Job Performance Optimization Method Based on Data Balance in the Wireless Sensor Networks

https://doi.org/10.3991/ijoe.v13i12.7882

Zeyu Sun, Guozeng Zhao(✉), Meng Li
Luoyang Institute of Science and Technology, Luoyang, China
Ly_zgz@163.com

Zhiguo Lv
Xidian University, Xi'an, China

Abstract—In the wireless sensor network, the representative MapReduce computing model based on data center has been widely used in large-scale data processing. In the data transmission phase, the wireless sensor network system uses the hash method to distribute data for each Reduce task based on the number of Reduce tasks. This data partitioning method based on the hash function results in non-uniform distribution of the output data in the data transmission phase and further leads to skewing of the input data in the Reduce task. Data skew will result in load imbalance in the Reduce phase and causes the system performance to degrade. In order to eliminate the data skew problem in the Reduce phase, this paper presents a load balancing method, which consists of two parts: the virtual partitioning method based on the consistent hashing and the heterogeneity-aware loads balancing (HLB) algorithm. The experimental results show that the proposed method can eliminate the data skew in the Reduce phase and distribute the load equitably for each Reduce task. In addition, the method produces less system overhead.

Keywords—wireless sensor networks, MapReduce, hash function, data skew, heterogeneity-aware

1 Introduction

In the MapReduce computing model, there are two types of tasks in a job: Map task and Reduce task. These tasks are dispatched by the system dispatcher to various work nodes in the cluster for execution [1-2]. Obviously, the latest completion time for a MapReduce job depends on when the last Reduce task is completed. If a work node spends a long time in processing the assigned Reduce tasks, the completion time of the entire MapReduce job will be affected [3]. Therefore, when a MapReduce job is running, allocating the appropriate size of input data for each Reduce task and keeping balance of the load at each work node are critical to the performance improvement of the entire MapReduce system. However, in the actual MapReduce cluster, two factors can cause the input data skew in the Reduce task:

1. Partition method for Map task output data in the MapReduce system;
2. Different processing capacities of work nodes.

In order to eliminate the data skew problem in the heterogeneous MapReduce cluster in the Reduce phase, this paper presents a method to balance the skew load in the Reduce task, which consists of two parts: the virtual partitioning method based on the consistent hashing and the HLB algorithm. Firstly, it uses the consistent hash method to carry out virtual partitioning of the Map output data. Secondly, based on the balancing algorithm, it conducts skewed partition detection on the virtual partition sets. If there is an skewed partition, it will carry out partition merge according to the merge policy; otherwise, according to the partition allocation strategy, it will distribute the Map output data corresponding to the virtual partition to each Reducer equally.

2 Speculation execution in the MapReduce system

The speculation mechanism in the MapReduce system starts when two conditions are met: i) all tasks in the job have been started, ii) the system has spare work nodes. Once the speculation task starts, the system will have two identical tasks executed in parallel at the same time. If there is a task is completed, then another concurrent task will be terminated by the dispatcher [4-6]. The speculation mechanism speeds up the backward Map or Reduce task execution time, thereby improving the system performance.

There are many reasons for speculation in the Reduce task, one of which is the competition for resources. Multiple Reduce tasks running at the same node can cause competition for system resources, which further affects the execution speed of the task. Another reason is the skewed load. Skewed load makes the Reduce task with more input data a backward task, which led to the execution of the speculation task. For the execution of the speculative Reduce task due to data skew, the speculation task does not effectively reduce the execution time of the Reduce task for large-grained input loads because the speculative Reduce task also needs to handle the same size of input load [7-8]. Therefore, it is necessary to work out a new and effective method to balance the skew of the Reduce task input data.

2.1 Description of the load balance problem

First, we give formal description of the symbolic variables used in this section. Assuming that there is *m* Mappers and *r* Reducers in a MapReduce cluster, for a job *J* with *MN* Map tasks and *RN* Reduce tasks, the following symbolic variables are given:

