Gossip-Based Load Balance Strategy in Big Data Systems with Hierarchical Processors

Xiufeng Cao¹ · Shu Gao¹ · Liangchen Chen^{1,2}

Published online: 21 August 2017

© The Author(s) 2017. This article is an open access publication

Abstract In big data systems, data are assigned to different processors by the system manager, which has a large amount of work to perform, such as achieving load balances and allocating data to the system processors in a centralized way. To alleviate its load, we claim that load balancing can be conducted in a decentralized way, and thus, the system manager need not be in charge of this task anymore. Two decentralized approaches are proposed for load balancing schemes, namely, a utilization scheme based on a load balance algorithm (UBLB) and a number of layers scheme based on a load balance algorithm (NLBLB). In the UBLB scheme, considering the hierarchy of the processor's processing abilities, a gossip-based algorithm is proposed to achieve load balance using the jobs' utilizations as load balance indicators in addition to the number of jobs. The reason for this action is that the processor's process abilities are different from one another. Thus, the utilization indicator is more reasonable. In the NLBLB scheme, the processors are classified into different layers according to their processing abilities. In each layer, a sub-load balance is conducted, which means that the UBLB is achieved in a sub-region. The efficiencies of the two proposed schemes are validated by simulation, which proves their positive effect.

Keywords Big data system · Hierarchical processors · Load balance · Gossip algorithm

✉ Xiufeng Cao
cxf0417@hotmail.com

¹ School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430000, China

² Department of Computer Application, China Institute of Industrial Relations, Beijing 100048, China

1 Introduction

Currently, a large amount of data is produced every minute, which has brought about the big data era. People have already realized the wealth of useful information in big data systems, but they cannot obtain valuable information using traditional data processes or tools; the data format in a big data system involves regularity and could be structured, semi-structured, or unstructured. Thus, a simple big data definition [1] was defined as follows: “The term Big Data applies to information that can’t be processed or analysed using traditional processes or tools”. According to the description in [1–4], there exist three main characteristics in a big data system, namely, volume, velocity, and variety (3V). Volume indicates that the scale of the data is very large. Velocity means that the data should be processed rapidly, timely, and efficiently. Variety stands for the variety of the data. Three more big data characteristics are mentioned in [2, 3], which are variability, veracity, and complexity.

Big data development faces some challenges [5, 6], which are listed as follows:

1. Data representation: how to make raw data be meaningful for computer processing or human interpretation.
2. Redundancy reduction and data compression: how to reduce the data redundancy and compress the data to reduce the cost (e.g., storage space, processing time).
3. Data lifecycle management: numerous data are produced every minute, and the storage space or process ability is limited; thus, how to decide what data should be stored or processed is an important question.
4. Analytical mechanism: traditional data process tools lack scalability and expandability, and how to develop an effective processing architecture is very important.
5. Energy management: a big data system must work all day long; thus, how to make it be green is a vital research area.
6. Expandability and scalability: data are produced continuously, and thus, how to design an analytical algorithm to fit the future data is a working direction.
7. Security and privacy: some confidential and personal information is hidden in the raw data. How to make the information not be leaked is vitally important.

Many companies have proposed big data architectures to describe the characteristics of big data. Oracle has provided a Big Data and Analytics Reference Architecture, shown in Fig. 1 [7]. This system can be divided into the following three aspects (layers): unified Information Management, Real-Time Analytics, and Intelligent Processes.

The unified information management layer addresses the requirement of managing the information entirely rather than maintaining governed silos independently. It is composed of high volume data acquisition, multi-structured data organization and discovery, low latency data processing and a single version of the truth. Real-time analytics allow a business to leverage useful information and analysis as events are actively occurring. It is composed of speed of thought analysis, interactive dashboards, advanced analytics and event processing. The Intelligent Processes make the business process executions more effective and efficient. It is composed of Application-Embedded Analysis, Optimized Rules and Recommendations, Guided User Navigation and Performance and Strategy Management. Based on this architecture, the Oracle big data platform can address many different types of data, including Operational Data, COTS Data, Content, Authoritative Data, System-Generated Data, External Data, Historical Data and Analytical Data.

When raw data arrives, a big data system will process it using a big data platform, which is composed of a large number of processors. Different jobs arrive continuously. Optimal

Fig. 1 Oracle big data and analytics reference architecture [7]

assignment of these jobs to processors is a vital challenge and an important problem. Traditionally, it can be solved by a centralized scheme. A job assigner in a big data system will monitor all of the processors and assign a newly arrived job to the processor that has the lowest load. However, the job assigner must perform a large amount of work, such as allocating data to the system processors, balancing the load among the processors, maintaining all of the nodes' states, and so on. Then, in this paper, decentralized algorithms are proposed to alleviate the load of the big data job assigner.

