    (4) as for every “tURLi”, output the result as the format like<tURLi, $PR_k$ /Out>, at the same time, output the source URL and its PageRank value which will be used as template later, the format is <sURL, $PR_k$ >.
    Step 2.1.2（Reduce1）:
    (1) The output data are aggregated by “tURLi”, and all the part PR value of the same tURLi are added up together here. Then there are the <tURLi, sum($PR_k$ /Out)> and <sURL, $PR_k$ > passed over from Map Job in this processing node. It is worth mentioning that the “tURLi” and “sURL” are the same in value;
```

(2) Use the formular $PR_{k+1}=\beta * \mathrm{sum}(PR_k/\mathrm{Out}) + (1-\beta)/n$ to calculate the new PageRank value of the sURL, as PR_{k+1};

(3) Construct the output format < sURL, (PR_k,PR_{k+1},tURL1,tURL2,…,tURLm)> and write it to file F for next MaprReduce job.

Step 2.2: the second iterative computation $P_{k+2}i$, using the accelerating model to modify $P_k i$'

Step 2.2.1（Map2）:

(1) Read one line in F, "sURL", "PR_k", "PR_{k+1}" and "tURL1,tURL2,…,tURLm " are separated;

(2) Use the "Out" to represent the "tURL" statistics count of "sURL";

(3) as for one "sURL", calculate the contrubition PageRank value to each target URL of the source URL: sURLpart= PR_{k+1}/Out;

(4) as for every "tURLi", output the result as the format like<tURLi, PR_{k+1} /Out>, at the same time, output the source URL and its PageRank value which will be used as template later, the format is <sURL, PR_{k+1} >.

Step2.2.2（Reduce2）:

(1) The output data are aggregated by tURLi, and all the part PR value of the same tURLi are added up together here. Then there are the <tURLi, sum（PR_{K+1}/Out）> and the template <sURL, (PR_k , PR_{k+1})> passed over from Map in this processing node. It is worth mentioning that the tURLi and sURL are the same.

(2) Use the formular $PR_{k+2}=\beta * \mathrm{sum}(PR_{k+1}/\mathrm{Out}) + (1-\beta)/n$ to calculate the new PageRank value of the sURL, as PR_{k+2};

(3) Abstract the PR_k, PR_{k+1} from the template <sURL, (PR_k , PR_{k+1})>, and set $P^{(k)}_i = PR_k$, $P^{(k+1)}_i = PR_{k+1}$, and $P^{(k+2)}_i=PR_{k+2}$. Then use the acceleration model formula $p^{(k)}{}_i{}' = \frac{\left(p^{(k+2)}{}_i - p^{(k+1)}{}_i\right)^2}{p^{(k)}{}_i + p^{(k+2)}{}_i - 2p^{(k+1)}{}_i}$ to calculate the new $P^{(k)}_i$, as $(P^{(k)}_i)'$. If the formula $p^{(k+2)}{}_i - 2p^{(k+1)}{}_i + p^{(k)}{}_i$ is equal to 0, then $p^{(k)}{}_i{}' = \frac{\left(p^{(k+2)}{}_i - p^{(k+1)}{}_i\right)^2}{p^{(k)}{}_i + p^{(k+2)}{}_i - 2p^{(k+1)}{}_i}$, else $p^{(k)}{}_i{}' = p^{(k+2)}{}_i$。Thus set $PR_{k+1}=p^{(k)}{}_i{}'$

(4) Compare $p^{(k)}{}_i{}'$ and $p^{(k+1)}{}_i$, if the difference between $p^{(k)}{}_i{}'$ and $p^{(k+1)}{}_i$ is less than ε, then it illustrates that the convergence value of the currently being calculated page has been reached, so the value of Counter increases1。

(5) Construct the output data as format < sURL, (PR_{k+1},tURL1,tURL2,…,tURLm)> and write them into file F for next iteration.

Parallel computing ends.

Step 3: Read the value from the Counter, if it is less than n, thus it illustrates that there are still some nodes have not reached its convergence value, then the algorithm jumps to step 2 for the next round parallel iterative operation. If it is equal to n, thus it illustrates all nodes have already achieved the condition of convergence, the algorithm ends. The final results are saved in file F.

5. The Discussion of Experimental Results

In the laboratory using 8 machines with different models and configurations (desktops and laptops), a Hadoop distributed computing environments is set up. Each computer has been installed Ubuntu 12.04 Linux operating systems, Java Development Kit JDK1.7.045 and hadoop1.0.3. Algorithm 1 and 2 will be carried on in the distributed computing environment, in order to verify whether algorithm 2 can effectively reduce the number of iterations, and enhance the algorithm efficiency.