Set of *m* Mappers:

$$MP=\{m_1,m_2,m_3,...,m_m\} \tag{1}$$

Set of Mapper mi output partitions:

$$LF_i=\{LF_{i1},LF_{i2},LF_{i3},...,LF_{iq}\} \tag{2}$$

Set of MN Map task output partitions:

$$LM = \{LF_1, LF_2, LF_3, ..., LF_{MN}\} \tag{3}$$

Set of *r* Reducers:

$$R = \{r_1, r_2, r_3, ..., r_r\} \tag{4}$$

Set of RN Reduce task input partitions:

$$LR = \{LR_1, LR_2, LR_3, ..., LR_{RN}\} \tag{5}$$

Execution time of RN Reduce tasks:

$$T = \{T_1, T_2, T_3, ..., T_{RN}\} \tag{6}$$

Definition 1 Allocation thresh hold The proportion of the input data size allocated to Reducer r_i in the size of all Map task output data. For a set *LR* containing *RN* elements, the allocation threshold for Reducer r_i can be obtained by the following formula:

$$\chi_i = \frac{LR_i}{LM} \tag{7}$$

In Formula (7):
LM—— total data size of the MN Map task output partitions in Job *J*;
LR_i——data size of the Reducer r_i input partition in Job *J*.
For Job *J*, the target of balancing the skewed Reduce input load is to minimize the execution time of the job in the Reduce phase, which can be translated into the following optimization problem:

$$\min(\max_{1 \le i \le RN}(T_i))$$
$$s.t.\ \ \varphi \notin LF_i, \bigcup_{i=1}^{MN} LF_i = LM, (A \subseteq LF_i \wedge B \subseteq LF_i \wedge A \neq B) \Rightarrow A \,\mathrm{I}\, B = \varphi,$$
$$\varphi \notin LF_{ij}, \bigcup_{j=1}^{q} LF_{ij} = LF_i, (C \subseteq LF_{ij} \wedge D \subseteq LF_{ij} \wedge C \neq D) \Rightarrow C \,\mathrm{I}\, D = \varphi,$$
$$LR_i = H(LM,\ r_i) \tag{8}$$

For job *J*, the output data size for all Map tasks is equal to the input data size for all Reduce tasks. The three constraints in Formula (8) describe the different components of the Reducer r_i input load. The first constraint indicates that the input data of the *RN* Reducers are from the output of each Map task [9]. The second constraint indicates that for any Mapper m_i, the output consists of multiple partitions. The third constraint indicates that for any Reducer r_i, the input partition size is determined by the partition function $H(LM, r_i)$.

Let's suppose there are three Reduce tasks in MapReduce job J. Fig.1 depicts the execution time of each task in the Reduce phase of the job. As shown in the figure, the input load of Reducer r_i consists of four parts: α_{i1}, α_{i2}, α_{i3} and α_{i4}, the time spent by Reducer r_i in processing the loads in the four parts is T_{i1}, T_{i2}, T_{i3} and T_{i4}, respectively [10-11]. It can be seen from Fig.1 that the completion time of job J depends on the completion time of Reducer r_3. If less input load is allocated to Reducer r_3 through the optimized load allocation method before it is executed, the execution time of the whole job will be shortened.

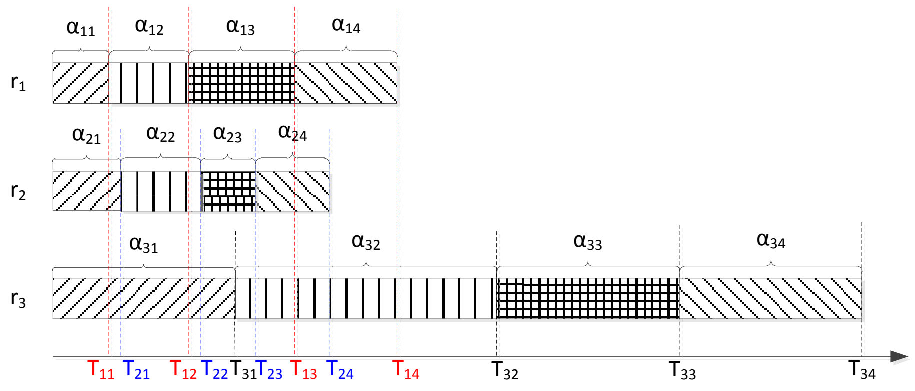


Fig. 1. Effects of the load distribution strategy on the execution time of Reduce tasks

In order to solve the optimization problem described in Formula (8), we propose a heterogeneity-aware loads balancing algorithm. Through this algorithm, the output data of the Map task can be assigned to each Reduce task fairly.

2.2 Balancing methods for Reduce input data

Definition 2 Partition granularity. The granularity of partition stands for the size of the storage space occupied by the key/value data contained in the partition.

