Send Orders for Reprints to reprints@benthamscience.ae

1748 *The Open Automation and Control Systems Journal*, 2014, *6*, 1748-1753

Open Access

An Algorithm of Association Rule Based on Cloud Computing

Fei Long[1,*] and Fang Luo[2]

[1]*Department of Economics and Management, Changsha University, Changsha, Hunan, 410000, China*
[2]*School of Information Science and Engineering, Central South University, Changsha, Hunan, 410083, China*

Abstract: Large-scale data processing is one of the focal points of research in information technology. The traditional algorithm of association rule is in a large overhead, due to the frequent itemsets being computed on the dataset. The rapid development of distributed technology makes cloud computing a reality in the implementation of data processing algorithms. To improve the traditional association rule algorithm, in this paper, an AprioriMR algorithm for mining association rule based on cloud computing is proposed. The AprioriMR algorithm takes HDFS to store data and is well adapted to the Hadoop's Map-Reduce computing model. It divides into two parts, deals with Map-Reduce operation, and combines to produce the frequent patterns. The AprioriMR algorithm inherits the Map-Reduce scalability to huge datasets and to thousands of processing nodes. Experimental results show that it is very efficiently compared with the traditional association rules algorithm and has a good speedup when deals with massive data.

Keywords: Apriori, cloud computing, HDFS, map-reduce.

1. INTRODUCTION

Internet has experienced since its birth nearly half a century, and penetrated into every aspect of our lives. With the rapid development of computer technology and the internet, the widely used Web2.0 data growth appears explosively. The data generated by the various fields has become a massive scale because of the mushroom storage technology development of data collection in recent years. Accurately picking up useful information and knowledge from these data in a high speed allows enterprises to step ahead in the highly competitive commercial so that they can obtain commercial success and economic benefit. Nowadays the explosive growth of the internet data even makes the point where a single compute can hardly be handled, so that the processing of vast amounts of data is becoming a thorny issue [1].

Data mining, a technique to understand and convert raw data into useful information, is increasingly being used in a variety of fields like marketing, business intelligence, scientific discoveries. Data mining as a new technology, combines the traditional data analysis method with the complex data processing algorithm to help people get the available information efficiently. Therefore, it is widely used in various fields. However, it will become a long time-consuming process with increasing size of the input data. Association rule mining plays an important role in the field of data mining, which is widely used in various fields. How to mine and use association rules properly is an important task of data mining. There is a very important theoretical and practical significance of association rules in the research [2].

As a hotspot of recent study at home and abroad, cloud computing originates from grid computing, parallel computing and distribution computing. Cloud computing could be operated on an ordinary cluster of personal computers, and generates efficient data processing result as well. The cloud computing technology has been relatively mature which enforcing computing power in the meantime lowering the cost. Cloud computing platform has a very high scalability which is ideal for handling large-scale data, and its storage and computing power can be enhanced by dynamically increasing computing nodes of the platform [3]. The superior performance of traditional data processing methods and techniques in the case of a single server, no longer adapts to the face of the cloud platform distributed processing mode. Traditional data processing method is used for the transformation of distributed computing, and on this basis the improved algorithm will have major significance for massive data processing.

If traditional data mining algorithms could be transformed and deployed to the cloud computing platform, there is no doubt that the problem of large-scale data mining can be solved. That is, combining data mining algorithms with cloud computing technology is the future trend. Unfortunately, there are less research findings for certain difficult algorithms and not all the algorithms can be executed on the platform [4]. In short, theoretical results and experiments show that data mining algorithm plays a very important role exploring the value of data under cloud computing. It produces theoretical and economic significance for academic research and commercial operations [5].

Based on this, our paper first integrates the cloud computing technology with the association rule algorithm, analyzes the Map-Reduce programming model and operation mechanism, and summarizes the cloud computing platform advantages. It then presents the idea of parallel improvement based on the partition. Through designing the improved association rule algorithm, this paper demonstrates the feasibility of the improved association rule algorithm and analyses it. Finally, experimental results show that the improvement

*Address correspondence to this author at the Department of Economics and Management, Changsha University, Changsha, Hunan, 410000, China; E-mail: 51441980@qq.com

1874-4443/14 2014 Bentham Open

algorithm is more efficient than the traditional algorithms, which reduces the time complexity and space complexity simultaneously.