In our opinion, the load balance problem can be solved in a decentralized way. The outcome is that there is no need for the job assigner to monitor the load on every processor. The decentralized gossip algorithm is employed to achieve load balance. Facing the fact that the processors that are used in a big data system could have different process abilities because the system could be upgraded after a period of time. Two algorithms are proposed. The first is a utilization scheme that is based on a load balance algorithm (UBLB). A traditional gossip algorithm exchanges and updates the load information between each pair of neighbours to make the load balance. The utilization is used to instead of the load to copy the hierarchical process abilities of the processors. The second algorithm is a number of layers based load balance algorithm (NLBLB). This algorithm exchanges and updates the load information as in a traditional scheme. However, the processor abilities are classified into different layers. In each layer, the traditional gossip algorithm is applied to achieve the load balance. In the simulation part, some simulations are given to verify the efficiency of our proposed schemes. Firstly, the UBLB is compared with the traditional decentralized load balance schemes. It is proven that the UBLB can work more efficiently than the traditional algorithms. Then, the NLBLB is simulated to demonstrate that it can achieve efficient load balancing.

The remainder of this paper is organized as follows. Section 2 briefly summarizes the related work. Section 3 formally presents the system model. Section 4 describes two load balance algorithms with hierarchical processors. Numerical simulations are given in Sect. 5, and conclusions are provided in Sect. 6.

2 Related Work

In this section, some related work is introduced. Load balancing is very important for multi-processor systems. It can minimize the execution time of jobs running in parallel [8]. A big data system is a multi-processor system. Thus, the load balance problem should be solved properly. However, to the best of our knowledge, there is no related work that has researched the hierarchical processors in big data systems until now. Some researchers have conducted work on similar topics, such as [9–15].

In [9], the authors designed a distributed big data infrastructure, which has remote agents control access points. Users in different geographical locations can have access to the multiple big data resources. They used the designed infrastructure to monitor the execution, collect the useful statistical data, and upload the outcomes from a remote HDFS to a certain storage machine. The results showed that this infrastructure could achieve a load balance in an efficient way.

The authors of [10] stated that when making scheduling decisions in the condition of multiple schedulers, load balance techniques can optimize the performance. They proposed a distributed task scheduler for multiple task computing. They used the distributed key-value store to organize and scale the task data-locality, task dependency, and metadata. The simulation results demonstrated that it could achieve an optimized load balance.

The authors in [11] noted that many companies faced a big data processing bottleneck. Then, they moved their data to cloud storage. In this way, other questions were produced, especially security and privacy issues, because the cloud owners could not be trusted continuously. Turning to the ORAM algorithm, the authors deployed privacy-preserving access to a big data system. To solve the load balance problem in this design, they proposed a data placement scheme that could improve the availability and responsiveness. Simulation results showed that the proposed algorithm could find outcomes that were close to the optimal solution in the condition of maintaining the load balance.

The authors in [12] claimed that a distributed storage system faced the data load skewed problem. A commonly used method is assigning data using uniform consistent hashing, which has the loss of data-locality problem. The authors proposed a new approach to achieving a load balance via partitioning the data into small subsets, and these subsets can be relocated independently. They designed a novel distributed key-value store that can make a load balance while retaining flexibility.

The authors in [13] stated that Hadoop is one of the successful applications of big data. In some conditions, the Distributed File System can accomplish the load balancing. However, it cannot address an overload rack. Thus, they proposed a new algorithm to make the system load balance in the overload racks a preferential condition. They classified the balance list into the following two categories: PriorBalanceList and NextForBalanceList. Simulation results demonstrated that the new scheme could balance the overload racks efficiently.

The above-mentioned achievements all solve the load balance of big data from different aspects. However, they did not consider a hierarchy of the processors in the big data system. Thus, the gossip algorithm will be turned to design new load balance schemes. Next, some related work is introduced on the gossip algorithm being used in different applications, such as [16–22].

In [16], the authors studied the load balance problem of networks whose nodes' speeds are different from one another. The case of finite tasks with different costs was considered. They proposed a gossip-based solution. The main idea was that swaps were executed

randomly between nodes. They examined two different net topologies: fully connected and generalized ring topologies. Finally, they also examined the convergence properties of the proposed scheme.

