Advances in Intelligent Systems Research, volume 147

International Conference on Network, Communication, Computer Engineering (NCCE 2018)

An Improved HBase Load Balancing Algorithm

Wei Xin [a)]

Beijing University of Technology, Beijing 100124, China.

[a)] Corresponding author: 308007367@qq.com

Abstract. This article briefly introduced the origin and development of distributed database, HBase in detail the load balancing problem and the original load balancing algorithm. The original load balancing algorithm considers only the space usage load of each node in the cluster, does not consider the hot spot access load problem of each node in the cluster, and makes the node that has passed the original algorithm load balancing remain in file access time Big difference. The improved algorithm proposed in this paper, taking into account the load on the space utilization, taking into account the file access frequency factors. The experimental results show that the improved algorithm proposed in this paper can balance the space utilization load and hot spot access load of each node in the cluster at the same time, so that the load balancing problem in HBase can be better solved.

Key words: HBase; load balancing; Hot spot load; distributed database.

INTRODUCTION

With the rapid development of network technology, especially the development of information technology with Internet as the core, information technology such as communication technology and computer technology has become increasingly important in people's life, work and scientific research [1]. Transmission services provided by the Internet started from the original single data services to the integrated business direction, audio and video and other multimedia services are growing. At the same time, many complex projects use large amounts of computer data to store and read, and such a large amount of data is impossible on a single computer and within a specified time frame [2]. In order to solve these problems and meet the development needs, a distributed database is proposed.

Distributed data is the traditional physical decentralized database into a number of data storage unit, stored in a number of physically dispersed database, and use the network to connect these scattered data storage nodes to form a logical database. Compared with the traditional database, distributed database has more storage capacity and higher concurrent traffic [3].

HBase is Apache's open source distributed database based on Google's paper BigTable, which uses the underlying HDFS file system to complement the Hadoop ecosystem and provides highly efficient random read and write capabilities [4].

With the continuous development of distributed technologies, load balancing in distributed databases becomes more and more important [5]. In a multi-node network environment, due to the randomness of task arrival and the difference in processing capabilities of each node[6], it is often the case that at some point, the access frequency of some nodes is high and the load of nodes is heavy , While other nodes have low access frequency, nodes are lightly loaded and even idle, and the load on each node is very unbalanced. In this way, some nodes in the system may not be able to respond quickly due to heavy load and may even cause node crashes. On the other hand, idleness of some nodes is also a waste of resources [7]. Therefore, certain load balancing measures need to be adopted to adjust the load of each node in the cluster to achieve an almost even distribution and to improve the performance of the distributed database cluster. Load balancing technology can consider in real time the current computing capacity of the node and node storage capacity and other information to try to assign new tasks to idle nodes, the strategy is assigned to the nodes in the system, trying to make the loads on all nodes roughly equal [8]. Distributed database

load balancing measures are mainly hardware and software approach, the hardware approach is to increase the number of nodes to achieve, but in reality, there can be no limit to increase the number of hardware, and excessive increase in the number of hardware will result in a waste of resources. Software way through an effective load balancing algorithm, so that all nodes in the cluster load all the balanced distribution, reduce the average task response time and improve the performance of the cluster [9].

This paper first introduces the original load balancing algorithm of distributed database HBase in detail, proposes the improved strategy and the improved algorithm, and verifies the improved algorithm through experiments. The experimental results show the effectiveness of the improved algorithm.

HBASE LOAD BALANCING ALGORITHM

The HBase original load balancing algorithm takes the region in each node as a unit and performs load balancing according to the current load of the cluster. HMaster in the cluster runs a specific thread to periodically check the cluster to determine if the cluster needs to be load balanced. Load balancer balancer will be based on the number of regions on each region server node to determine whether the need for load balancing, in particular, is to calculate the average number of regions on all region server, and then balancer configuration item hbase.regions.slop A threshold interval.

$$Floor = average \times (1 - slop) \tag{1}$$

$$Ceil = average \times (1 + slop) \tag{2}$$

Floor and Ceil in (1) and (2) are the lower and upper limits of the interval respectively. If the number of regions in a node is within (Floor, Ceil), it indicates that the node is not overloaded or hungry, that is, no load balancing is performed. Otherwise, it will join the load balancing queue. After the filter is completed, the balancer calculates the cost value that reflects the load status of each node in the current cluster. The calculation of the cost value takes into account three items: the region migration cost, the region localization policy, and the region count distribution. This operation is iteratively calculated multiple times. Each iteration, will randomly choose a pick region strategy, a total of three, namely, Random Region Picker, Load Picker and Locality Based Picker. Balancer randomly selects a picker policy and executes it once according to the picker policy and calculates the cost of the cluster. If the calculated cost is smaller than the original, it indicates that the exchange or migration of this region is valid. Each iteration is based on a new cluster, making a total of computed MaxSteps. After the iteration is completed, the load migration strategy plan table is obtained. In the table, each plan represents a region of a certain two nodes for region exchange, or a region in one node migrates to another node, and the region exchange and the region are generated. The migration operation will call the corresponding function to execute. Finally, when all plan execution is complete, a complete load balancing operation is completed.