The rest of the paper is organized as follows: Section 2 describes the Apriori algorithm and cloud computing in detail. The proposed algorithm is depicted in Section 3. Experiments and results analysis are presented in Section 4. We finish in Section 5 with conclusions and an outlook on future work.

2. BACKGROUND

2.1. Association Rule

In data mining, the association rule is a popular and well researched method for discovering interesting relations between variables in large databases. Agrawal [6] introduced associations for discovering regularities between products in large-scale transaction data recorded in supermarkets. This section describes some basic definitions and concepts which are used in the design of the proposed model.

Let $I=\{I_1,I_2,\ldots,I_m\}$ be a set of literals, called items. Let $D=\{t_1,t_2,\ldots,t_n\}$ be a set of transactions, where each transaction t_i is a set of items such that $ti \subseteq I$. Each transaction has a unique identifier TID. Given an itemset $X \subseteq I$, a transaction T contains X if and only if $X \subseteq T$. An association rule is an implication of the form "X⇒Y", where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \varphi$. The Support and Confidence are common indicators to measure strength of association rules. To select interesting rules from the sets of all possible rules, constraints on various measures of significance and interest can be used. The best-known constraints are minimum thresholds on support and confidence. The support supp(X) of an itemset X is defined as the proportion of transactions in the dataset which contain the itemset. The confidence of a rule is defined as $conf(X \subseteq Y) = \text{sup}p(X \cup Y)/\text{sup}p(X)$.

As one of the most influential algorithms of data mining, the Apriori algorithm is to find frequent itemsets from a transaction dataset and derive association rules. The Apriori algorithm is simple and easy to understand and easy to implement. In this algorithm, the k-itemset is used to generate the (k+1)-itemset, and the frequent k-itemsets are extracted from the candidate k-itemsets. Once frequent itemsets are obtained, it is straightforward to generate association rules with confidence larger than or equal to a user specified minimum confidence [7].

Let the set of frequent itemsets of size k be F_k and their candidates be C_k. Apriori algorithm scans the database at first and searches for frequent itemsets of size 1 by accumulating the count for each item and collecting those itemsets that satisfy the minimum support requirement [8].The Apriori algorithm iterates on the following three steps and extracts all the frequent itemsets. Generate C_{k+1}, the candidates of frequent itemsets of size k+1, from the frequent itemsets of size k. Scan the database for calculating the support of each candidate of frequent itemsets. Add those that satisfy the minimum support requirement to F_{k+1}. The Apriori algorithm is shown in Fig. (**1**).

However, the Apriori algorithm has some disadvantages. The Apriori algorithm needs to repeatedly scan the transaction database which may be very large and thus the scale of the database is the important factor for the Apriori performance [9]. The Apriori algorithm may generate vast intermediate itemsets. The Apriori algorithm generates C_{k+1} candidate of frequent itemsets of size k+1, from the frequent itemsets of size k. The number of the C_{k+1} may be very vast. The Apriori algorithm then scans the database to calculate the support of each candidate of frequent itemsets [10]. Because of the huge number of the candidates frequent itemsets, the process of confirming the candidates will take a lot of time.

```
F1=(Frequent itemsets of cardinality 1);
for(k=1;Fk≠Φ;k++) do begin
    Ck+1=aprio-gen(Fk);
    for all transactions t∈Database do begin
      Ct' =subset(Ck+1, t)
         for all candidate c∈Ct' do
           c.count++;
         end
      Fk+1={C∈Ck+1 | c.count≧minimum support}
    end
end
Answer ∪kFk;
```

Fig. (1). Pseudo code for Apriori algorithm.

2.2. Cloud Computing

The concept of cloud computing by Google in 2006 pointed out the direction for the processing of massive data developed by the Apache Foundation open source Hadoop cloud platform features. Also, the majority of researchers brought about the dawn of the low-cost mass data processing. Cloud computing is basically services-on-demand over the Internet. It is a natural evolution of the widespread adoption of virtualization, service-oriented architecture and utility computing.

Cloud computing is a computing capability that provides an abstraction between the computing resource and its underlying technical architecture. There is no formal definition commonly shared in industry, unlike for Web 2.0, and it is very broadly defined as on-demand provisioning of application, resources, and services that allow resources to be scaled up and down. Clouds are a large pool of easily usable and accessible virtualized resources. These resources can be dynamically reconfigured to adjust to a variable load, allowing also for optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs [11].