Definition 3 Virtual partition Virtual partition is a partition of the Map output data processed by the consistent hashing algorithm.

With the consistent hashing method, the output data of Map can be divided into more data partitions. As the partition granularity decreases, the chance of balancing the Reduce input load increases.

Definition 4 Virtual partition adjustment factors the threshold that controls the number of virtual partitions. This value can be adjusted according to different MapReduce applications.

Let δ represent virtual partition adjustment factor and RN be the number of Reducers. The number of virtual partitions VN can be obtained through the following formula:

$$VN = RN \times \delta \tag{9}$$

Assuming K_i is the key value of a key-value pair in the Mapper output data, the corresponding key-value pair of K_i can be obtained through the following virtual partition function. After virtual partitioning, the *number* v_i of the partition which the key-value pair belongs to is as follows:

$$V_i = \text{hash}(K_i)\text{mod}(RN) \tag{10}$$

Fig.2 shows the Reduce input load distribution process based on virtual partitions. First, all the output data of the Map function are processed through the virtual partition function, forming the mapping relationships between multiple virtual partitions and Reduce tasks.

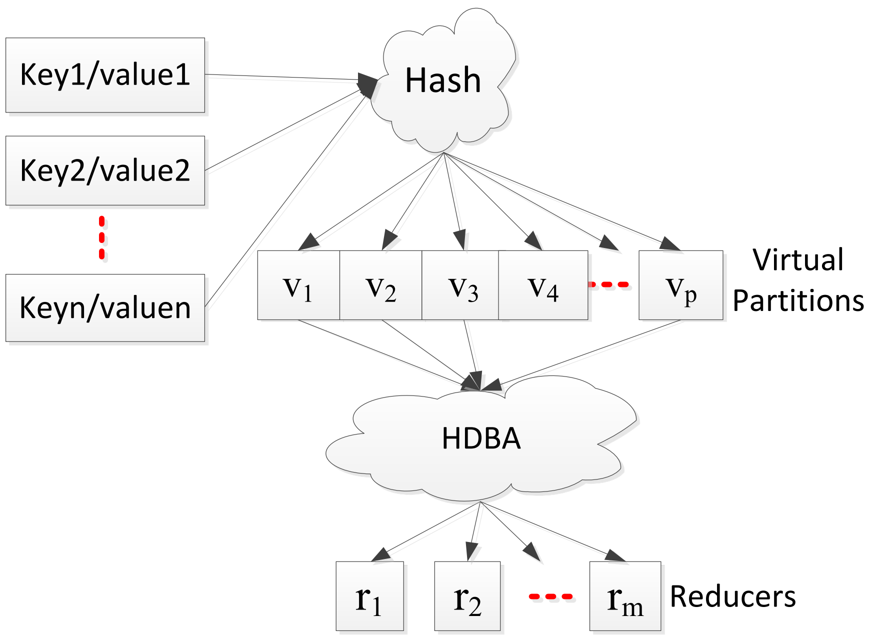


Fig. 2. Reduce load distribution based on virtual partition

2.3 Heterogeneity-aware loads balancing

This paper analyzes this optimization problem and converts it to a set partitioning method [12-13]. When the output data of the Map task contains skewed keys or skewed partitions, it uses the partition merging method to save system resources. Therefore, HSLB consists of three parts: skewed partition detection, partition merging and partition allocation.

In the MapReduce system, the output key/value of the Map function establishes the correspondence relationship with the Reduce task according to the partition function.

From Formulas (1) to (8), it can be seen that the size of each partition is determined by the number of keys contained in the partition. However, if a key contains many value data, this key is called an skewed key, and the partition to which the skewed key belongs is called an skewed partition.

