Acta Technica *61*, No. 4A/2016, 247–256

Research on virtual technology of data caching and data real-time allocation in cloud computing

Zeyu Yu[1], Huayun Yu[1], Yuncai Zhou[1]

Abstract. A distributed caching technology based on cloud computing is proposed in order to enable massive data processing and high concurrent access. On this basis, the LAB equalization algorithm is used to distribute equally the intermediate data output by the Map output to improve the load imbalance of the processing data in Reduce end. Through the parallel sampling test for each block data set, it is found that the distributed caching technology can relieve the server pressure. The MapReduce model can effectively reduce the running time of the system, which not only improves the system performance, access speed, reliability and response delay, but also makes the input data of the Reduce side achieve load balancing.

Key words. Cloud computing, distributed caching technology, massive data, data distribution, Map Reduce.

1. Introduction

With the development of cloud computing, more and more enterprises will build their own cloud platform, while more and more applications are supported by the cloud platform to provide users with cloud services [1]. With the increase in the number of users accessing the Internet and the large amount of data generated by sales promotion of certain e-commerce websites at certain times, making the cloud services popular, at the same time, the needs of the data visits, access speed and access security rapidly increase. And the distributed processing, distributed database and cloud storage, virtualization technology of the cloud computing for provide technical support and basic guarantee for the massive data analysis and processing [2].

Cloud computing system usually uses MapReduce model to realize the parallel computing and processing of large-scale data sets. The system uses virtual technology to generate virtual resource pool which can be configured and managed independently. Virtual resources and computing services are provided for various application

[1]Yangtze University, Hubei, 434023, China

systems according to actual demand. MapReduce model is a simplified distributed programming model and efficient task scheduling model, the model allows developers do not need to perceive the complex parallel computing and task scheduling of the background, thereby reducing the programming complexity [3]. For the processing of the large-scale data accessing the database with high real-life solution, it needs to be based on the guiding research of the distributed cache technology research and the integration of the related content. The following topics are discussed in this paper.

2. Literature review

In recent years, caching system technology has caused quite a stir at home and abroad, such as virtual data caching technology, cooperative data caching technology, network shared caching technology and other distributed caching technology. At present, the main study focus on two main aspects, the first is that the technical research of cache theory—to improve the cache algorithm and the innovative application of architecture, and the second is that theory is applied to the actual environment, that is, the verification process.

Colorado University in the United States proposed a cache structure, known as hierarchical cache architecture harvest, the main purpose was to greatly increase the hit rate range, but the response time to the user also increased [4]. A high-performance distributed architecture was mentioned, it was a Web proxy server based on a global memory object buffer pool, the buffer pool was established, in which data can be shared [5]; Duke University in the Crisp project proposed a theoretical method based on central map cooperative cache, but at present only in the theoretical research stage. Sinisa Srbljic et al. [6] used a distributed network caching technique to avoid the conflict of network congestion and the excessive load of proxy proxy [7]. The aim was to find out the optimal number of Proxy Server.Distributed caching technology is applied to distributed spatial database, P2P, SOA and many other applications [8].

For Google, Amazon, Alibaba and others Internet companies provide hundreds of millions of users at all times with demand, so in this case, it will bring massive data storage and processing problems, but also is an important moment that.test database load balancing ability. Therefore, the demand in this case, just relying on a few servers is far from being able to meet these data storage and processing requirements, if only enhancing the server performance can not change the requirements of this case. Google and Amazon are the national top technology aggregation, their solution is to abandon the traditional relational database, and use the storage with the key/Value form. Because the distributed cache system of key/Value pair slowly enter People's attention due to Google's big data epoch-making announcement, and the corresponding papers published by Dynamo which is launched by Amazon and Cassandra which is published by FaceBook, the Tair cache system developed independently by Alibaba plays a decisive role in the domesticIn [9]. These successful practices of the top Internet giants in the distributed cache system make the distributed cache system technology become one of the core technology of cloud

computing.

At present, the distributed cache system products have been quite mature and complete, the static cache products are more than the dynamic cache products in quantity, and the technology is relatively mature than the dynamic. Although the dynamic cache technology obtain some achievements in some ways, but still in a stage of exploration and development, namely the confused period, the current products are mainly JCS, OSCache and Memcached, etc. [10]. From the emergence of distributed cache technology until now, the technology is mature, the products are constantly updated, in fact, they emerge because a scientific problem cannot be solved, and Memcached is a key/value caching system which is wildly used and emerges to solve these problems.

