Advanced Science and Technology Letters
Vol.78 (MulGrab 2014), pp.55-59
http://dx.doi.org/10.14257/astl.2014.78.10

A Cache Management Scheme for Object-based Storage

Ying Shan [1], Dan Liu [1], Nian-min Yao [2]

[1] College of Information and Computer Engineering, Northeast Forestry University, Harbin 150040, China
[2] College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
{shanyingsc}@163.com

Abstract. This paper presents a cache management scheme for object-based storage, which consists of three main aspects. First, the cache update policy is proposed. The strategy considers the overall cache as a pool, consistency and accuracy of updating is ensured through the locking mechanism, while using Bloom Filter combined with the updated time series to achieve updating simplification. Secondly, the cache replacement strategy is introduced which achieving the cache replacement policy using cache spanning tree combined with cache stack. Finally, cache management model is proposed based on energy consumption. Experimental results show that the proposed cache management scheme can improve the efficiency of cache management in object-based storage system, while reducing energy consumption of cache management.

Keywords: object-based storage system; cache design; updating; replacement; energy consumption

1 Introduction

With the expanding scale of object storage system (OBSS), the need of services is increasing which promote higher performance demanding for the size of the OBSS[1-4]., different object has different size and property, therefore, the hierarchy cache scheme from the size of object is proposed by us to solve the problem[5-8].

Based on the above analysis, a cache design of load balancing system for object-based storage is proposed. The cache design adopt effective cache scheme through establishing L1 and L2 level cache in MDS and OSD, including updating replacement policy and locking mechanisms of cache, in order to achieve collaborative cache, while energy consumption cache design is adopted which ensure consistency, isolation, durability and stability of cache operations to the greatest extent to obtain the best performance.

2 Main Ideas of cache Design

2.1 Cache updating scheme

In structure of cache pool for the OBSS, all of cache in MDS construct cache pool, namely L1, all of cache in OSD constitute cache pool, namely L2.

The key issue is the timing of the accumulation update, take the write OSD operation into account to merge the cumulative update and simplification is an effective way. Bloom filter matrix storing metadata hash values and unconsolidated update sequence is adopted before Simplification and update operations .

A lock mechanism is adopted during update process. Mutex lock is set during the update operation for the same object, which means multiple update operations to the same object can not be performed simultaneously on the same time. Update operations is accomplished by update queue, while there is only one update operate effectively at a time. The update exclusive lock is set during the process of consolidation and simplification update operations which writes has the highest priority, while other operations has to wait until the completion of writes operation.

2.2 Cache Replacement Strategy

We design collaboration L1 cache pools structure with multiple cache, using B-balanced tree structure as a tree topology between cache of L1 cache pool. The cache which MDS requests is regarded as the root node of the tree topology, namely C1 cache. Topology tree is established based on the query cost between nodes, thus ensuring minimal overall cost of the query.

In the cache pool topology, the client sends the request to its neighboring MDS, while MDS searches the local cache according to customer request, if the request does not exist in the local cache, the cache topology need to establish topology tree until you find the requested metadata information. If not, the metadata information will be updated.The topology tree structure is corresponding to a stack buffer pool, the structure determines the replacement policy of cache data.

The stack is set in the order of the tree hierarchy traversal according to the topology which arranged from top to bottom, i.e., a collection of stack cache. The metadata information stored inside the stack top-down are according to the principle of metadata most recently accessed objects. Adjacent cache stack can be exchanged between the accessed cached data, the data from the mobile to the arrangement of the principles remain as LRU principle. If metadata which the client requests is not contained in L1 cache pool, then the metadata storage device call the appropriate metadata information stored in the local cache MDS, while metadata information in the bottom of the stack is moved to an adjacent metadata information stack cache.

2.3 Evaluation of energy-consuming cache

Cache in different update and replace operations will cause energy-consuming. The aim of our cache design is for purposes of reducing the energy loss in cache operations. Energy consumption of cache management is regarded as mark for performance evaluation of OBSS, which is reflected by the average response time for requesting services indirectly.

3 Performance Evaluation

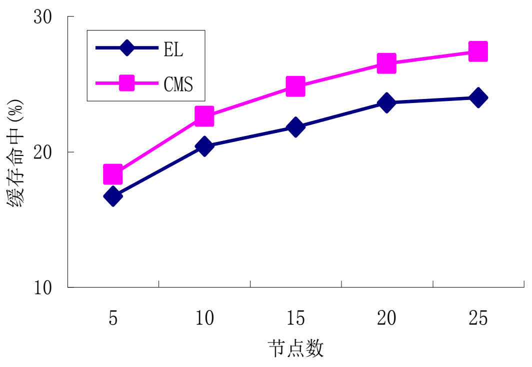

Fig. 6. Cache hit of different node number

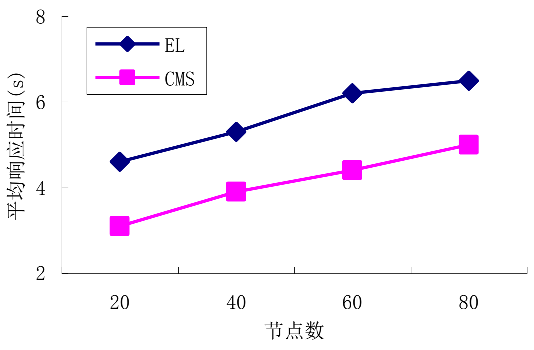

Fig. 7. Average response time of different node number

In our experiment, we employ a simulation to evaluate the proposed architecture. We first introduce experimental settings. We used NS-2 as the network simulation tool and disksim 4.0 as tool to realize the OSD nodes. We use scheme in this article (referred to as CMS) with the use of replacement LRU algorithm based on edge updating program (referred to as EL) for comparison.

