Research of Dynamic Load Balancing Strategy on HBase

An-ping Xiong & Jiao Zou

College of Computer Science & Technology, Chongqing University of Posts & Telecommunications, Chongqing 400065, Chinas

Keyword: HBase column type storage; Hotspot issues; Load balancing

Abstract. As an open source distributed column store database, HBase is widely used because of its low cost and high scalability. According to the problem that HBase can't spread requests to each server greatly be-cause of its load balancing strategy did not consider Region Server's dynamic load factor. This paper proposed a dynamic load balancing strategy based on Table, The strategy can disperse the request and the settlement of hotspot issues better. The experimental results show that the improvement load balancing strategy can improve the overall performance of the system.

1. INTRODUCTION

With the development of information technology, especially the rapid development of mobile communication and social networks, TB or even PB structured, semi-structured and unstructured data need to be deal. The traditional relational database is unable to meet the need of such a massive amount of data processing. In such a case, NOSQL comes into being, such as, MongoDB, CouchDB, Cassandra, Hypertable etc[4]. As a member of NOSQL, HBase is a distributed, column-oriented open source database. It is a high reliability, high-performance, column-oriented, scalable distributed storage system, using Hadoop HDFS as its file storage system, using Hadoop MapReduce to process the vast amounts of data, at the same time, using the Zookeeper as collaborative services[10]. HBase has been used in many large networks such as Facebook, Taobao, Twither. HBase processing framework uses a load balancing policy to improve the performance of parallel processing systems[1]. Because the single consideration of its own strategy, it can't good to do load balancing between the server. And efficient load balancing strategy, can effectively expand the server bandwidth, increase system throughput, and strengthen the system of data processing ability. This paper proposes a dynamic load balancing strategy based on table according to HBase own storage mechanism. The requests are evenly distributed to each server, improving the performance of HBase.

2. RELATED TECHNOLOGY OF HBASE

The HBase can run on ordinary commercial server, supporting data sets from the medium scale to millions of rows and billions of columns. It has been successfully applied to a typical Internet search, crawl incremental data, content services, information exchange and other fields[7].

2.1 Research status

The related research about the distributed load balancing technology has been carried out. Li Bin, Yingwei Zhang et al[7]analyze the characteristics of NoSQL database, such as supporting high concurrent read and write data, high efficiency storage and access mass data, and high scalability and high availability. And through researching the load balancing for distributed storage systems, they proposed to use of consistent hashing function to achieve load balancing system. Literature [2] proposed a distributed dynamic load balancing scheduling algorithm based on general load balancing scheduling algorithm. The algorithm is dynamic collection, statistics and analysis server information, to complete the server load balancing task, achieving clusters normally scheduling. On this basis, taking distributed scheduling strategy enhance the efficiency and performance of the load balancing scheduling to minimize its effect on the performance of the whole business system. Liyan Dong, Huang Le, Limin Wang et al[3] proposed an improved distributed system load balancing

strategy for the system load imbalance problem caused by heterogeneous of the server computing power and bandwidth. The fuzzy comprehensive evaluation theory is applied to the server performance evaluation. It helps to select the best performance of the server, making the load balance, and it improved the utilization rate of distributed resources. The literature [9] analyzed the existing data placement strategy and dynamic load balancing algorithm in distributed storage system. In view of the current distributed storage of data distribution policy scalability and flexibility, it proposed an improved hash scheme and a load balancing strategy which is based on node capacity of perception. The literature [10] designed and implemented a dynamic load balancing strategy based on DTS technology, combining with load balancing algorithm based on middleware. This strategy can uniformly distributed load to each server, and has the characteristics of strong portability, flexible expansion, to further improve the execution efficiency of distributed server.

2.2 HBase data model

HBase data table is a sparse, persistent, multi-dimensional, ordered mapping table. As shown the left table of Figure.1, the row of table has three basic types of definition: Row Key, Time Stamp, Column. The Row Key is the unique identifier of the row of the table, the Time Stamp is the corresponding timestamp of each data operation, can be seen as similar to the version information, the Column is defined as: <family>:<label>,through the two parts can only define a storage column of a data.[6]

Although in concept, the HBase data table can be represented the sparse table structure as shown in the left table of Figure 1. But in physics, HBase data is to be stored in accordance with the column family. Each column family values, combined with a key value and a corresponding timestamp stored in a separate file. Different column family data even belong to the same row will also be stored in different files. The right table of Figure 1 expressed the actual storage structure of the left table of Figure 1, the blank cells does not occupy any physical unit in real memory.