3. Research contents and methods

3.1. Distributed caching technology

Distributed cache has high performance, dynamic scalability, ease of use, high availability, distributed code execution and other characteristics [11]. And in order to solve the performance problem under large concurrent and avoid high response latency, distributed cache abandons the original relational database, and uses the key/value form of data storage, accompanied by high-speed memory as storage medium, which can guarantee the high performance, dynamic scalability of the system. Moreover, the distributed cache uses multiple copies of NRW mechanism to avoid the cache single point failure, so as to further ensure the data reliability, and ultimately data consistency on the basis of improving the speed of data access. Distributed caching system should also implement the data redundancy mechanism, which ultimately ensure the security of the system [12].

3.2. MapReduce data distribution problems

MapReduce model take a key value pair (keyin, valuein) as input to output another key value pairs (keyout, valueout), the above operation is achieved by the user-defined Map function and Reduce function. The existing processing mechanism first splits the input data of each Map task, the slice data blocks are executed in parallel on different nodes, and the slice size is specified by the user according to the actual situation. Therefore, the data size processed by each Map task is determinate and basically the same [13]. Based on the intermediate results generated in the calculation, the Map tasks are partitioned by the default Hash allocation algorithm. The data set under the same key value are assigned to the same Reduce node to deal with. Since the size of the data set can be determined after the Map phase is completed. Therefore, the data volume of each Reduce task has dynamic uncertainties which shows that the intermediate data output by the Map end is unbalanced, which leads to the imbalance of the data processing load of the Reduce end [14].

3.3. Data allocation algorithm

We use the above principle to evaluate the locality of the data, which is the ratio of the number of key_s at the local node to the number of key_s distributed at each node. Locality of the LAB data allocation algorithm is:

$$\text{Locality}_{\text{LAB}} = \frac{\sum_{i=1}^{N}(k_i)^j}{\sum_{i=1}^{K} c(k_i)}\,, \tag{1}$$

where $(k_i)^j$ is the number of key_i at the n_jth node and $c(k_i)$ is the sum of the number of key_i at all nodes.

The balance of input data of the Reduce shows that the data size processed by Reduce end is basically the same. In the MapReduce system, the end of the job depends on the slowest sub-tasks, so the operating efficiency is limited by the slowest Reduce tasks. In the LAB algorithm, the difference in the balance of D_{overload} is expressed with overload data, the size is the largest input data volume of the Reduce end minus the average.

$$D_{\text{overload}} = \max(\text{Data}_i^j) - \frac{\text{TotalData}}{M}\,, \tag{2}$$

where Data_i^j represents the amount of data at the node n_j after assigning the key_i, which is calculated as follows:

$$\text{Data}_i^j = \begin{cases} \text{Sum}\, N^j = \sum_{i=1}^{N} c(k_i)^j, \\ \text{Data}\, N_{i-1} + (c(k_i) - c(k_i)^j), \\ \text{Data}\, N_{i-1} - c(k_i)^j. \end{cases} \tag{3}$$

Here $\text{Sum}\, N^j$ is the sum of all keys at node n_j, and when the key is assigned to a node, the standard deviation of each node is used to evaluate the equalization of the assignment, expressed as:

$$\text{Balance} - \text{score}\, N_i = \sqrt{\frac{\sum_{i=1}^{M} (\text{Data}\, N_i - \text{Mean})^2}{M}}\,. \tag{4}$$

The LAB algorithm assigns keys with different sets of numbers to the nodes with the smallest standard deviation. The key_s are arranged in descending order of $c(k_i)^j$ values. In order to improve the locality of data allocation, the key is assigned to the node with the largest number of key_s. After the completion of a key value distribution, the new data on each node is calculated, and then is the next key allocation. the heuristic method is used to assign the data set in turn which is correspond to the N key values distributed in the M nodes to the desired Reduce nodes.

4. Results analysis

4.1. Distributed cache

4.1.1. Overall structure diagram. Considering the function of distributed caches and the model architecture of distributed memory data, the specific and effective concordance is achieved by the distributed and coordinated mechanism of Zookeeper, and a distributed cache architecture is realized. Distributed cache overall structure is shown in Fig. 1 [15].