The authors of [17] focused on arbitrarily connected graphs and studied the load balance optimization problem. They treated the connected network as a homogeneous graph and proposed a randomized load balance algorithm that is gossip based. They showed that a partial swap was a key issue to ensure good performance of the proposed procedure. Then, they proposed a simple sub-optimal heuristic and demonstrated its convergence properties within the same bounds of GQG-IPP. Their simulation results demonstrated that the achieved consensus was optimal.

The authors of [18] studied the dynamic load balancing problem when scheduling in the self-organized desktop Grid environment. The proposed algorithm has the feature of nodes joining or leaving the Grid freely at runtime. Turning to the transparent process migration, user tasks could be optimally launched. The gossip-based protocol was used to aggregate the runtime load information.

The authors of [19] stated that computer networks are heterogeneous, large-scale and dynamic. The aggregation of information has become increasingly important. Thus, they proposed a gossip-based algorithm in a decentralized way to compute the network aggregate values. This scheme worked proactively to track the changes in the network. Simulation results showed that the proposed algorithm is efficient and robust.

Overall, there are many outstanding achievements in studying big data and the gossip algorithm. In this paper, the gossip algorithm is applied to solve the load balance problem in big data systems in a distributed way, which can alleviate the load of a big data system. In this way, the performance of the system is improved dramatically.

3 System Model

In a big data system, many different types of data are processed, as shown in Fig. 2. The job assigner assigns jobs to the processors. Usually, the job assigner has many jobs to conduct, such as allocating data to the system processors, making the load balanced among the processors, and maintaining all of the nodes' states. To alleviate the load of the data assigner, a gossip-based strategy is used to achieve load balance in a distributed way, and the data assigner must not care about the load balance anymore.

In general, there are two definitions of load balance. The first definition is that if all of the processors receive the same number of jobs, we say that load balance is achieved. According to this definition, the assigned number of jobs is equal, but the differences from job heterogeneity for the big data processor are ignored. The other definition is that if all of the processors have the same utilization (which is calculated by dividing the number of jobs of a processor by its process ability), we state that it achieves load balance. Because the jobs are processed by the processors, this definition is reasonable. In this paper, the second definition will be applied to define the load balance.

The reason for the processors' process abilities being different is that the big data system is flexible, and its scale is becoming larger and larger. During the upgrading, the processors used could have different process abilities. Then, the load balance problem is solved in the condition of hierarchical process abilities. Assuming that there are m processors, the process ability and the load of processor i can be written as A_i and L_i . Thus, the utilization rate of processor i can be written as $P_i = \frac{L_i}{A_i}$. To achieve load balance, the

Fig. 2 A big data system

standard deviation of the utilization rate is optimized, which is the same as the approach proposed in [18] and is shown in formula (1).

$$\min \sigma = \sqrt{\frac{1}{m} \sum_{i=1}^m \left(P_i - \frac{\sum_{i=1}^m P_i}{m} \right)^2} \quad (1)$$

In this formula, m is the number of processors. P_i is the utilization rate of processor i , and σ is the standard deviation. If the standard deviation is minimized, then load balance is achieved. In the following section, two schemes will be given to achieve a decentralized load balance.

4 Two Load Balance Algorithms with Hierarchical Processors

In this section, two different load balance algorithms will be introduced to solve the above-mentioned problem. The scenario is that the processors' process abilities are different and the optimized problem is achieving a similar processor utilization. The first approach is called a utilization-based load balance algorithm, which can achieve load balance in the entire big data system but introduces more communication costs and has a higher convergence time. The second approach is called the number of layers based load balance algorithm, which can achieve load balance in the different layers of a big data system. The

Fig. 3 Sub-load balance process of using a utilization-based load balance algorithm

advantage is that it needs less communicating cost and a lower convergence time, while the performance of the load balance is not as good as a utilization-based load balance scheme.

4.1 Utilization-Based Load Balance Algorithm

According to the proposed scenario, the abilities of different processors could be different. Firstly, a definition is presented to introduce the load balance between two processors.

Definition 1 (*Sub-load balance*) Given that the number of jobs assigned to two processors i and j is L_i and L_j , respectively, and the process abilities of these two processors are A_i and A_j . We say that the sub-load balance is obtained if formula (2) is achieved.

$$\frac{L_i}{A_i} = \frac{L_j}{A_j} \quad (2)$$

To achieve the sub-load balance, processors i and j must perform the following steps, and the schematic process is shown in Fig. 3.