Cloud computing includes a number of technologies such as Web services, Service Oriented Architecture, Web 2.0, Web Mashup etc. One of the key new technologies used in the cloud are scalable batch processing systems. Distribution and scalability are playing important roles in the cloud, and for that Hadoop can be used. Many companies, especially ones that operate through demanding websites, *e.g.* Amazon, Facebook, use Hadoop.

Apache Hadoop is a collection of related subprojects that compose an infrastructure for distributed computing. Hadoop is written in Java language; any machine that supports Java

can run the Hadoop software. Hadoop enables the development of scalable, efficient and distributed computing using very simple Java interfaces. Using the Hadoop platform, programs can be developed that facilitate the processing of large amounts of data. The main applications for Hadoop seem to be log analysis, Web indexing, and various data mining and customer analysis applications [12, 13].

3. APRIORIMR ALGORITHMS

The association rule mining algorithms will fail if we use the existing centralized association rule mining algorithms. Although there exist many improved association rule mining algorithms, which have the ability to reduce the times of scanning original dataset, yet almost all of them are not suitable to deal with large-scale data.

For the deficiency of traditional Apriori algorithm, to solve these problems, we proposed a new algorithm of Apriori algorithm under cloud computing environment, called AprioriMR algorithm. Number of iterations is same in AprioriMR algorithm as number of iterations in traditional Apriori. But AprioriMR algorithm scans less data rather than scanning whole data as in Apriori. The differences between Apriori algorithm and AprioriMR algorithm is: AprioriMR uses new data structure to represent the dataset and Map-Reduce framework, whereas Apriori uses counting. The workflow of the AprioriMR algorithm can be seen in Fig. (**2**).

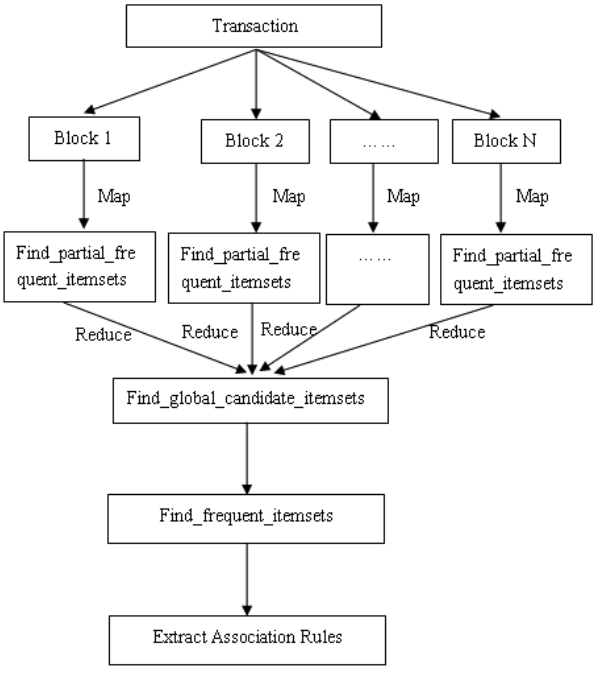

Fig. (2). The workflow of AprioriMR.

The AprioriMR algorithm consists of three steps: data initialization, finding frequent itemsets and extracting association rules from the frequent itemsets. The implementation of AprioriMR algorithm based on Map-Reduce mainly includes seven steps as follows:

In the data initialization step, the task is to scan each record of the input transaction. Line numbers were used as transaction_id. The definition of <key,value> is the first step to implement the AprioriMR. The pairs of key/value are shown in Table **1**; we can design the map function and reduce function using the <key, value>pairs.

Table 1. Key/value pairs for map and reduce function.

Input/ output	**Input: <Key,Value> pairs**	**Output: <Key,Value> pairs**
Map function	Key: transaction_id Value: one row of data	Key: candidate items Value: 1
Reduce function	Key: candidate items Value: 1	Key: frequent items Value: support

The AprioriMR algorithm uses Hadoop components to perform job execution and information storage. It first stores the data in the file rather than the database in order to deal with in the Hadoop Distributed File System, and then deploys Map-Reduce to parallel the first step. The original transaction data is automatically divided into N data subsets by the HDFS, and then these subsets are sent to N nodes. For formatting the N data subsets, each of the N nodes processes the data subset independently, and then generates <key,value> pairs where key is transaction_id and the input of the Map function, and their formats are <transaction_id, one row of data >.

