International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.7, No.2 (2014), pp.131-142
http://dx.doi.org/10.14257/ijsip.2014.7.2.13

A New Data Mining Algorithm based on MapReduce and Hadoop

Xianfeng Yang [1] and Liming Lian [2]

[1] *School of Information Engineering, Henan Institute of Science and Technology, Xinxiang, Henan, P.R.CHINA*
[2] *Department of Mechanical and Electrical Engineering, Xinxiang University, Xinxiang Henan, P.R.CHINA*
[1]*49377535@qq.com*, [2] *llm123513@163.com*

Abstract

The goal of data mining is to discover hidden useful information in large databases. Mining frequent patterns from transaction databases is an important problem in data mining. As the database size increases, the computation time and required memory also increase. Base on this, we use the MapReduce programming mode which has parallel processing ability to analysis the large-scale network. All the experiments were taken under hadoop, deployed on a cluster which consists of commodity servers. Through empirical evaluations in various simulation conditions, the proposed algorithms are shown to deliver excellent performance with respect to scalability and execution time.

Keywords: *Parallel algorithm, MapReduce, Hadoop, data mining*

1. Introduction

Business intelligence and data warehouse can handle TB level data or even higher level. Although many methods have been proposed to deal with high-dimensional data, but the query process is a bottleneck [1]. The emergence of cloud computing to the massive data mining, Hadoop is a MapReduce programming model and mass data [2]. It has made a lot of simulation system in the cloud computing, such a calculation based on the concept of cloud modeling and simulation platform of COSIM-CSP system [3], a new mode of the networked manufacturing [4], private cloud framework for visual simulation [5] ,and the military training system [6]. A simple MapReduce index is completed by McCreadie on the Hadoop [7].Ralf proposed a basic program designed to support cloud computing [8]. Moretti introduced a scalable data mining method, the data and computation is distributed to a cloud computing [9]. Gillick implementation of the inquiry learning with Hadoop[10].Many algorithms are using the tree structure in these application systems, such as ID3, C4.5, Fpgrowth KD-tree, PrefixSpan and BIRCH. Most data mining algorithms are based on object oriented programmings (OOP) which are usually run on a single computer. Some literatures have been described the detail of method [11-14].However, the characteristics of the MapReduce mode is not suitable for data mining.

First of all, MapReduce is lack of overall. The lack of data sharing between the tasks nodes in Hadoop, such as shared memory. The KD tree model and cluster tree model is a model of data mining that requirement for obtaining the global access from the training data. Each node of the Hadoop only processes the data block which is allocated, and output the results of data block. No correlation between the Map sub tasks and between sub tasks of Reduce also unrelated. Therefore, to complete the task of global way linked list and tree structure is very difficult. Secondly, HDFS does not allow random write operation. Because the Hadoop object

ISSN: 2005-4254 IJSIP

is GB, or even TB level data. Each file in a block unit distribution is stored in different nodes. It will cause the access failure of subsequent block when we insert or update the data in the middle of the file, so the random write operation is not allowed. Massive data once written into the HDFS (Hadoop distributed file system) will not modify, only can be added or deleted. This ensures that large files can be distributed in each node, and the node only processes the most easily accessible file fragments. Therefore, it is impossible to simulate the list and tree structure by using the HDFS file system. Thirdly, the task has a short life cycle. Each data block is node scanning one time; it will no longer be accessed. In data mining, often training model for data set, and the test set using the model to calculate the effectiveness. As mentioned before, there is no public area to accumulate the results of data analysis in Hadoop, so it is unable to realize the reference in front of the data mining results.