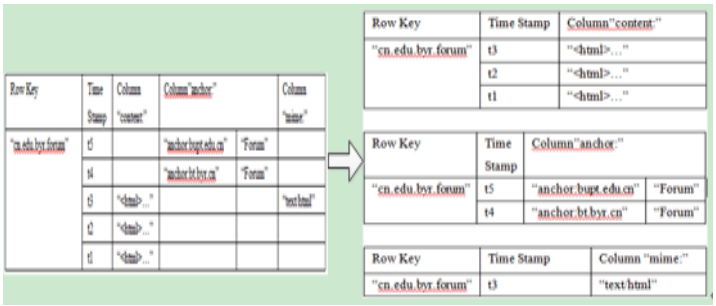

Figure1. HBase logical data model

2.3 HBase load balancing strategy

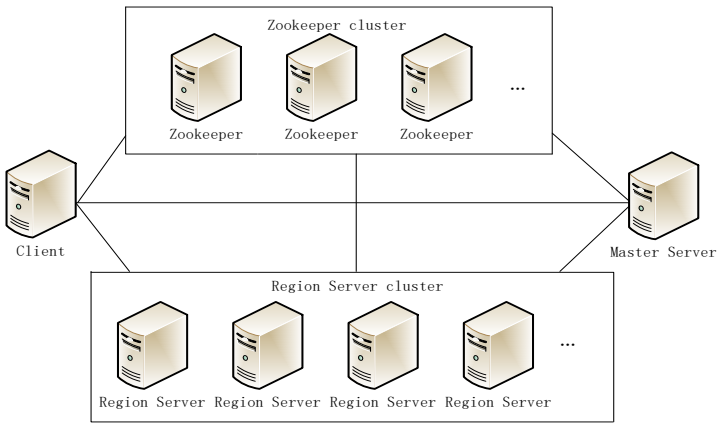

Figure2. HBase system architecture

HBase system architecture has a Client, Zookeeper, Master Server,Region Server components as shown in figure 2. In HBase, the Master Server is responsible for the management of the Region

Server's load balancing, adjust Region distribution. The Master opened a separate thread for the load balancing of the whole cluster. A cluster load balancing algorithm is mainly realized by the method of balanceCluster. The load balancing algorithm steps are shown in figure 2.

The basic idea of the method balance Cluster is computing the total number of region of each Region Server loading in the whole HBase cluster. These Region Servers are sorted according to the number of its region load. Calculating the average number of Region all the Region Server hosting, then the average takes ceil and floor operation that is calculating the maximum number and the minimum number of region each Region Server needs to load. The algorithm to achieve the final effect is that each Region Server loads the same number of the Region in the cluster.

3. IMPROVED HBASE LOAD BALANCING STRATEGY

The existing HBase load balancing strategy only take region's number into consideration when balance on each Region Server, existing two following problem: (1) HBase data is stored in the form of table. When a Table is very large, it will be split into multiple regions.[5] If this table's regions are allocated on the same Region Server, when this table is accessed frequently, reading efficiency will not be ideal. (2) Due to the arbitrariness of user access, it may be some region hot, causing some Region Server overload.

To solve above problems, this section propose a dynamic load balancing strategy based on the table.

3.1 Dynamic load balancing strategy based on the Table

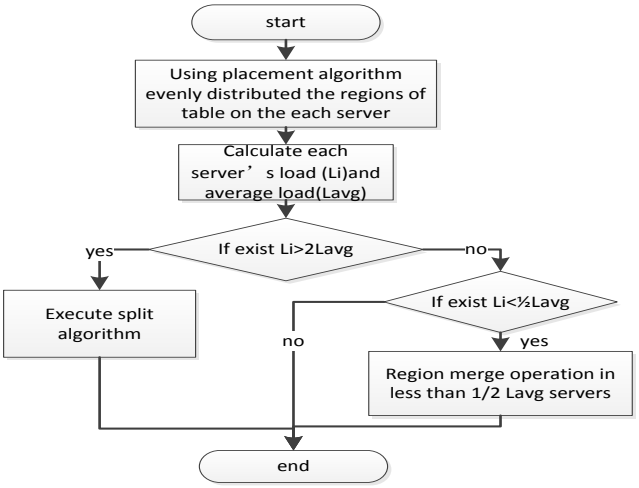

Figure3. the flow chart of dynamic load balancing strategy based on the table

On the basis of existing HBase load balancing strategy, Firstly, the table's region will be evenly distributed to each Region Server, and monitor the load of all Region Server. If there is heavy load of Region Server, split the hot region on the server, then assign sub-region to the load lighter Region Server. At the same time, in order to avoid too much too small region, execute region merge operation in a lightly load Region Server.