In order to detect skewed partitions, the following definition is given:

Definition 5 Computational capability Computational capability represents the processing capacity of a Reducer at a work node. The computational capability of a Reducer r_i can be calculated through the following formula based on the Reduce task log information completed by the Reducer:

$$CP(\mathrm{r}_i) = \frac{\sum_{i=1}^{n} Input(\mathrm{R}_i) / \mathrm{T}(\mathrm{R}_i)}{n} \tag{11}$$

The optimal solution for the optimization problem shown in Formula (8) is to allocate the input load to each Reducer based on the result of the set partitioning so that the Reduce task run on each Reducer takes the same time. This is because when all tasks in the Reduce stage are completed at the same time, the execution time will be the shortest. Assuming that a Reduce task R_i is dispatched to the working node where Reducer r_i is located, the execution time of the task can be determined by the following formula:

$$Exe\_time(R_i) = TS(r_i) / \mathrm{CP}(r_i) \tag{12}$$

The skewed partition detection algorithm (SPD) consists of three steps: first, it evaluates Reducer's computational capability based on the Reducer task log information successfully executed on each Reducer; after that, it calculates the load distribution threshold for each Reducer based on the objective of balancing the skewed Reduce input load; finally, it detects whether there is any skewed partition according to the load allocation threshold of each Reducer and the sizes of virtual partitions corresponding to all Map task output data [14-15].

When the output data of the Map contains any skewed key, the partition strategy based on the hash function cannot divide the skewed key. In this case, this paper proposes a partition merging strategy. By combining small-granularity partitions, we can release some of the computing resources at the work node and improve the system resource utilization. For example, for MapReduce application A, we assume that the output data of the Map task in A is divided into five partitions with different granularities, where the partition P_1 contains a skewed key. The five partitions are assigned to five Reducers as input loads, respectively.

Let the virtual partition set $L=\{L_1, L_2, \dots, L_P\}$, where $L1$, is the size of the i-th virtual partition. Let all Reducer load distribution threshold set $\chi=\{\chi_1, \chi_2, \chi_3, \dots, \chi_r\}$, where χ_q is the load distribution threshold for Reducer r_q. Then the optimization problem in Formula (4-10) can be transformed into a problem of finding a partition PA for the set L:

$$\begin{aligned}
&PA = \left\{PA_1, PA_2, PA_3, ..., PA_r\right\} \\
&s.t.\ PA_i \cap PA_j = \varphi(1 \le i \ne j \le r) \\
&SUM(PA_i) < \mu \times \chi_i \times \sum_{j=1}^{p} L_j (1 \le i \le r) \\
&\bigcup_{1 \le i \le r} PA_i = L
\end{aligned} \tag{13}$$

When the virtual partition output by the Map task does not contain any skewed partition, the algorithm distributes appropriate size of input load to each Reducer according to its computational capability. Similar to the partition merging process, the partition distribution process also adopts the idea of Bin-Packing. But there are two differences between the two: 1) The number of Reducers is fixed in the partition distribution, while the objective of partition merging is to minimize the number of Reducers to improve the utilization of system resources; so during the partition merging, the number of Reducers is not fixed; 2) The input load thresholds for Reducers are different.

The HSLB algorithm consists of three steps: Firstly, it transforms the Reducer input load distribution problem into the set partitioning problem, solves this set partitioning problem by integrating the Reducer performance model, and obtains the distribution threshold for the Reducer input load. Then, based on the Reducer load distribution threshold obtained, it obtains the set of the Reducer input data size, compares the maximum value in the set with the partition with the largest granularity among all the current virtual partitions, and detects whether there is any skewed partition according to the comparison results. Finally, if there is a skewed partition, it will merge the virtual partitions, and if not, it distributes the appropriate load to each Reducer according to the load distribution threshold.

3 Prediction analysis of the HLB algorithm

The Hadoop platform used in this paper is Version 0.21.In the Hadoop cluster, the virtual machine host is configured as a Job Tracker. This node also works as the Name Node. It configures the remaining 18 virtual machine nodes and 2 physical nodes as Task Tracker nodes, which at the same time are also Data Nodes. In order to ensure that the Hadoop system distributes diverse computation resources for Reduce tasks, it provides different Reduce computation resource pools for some work nodes. In this experiment, HDFS maintains the system default configuration.

We compare the experimental results with the results of the Hadoop Speculative (Hadoop-SP) algorithm. The evaluation on the experiment consists of the following parts: 1) analysis of the virtual partition adjustment factor, to verify how the size of virtual partition adjustment factor affects the performance of the Load Balancer; 2) performance analysis of the Load Balancer, to verify how effective the Load Balancer is in improving the job performance when there is any or no skewed partition in the virtual partitions; 3) analysis of the effects of the performance prediction model on the balance results, to verify the job performance improvement after the Load Balancer uses the performance prediction model proposed in the second section; 4) Balancing overheads, to verify the system overhead incurred by the Load Balancer during operation.