Step 1: Each processor calculates its own utilization rate;

Step 2: Processors share the utilization rate information with each other;

Step 3: The processor who has a larger utilization rate (assume that processor i has the larger utilization rate) gives the other processor some jobs, and the number of jobs is calculated using formula (3)

$$L_i - \frac{A_i \times (L_i + L_j)}{A_i + A_j} \quad (3)$$

Using this method, the sub-load balance is achieved. Then, it can be used in each pair of processors to achieve the load balance between the two processors. This procedure can be repeated to achieve load balance in the entire big data system in a distributed way.

The generalized utilization-based load balance algorithm (UBLB) is presented. The mechanism is as follows:

Algorithm 1 UBLB Pseudo-code

Initialization

$A = \{A_1, A_2, \dots, A_m\}$ // process abilities of m processors

$L = \{L_1, L_2, \dots, L_m\}$ // numbers of jobs assigned to the m processors

I // indicator, if a processor is performing a sub-load balance, it is set to 1; otherwise, it is set to 0

Do

For each processor whose indicator is 0 // achieve the sub-load balance

Find a neighbour whose indicator is 0

Set the indicators of it and its neighbour to 1

Do sub-load balance between these two processors

End FIND

Set the indicators of it and its neighbour to 0

Return to **FOR**

End FOR

End DO

According to the UBLB algorithm, the big data system works in the following way. Once the data assigner obtains a task that can be divided into n jobs, it randomly assigns those jobs to different processors. The UBLB algorithm is running all of the time to make each processor obtain the same share of job utilization. In this way, the data assigner performs the load balance task, which can alleviate its load.

4.2 Number of Layers Based Load Balance Algorithm

Because of upgrading, processors process abilities are different; another way to achieve load balancing in a big data system is to classify the process abilities into different layers.

An example is used to show the layer-forming process. Table 1 lists the eight different processors as well as their corresponding process abilities. The processors are classified into three layers, as shown in Fig. 4. The process ability of the processors in the first layer

Table 1 Processor IDs and corresponding process abilities

Processor IDs	Process abilities (GB/s)
P_1	2.1
P_2	1.3
P_3	2.4
P_4	3.2
P_5	1.7
P_6	2.4
P_7	3.6
P_8	3.7

Fig. 4 Layered processor zones **Fig. 5** Sub-load balance process of the number of layers based load balance algorithm

is between 1 and 2, the process ability of the processors in the second layer is between 2 and 3, and the process ability of the processors in the third layer is larger than 3.

Another definition is presented to demonstrate the load balance between two processors in a certain processor zone.

After classifying the processors into a different category, the number of jobs initially assigned to different layers is according to the process abilities of the processors in each layer. Then, the sub-load balancing can be conducted to apply the following definition.

Definition 2 (*Layered sub-load balance*) Given that the number of jobs assigned to the processors i and j in layer k are L_i^k and L_j^k , respectively. We state that they achieve a layered sub-load balance in the condition of $L_i^k \approx L_j^k$.

To achieve the layered sub-load balance, processors i and j in layer k must perform the following steps, as shown in Fig. 5.

Step 1: The processors find each other using their neighbours;

Step 2: The processors share the load information with each other;

Step 3: The processor that has a larger load (assuming that processor i has a larger load) gives the other processor some jobs, in the amount of $\frac{L_i^k - L_j^k}{2}$.

Next, the number of layers based load balance algorithm (NLBLB) is also presented. The mechanism is as follows:

Algorithm 2 NLBLB Pseudo-code

Initialization

m // the number of processors

$A = \{A_1, A_2, \dots, A_m\}$ // the process abilities of the m processors

$L = \{L_1, L_2, \dots, L_m\}$ //the number of jobs assigned to the m processors

I // indicator, if a processor is performing layered sub-load balancing, it is set to 1; otherwise, it is set to 0

Do

Classify the processors into different layers

For each processor whose indicator is 0 // achieve the layered sub-load balance

Find a processor that is in the same layer with its indicator being 0 using a neighbour

Set the its indicators and the selected processor to be 1

Do layered sub-load balancing between these two processors

End FIND

Set its indicators and its neighbour to be 0

Return to **FOR**

End FOR

End DO

According to the NLBLB algorithm, the big data system works as follows. There are k layers in the system. Once the data assigner obtains a task that can be divided into n jobs, it randomly assigns $(g \times 2n)/(1 + k)k$ jobs to the g 'th layer's processors. The NLBLB algorithm is running all the time to make each processor obtain the same share of job utilization. In this way, the data assigner also must not perform the load balance task that can alleviate its load. However, it must assign the number of jobs to every layer according to its layer number.