With the increase number of nodes increased in CMS, performance will change accordingly. For this situation, we tested the changes hit rate while the nodes number were 5, 10, 15, 20 and 25. The result in Fig. 6. shows when nodes number is little, EL and the CMS cache hit has little difference, with the increasing number of nodes, the difference cache hit is increasing, while growth of EL and CMS in cache hits is constantly on the increase. Overall, CMS has been greater than EL cache hit. As we can see from the experimental results, when number of nodes increases, the complexity of the advantages of CMS are apparent.

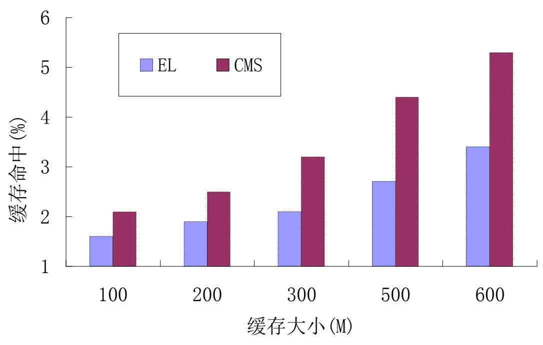

Fig. 8. Cache hit of different cache size

The average response time are tested with the number of different nodes in Fig. 7., The figure shows the impact of average response time in CMS and EL with the nodes number increasing. As the nodes number in the same circumstances, CMS obtains less average response time than EL. As the number of nodes increased, the average response time is increasing, while the average response time of EL is also constantly increasing. Overall, CMS has less average response time than that of EL. From the experimental results, when number of nodes increases, performance of CMS affected by the number of nodes is not obvious.

In our tests, as taking into account performance of different cache size, we adopted cache hit of cache size as mark. From the result of our tests in Fig. 8., CMS continues to show its better than EL with the increasing cache size. From the figure, the trend curves shows that the growth rate of CMS shows faster than that of the EL. Judging from the overall trend, CMS cache hit growth rate is always greater than that of EL, it can also be seen from the figure, with the request arrival rate increases, our scheme has more influence on growth rate of CMS than EL.

4 Conclusion

This paper presents a cache management scheme for object-based storage, Our design takes several aspects. First, the proposed cache update strategy. The policy will be fully considered as a cache pool, ensuring the consistency and accuracy of the update by locking mechanism, while using Bloom Filter and updating time-series to accomplish simplified updates. Secondly, the proposed cache replacement policy, combination of cache spanning and stack structure to achieve cache replacement policy. Finally, the proposed cache design model take energy consumption into account. The cache scheme become an effective way to promote performance of OBSS and reduces energy consumption.

In our immediate future work, we focused on the energy issues of cache for OBSS, consider how to minimum the energy in the cache optimization according to the distribution of the load, thereby using the server cache design to achieve efficiency and improve service performance.

Acknowledgments: This work is supported by the Fundamental Research Funds for the Central Universities No.DL13BBX03 and Fundamental Research Funds for the Central Universities: HEUCFT1202,HEUCF100609.

References

1. Mesnier M, Ganger G R, Riedel E, Object-Based Storage, IEEE Communications Magazine, 2003, 41(8), pp. 84-90.
2. A. Azagury, V. Dreizin, M. Factor, , E. Henis, D. Naor, N. Rinetzky, O. Rodeh, J. Satran, A. Tavory, L. Yerushalmi, Towards an object store , Proc. 20th IEEE/11th NASA Goddard Conf. Mass Storage Systems and Technologies, 2003(MSST 2003), pp. 165--176 (2003)
3. M. M. FACTOR, K. METH, D. NAOR, et al, Object Storage: The Future Building Block for Storage Systems, Proceedings of the 2nd International IEEE Symposium on Mass Storage Systems and Technologies. pp. 119--123 (2005)
4. David H. C. Du and Recent. Advancements and Future Challenges of Storage Systems. Vol. 96, No. 11,Proceedings of the IEEE, pp. 1875--1886 (2008)
5. T.Gonzalez, Clustering to minimize the maximum inter-cluster distance, Theoretical Computer Science,Vol 38, pp. 293--306 (1985)
6. BHARADWAJ VEERAVALLI. WPAR: A Weight-based Metadata Management Strategy for Petabyte-scale Object Storage Systems. Proceedings of the fourth international workshop on Storage Network Architecture and Parallel I/Os, pp. 99--106 (2007)
7. SCOTT A. BRANDT, ETHAN L. MILLER, DARRELL D. E. LONG, XUE Lan. Efficient Metadata Management in Large Distributed Storage Systems, in Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage System and Technologies (MSS’ 03). pp. 290--298 (2003)
8. LIU Qun, FENG Dan, WANG Fang. Research on Metadata Server of High Reliability, Computer Engineering, 34(17) , pp. 88--90 (2008)