IMPROVED HBASE LOAD BALANCING ALGORITHM

Problem Description

For the load problem caused by accessing hotspots in HBase, other improved indicators are introduced in the current improved algorithms, such as disk I/O access rate, CPU utilization and memory utilization, and file access times. And use these data to calculate the final load evaluation index according to the proportion coefficient developed by each one to carry out the load balancing of each node. However, there are still defects in these load balancing solutions: (1) The load evaluation method is too restrictive. That is, the load obtained by adding a certain proportion of coefficients can only reflect the load under certain conditions. This is obviously not enough. (2) The current load evaluation method can only obtain the load of each node in the distributed database and cannot obtain the load status of each region in each node. The load balancing strategy finally balances the load through the region migration. If the region's load cannot be obtained, corresponding load migration strategies cannot be used to balance

the load of each node, and a migration strategy cannot be derived, that is, which region among which nodes to migrate to balance the load.

Improve Algorithm

The improved algorithm is based on the original algorithm, that is, through the number of regions to solve the space load balancing strategy plan based on the next round of new cost value iteration. This iteration will consider Requests per second distribution, region mobile cost and region localization strategy three. The region count distribution is no longer considered because plan has solved the problem of space load, the cost of moving region and the strategy of region localization continue. Requests per second reflects the number of requests in the region per unit time, which reflects the hot spot visits in the region. Specifically, calculate the average number of requests on all region server, and then according to balancer configuration item requests. Slop draw a threshold interval.

$$Floor = request.average \times (1 - request.slop) \tag{3}$$

$$Ceil = request.average \times (1 + request.slop) \tag{4}$$

Floor and Ceil in (3) and (4), respectively, are the lower and upper bounds of the filter interval. If requests are not in this interval, then load balancing is required. The new pick region strategy retains only the region switching strategy, that is, randomly exchanges two regions in two region servers to calculate the new cost value. If the calculated cost value is smaller than the original value, then the exchange of this region or Migration is effective. Each iteration is based on a new cluster and a total of computed MaxSteps times. Finally, get the final load migration strategy plan table.

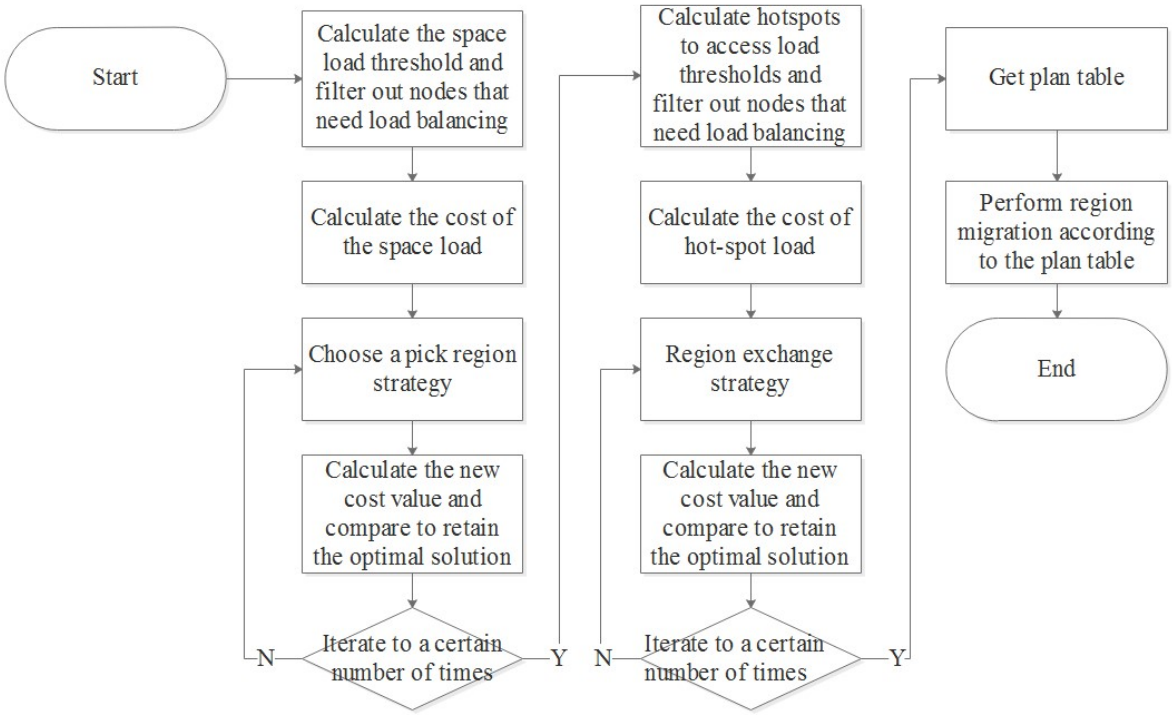

FIGURE 1. Improve algorithm flow chart

EXPERIMENT

Experimental test environment vmware created. Use vmware to create four virtual machine nodes, one master node, and three slave nodes. Table 1 shows the configuration information of each node.