Finally, the MapReduce has also been shown to have significant problems [15] with more complex algorithms, like conjugate gradient, fast Fourier transform and block tri-diagonal linear system solver. Moreover, most of these problems use iterative methods to solve them, indicating that MapReduce may not be well suited for algorithms that have an iterative nature. However, there is more than one type of iterative algorithm. To study if MapReduce model is unsuitable for all iterative algorithms or only a certain subset of them, we devised a set of classes for scientific algorithms. Algorithms are divided between these classes by how difficult it is to adapt them to the MapReduce model and their resulting structure. To be able to compare the classes to each other, we selected and adapted algorithms from each class to the MapReduce model and studied their efficiency and scalability. Such a classification allows us to precisely judge which algorithms are more easily adaptable to the MapReduce model and what kind of effect belonging to a specific class has on the parallel efficiency and scalability of the adapted algorithms.

In order to solve the shortcomings and the problems, this paper uses a Newman parallel search algorithm based on MapReduce, to improve the processing speed of massive data. The test proves that the parallel algorithm is scalable to large data sets in this paper.

2. Fast Newman Parallel Algorithm

2.1. The Newman Algorithm with Modularity

The fast Newman algorithm is a clustering algorithm based on greedy algorithm. Its steps are as follows：

Step 1: Initialize the network for N communities. It means that every point is an independent association. The initial element e_{ij} and a_i is satisfies the following formula:

$$e_{ij} = \begin{cases} \dfrac{1}{2m} \\ 0 \end{cases}$$

When the side is connected between i and j, the $e_{ij} = \dfrac{1}{2m}$. When the side is not connected between i and j, the $e_{ij} = 0$.

$$a_i = \frac{k_i}{2m}$$

Where the k_i is the degree of node; m is the total number of edges in the network.

Step 2: Merge the communities successively which is connected by the side, and calculation the increment of module.

$$\Delta Q = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j)$$

According to the principle of greedy algorithm, each time with the Q should be increased most or reduce the minimum direction. After the merger by each time, we will update the corresponding elements of e_{ij}, and add the row and column which is correspond to community.

Step 3: Repeat step 2, constantly consolidated community, until the Q value is no longer increases. This has been the best network community structure

Due to the time complexity of the algorithm is $O((m+n)n)$, so when the data quantity exceeds 10000, the required memory for $1*10^8$, then the memory overflow.

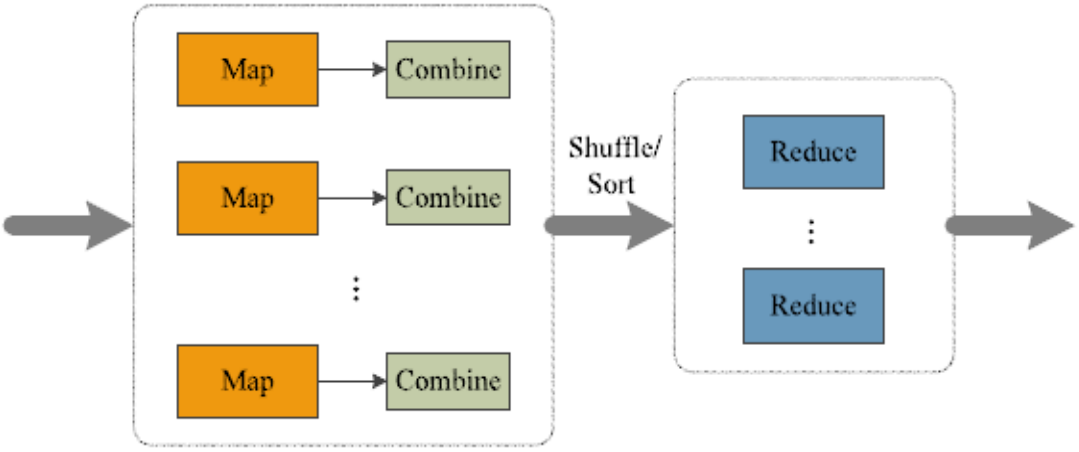

Figure 1. The MapReduce Model

2.2. Newman Parallel Algorithm and Modularity

First initialize
MAP stage
Read the side information

$$Emit(< key_1, key_2 >, value)$$

key_1 and key_2 is the Edge between vertices, $value$ is the number of the edge.
Then, we will repeat the following procedure:
Step 1:
the edge sequence files will be converted into the degree sequence file