5.1. Experimental Scheme Design

The algorithm will be verified in two aspects: (1) the number of iterations and the efficiency of implementation; (2) the suitability of algorithm for mass data processing.

In aspect one, algorithms 1 and 2 will calculate the same data set, and the calculated results will be compared to verify whether the algorithm 2 can reduce the number of iterations and enhance the efficiency of algorithms than algorithm 1;

In aspect two, three different data sets sogouT-mini, sogouT-reduce and sogouT-link will be chosen from Sogou laboratory (http://www.sogou.com/labs/dl/t-link.html), and algorithm 1 and 2 will be carried out with the three datasets respectively. The result calculated from the experiments can show whether algorithm 1 and 2 can effectively meet the processing requirements of big data sets.

5.2. Experimental Results Analysis

1. The Analysis of Iteration Numbers and Execution Efficiency

A web graph with 7 nodes is designed, whose link relationships are shown in Figure 1. In order to better test the new algorithm can reduce iteration numbers of PageRank, the iteration numbers are increased through lowering the threshold from 1E-1 to 1E-7. Experimental iteration numbers and run time of the algorithm 1 and 2 are recorded, as shown in Table 1. The comparison analysis about iteration numbers of two algorithms is shown in Figure 2, and the comparison analysis about run time of two algorithms is shown in Figure 3.

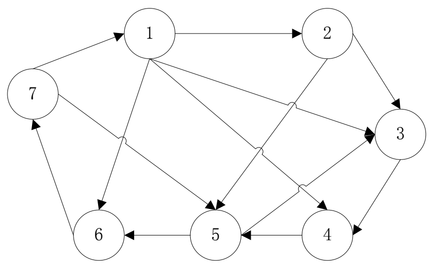

Figure 1. Experimental Web Graph Data Set

Table 1. The Run Results

threshold	Algorithm 1		Algorithm 2	
	Iteration Numbers	Run time (s)	Iteration Numbers	Run time (s)
1.00E-01	3	22	1	15
1.00E-02	15	107.1	6	92.3
1.00E-03	30	213.5	10	141.6
1.00E-04	44	312.3	16	249.5
1.00E-05	58	411.5	21	325.7
1.00E-06	72	510.8	25	395.6
1.00E-07	87	617	33	495.2

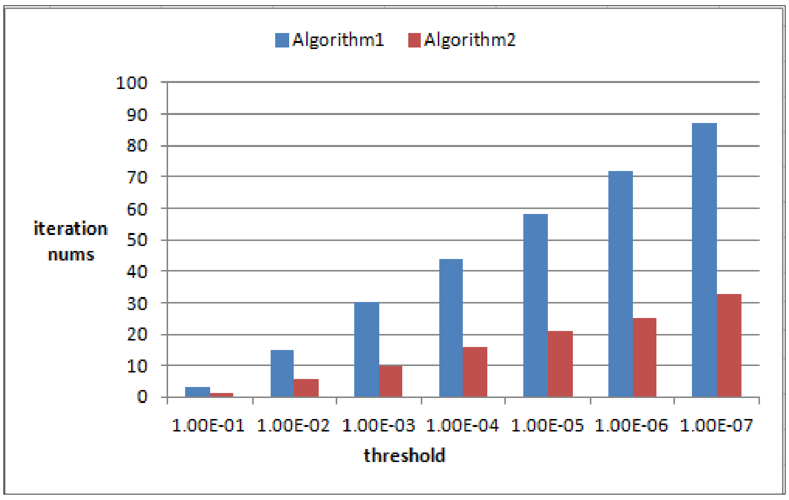

Figure 2. Comparison of Iteration Numbers

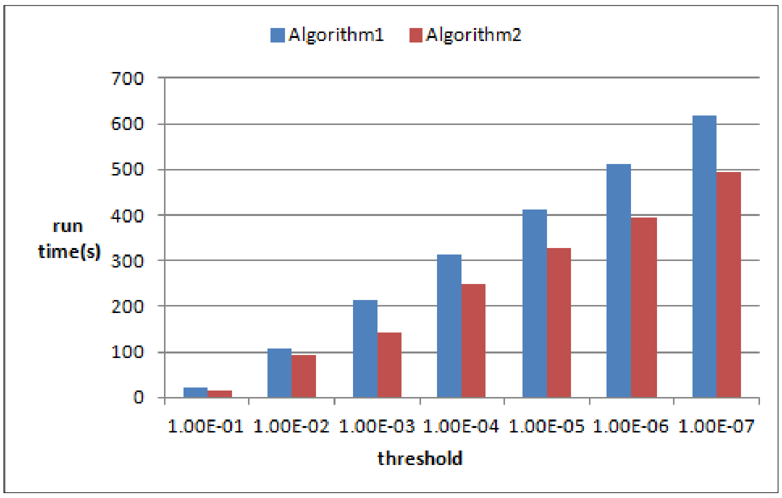

Figure 3. Comparison of Run Time

As shown in Table 1, Figure 2 and Figure 3, the following conclusions can be drawn.