3.1 Performance analysis of Sort

In the case of Sort, after the system integrates the Load Balancer, the execution speed of the job is 1.04 to 1.13 times than that of the speculative execution algorithm Hadoop-SP. In the case of RII, after the integration of the Load Balancer, in the worst case (δ=1), the job performance is improved by 5% compared to Hadoop-SP, and in the best case (δ=25), the performance is improved by 11%. When CB is operated, in the best case (δ=10), the Load Balancer shortens the job execution time by 23% compared with Hadoop-SP, and in the worst case (δ=1), it shortens the job execution time by 8%.In the use case of KM, after the integration of Load Balancer, the execution speed of the job is 1.19 to 1.28 times that of Hadoop-SP. The experimental results show that the HSLB algorithm has better performance. When the value of the virtual partition adjustment factor δ is between 5 and 30, the performance of the Load Balancer is stable.

As shown in Fig.3, when δ=1, the number of virtual partitions is the same with the number of partitions obtained by the hash-based partitioning method, but when the hash-based Hadoop-SP partitioning method is used, the input load size obtained by each Reducer is random, which means the Reducer with the weakest computional capability will have the chance to obtain the input data partition with the largest granularity. However, the HSLB algorithm considers the heterogeneity of the node where each Reducer is located and distributes the load based on the Reducer's computational capability. Therefore, it can prevent the Reducer with the weakest computational capability from receiving the input data partition with the largest granularity, reduce the differences in the execution progress of the Reduce tasks and lower the probability of the implementation of speculative tasks. As can be seen from Fig.3, when a virtual partition does not contain any skewed partition (such as: CB, SORT and RII), with the increase in the value of δ, the HLSB algorithm ensures that the Reducer input load is distributed more equitably and that the execution efficiency is further improved. For KM, the Load Balancer also greatly improves the job performance, because the output data in the Map task contain skewed partitions, which Hadoop-SP cannot identify, but it still partitions the output data of Map task based on the hash method. However, the Load Balancer proposed in this paper can identify the skewed partitions, and distribute the skewed partition to the Reducer with the strongest computational capability so as to shorten the job execution time. In addition, after mergering partitions, the Load Balancer releases some of Reducer's computation resources so as to improve the utilization rate of system resources. However, if the virtual partition adjustment factor is too large, it will take a larger amount of time to acquire partition meta information; therefore, for the sake of generality, δ is set to be 15 in the following experiment in this paper.