5 Simulation Results

In this section, some simulations are conducted to validate the efficiency of our proposed schemes. In the first simulation, the traditional load balance algorithm is compared with our proposed UBLB algorithm, which is used in a big data system. The scenario is as follows. There are six processors, with the process abilities being $A = \{1.2 \text{ GB/s}, 3.8 \text{ GB/s}, 1.4 \text{ GB/s}, 3.2 \text{ GB/s}, 1.5 \text{ GB/s}, 3.4 \text{ GB/s}\}$ in the big data system. The amount of data that must be processed is 400 GB. At the beginning, the resources are randomly distributed among the processors.

Figure 6 shows that in the traditional load balance scheme, the indicator of the big data system being load balanced is each processor being assigned approximately the same amount of data. This approach ignores the differences in the process abilities of the

Fig. 6 Traditional load balance scheme using the number of jobs as the load balance indicator

different processors. In Fig. 7, if the utilization is considered to be the load balance indicator, the outcome is that the big data system is not load balanced anymore.

Next, the proposed UBLB scheme is simulated to use the same scenario. The difference is that it treats the utilization as a load balance indicator. From Fig. 8, it achieves load balance after a few steps. It is more reasonable for the processor with the stronger ability to process more data.

Fig. 7 The traditional load balance scheme using the utilization of jobs as the load balance indicator

Fig. 8 The proposed UBLB load balance scheme using the utilization of jobs as the load balance indicator

Fig. 9 Proposed NLBLB load balance scheme using the number of jobs as the load balance indicator

Then, the second proposed load balance scheme NLBLB is also emulated. The scenario is similar to the above scenario. There are also 8 processors, with the process abilities being $A = \{1.2 \text{ GB/s}, 3.8 \text{ GB/s}, 1.4 \text{ GB/s}, 3.2 \text{ GB/s}, 1.5 \text{ GB/s}, 3.4 \text{ GB/s}, 1.6 \text{ GB/s}, 3.5 \text{ GB/s}\}$.

The amount of data that must be processed is 400 GB. From this statement, the processors should be classified into two categories. The first one's process ability is approximately 1.5 GB/s, and the second's is approximately 3.5 GB/s. The processors in

Fig. 10 Proposed NLBLB load balance scheme using the utilization of jobs as the load balance indicator

each layer are the same. Therefore, either the number of jobs or the utilization of the jobs is applied as the indicator of the load balance.

Figure 9 shows that the processors in either layer are assigned the same number of jobs, which means that the big data system achieves sub-load balance. Additionally in Fig. 10, the utilization of the jobs is applied as the load balance indicator, and the conclusion is that the NLBLB scheme can also achieve utilization-based load balance.

According to the above simulations, the following conclusions are obtained:

1. Using the utilization of jobs as the load balance indicator is more reasonable.
2. The two proposed load balance schemes work efficiently.
3. If the processor's abilities in a big data system can be classified into several layers, it would be better to use the NLBLB scheme; if the processor's abilities are different from each other, then it would be better to use the UBLB scheme.

6 Conclusions

Big data processing has brought a large benefit to people's daily lives. Today, how to optimize a big data system is a hot research topic. In a traditional big data system, the load balancing is conducted by the job assigner, which has many jobs to complete. If the job assigner is overloaded, then the big data system breaks down. The two decentralized load balancing schemes are proposed that aim to alleviate the load of the job assigner. The first scheme is called UBLB. Considering the hierarchy of the processor's process abilities, we claimed that the load balance indicator should be the utilization of jobs rather than the number of jobs. The second scheme is called NLBLB. In this scheme, the processors are classified into different layers according to their process abilities. In each layer, a sub-load balance could be performed. In the simulations, some scenarios are built to verify the

efficiency of the proposed schemes. From the outcomes, it is concluded that, indeed, these two schemes have good performance.

