International Journal of Hybrid Information Technology
Vol.8, No.2 (2015), pp.301-310
http://dx.doi.org/10.14257/ijhit.2015.8.2.28

SnIClustering Algorithm Based on Sampling and Filtering under the MapReduce Framework

Fei Yang[1], Wan-zhen Zhang[2] and Wei Dai[*3]

[1]*School of Computer Science and Technology, Hubei Polytechnic University, Huangshi 435003, Hubei, China*
[2]*Guilin University of Electronic Technology, Guilin 541004, Guangxi, China*
[3]*School of Economics and Management, Hubei Polytechnic University, Huangshi 435003, Hubei, China*
[1]*yfvs07@163.com,*[2]*wan_zer@163.com,*[*3]*dweisky@163.com (Corresponding Author)*

Abstract

SnIClustering Algorithm is put forward to deal with the large number of intermediate values when processing MapReduce. SnIClustering Algorithm picks up a few representative data through cluster sampling, and then retains the useful data through filtration according to the distribution characteristics. By doing so, intermediate values of MapReduce can be reduced sharply, saving time and easing network load. The last step is to cluster the selected data and samples. Experimental results show that SnIClustering is suitable to process large-scale data, since it can both process large-scale data within a short time and maintain fine clustering effect.

Keywords*: Data mining; MapReduce; Clustering; Hadoop*

1. Introduction

Data mining is an eye-catching data analysis technique, since it can help users to target hidden but useful mode or knowledge, in the hope of improving the service. For instance, Amazon and Taobao apply this technique to analyze product association, while Youku recommends videos for users based on its processing data. Mass data mining is contributive to providing better Internet service, making the service more people-oriented. Consequently, data mining is of great importance to modern service industry and national security. Clustering, as an essential component in data mining, plays a crucial part in mass data.

MapReduce is still an emerging programming model to analyze large-scale data. Originally, it was designed to analyze linear data stream. Thus, it is ineffective to process iterative operation on MapReduce. The thesis is aimed to design a new clustering algorithm to analyze large-scale data aided by MapReduce. The new algorithm needs to maintain quality and time-efficiency of clustering in the serial communication and realize parallelization with MapReduce.

In MapReduce, all data are presented in key-value pairs (KVP). A value which is the data of users, and the key, is the unique reducer of that data. MapReduce works in three phases. Firstly, Map. Preprocess the data; get a series of intermediate values (KVPs), temporarily storage the data in local disk and output. Secondly, Shuffle. Transmit the KVPs with same keys to the same reducer. Thirdly, reduce. Process the data and output. Thus, when processing the large-scale data with MapReduce, the following questions need to be thought:

ISSN: 1738-9968 IJHIT

1) How to minimize *I/O* expense? 2) How to minimize network load? 3) How to optimize clustering output?

Given the above questions, this paper learned from the thought of effective sampling [1] and filtering [2], and designed the clustering Algorithm SnIClustering. This algorithm first samples the data set entered rapidly through probability proportional sampling. And then effectively filters the data set so as to greatly reduce the mediate results produced by mapper, the disk operation time of these mediate results and the network flow caused when shuffling the results to reducer. Last, execute clustering to the remaining data and sample after filtering. The results proves that, compared by reference algorithm, the SnIClustering Algorithm not only works faster but also yields favorable clustering results.

2. Relevant Works

Robson, together with some others [2], proposed a clustering algorithm SnI based on MapReduce. One of its major characteristics is that it takes the features of MapReduce programming framework into full consideration so as to cut the *I/O* expense and online expanse as much as possible. SnI assumes that most data in physical data set are included in some major clusters, then try to recognize the distribution features of these clusters through random sampling, and lastly execute the following clustering. In this way, it notably reduces the mediate data produced during the map stage in SnI and the *I/O* operation time of the temporary storage and visits to these data. Meanwhile, the amounts of data transmitted during shuffle stage are also greatly cut and the network loads between mapper and reducer are alleviated. The sampling and filtering method adopted in SnI have distinct disadvantages as well: 1) with large amounts of data, even the sampling is done in a small percentage, the scale of the samples will be huge and thus leads to a great expense in time and storage space; 2) the quality of the sample is not promising and the filtering will end up poorly if the samples are closely distributed.

k-meansII [1] always chooses the data points which is the farthest from the collected samples. Therefore, the samples collected can better reflect the distribution features of the total data set and the points away from the clusters can be well-handle. The sampling process of k-meansII itself is a constant approximation algorithm to the k-means problem. During the real operation, the original clustering center of k-means algorithm produced by the few samples produced through this way can effectively reduce the iterations needed for the algorithm to reach the convergence condition and the quality of clustering can be improved. K-meansII adopts k-means algorithm during the final clustering stage while MapReduce produces many extra expanses when handling iterative operation, and thus reduces the overall time performance of the algorithm.

