Design and Implementation of a Random Data-Placement System with High Scalability, Reliability and Performance

Kun Liu, Wei Xue, Di Wang, and Jiwu Shu

Dept. of Computer Science and Technology, Tsinghua University, Beijing
Liukun04@mails.tsinghua.edu.cn

Abstract. As storage system scales to thousands of disks, data distribution, load balance and the support for heterogeneous disks become increasingly important. In this paper, we present a new data-placement method named Weighted Interval Algorithm (WIA) for heterogeneous disks. Through it is not optimal in some circumstances, the difference between WIA and the optimal algorithm is trivial. Combined with replication, WIA can nearly balance access load and space utilization and improve reliability simultaneously. For the first time, we implement a data-placement system with high scalability, reliability and performance. The experimental results show that WIA reduces the average response time by 14.8% and decreases coefficient of relative load from 78.09% to 47.46% while the difference of the ratio of space utilization between disks is not more than 0.79%.

1 Introduction

With the development of computer technology, more and more information is created and used in a digital way. When this trend makes it easier for us to utilize information, it also makes the storage system's scalability, reliability and performance become increasingly important. This domain has become a focus and difficulty of research [1], [2], [3], [4], and [5].

Data placement is an effective way to improve scalability of storage system. It maps data objects to different disks to balance access load and space utilization. There are two well-known data-placement policies: striping and random data-placement [4]. Compared to striping, random data-placement moves almost optimal data objects to achieve new balance when the characteristic of storage system changes and so has better scalability. What's more, its performance is good and comparable to striping [3].

At the same time, data placement introduces the problem of reliability. It may place logically related objects to different disks and each disk can become single point of failure. And heterogeneous disks are inevitable in a massive storage system, so determining how to distribute data to heterogeneous disks is another problem faced by data placement.

At present, most researches on data placement are about the theoretic model and little attention is paid to the reliability of data-placement system. Furthermore, data placement based on disk capacity is hard to balance access load and space utilization at the same time. In this paper, in order to describe the power of disks, we define the

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 695 – 702, 2006.

weight of disk, which can be the capacity, throughput or some combination of the two. Then a new data placement method named Weight Interval Algorithm (WIA for short) is proposed. Combined with replication, WIA can balance the access load and space utilization and improve reliability at the same time. For the first time, we implement the data-placement method in a real storage system AXUM, which is an in-band virtualization system and introduced in section 3, and get some valuable experimental data to support this method.

2 Related Work

Liner Hashing and it variants [9], [10], [11], and [12] adopt scalable distributed data structure. This method does not take into account the differences between disks and so does not support heterogeneous disks. An algorithm for pseudo-random distribution of data to multiple disks using portioning of the unit range is proposed in [2]. But it doesn't allow for the placement of replicas and isn't optimal even in theory.

The algorithm in [1] can place data objects optimally. One flaw of this method is that it supports heterogeneous disks in a restricted way. The concept of cluster is introduced in this algorithm, which is a group of homogeneous disks. Disks are added into the system in a cluster way, which means only homogeneous disks can be added to the system at a time. And the data object and its replicas must be placed at the same cluster. This may introduce heavy load imbalance between clusters. Another flaw is that though heterogeneous disks can be added, the construction of a storage system must begin with a lot of homogeneous disks. The reason why cluster is introduced is that under some circumstances the optimal algorithm for replicas placement does not exist but with the prerequisite of cluster, the algorithm in [1] is optimal. On the contrary, our algorithm, WIA, has no prerequisites. Disks can be added into storage system in any way. And the difference between WIA and the optimal algorithm is rather trivial.

At present, data-placement algorithms are mostly aimed to balance the space utilization of disks; little attention is paid to access load. The method in [13] is to place data objects based on B-ZBSR, bandwidth-zone bandwidth to space ratio. Though it can alleviate the imbalance of access load, it wastes a lot of disk space.

Generally speaking, there are two policies to improve reliability: replication and parity. The latter is complicated and hard to scale. In [8], data object and its replica are placed to neighboring disks to improve reliability. This method can not utilize the parallelism of multiple disks adequately. If a disk has many hotspots, then most of the access is focused on two disks while others are free.