Map：

```
if (key1 == key2)
   Emit(key1, value)
else
{
```

$Emit(key_1, value)$;

$Emit(key_2, value)$;

}

Reduce：

$sum = 0$;

$for(IntWritable \quad value : values)$

$sum += value.get()$;

$SumValue.set(sum)$;

$Emit(key, SumValue)$;

Step 2:

Calculated the value of δQ

Map：

Read the edge sequential file

$Emit(< key_1, key_2 >< value, (), () >)$;

Reduce:

Read the degree sequential file

$Emit(< key, -1 >< (), value, () >)$;

$weight = 0$;

$degree_1 = 0$;

$degree_2 = 0$;

$\delta Q = 0$;

$for(ValuePair \quad value : values)$

{

$if\ (value.getFirst != 0)$

$weight = value.getFirst$;

$else \quad if \ \ (value.getSecond != 0)$

{

$if\ (degree_1 == 0)$

$degree_1 = value.getSecond$;

$else$

$degree_2 = value.getThird$;

}

$\delta Q = weight/edge - \frac{1}{2}(degree_1/edge) * (degree_2/edge)$;

}

$Emit(< key_1, key_2 >, \delta Q)$;

Step 3:

Find out the value of δQ which has the most vertices

Reduce:

$int \quad maxValue = 0$

$while(value.hasNext())$

{
$maxValue = Math.max(maxValue, values.next().get())$
}
$emit(< key_1, key_2 >, maxValue)$
$if\ (maxValue == 0)$
end

Repeat the Step 1 to Step 3, until the X value is no longer increases.

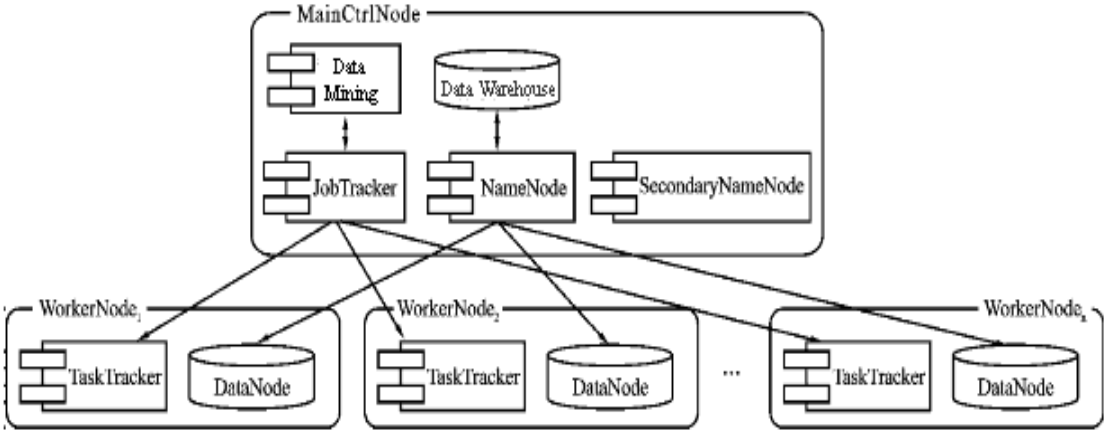

Figure 2. Parallel Data Processing Platform Based on the MapReduce

3. The Comparison Algorithm with this Article

We chose one algorithm from each of the algorithm classes outlined in Section 3 to illustrate the different design choices and problems that can arise when adapting scientific computing problems to the Hadoop MapReduce framework. These algorithms are:

A. Partitioning Around Medoids (PAM).
B. Clustering Large Application (CLARA1).
C. Clustering Large Application (CLARA2).