He and some others [3] used MapReduce to design MR-DBSCAN algorithm. They analyzed the parallel mechanism of the DBSCAN algorithm in detail, and then adopted an optimizing strategy in MR-DBSCAN algorithm, which reduced the visit frequency, time and space complexity of *I/O*. The algorithm designed a practical data partitioning strategy for the large scale space data set without index during the data partitioning stage to solve the load balancing problem. It also realized effective elasticity and speed.

3. SnIclustering Algorithms

3.1. Algorithmic Thinking

Sampling Process is as follows.

1. Selecting a random set of sample points *C* among the input data set *X*.

2. Calculating the initialization overhead of clustering $\varphi = \phi_X(C)$

3. For *i*=1 to $O(\log\phi)$ do

4. Independent Sampling each point $x \in X$ with probability of $p(x) = \frac{\ell \cdot d^2(x, C)}{\phi_X(C)}$ and get sample set C'

5. $C = C \cup C'$

6. End for

7. For each sample point $c_i \in C$, calculating the number of points *n* nearer to c_i than to $c_{j(j!=i)}$, and setting *n* as the weight of c_i.

8. Clustering the weighted sample set and get *k* clusters and *k* centroid. Among which,

$$\phi_X(C) = \sum_{x \in X} \min_{i=1,\cdots,k} \|x - c_i\|^2$$

is sampled factor, proportional to *k*[4-6].

Suppose that *k*=3, =3, total sample number is 7, then the sampling process of k-means II is presented in Figure 1, Figure 2 and Figure 3.

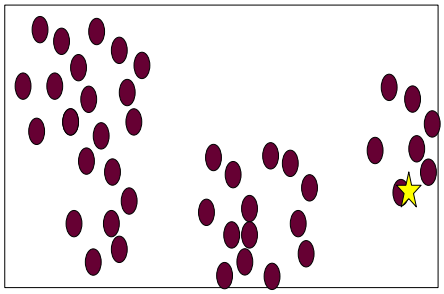

Figure 1. The Initialization of Samples

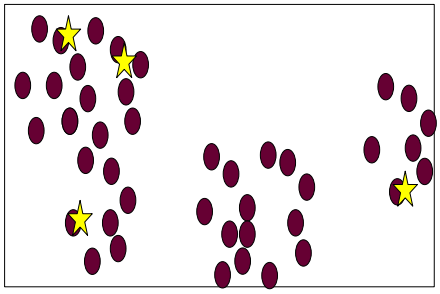

Figure 2. First Round of Iterative Sampling

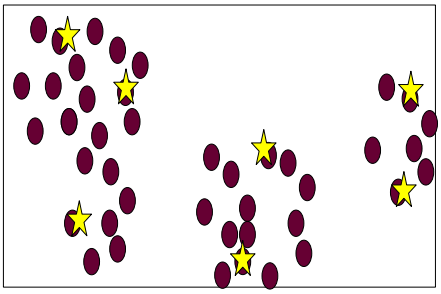

Figure 3. Second Round of Iterative Sampling

SnIClustering Algorithm first selects a small sample from large amounts of data, and clustering them to get clusters and their feature descriptions. The description of a cluster is a $2*d$ dimension vector $F=(x^i_{\min}, x^i_{\max})$. d is the dimension of original data, $x^i_{\min}$ and $x^i_{\max}$ represent the maximum and minimum of the dimension. Then filter the original data according to feature descriptions, i.e. filter the data within $[x^i_{\min}, x^i_{\max}]$. As shown in Figure 4 and Figure 5, data inside the rectangular will be filtered. At last, clustering the unfiltered data and sample data and get the final result.