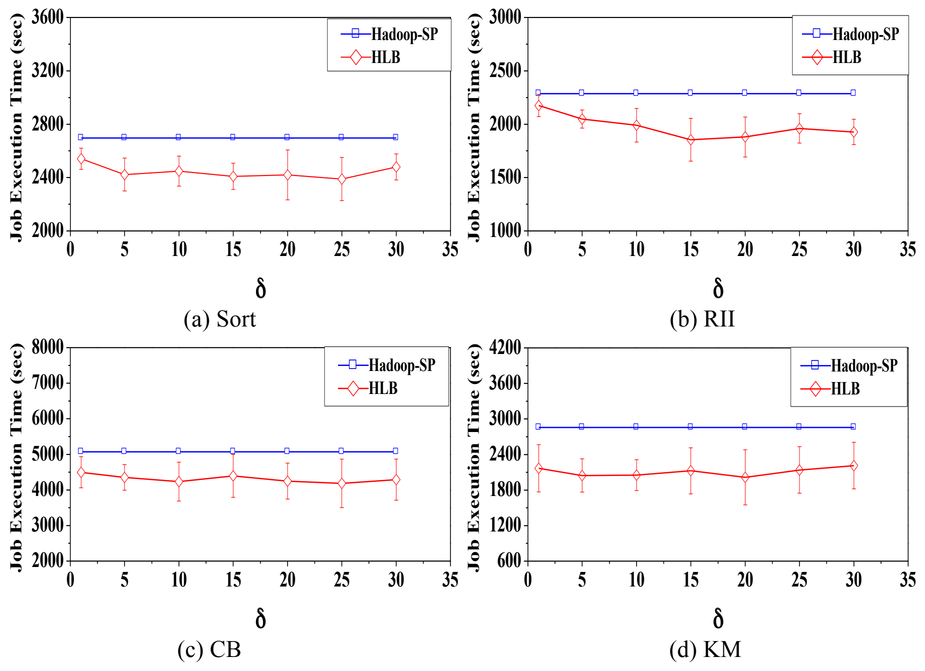


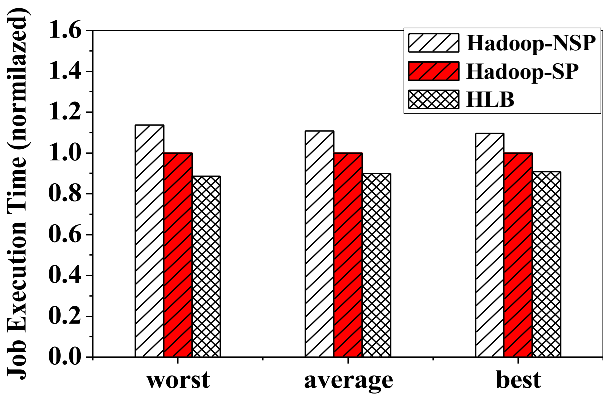
Fig. 3. Job performances under different partition adjustment factor δ

3.2 Load Balancer performance analysis

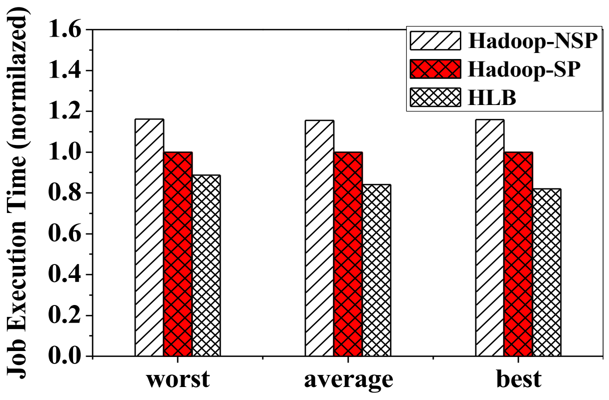
Effectiveness of the Load Balancer when there is no skewed partition. The virtual partition adjustment factor δ is set to be 15. Fig.4 shows the comparison of the experimental results under three methods (HLB, Hadoop-SP and Hadoop-NSP).

As shown in Fig.4, for Sort, the performance of HLB is improved by 15% on average compared with that of Hadoop-SP and by 28.5% compared with that of Hadoop-NSP. For RII, the job time of HLB is 9.5% shorter than that of Hadoop-SP on average and 19% shorter than that of Hadoop-NSP. For CB, compared with those under Hadoop-SP and Hadoop-NSP, the job performance under HLB is improved by 10.2% and 15.5% on average.

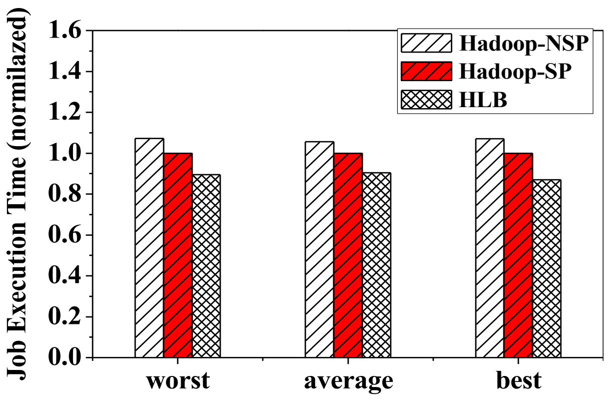
Unlike in the cases of RII and CB, the input data in the Sort case are randomly generated by the Random Writer program, so that the output key pairs of the Map are evenly distributed. In this case, for Hadoop-SP and Hadoop-NSP, the input load is evenly distributed among the various Reducers due to the use of the hash partitioning method. However, due to the different computational capabilities, the completion time of Reducers is different. HLB considers the different computational capabilities of Reducers and distributes different sizes of input load to Reducers so that the node with the weakest computational capability obtains the least input load, thereby improving the task performance in the entire Reduce stage.