（1）Through the application of acceleration model, algorithm 2 can effectively reduce the iteration numbers of PageRank algorithm, with the rate of 50%-70%.

（2）For a given data set, one iteration in algorithm 1 costs approximately 7 seconds, while algorithm 2 takes about 15 seconds, because that the algorithms 2 requires one more iteration to get $P_{k+2}i$, but on the whole, the algorithm 2 shortens the run time and improves the efficiency compared to the algorithm1.

2. The Suitability of Two Algorithms for Mass Data Processing

Select three datasets which are sogouT-mini (with 5,604 nodes), sogouT-reduce(with 93,443 nodes) and sogouT-link(with 3,537,374 nodes) to run algorithm 1 and 2 separately. Original PageRank values are given by 0.5, and the convergence threshold is set to 1E-1.

The experimental iteration numbers and run time result of sogouT-mini is shown in Table 2; sogouT-reduce, Table 3, and sogouT-link, Table 4. The comparison analysis about iteration numbers of two algorithms is shown in Figure 4, and the comparison analysis about run time of two algorithms is shown in Figure 5.

Table 2. The Result of SogouT-mini

sogouT-mini (5604 nodes)	Run Time(s)	Iterantion Numbers
Algorithm 1	70	10
Algorithm 2	56	4

Table 3. The Result of SogouT-Reduce

sogouT-reduce (93443 nodes)	Run Time(s)	Iterantion Numbers
Algorithm 1	369.2	52
Algorithm 2	308.5	22

Table 4. The Result of SogouT-link

sogouT-link (3537374nodes)	Run Time(s)	Iterantion Numbers
Algorithm 1	17719.2	70
Algorithm 2	14674.8	29

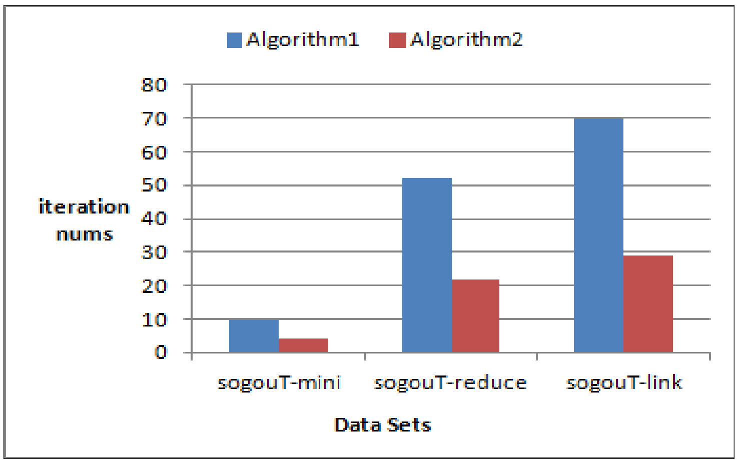

Figure 4. Comparison of Iteration Numbers

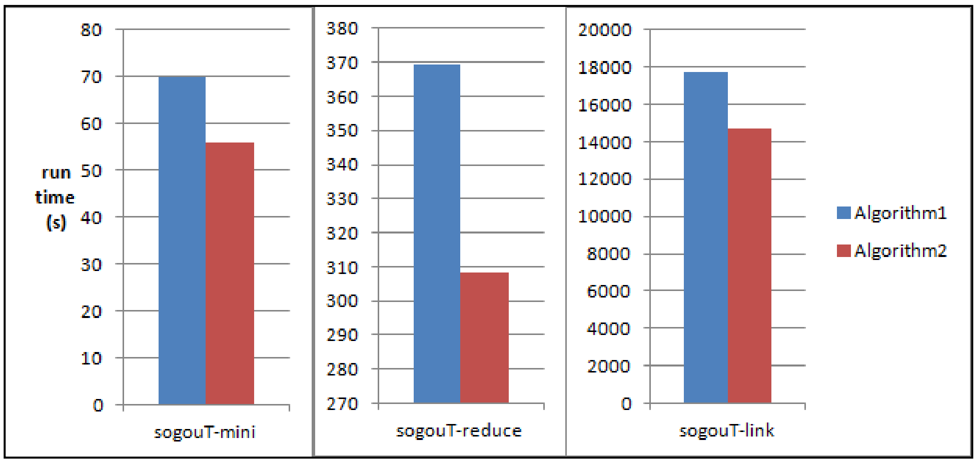

Figure 5. Comparison of Iteration Numbers

From Figure 4 and Figure 5, it can be found:

（1）With increasing data sets (the sogouT-link data set is close to the G magnitude), both algorithm1 and algorithm2 are able to carry out its operations effectively, highlighting the advantages of parallel algorithms

（2）For a variety of practical data sets, algorithm 2 in terms of the iteration number and run time of the algorithm were less than the algorithm 1, so the proposed algorithm 2 can effectively accelerate the computation of PageRank algorithm, and reduce execution time by 20%.