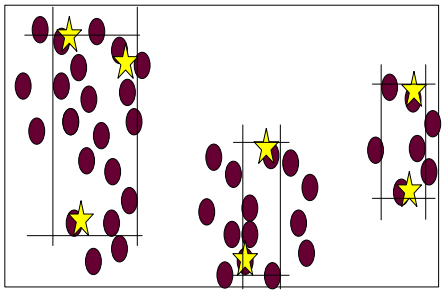

Figure 4. Before Data Filtering

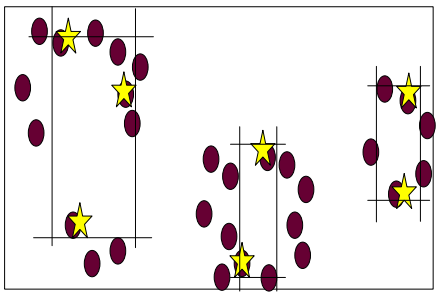

Figure 5. After Data Filtering

3.2. Algorithm Description

Pseudocode of SnIClustering (X, k, ℓ) is as follows[7-10].

Input: X(Initial Set), k(number of clusters), ℓ (sampled factors)

Output: Final clustering result.

Procedures:

1. Sampling.

2. Selecting a random set of sample point C.

3. Each mapper is responsible for a data section $X' \subseteq X$ and calculating the sectional initialization overhead of clustering $\phi_{X'}(C)$.

4. A reducer will be responsible for adding up all the $\phi_{X'}(C)$ and get the overall initialization clustering overhead $\phi_{X'}(C)$.

5. Making $\varphi = \phi_X(C)$.

6. For i=1 to $O(\log\phi)$ do

7. Each mapper will conduct independent sampling $x \in X'$ with the probability of $p(x) = \frac{\ell \cdot d^2(x,C)}{\phi_X(C)}$

8. A reducer is responsible for collecting the sample points from each mapper and including them in C.

9. Recalculating clustering overhead $\phi_X(C)$ according to the new sample set C, updating φ as in procedure 3 and 4.

10. End for

11. For each sample point $c_i \in C$, each mapper will calculate the number of points n nearer to ci than to $c_{j(j!=i)}$ (Using combiner to conduct local statistics)

12. On the reducer side, for each sample point $c_i \in C$, adding up all the results derived from mapper side. The sum will be the weight of c_i.

13. Clustering weighted sample points and get k clusters, get the $Center_{n\,(n=1,\ldots,k)}$ and feature description $F_{n\,(n=1,\ldots,k)}$.

14. Data filtering and final clustering.

15. In each mapper, for any x, $x \in X'$, if x falls within the boundary of $F_{n\,(n=1,\ldots,k)}$, it will be filtered, otherwise, it is sent to reducer.

16. Reducer receives data and sample data from mapper, and clustering the data locally.

17. return the clustering results

In step 13, k-means++ algorithm is used to clustering sample data. In step 16, $Center_n$ $(n=1,\ldots,k)$ is initial clustering center and k-means algorithm is used [11-12]. That can improve the clustering effect of k-means and reduce the iteration times.

4. Experimental Evaluation

In this essay, the experimental environment is mainly determined by 3 computers, which are the composition of Hadoop cluster. Among this environment, the Hadoop version is 1.0.0. Computer configuration is Intel(R)Core(TM)i3-2100 CPU 3.1GHz, Dual-Core, 4.00GB memory, 1TB external memory and Windows 7 operating system.

4.1. Data-set and Parameter

To make the testing algorithm more efficient, we generate a set of semantic data sets of 8 clusters. Each data set has 10,000 records and has the number of dimensions as 2, 4, 8 respectively. The data in each dimension is in accordance with the normal distribution and is standard in the range of [0, 1]. To the elasticity and speedup ratio of the testing algorithm, 3 data sets contain 10 number of dimensions has been generated. The corresponding record number is 1*108(about 1GB), 2*108, 3*108 respectively. About the parameter setting of K-means II algorithm, we perform the iteration for 5 times and receive 50*k samples, each iteration can give us 10*k samples. Then, we use the k-means++ algorithm to cluster those samples. The parameter setting of SnI algorithm stays the same with the primary sources. The experimental result is the average value of 10 experimental results.

4.2. Experimental Results and Analysis

In this essay, we use clustering cost, adopted in k-means II, as the assessment criteria towards clustering quality. We also test the speedup ratio and elasticity in algorithms. The experimental results are shown in Figures 6-9.