(a) Sort



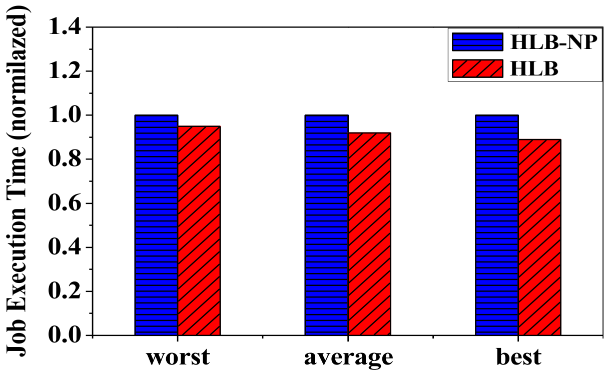
(b) RII



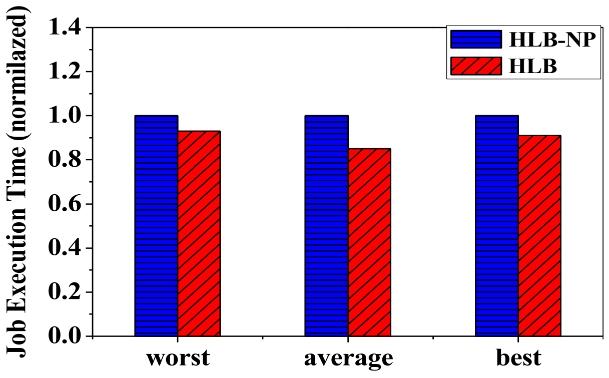
(c) CB

Fig. 4. Comparison of job performances when there is no skewed partition

Effects of the performance prediction model on the balancing results. In order to verify the effects of the Reducer performance prediction model on the balancing results, for the sake of generality, we use the KM case which contains the skewed partitions and the Sort case which does not contain any skewed partition as the experiment subjects. In order to verify the effects of different performance prediction methods on the balancing results, we compare the performance prediction model based on LS-SVM adopted in the HLB and the method based on linear model (HLB-NP) used in the LATE scheduler. The experimental results are shown in Fig.5. As can be seen from this figure, the LS-SVM-based performance prediction model can better improve the performance of the system. In the use case of Sort, the HLB can improve the job performance by up to 8.5% on average and in the KM use case; the job performance is improved by 15% on average. This is because compared with the performance prediction method based on the linear performance model; the performance prediction model proposed in this paper can more accurately predict the performance of Reducers.



(a) Sort



(b) RII

Fig. 5. Effects of the performance prediction model on the balancing results

4 Conclusions

Regarding the data skew problem in the heterogeneous MapReduce cluster in the Reduce phase, this paper presents a MapReduce job performance optimization method based on load balancing, which consists of two parts: the virtual partitioning method based on the consistent hashing and the HLB algorithm. The former uses the consistent hashing method to divide virtual partitions of the Map output data to reduce the granularity of each virtual partition. The latter, based on the job log information, evaluates the processing capability of Reducers in the cluster and detects the skewed partitions in the virtual partitions. By solving the set partitioning optimization problem, the algorithm mergers or distributes virtual partitions to balance the input load of Reducers. The experimental results show that the proposed method can divide the Map output data into fine-grained partitions, effectively detect the skewed partitions to eliminate the data skew in the Reduce stage, and distribute the load equitably according to the Reducers' heterogeneity. Under the circumstance where there is no skewed partition, compared with the existing algorithm, the proposed algorithm can improve the execution performance by 28.5%; in the case where there are skewed partitions, the job execution performance is improved by 22.8%. This means, when there are skewed partitions, this method can effectively improve the system resource utilization.

5 Acknowledgment

Projects (No. 2016GGJS-158) Supported by Cultivation Young Key Teachers in Colleges and Universities of Henan; Projects (No.142102210063 and 162102410051) Supported by the Henan Province Department Science Natural and Technology; Projects (No.17A520044) Supported by the Henan Province Education Department Natural Science Foundation;

6 References

[1] Sun, Z.Y, Zhang, Y.S., Nie, Y.L.,Wei, W., Jaime, L., Song, H.B. (2017). CASMOC: A novel complex alliance strategy with multi-objective optimization of coverage in wireless sensor networks. Wireless Networks, 23:1201-1222. https://doi.org/10.1007/s11276-016-1213-3

[2] Rao, R. V., Savsani, V.J., Vakharia, D.P. (2012). Teaching-learning-based optimization: An optimization method for continuous non-linear large scale problems. Information Sciences, 183:1-15. https://doi.org/10.1016/j.ins.2011.08.006

[3] Xu, B.M., Zhao, C.Y., Hu, E.Z., Hu, B. (2011). Job scheduling algorithm based on Berger model in cloud environment. Advances in Engineering Software, 42:419-425. https://doi.org/10.1016/j.advengsoft.2011.03.007

[4] Hadi, M. (2012). Adapting a heuristic oriented methodology for achieving minimum number of late jobs with identical processing machines. Research Journal of Applied Sciences, Engineering and Technology, 4:245-248.