Hadoop Distributed File System is a master/slave architecture that consists mainly of Namenode and Datanodes. In the files each line can be seen as a transaction, and each item is separated with white space. The workflow of HDFS is shown in Fig. (**3**). The master server is responsible for managing the namespace and file operations of datanode software. The Namenode and Datanodes are designed to run on machines [14]. A file is divided into multiple blocks and is stored on a group of datanode software programs.

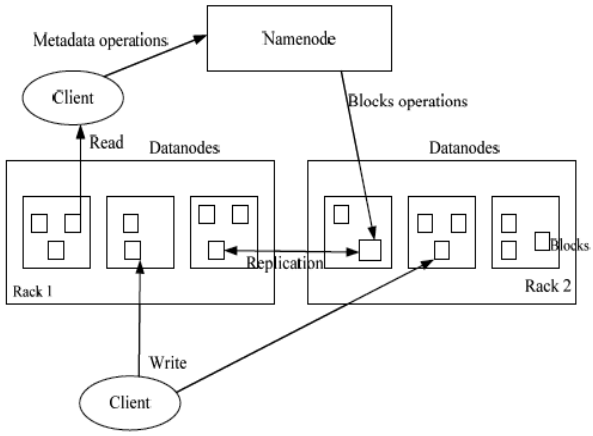

Fig. (3). The workflow of HDFS.

Using Map-Reduce for frequent itemsets counting in Apriori showed good scalability for algorithm. The Map-

Reduce model provides sufficient high-level parallelization. The corn idea of cloud computing technology is Map-Reduce in which all the strategies of improving algorithms under cloud computing have two different parts, which are map and reduce, respectively [15].

The Map function receives input pairs and produces a set of intermediate key/value pairs, and then sends intermediate key/value pairs to reduce nodes. The datasets in files are split into smaller segments automatically after storing in HDFS and the map function is executed on each of these data segments.

The Reduce function accepts an intermediate key and a set of values for that key and merges these values together to form a possibly smaller set of values. All algorithms must separate into two such functions; otherwise they would become extremely difficult to run on the platform. It only needs two Map-Reduce phases to find frequent itemsets. The workflow of Map-Reduce is shown in Fig. (**4**).

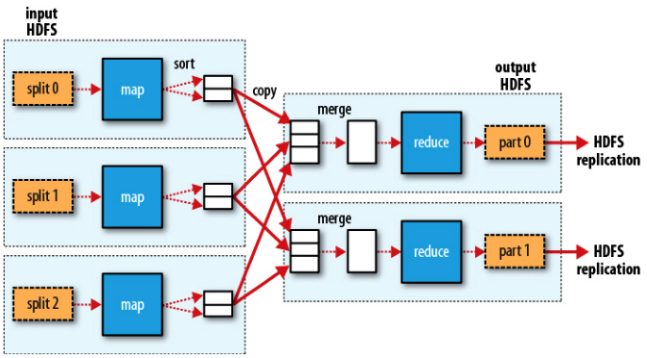

Fig. (4). The workflow of Map-Reduce.

The key step in association rule algorithm is to find the frequent itemsets. In the iteration, it computes the occurrences of partial frequent itemsets in each of the data blocks. It is obvious that the occurrences counting of candidate itemsets in one transaction is irrelevant to the counting in another transaction in the same iteration. Therefore, the occurrences computation process in one iteration could be executed in parallel.

Map function is executed on these data segments and produces <key,value> pairs. The framework groups all the pairs, which have the same item and invokes the reduce function passing the list of values for candidate items. The map's output is a list of intermediate <key,value> pairs: grouped by the key *via* combiner, and stored in the map worker, where the key is an element of partial frequent k-itemsets and the value is its partial count. When all map tasks are finished, the reduce task is started. The map's output are shuffled to the reduce worker that calls a reduce function. The pseudo-code for Map function is shown in Fig. (**5**).

The Reduce function adds up all the values and produces a count for the candidate item as a one-time synchronization. This algorithm's advantage is that it doesn't exchange data between data nodes, it only exchanges the counts. In every scan, each map function generates its local candidate items, and then the reduce function gets global counts by adding local counts. The output of reduce function is a list <key,value> pairs , where the key is an element of partial frequent k-itemsets and the value equals to one, stored in HDFS. Then, candidate itemsets are generated, setting the support count of candidate itemsets to 1. Each of the N nodes separately merges the support count by using Combiner function, if each candidate itemsets is the same. The same candidate itemsets in N nodes are sent to the same node by using the Hash function. The pseudo-code for Reduce function is shown in Fig. (**6**).