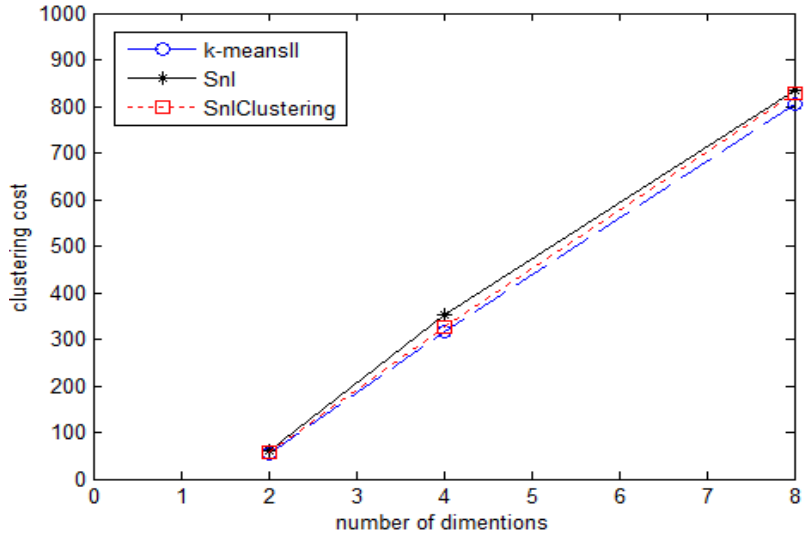

Figure 6. Clustering Cost

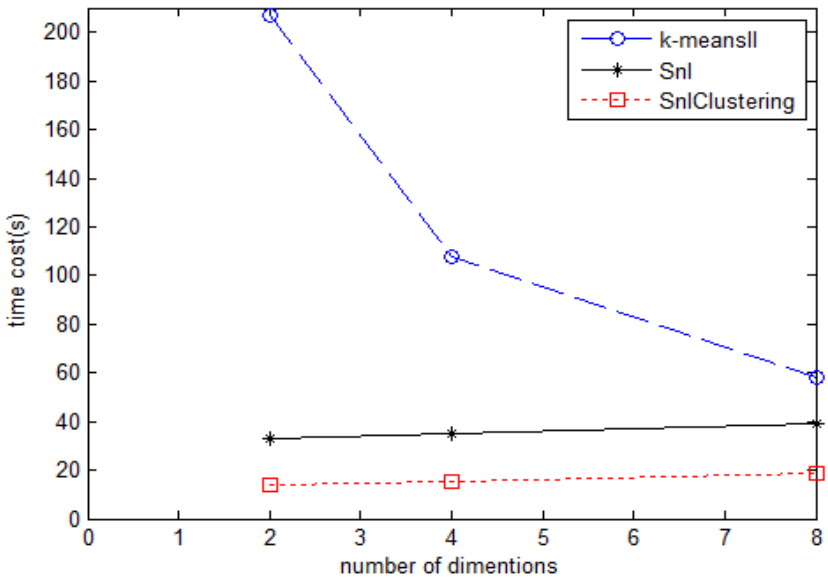

Figure 7. Time Cost

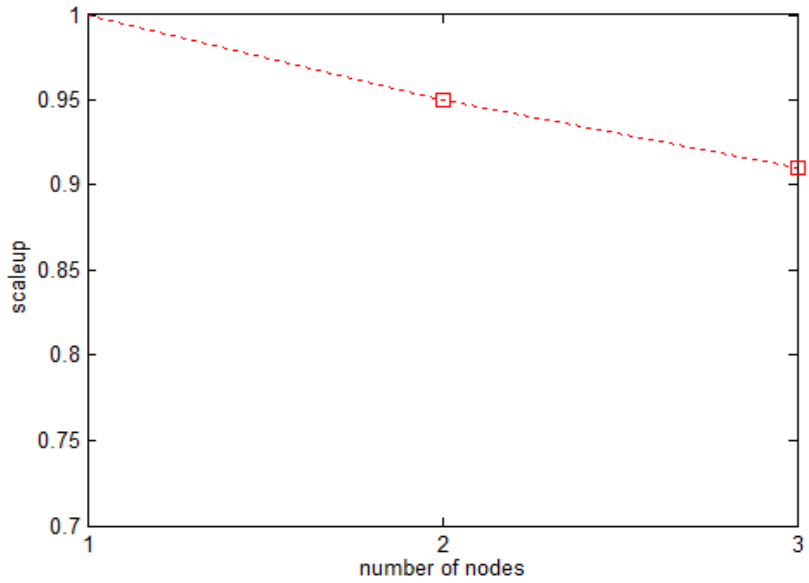

Figure 8. Elassticity

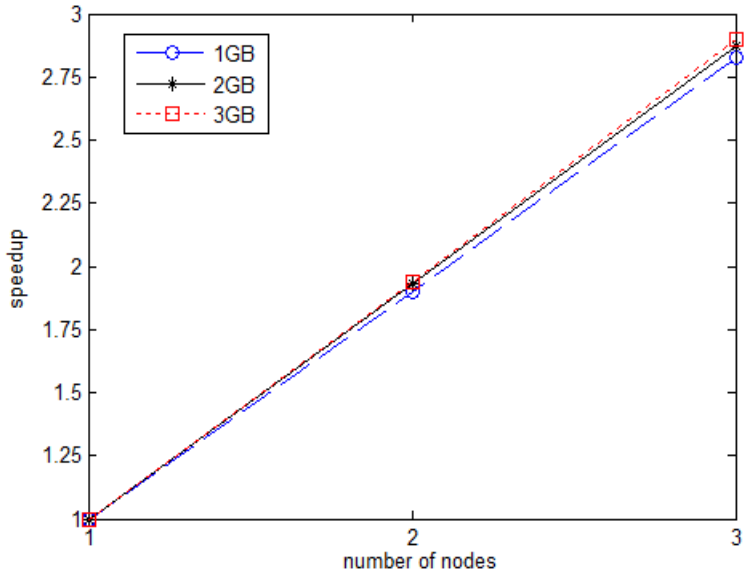

Figure 9. Speedup Ratio

From the results, we can see that compared with k-means II algorithm, the other two algorithms using filtering ideas have good time performance. In theory, the bigger of data-set, the more comparative advantage in time, which are gained by SnIClustering and SnI. This is because the frame of MapReduce has some implication operation which will cost some time. When the data-set is small, the percentage of this time in the whole time is big, however, the condition will be totally different and the small number can even be neglected. In the clustering quality, our calculation method is near to k-means II, better than SnI and has well-performed speedup ratio and elasticity, which worth to dig the mass data set [13-14].

