Acta Technica *62* No. 2A/2017, 445–454

A priority based dynamic bandwidth scheduling in SDN networks[1]

Zun Wang[2]

Abstract. In order to solve the problems of effective storage and the large scale data processing, the data centers running large-scale distributed computing are being constructed all over the world. Hadoop, a distributed computing framework that serves as the core infrastructure of data centers, is proposed. For reducing the time cost brought about by the job implementation in the data migration process, the job scheduling algorithm that plays an important role in the Hadoop assigns the task to task required node data for the implementation as much as possible, so as to shorten the operation response time and improve the performance of the cluster. Through the use of SDN (Software Defined Network) to control the network flexibility, the network control is conducted for the data transfer, so as to avoid the influence of task response time caused by network load change. The experimental results showed that the BS-IDS algorithm can adapt to the dynamic load change of data center effectively, and it has better performance in job response time compared with the traditional delay scheduling algorithms.

Key words. Software defined network, scheduling algorithm, Hadoop.

1. Introduction

In recent years, the Internet has brought great changes to people's daily life, and a large number of applications are being generated every day in the Internet, which has led to the rapid development of Internet [1]. As the most widely used cloud computing and large data processing platform, Hadoop has obtained the world's attention. But in many ways, it still exist the space to improve and enhance the performance, and one crucial issue is the job scheduling [2]. As one of the core technologies of Hadoop, Hadoop job scheduling algorithm is mainly responsible for decision-making for scheduling which job and a suitable task in the job to free computing nodes so that the scheduling is reasonable and the cluster is optimal. However, for the continuously improved job scheduling algorithm, when faced with increasing data and a variety of application scenarios, there are still some shortcomings. As one

[1]The author acknowledges the National Natural Science Foundation of China (Grant No. 51578109) and the National Natural Science Foundation of China (Grant No. 51121005).

[2]Beijing University of Posts and Telecommunications, 100876, China

of the most widely used Hadoop scheduling algorithms, the basis for FIFO (first in first out) to choose the job is based on the time sequence of operations arrived. Its advantages are simple and easy to implement, and the disadvantage is that it treats all the jobs the same, and ignores the urgency of the operation, which may result in that small operations are in the waiting state for a long time that they cannot be scheduled for implementation. In the data center, because the network bandwidth is a scarce resource, under the premise of ensuring the fairness of the operation, in order to improve the probability of task implementation on the local node, researchers at the University of California at Berkeley, based on Fair Scheduler, studied the corresponding delay scheduling algorithm. However, the algorithm only uses a static value as the waiting time threshold for the team's first operation, so it cannot effectively adapt to the dynamic load changes and network changes of the cluster. On this basis, a dynamic waiting time threshold based on network bandwidth and data center load variation is set as the waiting time of the team's first operation [3]. Nevertheless, these algorithms do not take into account the completion time of first team job from a global perspective, ignoring the effects of network load change on the data transfer. As a result, the performance of the algorithms greatly reduced to a certain extent. Therefore, it has important significance to make a detailed analysis of various scheduling algorithms at present, to sum up the corresponding advantages and disadvantages, and through a combination of new technology, and to correct the corresponding defects, essential for reducing the job response time of Hadoop systems and improving the Hadoop cluster performance and so on.