Map<key, value>

Input: Transaction, min_sup
Output: <*key, value*> pair
L1=find_1_itemsets(Transaction)
for（k=2; $L_{k-1}\neq\Phi$; k++){
Ck=Apriori(L_{k-1},min_sup)
for each t∈D{
Ct=subset(Ck，t)
for c∈Ct
c.count ++
}
<c,c.support>
}
Return ⋃ <c,c.support>

Fig. (5). Pseudo-code of map function.

Reduce <key, value>

Input: <c,c.support>
Output: L_k(global frequent itemsets)
Compare c for all<c,c.support>
Count=count_c(c,sum_support)
If(count>=min_sup)
Insert(L_k,c)
Return ⋃ L_k

Fig. (6). Pseudo-code of reduce function.

All the occurrences of candidate itemsets are summed up. Furthermore, the joint actions are done on the frequent (k-1)-itemsets and prune actions are performed on the candidate (k-1)-itemsets. Finally, the frequent k-itemsets are found. According to the frequent itemsets, the rules that have a support and confidence greater than given thresholds, are generated.

4. PERFORMANCE EVALUATION

In this section, the experiments and performance evaluation results analysis are presented. In order to test the performance of the AprioriMR algorithm designed by us, Hadoop platform is used to perform experiments. In our experiments, we used Hadoop version 0.19.2, running on a cluster with 10 machines (1 master, 9 slaves). Each machine consisted of duplicate-core processors (running at 2.60GH) and 2GB memory.

Dataset mainly determines the performance of the parallel algorithm. This experiment is based on real datasets, to calculate the correlation of products. If the experiment is

done with the small datasets, its performance turned out to be lower for some reasons; hence extra communication time occupying a large proportion is compared to the total execution time.

With the length of execution time, number of transactions and Datanodes as the evaluation criteria, the overall performance of Apriori and AprioriMR algorithm based on the Map-Reduce was tested, and the performances of Apriori in the single machine environment and AprioriMR algorithm based on the Map-Reduce, were compared and analyzed. All experiments were executed three times and the average was taken.

The performance of our proposed AprioriMR algorithm by comparing its execution time with the execution time of the traditional Apriori algorithm was evaluated. Fig. (**4**) shows the running time of the traditional Apriori algorithm and the proposed AprioriMR algorithm.

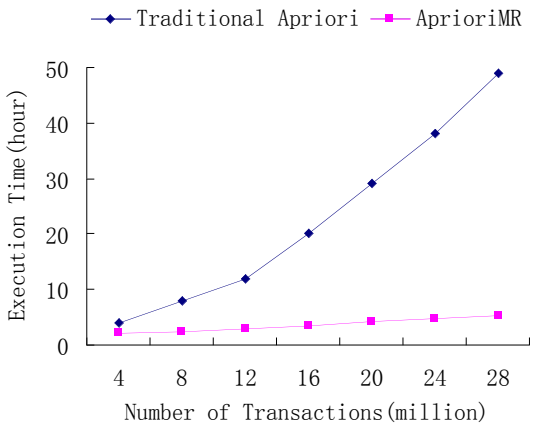

Fig. (7). Comparison of algorithms based on execution time.

As shown in Fig. (**7**), our proposed AprioriMR algorithm outperforms the traditional Apriori algorithms and it will continue to outperform them as the datasets size is increasing. When the processing datasets are proportionally increased, the running time of the AprioriMR algorithm has little change before and after, which shows that the AprioriMR algorithm has good scalability in the Map-Reduce environment. The results show that under the cloud computing environment, the improved algorithm can effectively mine the frequent itemsets from mass data, and the dataset partition method and distribution method can improve the efficiency of the improved algorithm in the heterogeneous cluster environment. In the Map-Reduce environment, AprioriMR algorithm eliminates the need to iterative scanning of the data to find all frequent items.

The Speedup of our proposed AprioriMR algorithm is evaluated by comparing its execution time with Datanodes varied on datasets. Fig. (**8**) shows test results of the Speedup of the AprioriMR algorithm based on Map-Reduce.