Besides, we have examined the filtration efficiency (data volume has been filtered/original data volume) of the two algorithms. Under the condition of efficient sampling, the SnIClustering can filter the data stably and efficiently. But to SnI, because of stochastic sampling technique, the filtration efficiency is unstable, sometimes even worse. The filtration efficiency of the two algorithms are shown in Figure 10.

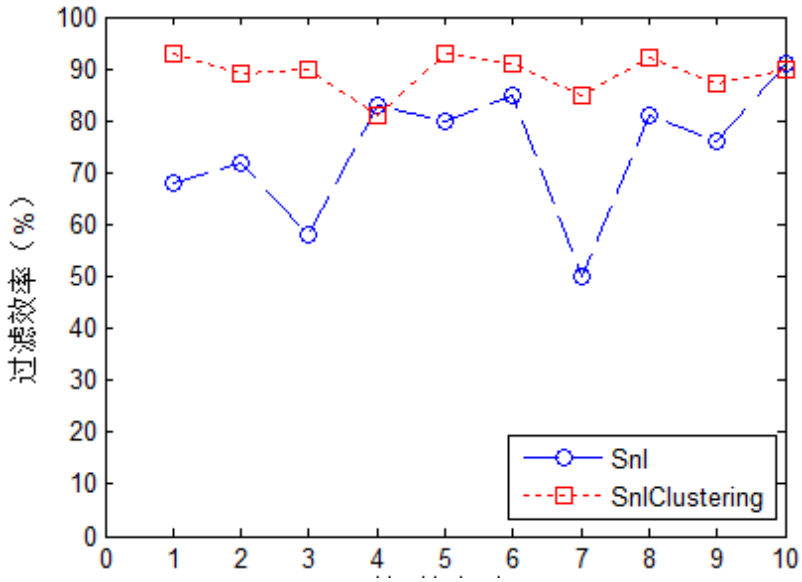

Figure 10. Filtration Efficiency

5. Conclusions

To the problem of intermediate value in the process of carrying out the algorithm in MapReduce, SnIClustering is proposed in this essay, which is based on sample and filtration. First, we introduce the necessity of using MapReduce to cope mass date-set and some major problems concerning the design of clustering method and some works in the trial of problem-solving. Then, the basic idea and pseudocode in SnIClustering are described. Finally, we form the experimental platform to exam the algorithm efficiency. The experiment results show that the SnIClustering can reduce intermediate data, the cost of Read and write in *I/O*, and communication cost. So the algorithm can gain a good time performance. At the same time, this algorithm performs greater clustering quality, elasticity and speedup ratio.