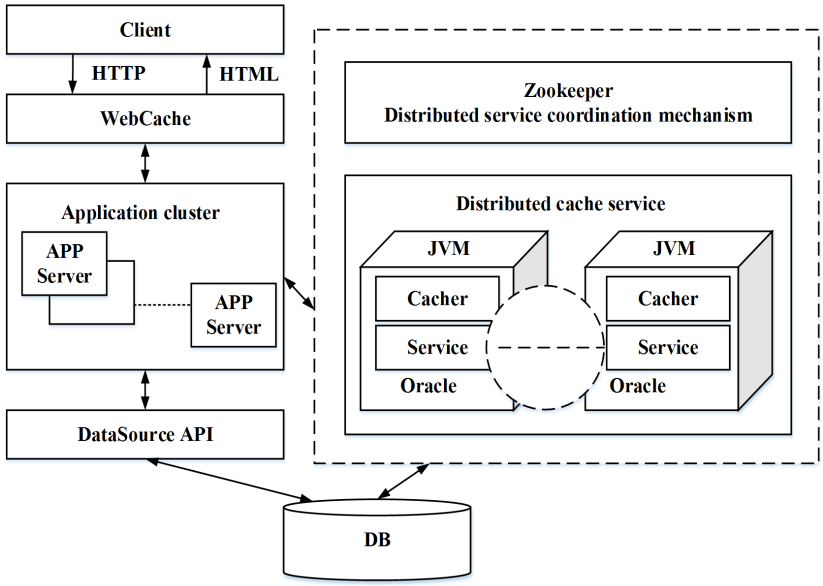

Fig. 1. Distributed cache overall architecture diagram

4.1.2. Reliability testing. The distributed cache system environment is tested for reliability, including data recovery and data integrity testing for each cache node. The test results are shown in Table 1.

Table 1. Data recovery time table

Add or delete the cache nodes	Data recovery time (s)
No-operation	About 2.2
Commodity fuzzy query	About 3

It can be seen that the distributed cache system can meet the requirements of application and meet the system performance requirements. The distributed memory data management has high reliability and high expansibility, and contains powerful fault tolerance and self-repair function.

4.2. Data distribution

4.2.1. Working time. In different allocation strategies, the execution time of the system varies with the inclination of the data set, as shown in Fig. 2. The data set size is 1 G, and depicted is an enlarged view of the LABWC working change curve.

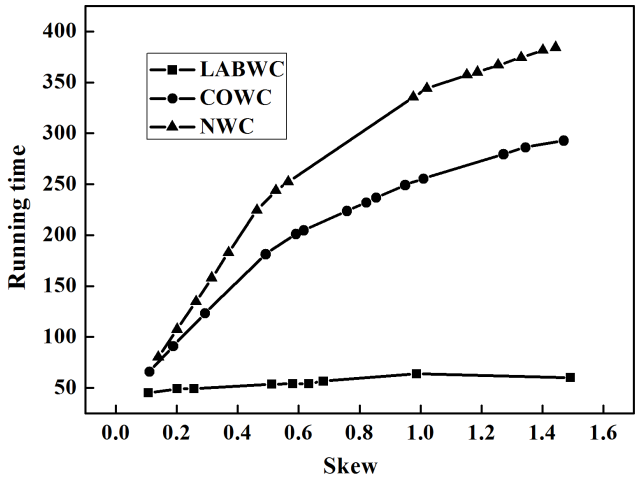

Fig. 2. Running time under different inclination

It can be seen that for different size of data set the running time increases with increasing tilt and the time difference tends to increase. This is due to the increase of the inclination. In the NWC mode, it takes more time to wait for the data from the Reduce side to complete the job. The COWC method needs to merge more data sets, and the LABWC method needs to balance the more inclined data. At the same time, the overall operation time of LABWC operation mode is smaller than that of NWC and COWC, and the change of the time difference of the different inclination is small, and the time difference shows a decreasing trend.

WordCount experiment was conducted on the data set with 1.5 degree inclination. The relationship between the running time of the system and the size of the data set is shown in Fig. 3.

It can be seen that the run-time increases with the increase of data set for data sets with different data set but different inclination. The run-time gap between LABWC and NWC and COWC is significantly increased with the increase of processing data set. It can be seen from the 2G data set experiment that the LABWC running time is about half of the NWC running time and about one third less than the COWC operation. It is shown that the improved LAB algorithm can effectively improve the efficiency of parallel operations.