3.1. Partitioning Around Medoids

The PAM clustering algorithm [16] is not sensitive to noise and outlier data, and the input sequence of cluster and cluster data is irrelevant which is found by this algorithm. The PAM algorithm can handle different types of data points. The basic idea of the algorithm is that in order to find the K cluster, the PAM cluster algorithm to define an object for each cluster. This object is known as the center, it is the object in the cluster's center position. When the k center point is selected, the residual n-k (n is the number of data objects) non selected objects are included in the selected object from its recent representative cluster. In order to find out the K centers, the algorithm first randomly selected K objects. Then, we use a non selected object X to replace a selected object Y at each step,if such a replacement can improve the clustering quality.

- Map:

– Find the closest medoid and assign the object to its cluster.
– Input: (cluster id, object).

- Output: (new cluster id, object).

• Reduce:

- Find which object is the most central and assign it as a new medoid to the cluster.
- Input: (cluster id, (list of all objects in the cluster)).
- Output: (cluster id, new medoid).

3.2. Clustering Large Applications

In order to deal with the large amount of data, Kaufman and Rousseeuw proposed K medoid algorithm(CLARA) [17]. CLARA is not found representative objects from the whole data. It is found representative objects from the sample dataset. Then, it select the center point from the sample calculation method for replacement costs. If the sample is randomly selected, it should represent the original data set. However, the algorithm does not guarantee a sample which is truly random. The quality of clustering is that all objects of data set on the average non similarity, not just the sample of these objects on the average non similarity. In order to achieve better approximation, the best clustering is used as the output.

First CLARA MapReduce job:

• Map:

- Assign a random key to each object.
- Input: (key, object).
- Output: (random key, object).

• Reduce:

- Read first n objects, which are sorted in the ascending order of the keys. Because the keys were assigned randomly, the order of the objects is random after sorting. Perform PAM clustering on the n objects to find k different candidate medoids.
- Input: (key, list of objects).
- Output: (key, list of k medoids).

Second CLARA MapReduce job:

• Map:

- For each object, find the closest medoid and calculate the distance from it. For each object, this is done as many times as there were candidate sets of medoids, and one output is generated for each.
- Input: (cluster, object).
- Output: (candidate set id, distance from the closest medoid) [One output for each candidate set].

• Reduce:

- Sum the distances with the same candidate set id.
- Input: (candidate set id, list of distances).
- Output: (candidate set id, sum (list of distances)).

From the experiment results (Tables 2, 3 and Figs. 2, 3) it is possible to see that the CLARA MapReduce algorithm works much faster than PAM.

4. The Simulation and Conclusion

The performance of parallel computing can be measured through its speedup and scalability. [18-20]. The speedup is the performance improvement that is obtained by the

parallel computing to reduce the running time. It is an important index to verify the performance of parallel computing. The formula is Sp = Ts /Tp, Where the Ts denotes the calculation time of serial algorithm (*i.e.*, in a single node), and Tp denotes calculation time of parallel algorithm (*i.e.*, in the same p node).The acceleration is bigger, the consumption of relative time of parallel computing is less, and the parallel efficiency and performance improvement is higher.

We continuously give four tables and four graphs as below. they are the test results in the different number of nodes of the four algorithms.

Table 1. Run Times for the Newman in MapReduce

Unknowns	24	500	1000	2000	4000	6000	8000
1 node	260	263	335	645	1955	3752	7561
2 nodes	256	260	287	505	1352	2489	4425
4 nodes	255	245	291	358	778	1403	2358
8 nodes	249	252	288	387	569	856	1562
16 nodes	229	238	268	300	358	568	954

Table 2. Run Times for the PAM in MapReduce

Objects	10000	25000	50000	750000	100000
1 node	1425	1466	2125	3785	7003
2 nodes	1154	1852	1995	2158	6251
4 nodes	811	793	1258	2368	2685
8 nodes	635	611	1325	1157	1774
16 nodes	288	512	427	788	1121

Table 3. Run Times for the CLARA1 in MapReduce

Objects (thousand)	25	50	100	500	1000	5000	10000
1 node	125	126	138	177	275	882	1658
2 nodes	80	88	90	153	231	512	814
4 nodes	60	66	72	125	133	342	475
8 nodes	52	66	61	124	132	253	325
16 nodes	44	50	58	100	102	121	168