TABLE 1. Node configuration information

Node	System	Hbase	JDK	RAM	Size
master	Ubantu16.04.3	Hbase1.2.6	Jdk-8u11-linux	1G	20G
Slave1	Ubantu16.04.3	Hbase1.2.6	Jdk-8u11-linux	1G	20G
Slave2	Ubantu16.04.3	Hbase1.2.6	Jdk-8u11-linux	1G	20G
Slave3	Ubantu16.04.3	Hbase1.2.6	Jdk-8u11-linux	1G	20G

Set up the environment, run start-hbase.sh, start Hbase. Hbasse start, the first shell statement to write data in each node, and then use the shell statement to read the data. Record data before and after load balancing.

The experimental data of the original algorithm is as follows:

TABLE 2. The original algorithm before and after load balancing the number of region in each node

Stage	Slave1	Slave2	Slave3
Before load balancing	24	3	5
After load balancing	11	10	11

TABLE 3. The number of requests per second for each node before and after the original algorithm load balancing

Stage	Slave1	Slave2	Slave3
Before load balancing	559.9	0	0
After load balancing	337.3	101.5	138.7

Through the experimental data, we can see that the number of regions in each node is effectively load-balanced by the original algorithm, but the requests per second in each node are not effectively load-balanced, and the results after many tests are also quite different.

The experimental data of the improved algorithm is as follows:

TABLE 4. The improved algorithm before and after load balancing the number of region in each node

Stage	Slave1	Slave2	Slave3
Before load balancing	24	3	5
After load balancing	11	11	10

TABLE 5. The number of requests per second for each node before and after the improved algorithm load balancing

Stage	Slave1	Slave2	Slave3
Before load balancing	573.6	0	0
After load balancing	197.1	178.6	199.0

Through the experimental data, we can see that the number of regions in each node count and requests per second through the improved algorithm for effective load balancing, load balance after the request per second number there are more than 20 differences in the number of because of each region visit the amount is different. The experimental data also prove that the improved algorithm proposed in this paper solves the problem of hotspot access load balancing while solving node space load balancing.

CONCLUSION

This paper introduces the original load balancing algorithm of distributed database HBase in detail, analyzes the problem that the original algorithm only considers the space utilization load, and proposes an improved algorithm for this problem. The improved algorithm presented in this paper can not only handle the load balancing of the space utilization load, but also can solve the load problem caused by the node's hotspot access, so that the load balancing problem in Hbase can be better resolved. Through experimental and experimental data, it also proves that the improved algorithm is effective in solving the problem of load balancing. The next step is to study the problem of load balancing under unbalanced conditions. The load evaluation criteria and plan generation of nodes under unbalanced conditions are relatively complex and will be difficult to study.

REFERENCES

1. Mcafee A, Brynjolfsson E. Big data: the management revolution. [J]. Harvard Business Review, 2012, 90(10):60-6, 68, 128.
2. Mehul, Nalin, Vora. Hadoop-HBase for large-scale data[C]// International Conference on Computer Science and Network Technology. IEEE, 2011:601-605.
3. Taylor R C. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. [J]. BMC Bioinformatics, 2010, 11 Suppl 12(12):3395-3407.
4. Fan Y, Wu W, Cao H, et al. LBVP: A load balance algorithm based on Virtual Partition in Hadoop cluster[C]// IEEE Asia Pacific Cloud Computing Congress. IEEE, 2012:37-41.
5. Hsiao H C, Chung H Y, Shen H, et al. Load Rebalancing for Distributed File Systems in Clouds[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(5):951-962.
6. Konstantinou I, Tsoumakos D, Mytilinis I, et al. DBalancer: Distributed Load Balancing for NoSQL Data-stores[C]// Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013:1037-1040.
7. Xia L, Chen H, Sun H. An optimized load balance based on data popularity on HBASE[C]// International Conference on Information Technology and Electronic Commerce. IEEE, 2015:234-238.
8. Susila N, Chandramathi S, Kishore R. A Fuzzy-based Firefly Algorithm for Dynamic Load Balancing in Cloud Computing Environment[J]. Journal of Emerging Technologies in Web Intelligence, 2014, 6(4).
9. Chen G B. A Distributed Dynamic Load Balancing Scheduling Algorithm[J]. Journal of Guangxi Teachers Education University, 2014.