4.2.2. Load balancing. The standard deviation is used to evaluate the load balance. The ordinate Stdev represents the standard deviation. The abscissa is the

inclination of the data set. The smaller the value of Stdev, the better the load balance of the processed data. The data set with the size of 1 G and the inclination between 0.1 and 1.5 was used to carry out the WordCount experiment. The experimental results show that the load balance results are shown in Fig. 4.

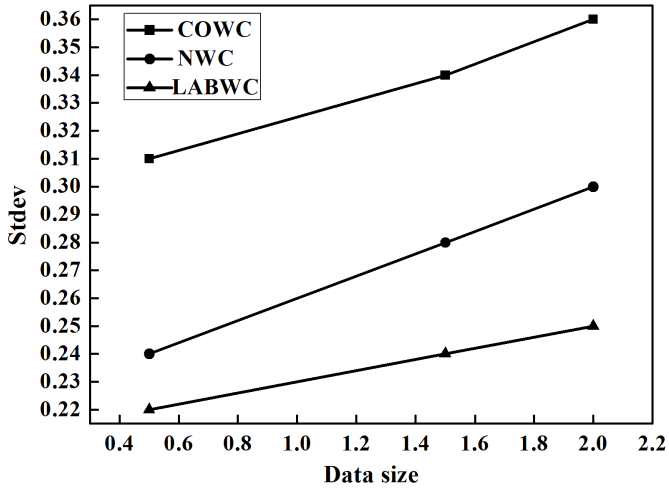

Fig. 3. Running time of data sets of different sizes

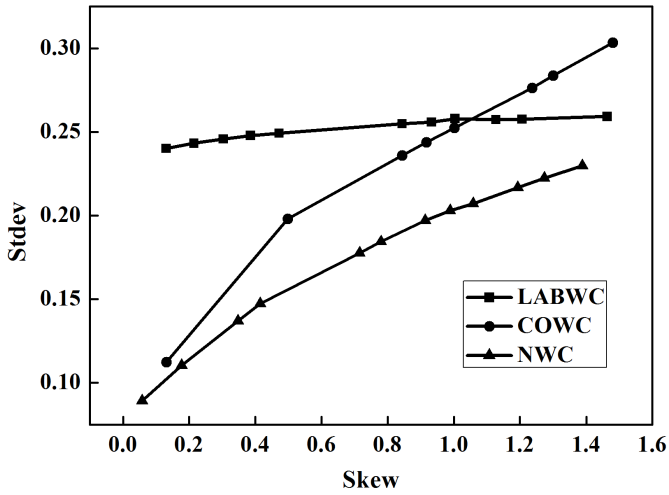

Fig. 4. Load balance of reducing end with different inclination

It can be seen from the figure, Stdev's change trend and execution time trends are basically consistent, while reflecting that the smaller degree of tilt, the load balance of the difference smaller. With the increase of the inclination degree, the

advantage of load balancing of Reduce end under LABWC mode appears. The results show that the LAB allocation strategy based on the sampling results can effectively improve the balanced distribution of the intermediate data of the Map end, and make the Reduce end handle the data size balance. When the inclination is 1.5, the load balance of different size data set is shown in Fig. 5.

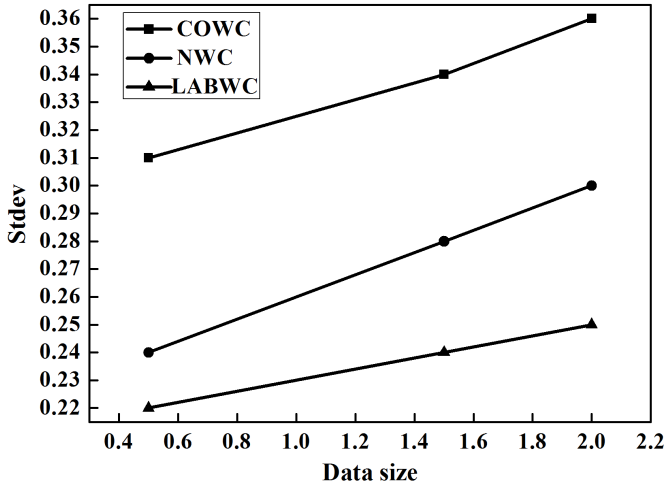

Fig. 5. 5 Load balance of different size data set

It can be seen from the figure that the Stdev values of LABWC under different data sets are smaller than those of other methods. With the increase of data set, the Stdev values of different operation modes show an increasing trend, but the increase of NWC and COWC modes is significantly greater than the LABWC operation mode, therefore, LABWC operation mode has obvious advantages of load balancing.