6. Conclusion

The parallel PageRank algorithm based on power iteration acceleration proposed in this paper can not only dramatically reduce storage consumption of data sets, which is an effective solution to the storage and calculation problems of current Web link big data, and can effectively reduce the PageRank iteration numbers and run time. And From the derivation of acceleration model, it is found that the acceleration model is independent with specific sequence $\{x_k\}$, as long as the sequence $\{x_k\}$ itself converges, the model can accelerate them. Therefore, it is not only suitable for PageRank algorithm, but also for other page ranking algorithms based on link analysis, such as the HITS algorithm.

Acknowledgements

The paper is supported by “the Fundamental Research Funds for the Central University” (WUT:2014-IV-106).

References

[1] J. Dean and S. Ghemawat, “MapReduce simplified data processing on large clusters”, Communications of the ACM, vol. 51, no. 1, **(2008)**.

[2] S. Ghemawat, H. Gobioff and S.-T. Leung, “The Google file system”, 19th ACM Symposium on Operating Systems Principles, **(2003)** October, Bolton Landing, NY, USA.

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes and R. E. Gruber, “Bigtable: a distributed storage system for structured data”, ACM Transactions on Computer Systems, vol. 26, no. 2, **(2008)**.

[4] L. Wang, J. Tao, R. Ranjan, H. M Achim, S. J. Chen and D. Chen, “G-Hadoop: MapReduce across distributed data centers for data-intensive computing”, Future Generation Computer Systems, vol. 29, no. 3, **(2013).**

[5] S. Brin and L. Page, “Reprint of the anatomy of a large-scale hyper textual web search engine”, vol. 56, no. 18, **(2012)**.

[6] B. Liu, Editor “Yu Yong Translator”, Web Data Mining (the 2nd Version), Beijign: Tsinghua University Press, **(2011)**.

[7] W. Yuan and D. David, “Computing page rank in a distributed internet search system”, Proceedings of the Thirtieth international conference on Very large databases, **(2004)** August – September, Toronto, Canada.

[8] S. Kamvar, T. Havelivala and G. Golub, “Adaptive Methods for the computation of PageRank”, 11th ILAS Conference, **(2004)** July 19-24, Coimbra Portugal.

[9] W. Jiaqi and T. Yongji, “PageRank algorithm optimization and improvement”, Computer Engineer and Applications, vol. 45, no. 16, **(2009).**

[10] I. Pavaloiu and E. Catinas, “On an Aitken---Newton type method”, Numerical Algorithms, vol. 62, no. 2, **(2013)**.

[11] J. H. Mathews and K. D. Fink, Z. Lu, C. Yu, Q. Fang, “Translator Numerical Methods (MATLAB version)”, Beijing: Publishing House of Electronics Industry, **(2007)**.

Authors

Chun Liu, she has being studying the Ph.D degree in Computer Application in Wuhan University of Technology (WHUT), and she also a LECTURER at WHUT. Her research interests are parallel computing and data mining.

Yuqiang Liu, he received the Ph.D. degree in Computer Application from Wuhan University of Technology (WHUT), Wuhan, China, in 2012.Now he is currently ASSOCIATE PROFESSOR at WHUT, and his research interests are computer application support technology, cloud computing and big data.