Acknowledgements

This study has been financially supported by Humanities and Social Sciences Youth Fund Project of Ministry of Education of China (No.13YJCZH028), Science and Technology Research Foundation of Education Bureau of Hubei Province (No. B2013064) and Hubei Polytechnic University Innovative Talents Project (No.12xjz20C).

References

[1] B. Bahmani, B. Moseley and A. Vattani, “Scalable k-means++[J]”, Proceedings of the VLDB Endowment, vol. 5, no. 7, (**2012**), pp. 622-633.
[2] R. L. Ferreira Cordeiro, C. Traina Junior and A. J. Machado Traina, “Clustering very large multi-dimensional datasets with mapreduce[C]”, Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, (**2011),** pp. 690-698.
[3] Y. He, H. Tan and W. Luo, “Mr-dbscan: An efficient parallel density-based clustering algorithm using MapReduce[C]”, Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on. IEEE, (**2011**), pp. 473-480.

[4] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding[C]", Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, (**2007**), pp. 1027-1035.
[5] R. Jin, A. Goswami and G. Agrawal, "Fast and exact out-of-core and distributed k-means clustering [J]. Knowledge and Information Systems, vol. 10, no. 1, (**2006**), pp. 17-40.
[6] S. Datta, C. Giannella and H. Kargupta, "K-Means Clustering Over a Large, Dynamic Network[C]", SDM, (**2006**), pp. 153-164.
[7] J. Ekanayake, H. Li and B. Zhang, "Twister: a runtime for iterative mapreduce[C]", Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM, (**2010**), pp. 810-818.
[8] A. Ene, S. Im and B. Moseley, "Fast clustering using MapReduce[C]", Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, (**2011**), pp. 681-689.
[9] G. Malewicz, M. H. Austern and A. J. C. Bik, "Pregel: a system for large-scale graph processing[C]", Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, (**2010),** pp. 135-146.
[10] T. Sun, C. Shu and F. Li, "An efficient hierarchical clustering method for large datasets with map-reduce[C]", Parallel and Distributed Computing, Applications and Technologies, 2009 International Conference on. IEEE, (**2009**), pp. 494-499.
[11] J. Ekanayake, T. Gunarathne and J. Qiu, "Cloud technologies for bioinformatics applications [J]", IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 6, (**2011**), pp. 998-1011.
[12] A. Dave, W. Lu and J. Jackson, "CloudClustering: Toward an iterative data processing pattern on the cloud[C]", Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on. IEEE, (**2011),** pp. 1132-1137.
[13] A. J. Dou, V. Kalogeraki and D. Gunopulos, "Data clustering on a network of mobile smartphones[C]", Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on. IEEE, (**2011),** pp. 118-127.
[14] P. Mell and T. Grance, "The NIST definition of cloud computing [J]", (**2011**).

Authors

Fei Yang, she received her B.S.in Computer Science and Technology Department (2003) from Hubei Normal University and M.S. in school of Computer Science and Engineering (2009) from Wuhan University of Technology. Now she is a full researcher of informatics at Computer Science and Technology Department, Hubei Polytechnic University. Her current research interests include different aspects of data mining, embedded technology and dependable Computing.

Wan-zhen Zhang, she received her B.S. in Computer Science and Technology department(2003) from Hubei Normal University and M.S. in school of Computer Science and Engineering (2011) from Guilin University of Electronic Technology.Now she works in Guilin University of Electronic Technology and her current research interests include different aspects of data mining.

Wei Dai, he received his M.S.E. in Computer Science and Technology (2009) and Ph.D. (2012) from Wuhan University of Technology. Now he is a full researcher of informatics at Economics and Management Department, Hubei Polytechnic University. His current research interests include different aspects of Intelligence Computing and Information Systems.