3.1.1Placement algorithm

As shown in Figure 4, because all rows in the HBase tables are sorted according to the Row Key dictionary. The table is divided into multiple regions in the row direction. A table which may be very large is divided into a number of regions. This paper will first evenly spread the table's regions to different servers. Acquiring data from multiple servers when reading the same table is helpful to improve the reading efficiency, achieve the equilibrium distribution of Table Region.

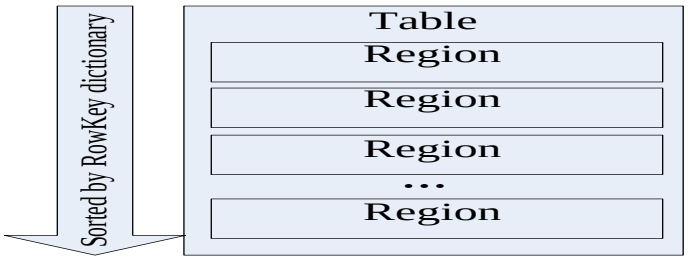

Figure4. HBase Table segmentation

3.1.2 Hot Region split algorithm

First, assume that the server performance, network bandwidth is the same, the algorithm calculate Region Server load. Factors considered in this paper have the number of regions, the visit quantity, the memory usage, the space usage. Related defined as follows:

Nrg：The number of region server storage

Nrq：The visits quantity of server;

M：The memory usage of server

S：The space usage of server

The effects of different parameters on the performance of the requested data is not the same, so the corresponding effect vector weights can be set as $a = (a^{Nrg}, a^{Nrq}, a^{M}, a^{S})$,the a=1,a^{Nrg},a^{Nrq},a^{M},a^{S}

Respectively: the weight of Nrg, Nrq, M,S. The weight vector of the performance parameters should be set according to the actual test results, so as to achieve better results.

Server I's loading can be expressed as the formula (1):

$$L_i = a^{Nrg} * \frac{Nrg_i}{\sum_i^n Nrg_i} + a^{Nrq} * \frac{Nrq_i}{\sum_i^n Nrq_i} + a^{M} * M + a^{S} * S \quad (1)$$

The calculation formula of the server average load as shown in formula (2):

$$L_{avg} = \frac{\sum_i^n L_i}{n} \quad (2)$$

The algorithm steps:

According to the Nrg, Nrq, M, S, computing server load and the average load;

The server $L_i > 2L_{avg}$ join the queue Q, in descending order, the server $L_i < 2L_{avg}$ join the queue S, in ascending order;

Selecting the first server from the Q as k, all regions on the server k sort descending according to the number of requests as queue R;

Selecting the first region from queue R as r, Split into two sub regions: r_A, r_B;

Selecting the first two servers from queue as x, y; Adding the sub regions r_A, r_B to servers x, y, recalculate the x, y load to get L_x, L_y ,if $L_x > 2L_{avg}$,delete server x from queue S, Otherwise the load on the server in the server S sort ascending again, the same as y; if $L_k < 2L_{avg}$,delete server k from queue Q, Otherwise the load on the server in the server S sort descending again;

If queue Q or queue S is empty, the algorithm terminates, or turn (3)

3.1.3 Region merging algorithm

With hot region split, the number of small regions will be more and more, increasing management overhead and affecting server performance. In order to avoid the above situation, at an appropriate time, the Region merging is a must. The relevant are defined as follows:

Rs: The size of the Region.

Rq: The visit quantity of region;

Arq:The average visit quantity of All Region;

The algorithm steps:

According to the key value to sequential search region, When the size of the region1 Rs1 is less than half of the specified size, compare the visit quantity of region1(Rq1) with the average visit quantity of the all regions(Arq1), Otherwise, find next Region.

If Rq1<(Arq1)/2, According to the key value, search the next region, if the size of Rs2 is less than half of the specified size and Rq2<(Arq2)/2,merge the region1 and region2, otherwise, turn (1)

4. EXPERIMENT

In this paper, experimental environment physical topology is shown in Figure 5, to build 8 node Hadoop cluster. Each node is configured as follows: 6-core Intel Xeon CPU 2.4GHz, 8G memory, 10T hard. The network environment is a Gigabit fiber.