2. Improved delay scheduling algorithm based on SDN

2.1. BS-IDS algorithm

The jobs that users submitted are arranged into multiple job queues, and each queue has a team first job. It is assumed that the selected jobs for scheduling have the remaining m tasks that are not performed, representing $T_1, T_2, ..., T_m$, respectively, then the size of the data block that the task corresponds to is Δ. The data center has n nodes, and they are denoted as $Node_1, Node_2, ..., Node_n$, respectively. The SDN controller periodically transfers the data link bandwidth of the data center to the Master nodes of the Hadoop system. At some points, a free node $Node_{\text{free}}$ requests a task assignment to the master node, so as to ensure the data locality in the task implementation and reduce the response time of the task. The strategy that the BS-IDS scheduling algorithm uses is: if the free node is not the local node of the team first work, the waiting time threshold $t_{\text{threshold}}$ in the team first job is calculated by using the waiting time threshold algorithm, and compared with the waited time t_{wait} respectively. The quantities $t_{\text{threshold}}$ and t_{wait} represent the calculated team's first job waiting time threshold and the waited time, respectively. If the waiting scheduling time of the team's first job does not exceed the waiting time threshold for the team first job, other jobs are scheduled firstly. The real-time available bandwidths of each uncompleted task required data blocks storage node to the free node in the team's first job is achieved by SDN, which are $B_1, B_2, ..., B_m$,

respectively [4]. Symbol B_1 represents the real-time bandwidth available to the free node $Node_{\text{free}}$ from the node stored by the data blocks required by the task T_1. Assuming that each teak in the team first job is scheduled to the current free node work $Node_{\text{free}}$, the data block transfer time overheads caused by the implementation of the task of non local node are $\frac{\Delta}{B_1}, ..., \frac{\Delta}{B_m}$, respectively. From these data block transfer time overhead, the task T_i that a minimum time overhead corresponds to is chosen, and it immediately schedules the task T_i. The calculation process is as follows:

$$T_i = \min\left(\frac{\Delta}{B_j}\right),\ 1 \leq j \leq m\,. \tag{1}$$

In this case, since the team's first task is scheduled to execute on a non local node, it is necessary to move the data block from the node where the task needs data storage to the computing node. As we all know, network resource is a scarce resource of data center. In order to avoid the effects of network load change on the data blocks transfer, we use SDN bandwidth control capability, to design a bandwidth allocation mechanism based on time section. And it assigns the bandwidth for the data blocks transfer, so as to ensure the efficient allocation of tasks, and improve the performance of BS-IDS scheduling algorithm.

2.2. Bandwidth allocation mechanism based on time section

The delay scheduling algorithm is based on the real-time available bandwidth between the current time nodes to calculate the waiting time threshold. And the network bandwidth resources of data center are the scarce resources, so between the various applications, they will compete for the network bandwidth. In order to ensure the performance of the delay scheduling algorithm, we introduce a time based bandwidth allocation scheme. The main idea is as follows: at the present time $T = t_0$, when there is a free node $Node_{\text{free}}$ requesting for task allocation, if the waited time of the team first job exceeds the waiting time threshold, the task T_i that a minimum data block migration time overhead corresponds to is selected to schedule to the node $Node_{\text{free}}$ for the implementation.

When a task T_i is assigned to a free node $Node_{\text{free}}$ for the implementation, it is assumed that the link Link_i is scheduled by the task T_i for implementation on a nonocal node, the data block that the task T_i corresponds to moves the path has passed over [5]. The Hadoop scheduler calls the interface of the SDN controller, leaving the link Link_i reserved for the task T_i in the time period of $\left(t_0, t_0 + \frac{\Delta}{B_i}\right)$, and the reserved bandwidth size is B_i.

When the data required by the task T_i has completed the migration from the storage node to the free node $Node_{\text{free}}$, the OpenFlow controller will terminate the occupying of the link Link_i.

With the help of bandwidth allocation mechanism based on time section, we can maximize the use of network bandwidth capability, to reduce the impacts of network load change on data transfer, which efficiently makes task scheduling and improves the performance of BS-IDS algorithm.

3. Method

As shown in Fig. 1, the BS-IDS scheduler is mainly divided into task scheduling module, job initialization module, job queue management module, team first job information pool module and waiting time threshold calculation module. Among them, the BS-IDS scheduler inherits and implements the team first modules interfaces of the Hadoop system. In addition, the BS-IDS scheduler designed the first operation information pool and the waiting time threshold calculation module based on the BS-IDS scheduling algorithm.