Table 4. Run Times for the CLARA2 in MapReduce

Objects (thousand)	25	50	100	500	1000	5000	10000
1 node	112	123	134	172	276	882	1658
2 nodes	80	88	95	153	235	518	817
4 nodes	60	66	72	124	134	343	476
8 nodes	53	65	67	125	132	253	324
16 nodes	45	51	57	98	99	115	151

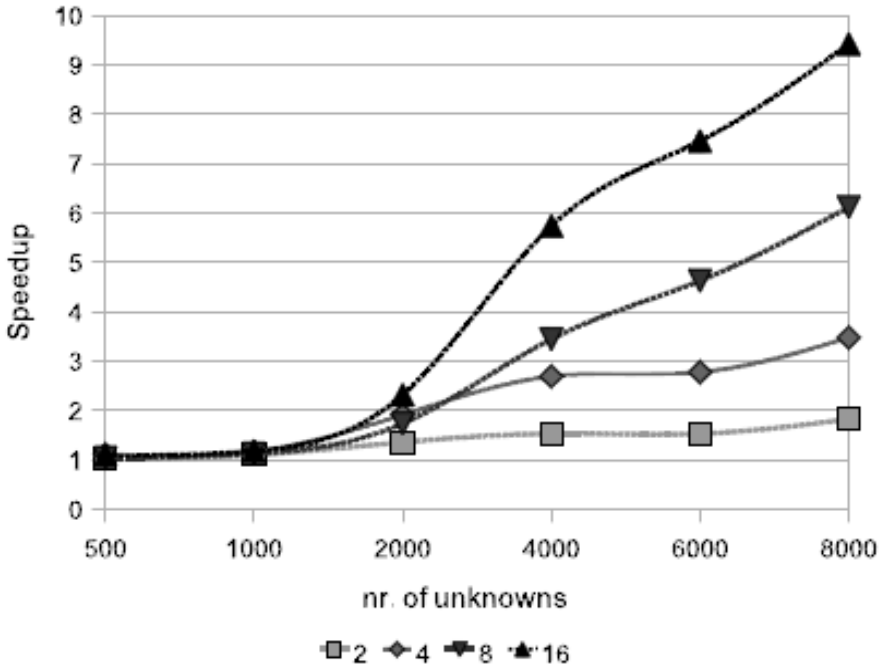

Figure 3. Speedup for Conjugate Gradient Algorithm with Different Number of Nodes