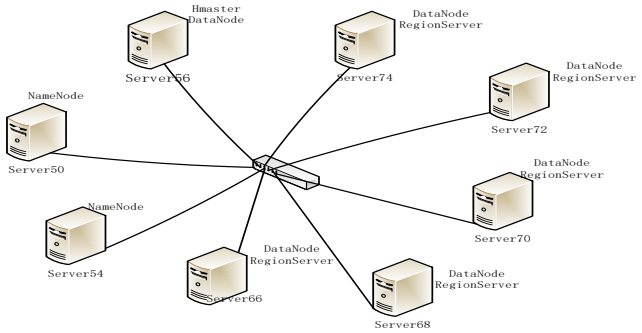

Figure5. Experiment of physical topology

Using HBase built-in load balancing algorithm and the load balancing algorithm based on the Table, compare the average response time of the 500M size Table under different amount of concurrent access

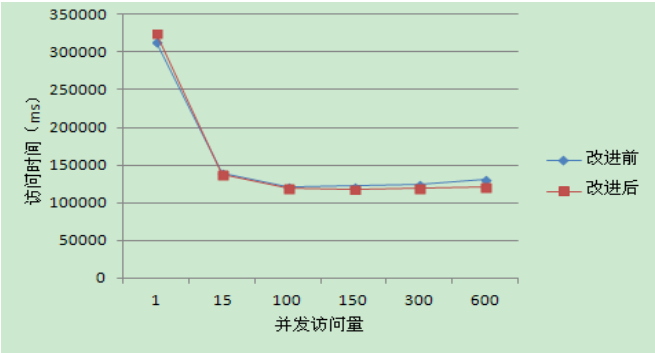

Figure6. To access a Table the average access time contrast figure

As shown in figure 6, the abscissa represents concurrent access number of threads; Ordinate represents access takes time. When the visit quantity is 1, because of the increase of the addressing time, improved algorithm of the access time will be slightly larger than before improvement; when traffic is 15, access time decreased obviously, but because the visit quantity is not large, the access time had little difference between before and after the improvement. When concurrent traffic increases to more than 300, under the original equilibrium strategies, time has a certain rise. And the improved equilibrium strategies effectively balance the read request, the access time no obvious change.

5. CONCLUSIONS

Good load balancing strategy can effectively improve the overall performance of distributed data storage system. According to the characteristics of the HBase database, this paper proposes a load balancing strategy based on table. This strategy effectively assigned requests to each server, improving the overall performance of the system. And the effectiveness of the proposed strategy is validated by experiment. In future work, in order to better dispersion requests to resolve hot issues, according to the distribution characteristics of hot data on the Region, further researching about hot region split standards will continue.

REFERENCES

[1] Cai L, Huang S, Chen L. Performance analysis and testing of HBase based on its architecture[C]. Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on. IEEE, 2013:353 - 358.

[2] Chen Guo-bing.A distributed dynamic load balancing scheduling algorithm[J] . journal of Guangxi Teachers Education University:Natural Science Edition. 2014, (1).

[3] Dong Li-yan, Huang le, Wang Li-min,et al. Imporyed Strategy of Distributed System's Load Balancing[J]. journal of beijing university of technology. 2011, (12):1903-1906.

[4] Han J, Haihong E, Le G. Survey on NoSQL database[C].Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on. IEEE, 2011:363 - 366. [1]Cai, Lizhi, et al. "Performance testing of HBase based on the potential cycle." Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on. IEEE, 2013:359 - 363.

[5]JavaChen.Regionsplitstrategy[EB/OL].http://blog.javachen.com/2014/01/16/hbase-region-split-policy,2014-01-16.

[6] Lars George, "HBase: The Definitive Guide", 1st edition, O'Reilly Media, September 2011, ISBN 9781449396107.

[7] Li Bin, Zhang Ying-wei. The Realization of Load Balancing in NoSQL[J]. Computer Knowledge and Technology,2012,(6).

[8] Nick Dimiduk, Amandeep Khurana. HBase in Action[M]. America: Manning Publications SMARTER, 2012:257-264.

[9] Pang Hui,Yan Jun-chen. Analysis and Research on the Distributed Storage of Data Placement Strategy[J]. JOURNAL OF HEBEI INSTITUTE OF ARCHITECTURE AND CIVIL ENGINEERING. 2012,30(3).

[10] Ronald C Taylor, "An overview of the Hadoop / MapReduce / HBase framework and its current applications in bioinformatics", 11th Annual Bioinformatics Open Source Conference (BOSC) 2010, Boston, MA, USA. July 2010.