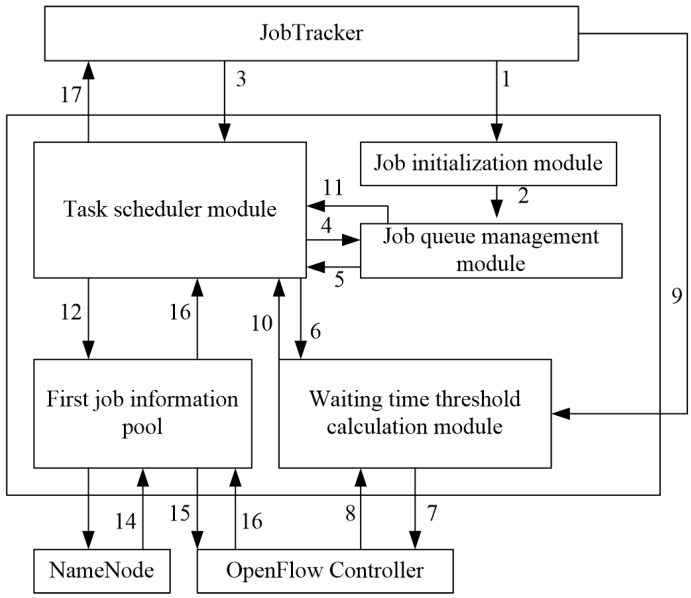

Fig. 1. BS-IDS scheduler module and scheduling flow chart

BS-IDS scheduler flow is shown in Figure 1, and its major process can be divided into the following steps:

Submission and initialization of the job, as shown in step 1 in the figure, when the user submits the job from the client, the BS-IDS scheduler first of all uses the JobTracker.initJob function to initialize it. As shown in step 2 in Fig. 1, a Map task is created for each data block based on the partitioning of the data blocks of the job file. The queue job management module adds successful initialization jobs to the appropriate queue and waits for the implementation.

Request and processing of tasks, as shown in step 3, 4, and 5 in the figure, when the data center has free nodes requesting the task allocation to JobTracker, JobTracker will call the job management and scheduling module to select a job to be scheduled. Secondly, as shown in step 6, 7, 8, 9, and 10, the task scheduling module will give a request to task scheduling module in the BS-IDS scheduler and the waiting time threshold calculation module. And then the calculated team's first job waiting time threshold is given feedback to the task scheduling module. The

task scheduling module team's first job is compared with the waited time and the waiting time threshold. As shown in step 11, if the waited time is less than the waiting time threshold, it first of all schedules other operations; otherwise, perform the following operations. Since the waited time of team's first job exceeds the waiting time threshold, a task in the team's first job needs to be allocated to the current free node. At this point, as shown in steps 12, 13, 14, and 15 in the figure, the task scheduler module sends the task assignment request to the team's job information pool [6]. The team's first job information pool, from the NameNode and OpenFlow controller in the HDFS system, obtained the data blocks backup information of team's first job and network load information of data center. And through calculation, the task is assigned to free nodes to perform, and the task that the data block transfer time overhead between the required block data storage node and current free node corresponds to.

As shown in step 16, the first team job information pool will deliver the task information for the task scheduling module, and allocate the task to the free node. As shown in step 17, the data block that the task corresponds to would be moved to free nodes from the storage nodes. And the MAC address of storage node and free node of the task required block and the transfer time section is transferred to the OpenFlow controller. Then the OpenFlow controller will reserve the data transfer link in the migration period for the task. As shown in step 16 in Fig. 1, the task scheduler module assigns the task to the current free node for the implementation.

4. Results and discussion

The experimental environment is built based on the Hadoop framework of SDN, and its effectiveness is verified. The network environment it builds is shown in Fig. 2, mainly including 6 nodes, two OpenFlow switches, and a router. The 4 nodes are taken as Slave, which is responsible for storing data and performing tasks, and the other two nodes are Master and OpenFlow controllers, respectively.