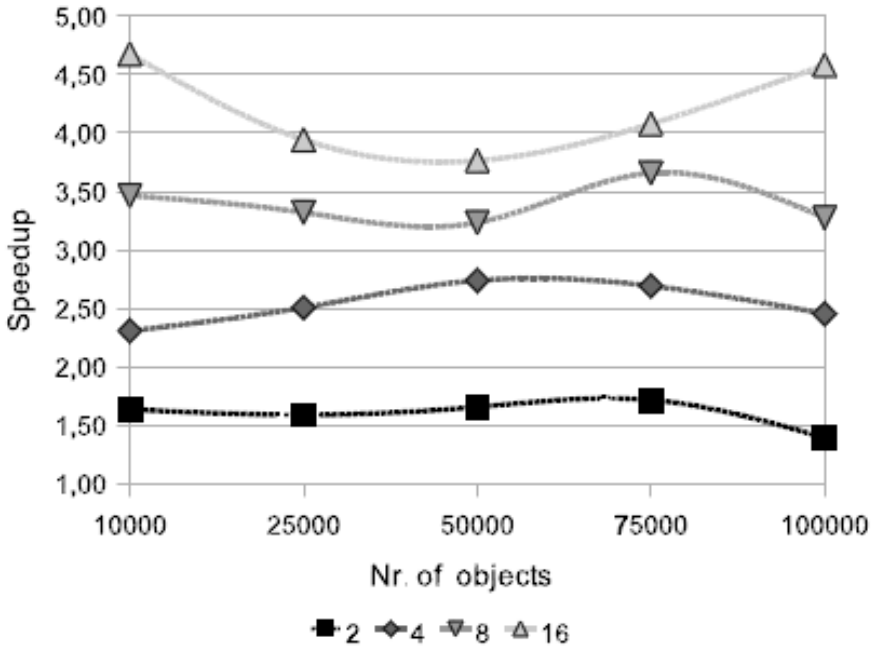

Figure 4. Parallel Speedup for PAM with Different Numbers of Nodes

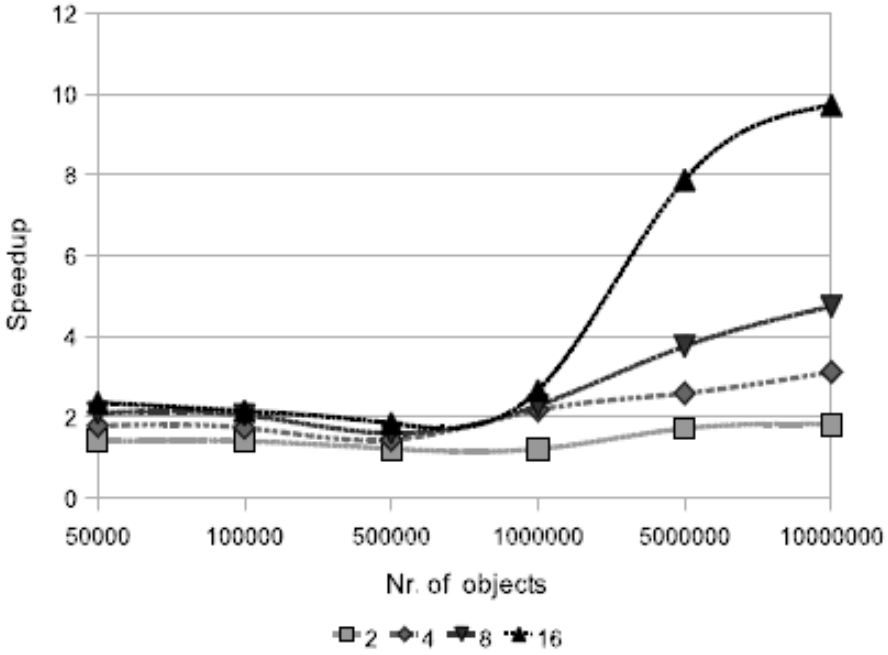

Figure 5. Parallel Speedup for the CLARA Algorithm with Different Number of Nodes

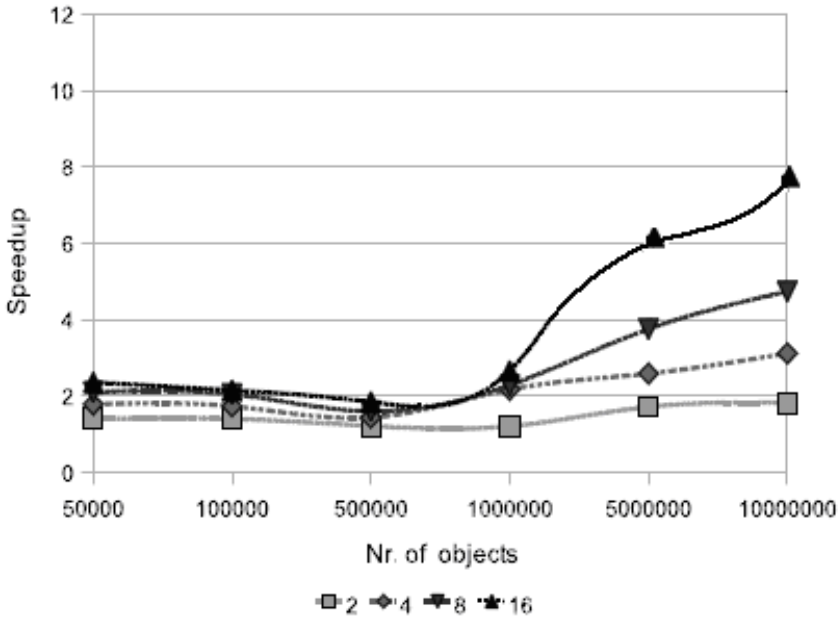

Figure 6. Parallel Speedup for the Integer Factorization with Different Numbers of Nodes