In the Map-Reduce environment, when AprioriMR algorithm deals with the datasets, with the increase of Datanodes, the execution time is proportionally decreased. In the cases of Apriori and AprioriMR algorithm dealing with large datasets, with the increase of Datanodes in the cluster, the running time is proportionally decreased. This is easily predicted from experiments where we noticed that the more data a node processes, the less significant proportion becomes the communication time.

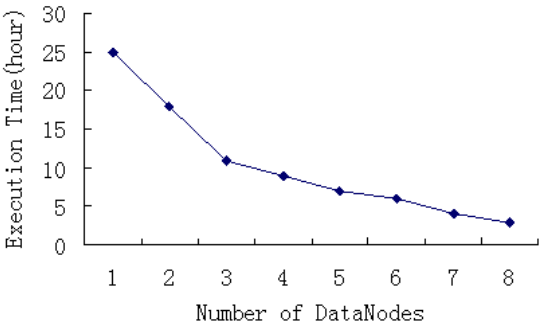

Fig. (8). Execution time for AprioriMR algorithm with Datanodes varied on datasets.

The larger datasets would have shown even better Speedup characteristics. In the case of more than one data node, Speedup increases with the increase of number of data in certain circumstances, which can prove the high efficiency of the parallel Apriori based on Map-Reduce. AprioriMR repeats scanning the other intermediate data that usually keep shrinking per iteration. The above analysis shows that the AprioriMR has good Speedup in the Map-Reduce environment.

CONCLUSION

The task of finding all association rules requires a lot of computation power and memory. In this paper, we proposed the AprioriMR algorithm based on cloud computing. The AprioriMR algorithm inherits the Map-Reduce scalability to huge datasets and to thousands of processing nodes. Experimental results show that AprioriMR algorithm is very efficient compared with the traditional Association Rules algorithm and have a good Speedup when deals with massive data.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (No. 61304184) and Changsha Municipal Science and the Technology Bureau (K1309015-11).

REFERENCES

[1] J. Dean, and S. Ghemawat, "Parallel mining of association rules," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 962-969, 1996.

[2] Y. Ye, "A parallel apriori algorithm for frequent itemsets mining," In: *Proceedings of the 4th International Conference on Software Engineering Research Management and Applications*, 2006, pp. 87-94,

[3] A. Weiss, "Computing in clouds," *ACM Networker*, vol. 11, no. 4, pp. 18-25, 2007.

[4] D. Jeffrey, and G. Sanjay, "Map-reduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, 2008.

[5] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules in large databases," *VLDB*, pp. 487-499, 1994.

[6] M. M. Gaber, A. B. Zaslavsky, and S. Krishnaswamy, "Data stream mining," *Data Mining and Knowledge Discovery Handbook*, pp. 759-787, 2010.

[7] U.M. Fayyad, G.P. Shapiro, and P. Smyth, "From data mining to knowledge discovery: an overview," *Advances in Knowledge Discovery and Data Mining*, 1996, pp. 1-34,

[8] Y. Gu, and R.L. Grossman, "Toward efficient and simplified distributed data intensive computing," *IEEE Transactions Parallel Distribution System*, vol. 22, no. 6, pp. 974-984, 2011.

[9] H. Toivonen, "Sampling large databases for association rules," In: *Proceedings of the 22th International Conference on Very Large Data Bases*, pp. 134-145, 1996.

[10] M. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, pp. 372-390, 2000.

[11] R.R. Exposito, G.L. Taboada, S. Ramos, J. Tourino, and R. Doallo, "Performance analysis of HPC applications in the cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218-229, 2013.

[12] V. C. Emeakaroha, M.A.S. Netto, R.N. Calheiros, I. Brandic, R. Buyya, and C. A. F. D. Rose, "Towards autonomic detection of SLA violations in cloud infrastructures," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1017-1029, 2012.

[13] Hadoop[EB/OL]. http://hadoop.apache.org/, 2010.

[14] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System,"In: *IEEE 26th Symposium on Mass Storage Systems and Technologies MSST*, 2010, pp. 1-10.

[15] M. Zaharia, A. Konwinski, and A.D. Joseph, "Improving MapReduce performance in heterogeneous environments," In: *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation,* New York: ACM Press, 2008, pp. 29-42.

Received: September 16, 2014 Revised: December 22, 2014 Accepted: March 03, 2015

© Long and Luo; Licensee *Bentham Open*.

This is an open access article licensed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/-licenses/by-nc/4.0/) which permits unrestricted, non-commercial use, distribution and reproduction in any medium, provided the work is properly cited.