Relevant allocation parameters of Hadoop system are shown in Table 1.

Table 1. Hadoop system parameters allocation table

Link bandwidth	100 Mbps–1000 Mbps
Data block size	64 M
Data block backup	2

In order to analyze the influence of different scale data on BS-IDS scheduling algorithm, this experiment, according to the number of tasks, divided the jobs into five groups, as shown in Table 2. This experiment used the Wordcount and Sort system of Hadoop system as the test operations. The Hadoop default scheduler FIFO, the delaying scheduler, and the BS-IDS scheduler are tested, respectively, and the test results are analyzed.

Local probability of job data:

As shown in Fig. 3, in the 100 Mbps network environment, the network transmis-

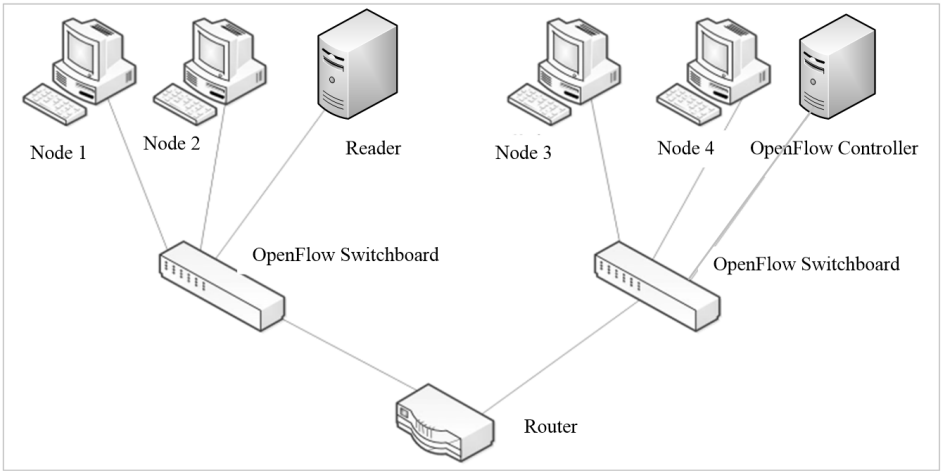

Fig. 2. Hadoop cluster topology based on SDN

sion overhead is great. And the BS-IDS scheduler adjusts the waiting time threshold so that the task is scheduled to run on the local node as far as possible. The rate of data localization is 100 %. In such networks, if the delay scheduling algorithm is adopted, because its setting is a static waiting time threshold, a small amount of data needs to be transmitted through the network. The data localization ratio of group 1, group 2, group 3 and group 4 is 95 %, 99 %, 94 %, and 93.5 %, respectively. For FIFO without adopting delay scheduling, part of the data needs to be transmitted over the network, increasing the network overhead of the data center.

Table 2. Job parameter configuration table

No.	The number of task	The number of job	The size of job
1	1	40	64 MB
2	4	10	256 MB
3	10	4	640 MB
4	20	2	1280 MB

As shown in Fig. 3, in the 1000 Mbps environment, data transmission in the network takes less time. Using the BS-IDS scheduler shortens the waiting time threshold and avoids invalid waiting time overhead. The proportion of the local data is relatively low. The delay scheduling algorithm is set to a fixed waiting time threshold, and there is only a small amount of data reaching the node implementing tasks through the data center network, maintained a higher proportion of local data. Because the BS-IDS scheduler waiting time threshold varies with data center load, when a job is submitted, in the team's first operations, there are a lot of tasks to be implemented [7]. The reaching intensity of the data center is great, thus it needs to set a larger waiting time threshold to schedule the team's first task on the local node, and then it can guarantee a certain number of tasks performed in the local