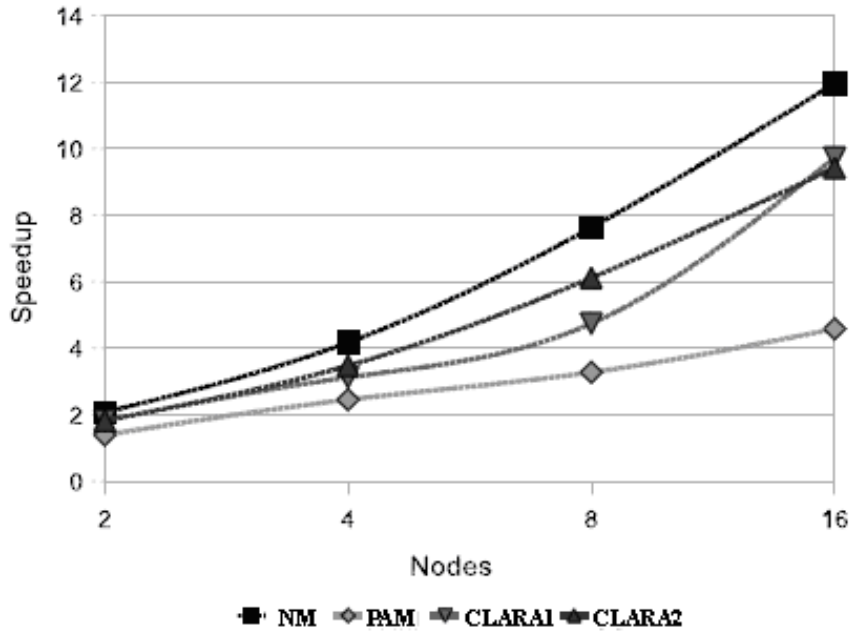

Figure 7. Achieved Speedup Comparison of the Four Algorithms

Regardless of the problems encountered, all implemented algorithms were able to achieve speedup from using multiple nodes, as shown in the Figure 7, with NM algorithm having the best and PAM the worst speedup in our tests.

In this paper, by using the Newman algorithm for parallel study, and the algorithm compared with the other three algorithms. The test results show that the parallel algorithm of error rate is smaller, and it has very good validity. Especially, the dataset size is bigger, the efficiency will be higher.

References

[1] C. Bohm, S. Berchtold and H. P. Kriegel, "Mul-tidimensional index structures in relational databases", Proceedings of the 1st International Conference on Data Warehousing and Knowledge Discovery (DaWak 99), Florence, Italy, F, **(1999)** August 30-September 01.

[2] J. Dean and S. Ghemawat, "Usenix. MapReduce: Sim-plified data processing on large clusters", Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI 04), San Francisco, CA, F, 2004 December 06-08.

[3] L. Bo-hu, C. Xu-dong and H. Bao-cun, "Networked Modeling & Simulation Platform Based on Concept of Cloud Computing—Cloud Simulation Platform", Journal of System Simulation (S1004-731X), vol. 21, no. 17, **(2009)**, pp. 5292-5299.

[4] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", Second Edition [M]. second ed. San Francisco, USA: Morgan Kaufmann, **(2006)**.

[5] H. Xiang, K. Feng-ju and T. Xue-wei, "Research on Framework of Private Cloud in Visual Simulation", Journal of System Simulation (S1004-731X), vol. 23, no. 08, **(2011)**, pp. 1652-1656.