[5] Sun, Z.Y., Shu, Y.X., Xing, X.F., Wei, W., Song, H.B. (2016). LPOCS: A novel linear programming optimization coverage scheme in wireless sensor networks. Ad-Hoc & Sensor Wireless Networks, 33:173-197.

[6] Kim, S.S., Byeon, J.H., Yu, H., Liu H.B. (2014). Biogeography-based optimization for optimal job scheduling in cloud computing. Applied Mathematics and Computation, 247:266-280. https://doi.org/10.1016/j.amc.2014.09.008

[7] Torkestani, J.A. (2012). A new approach to the job scheduling problem in computation grids. Cluster Computing, 15:201-210. https://doi.org/10.1007/s10586-011-0192-5

[8] Kim, S.S., Byenon, J.H., Liu, H., Abraham, A., Mcloone, S. (2013). Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization. Soft Computing, 17:867-882. https://doi.org/10.1007/s00500-012-0957-7

[9] Niu, S.H., Ong, S.K., Nee, A.Y.C. (2012). An improved intelligent water drops algorithm for achieving optimal job-shop scheduling soluting. International Journal of Production Research, 50:1-14. https://doi.org/10.1080/00207543.2011.600346

[10] Sun, Z.Y., Li, C.F., Xing, X.F., Wang, H.H., Yan, B., Li, X.L. (2016). k-degree coverage algorithm based on optimization nodes deployment in wireless sensor networks. Journal of Sensor, 16:1-16.

[11] Ebadi, A., Moslehi, G. (2013). An optimal method for the pre-emptive job shop scheduling problem. Computers & Operations Research, 40:1314-1327. https://doi.org/10.1016/j.cor.2012.12.004

[12] Yang, H.A., Lv, Y., Xia, C., Sun, S., Wang, H. (2014). Optimal computing budget allocation for ordinal optimization in solving stochastic job shop scheduling problems. Mathematical Problems in Engineering, 6:1-10.

[13] Vaddelle, J.L., Surekha, Y., Lakshmi, D.S. (2010). Adaptation of optimal methods in resource utilization, job scheduling in grids. International Journal on Computer Science & Engineering, 2:70-75.

[14] Quezada-pina, A., Tchernykh, A., Gonzalez-garcia, J.L., Hirales-carbajal, A., Ramirez-alcaraz, J.M. (2012). Adaptive parallel job scheduling with resource admissible allocation on two-level hierarchical grids. Future Generation Computer Systems, 28:965-976. https://doi.org/10.1016/j.future.2012.02.004

[15] Moon, Y.H., Youn,C.H. (2015). Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks. Computer Networks, 82:81-95. https://doi.org/10.1016/j.comnet.2015.02.030

7 Authors

Sun Zeyu is an associate professor in School of Computer and Information Engineering, Luoyang Institute of Science and Technology, Luoyang, China. His research interests include wireless sensor networks, internet of things, cloud computing. (E-mail:lylgszy@163.com)

Zhao Guozeng (Corresponding author) is a lecture in School of Computer and Information Engineering, Luoyang Institute of Science and Technology, Luoyang, China. His research interests include wireless sensor networks, cloud computing. (E-mail:ly_zgz@163.com)

Li Meng is a professor in School of Computer and Information Engineering, Luoyang Institute of Science and Technology, Luoyang, China. His research interests include wireless sensor networks, internet of things, cloud computing. (E-mail: 81634359@qq.com)

Lv Zhiguo received the B.S. degree in applied electronic technology from Henan Normal University, Xinxiang, China, in 2000, the M.S. degree in communications engineering from Guilin University of Electronic Technology, Guilin, China, in 2008. Since 2008, he has been on the faculty of Luoyang Institute of Science and Technology. He is currently working toward the Ph.D. degree at Xidian University, His research interests include networks communication, internet of things. (E-mail: 709009460@qq.com)

Article submitted 25 October 2017. Published as resubmitted by the authors 28 November 2017.