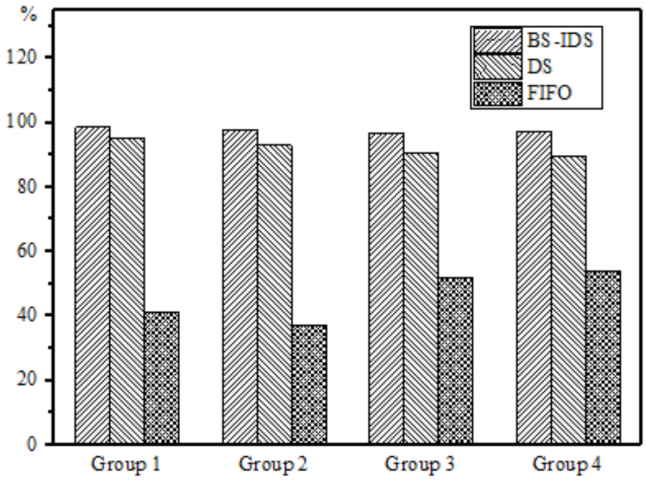

Fig. 3. Comparison of data local probability in the 100 Mbps environment

node. With the implementation of the task, the waiting time threshold is set to be close to 0, which is higher than FIFO.

Job response time comparison:

As shown in Fig. 4, in the 100 Mbps environment, the BS-IDS scheduler schedules the tasks to the nodes that the data is needed for the task, reducing the transmission time overhead of the data in the network. Moreover, during the data transmission time, the SDN bandwidth management and control ability is used to assign the bandwidth for the task and ensure the efficient allocation of tasks, which further reduces the response time of the job. Operation scale is increasing continuously, while the probability of local data by using delay scheduling is falling, which leads to the migration time overhead of data in the network increases, and a corresponding increase appears in job response time. As a result, the performance gap by BS-IDS is growing. While FIFO did not consider local implementation of tasks, it led to a large amount of data network transmission, and its performance was about 40 % of BS-IDS. Taking group 4 as an example, the completion time of the BS-IDS algorithm is reduced by 21 s compared to that of the delayed scheduling, and compared to FIFO, it is reduced by 150 s [8].

As shown in Fig. 4, in the 1000 Mbps network environment, the waiting time threshold of the BS-IDS scheduling algorithm is smaller because of the small network transmission overhead. At this point, the BS-IDS scheduling algorithm is slightly better than the FIFO, and the delay scheduling algorithm results in an extended response time due to the excessive waiting time overhead. Taking group 4 as an example, the job completion time of BS-IDS scheduling algorithm is 10 s less than FIFO, and it is reduced 130 s than that of delay scheduling.

BS-IDS scheduling algorithm's team's first job waiting time threshold variation rule:

In order to compare the effect of network bandwidth on the waiting time thresh-

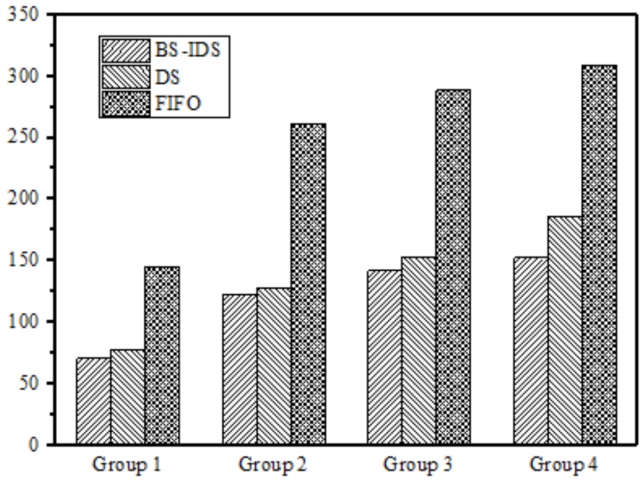

Fig. 4. Comparison of job response time in the 100 Mbps environment

old, this test compares and analyzes the operations with the number of group 1 under the 100 Mbps and 1000 Mbps networks. As shown in Fig. 5, it can be seen that, in the 1000 Mbps environment, the waiting time threshold of BS-IDS scheduling algorithm is smaller than the waiting time threshold of BS-IDS scheduling algorithm under the 100 Mbps network environment. This is because, in high bandwidth environments, BS-IDS scheduling algorithm will assign more tasks to non local nodes, to avoid wasting too much waiting time.