[6] H. An-xiang, F. Xiao-wen and L. Jin-song, "Aviation Simulation Architecture Based on Cloud Computing Platform", Journal of System Simulation (S1004-731X), vol. S1, **(2011)**, pp. 106-109.

[7] R. M. C. Mccreadie, C. Macdonald and I. Ounis, "On Single-Pass Indexing with MapReduce", New York, USA: Assoc Computing Machinery, **(2009)**.

[8] R. Lammel, "Google's MapReduce programming model – Revisited", Science of Computer Programming (S0167-6423), vol. 70, no. 1, **(2008)**, pp. 1-30.

[9] C. Moretti, K. Steinhaeuser and D. Thain, "Scaling Up Classifiers to Cloud Computers", Proceedings of the IEEE International Conference on Data Mining, Pisa, Italy, F, USA: IEEE Computer Society, **(2008)**.

[10] D. Gillick, A. Faria and J. Denero, "MapReduce: Dis-tributed Computing for Machine Learning", [2011-07]. http://www.icsi.berkeley.edu/~arlo/publications/gillick _cs262a _proj.pdf, **(2006)**.
[11] L. Yang and Z. Shi, "An Efficient Data Mining Framework on Hadoop using Java Persistence API", The 10th IEEE International Conference on Computer and In-formation Technology (CIT-2010). Bradford, UK. USA: IEEE, **(2010)**.
[12] S. Hinz, P. Dubois and J. Stephens, MySQL Cluster [M/OL] [08-02-2009] http:// dev.mysql.com/ doc/ refman/ 5.0/ en/ mysql-cluster -overview.html, **(2009)**.
[13] R. Biswas and E. Ort, "Java Persistence API - A Simpler Programming Model for Entity Persistence", http:// java.sun.com/ developer /technicalArticles/ J2EE/jpa/, **(2009)**.
[14] X. Lu, L. Zha and Z. Xu, "Asset-Leasing Model, Architecture, and Key Technology of Vega LingCloud", Journal of Computer Research and Development (S1000-1239), **(2010)**.
[15] C. Bunch, B. Drawert and M. Norman, "Mapscale: a cloud environment for scientific computing", Technical Report, University of California, Computer Science Department, **(2009)**.
[16] L. Kaufman and P. Rousseeuw, "Finding Groups in Data an Introduction to Cluster Analysis", Wiley Interscience, New York, **(1990)**.
[17] C. Pomerance, "A tale of two sieves", Notices of the American Mathematical Society, vol. 43, **(1996)**, pp. 1473-1485.
[18] R. Pike, S. Dorward, R. Griesemer and S. Quinlan, "Interpreting the data: parallel analysis with Sawzall", Scientific Programming, vol. 13, **(2005)**, pp. 277-298.
[19] K. van der Raadt, Y. Yang and H. Casanova, "Practical divisible load scheduling on grid platforms with APST-DV", Proc. of the 19th IPDPS'05, **(2005)**, pp. 29.b.
[20] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski and C. Kozyrakis, "Evaluating MapReduce for multi-core and multiprocessor systems", Proceedings ofInternational Symposium on High Performance Computer Architecture, HPCA, **(2007)**, pp. 13-24.

Authors

Xianfeng Yang, he received his bachelor's degree in Computer Application Technology from Henan Normal University, Xinxiang, Henan, China, in 2001,the master's degree in Computer Application Technology from China University of Petroleum, Dongying, China, in 2007. He is now a lecturer at School of Information Engineering, Henan Institute of Science and Technology, Xinxiang, China. His current research interests include pattern recognition, image processing, neural networks, natural language processing.

Liming Lian, he received the master degree (2011) in mechanical manufacturing and automation from Henan Polytechnic University, Jiaozuo, Henan, P.R. CHINA. He is now a lecturer at Xinxiang University, Xinxiang Henan, P.R. CHINA. His current research interests include computer aided design (CAD), system simulation and its application.