Acknowledgements The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper. This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 51479155 and 61502404), the Natural Science Foundation of Hubei Province of China (Grant No. 2014CFB190), the Science and Technology Department of Bijie City of Guizhou Province of China (Grant No. 29[2014]), and the Fundamental Research Foundation of the Central Universities (Grant Nos. 16ZY006, 15ZY015), and finished with the help of my Ph.D. supervisor and fellow students.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Zikopoulos, P., & Eaton, C. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. New York: McGraw-Hill Osborne Media.
2. Hilbert, M. (2016). Big data for development: A review of promises and challenges. *Development Policy Review*, 34(1), 135–174.
3. Hilbert, M. (2014). *Digital technology and social change*. <https://www.youtube.com/watch?v=XRV1h47sA&index=51&list=PLtjBSCvWCU3rNm46D3R85efM0hrzjuAIg>. Accessed 25 July 2015.
4. Big Data. (2012). https://en.wikipedia.org/wiki/Big_data#Definition. Accessed 14 March 2014.
5. Labrinidis, A., & Jagadish, H. V. (2012). Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12), 2032–2033.
6. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2), 171–209.
7. Sahu, B. K. (2015). *An Oracle White Paper, Big Data & Analytics Reference Architecture*. <http://www.oracle.com/technetwork/topics/entarch/oracle-wp-big-data-refarch-2015930.pdf>. Accessed 5 September 2016.
8. Willebeek-LeMair, M. H., & Reeves, A. P. (1993). Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, 4(9), 979–993.
9. Fernández, V., Méndez, V., & Pena, T. F. (2015). Federated Big Data for resource aggregation and load balancing with DIRAC. *Procedia Computer Science*, 51, 2769–2773.
10. Wang, K., Zhou, X., Li, T., Zhao, D., Lang, M., & Raicu, I. (2014). Optimizing load balancing and data-locality with data-aware scheduling. In *2014 IEEE international conference on big data* (pp. 119–128).
11. Li, P., & Guo, S. (2014). Load balancing for privacy-preserving access to big data in cloud. In *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)* (pp. 524–528).
12. Kroll, L. (2013). *Load balancing in a distributed storage system for big and small data*. <http://www.diva-portal.org/smash/get/diva2:651375/FULLTEXT01.pdf>. Accessed 17 February 2014.
13. Liu, K., Xu, G., & Yuan, J. E. (2013). An improved hadoop data load balancing algorithm. *Journal of Networks*, 8(12), 2816–2823.
14. Khayyat, Z., Awara, K., Alonazi, A., Jamjoom, H., Williams, D., & Kalnis, P. (2013). Mizan: A system for dynamic load balancing in large-scale graph processing. In *Proceedings of the 8th ACM European conference on computer systems* (pp. 169–182).
15. Bin, Y. (2015). Big data load balancing based on network architecture. In *International conference on intelligent computing* (pp. 307–317).
16. Franceschelli, M., Giua, A., & Seatzu, C. (2009). Load balancing over heterogeneous networks with gossip-based algorithms. In *American control conference, 2009. ACC'09* (pp. 1987–1993).
17. Franceschelli, M., Giua, A., & Seatzu, C. (2007). Load balancing on networks with gossip-based distributed algorithms. In *2007 46th IEEE conference on decision and control* (pp. 500–505).
18. Di, S., Wang, C. L., & Hu, D. H. (2009). Gossip-based dynamic load balancing in a self-organized desktop grid. In *Proceedings for the HPC Asia & APAN 2009 International Conference & Exhibition* (pp. 85–92).

19. Jelasity, M., Montresor, A., & Babaoglu, O. (2005). Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3), 219–252.
20. Johansen, Håvard D., Renesse, R. V., Vigfusson, Y., & Johansen, D. (2015). Fireflies: a secure and scalable membership and gossip service. *Acm Transactions on Computer Systems*, 33(2), 1–32.
21. Mousazadeh, M., & Tork Ladani, B. (2015). Gossip-based data aggregation in hostile environments. *Computer Communications*, 62, 1–12.
22. Matei, I., Somarakis, C., & Baras, J. S. (2015). A generalized gossip algorithm on convex metric spaces. *IEEE Transactions on Automatic Control*, 60(5), 1175–1187.

Xiufeng Cao is Ph.D. candidate in Computer science and technology from Wuhan University of Technology, Wuhan, China. His research interests include Big Data Analysis, Resource Schedule, Cloud Computing, Visual Analytics.

Shu Gao is professor and Ph.D. supervisor at Wuhan University of Technology, received Ph.D. degrees in Computer science and technology from Wuhan University of Technology, Wuhan, China, in 2007. Her research interests include Cloud Computing, Visual Analytics, Intelligent Transportation.

Liangchen Chen is Lecturer at China Institute of Industrial Relations and Ph.D. candidate in Computer Science and Technology from Wuhan University of Technology. His research interests include Big Data Analysis and Visual Analytics.