5. Conclusion

We studied how to use distributed caching technology in cloud computing to solve the problem in the massive data processing platform to ensure high performance, high access speed, low latency and high reliability of the system. The deployment and the overall architecture design of distributed caching system is proposed and discussed

The LAB algorithm is used to optimize the unbalanced distribution of the intermediate data. The improved LAB algorithm can balance the locality of the data and input of the Reduce end, which greatly improves the efficiency of parallel operations. Performance verification is performed on the distributed cache open source platform through an experimental cluster. The experimental results show that this method can distribute the load balance data for the Reduce end, and can solve the performance bottleneck of the unbalanced input data in the Reduce side.

References

[1] Y. Zhao, J. Wu, C. Liu: *Dache: A data aware caching for big-data applications using the MapReduce framework.* Tsinghua Science and Technology *19* (2014), No. 1, 39–50.

[2] C. L. P. Chen, C. Y. Zhang: *Data-intensive applications, challenges, techniques and technologies: A survey on big data.* Information Sciences *275* (2014), 314–347.

[3] K. Kambatla, G. Kollias, V. Kumar, A. Grama: *Trends in big data analytics.* J. Parallel and Distributed Computing *74* (2014), No. 7, 2561–2573.

[4] H. Cai, B. Xu, L. Jiang, A. V. Vasilakos: *IoT-based big data storage systems in cloud computing: Perspectives and challenges.* IEEE Internet of Things J. *PP* (2016), No. 99, 1–1.

[5] S. Jiang, T. Chen, J. Dong: *Application and implementation of private cloud in agriculture sensory data platform.* Computer and Computing Technologies in Agriculture IX, IFIP Advances in Information and Communication Technology *479* (2016), 60–67.

[6] K. Gai, M. Qiu, H, Zhao: *Security-aware efficient mass distributed storage approach for cloud systems in big data.* Proc. IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 8–10 April 2016, New York, USA, 140–145.

[7] E. C. Peters, S. Rabinowitz, H. R. Jacobs, P. J. Fasciano: *Computer system and process for transferring multiple high bandwidth streams of data between multiple storage units and multiple applications in a scalable and reliable manner.* Patent US 9432460 B2 (2016).

[8] Y. Shen, P. Lu, X. Qin, Y. Qian, S. Wang: *Efficient query algorithm of coallocation-parallel-hash-join in the cloud data center.* Cloud Computing and Security, series Lecture Notes in Computer Science *9483* (2016), 306–320.

[9] H. Duan, S. Yu, M. Mei, W. Zhan, L. Li: *CSTORE: A desktop-oriented distributed public cloud storage system.* Computers & Electrical Engineering *42* (2015), 60–73.

[10] A. Uta, A. Sandu, T. Kielmann: *Overcoming data locality: An in-memory runtime file system with symmetrical data distribution.* Future Generation Computer Systems *54*, (2016), 144–158.

[11] D. Dev, R. Patgiri: *A survey of different technologies and recent challenges of big data.* Proc. 3rd International Conference on Advanced Computing, Networking and Informatics, series Smart Innovation, Systems and Technologies, 23–25 June, Bhubaneswar, Orissa, India, 537–548.

[12] A. A. Yildirim, D. A. Watson: *A cloud-aware distributed object storage system to retrieve large data via HTML5-enabled web browsers.* Managing Big Data in Cloud Computing Environments (2016), Chapter 2, 25–45.

[13] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, W. Xiang: *Big data-driven optimization for mobile networks toward 5G.* IEEE Network *30* (2016), No. 1, 44–51.

[14] I. Stefanovici, E. Thereska, G. O'Shea, B. Schroeder, H. Ballani, T. Karagiannis, A. Rowstron, T. Talpey: *Software-defined caching: Managing caches in multi-tenant data centers.* ACM Symposium on Cloud Computing, 27–29 Aug. 2015, Kohala Coast, Hawaii, 174–181.

[15] C. Vitolo, Y. Elkhatib, D. Reusser, C. J. A. Macleod, W. Buytaert *Web technologies for environmental Big Data.* Environmental Modelling & Software *63* (2015), 185–198.

Received November 16, 2016