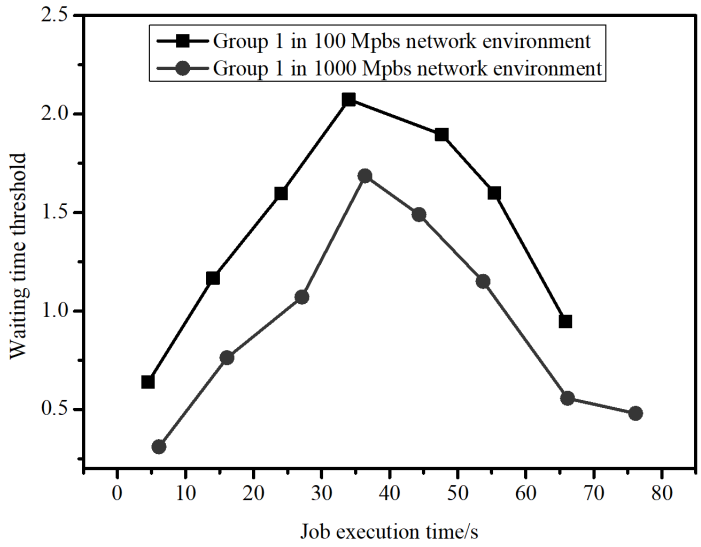

Fig. 5. Variation of waiting time threshold under different network environments

5. Conclusion

With the arrival of the big data era, the data centers operating large-scale distributed computing are built around the world. In the data center, nodes communicate with each other over the network. The limited bandwidth resources between nodes become the key factors that affect the performance of data centers. Using the bandwidth control capability provided by SDN, this paper proposes a scheme to improve the performance of large data processing by using link bandwidth. In the actual test environment, the Hadoop default scheduling algorithm FIFO, delaying scheduling algorithm, and BS-IDS scheduling algorithm are compared, and the efficiency of BS-IDS scheduling algorithm is proved.

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami: *Internet of Things (IoT): A vision, architectural elements, and future directions.* Future Generation Computer Systems *29* (2013), No. 7, 1645–1660.

[2] C. W. Tsai, W. C. Huang, M. H. Chiang, M. C. Chiang,C. S. Yang: *A hyper-heuristic scheduling algorithm for cloud.* IEEE Transactions on Cloud Computing *2* (2014), No. 2, 236–250.

[3] D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte, P. Leitão, J. Barbosa: *Benchmarking flexible job-shop scheduling and control systems.* Control Engineering Practice *21* (2013), No. 9, 1204–1225.

[4] H. Duan, C. Chen, G. Min, Y. Wu: *Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems.* Future Generation Computer Systems *74* (2017), 142–150.

[5] W. Wang, K. Zhu, L. Ying, J. Tan, L. Zhang: *Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality.* IEEE/ACM Transactions on Networking *24* (2016), No. 1, 190–203.

[6] K. Kaur, T. Dhand, N. Kumar, S. Zeadally: *Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers.* IEEE Wireless Communications *24* (2017), No. 3, 48–56.

[7] T. Vondra, J. Šedivý: *Cloud autoscaling simulation based on queueing network model.* Simulation Modelling Practice and Theory *70* (2017), 83–100.

[8] M. Kalra, S. Singh: *A review of metaheuristic scheduling techniques in cloud computing.* Egyptian Informatics Journal *16* (2015), No. 3, 275–295.

Received May 7, 2017