3 AXUM: An In-Band Virtualization System

AXUM is an in-band virtualization system and our platform for data placement. All physical disks (PD for short) are integrated into a storage pool named Source Container. Every PD is partitioned into several segments named Storage Granularity (SG for short) with the same size. According to the requirement, the administrator can allocate some storage space, that is to allocate some SG from SC to form a Virtual Disk (VD for short), which is used by users. So a VD is also composed of different SGs,

which are located in different PDs. Then the function of data placement on the basis of AXUM is to map each SG in VDs to a unique SG in different PDs to achieve a high scalability and performance.

4 Weighted Interval Algorithm

4.1 The Model and the Criterion

Assume that there are n physical disks PD_1 $PD_2 \ldots PD_n$ in the storage system and the weight of PD_i is w_i .The weight can be the capacity, throughput or some combination of the two. Define the weighted interval for PD_1 as $(0, w_1)$, PD_2 as $(w_1, w_1 + w_2)$, PD_n as $(\sum_{i=1}^{n-1} w_i, \sum_{i=1}^{n} w_i)$. The data object set to be distributed is $S = \{SG_1, SG_2 \ldots SG_m\}$ and m is the size of S.

We can evaluate different data placement algorithms according to the following criterions [2].

1. **Faithful distribution**, i.e. distributing a set of objects among a set of disks in such a way that the fraction of objects stored at a disk is equal (or at least close) to its share of the total weight of the system.
2. **Efficient localization**, i.e. computing the position of an object with a low time and space complexity.
3. **Fast adaptation**, i.e. adapting to changing weight with a near-minimal movement of objects.

4.2 Weighted Interval Algorithm

When replicas are not taken into account, WIA is similar to the algorithm in [1]; the difference is that we get rid of the restriction of cluster so it becomes much more flexible and simple. The basic idea of WIA is that for every data object SG_i in the set of S a random number r is generated, which distributes evenly between 0 and $\sum_{i=1}^{n} w_i$.

The object SG_i is placed to PD_i when r belongs to the weighted interval of PD_i . Under this circumstance, this algorithm can distribute data objects faithfully [1].

4.3 The Placement of Replicas

The principle of the placement of replicas is that no two replicas of a data object can be placed at a same disk. No algorithm can satisfy both the criterion in 3.1 and the principle above because under some circumstance replicas of a SG have to be placed at a same disk if faithful distribution is satisfied firstly.

So an approximation algorithm is proposed. Assume PD=$\{PD_1, PD_2 ... PD_n\}$; the data set to be placed is $S = \{SG_1, SG_2 ... SG_m\}$ and each SG_i has a certain number of replicas. For each SG_i, SG_i is placed by WIA at PD_1 for example. Then the first replica of SG_i is distributed to PD-{ PD_1 } by WIA, assume the result is PD_2. The rest replicas are distributed to PD-{ PD_1, PD_2 } in the same way.

Assume the number of data objects is N and each has a replica. Then the number of data objects placed at PD_i is $S_{oi} = N\frac{w_i}{w}$; the number of replicas placed at PD_i is $S_{ci} = \sum_{j \neq i} N\frac{w_j}{w} * \frac{w_i}{w - w_j} = N\frac{w_i}{w}\sum_{j \neq i}\frac{w_j}{w - w_j}$. So the total number of data objects placed at PD_i is $S_i = S_{oi} + S_{ci} = N\frac{w_i}{w}(1 + \sum_{j \neq i}\frac{w_j}{w - w_j})$.

Let $w_0 = \sum_{i=1}^{n}\frac{w_i}{w - w_i}$, then $S_i = N\frac{w_i}{w}(1 + w_0 - \frac{w_i}{w - w_i})$. This means this method trends to place data in disks with small capacity. The ratio of space utilization of PD_i is $\frac{S_i}{w_i} = \frac{N}{w}(1 + w_0 - \frac{w_i}{w - w_i})$ and the difference of ratio between disks is decided by w_i. In a massive storage system, w_0 and w are much bigger than w_i. So the difference is trivial.

5 Implementation of the Data-Placement System

5.1 Weight, Hotspot Replication and Load Balance

The weight in WIA is a significant parameter to determine the number of data objects placed at a disk. Generally speaking, two main parameters depicting a disk are the capacity and throughput. Throughput is not a good choice of weight. Firstly, the throughput of disk is determined by the disk interface, average seeks time, rotational speed and access pattern so it is not a constant. Secondly, the development of disk throughput is much slower than that of disk capacity and if objects are placed according to throughput, a lot of disk space will waste. Finally, under practical environment hotspot does exist so the weight of throughput can not ensure maximal bandwidth. The space utilization of each disk is the same if the weight is disk capacity. But because of the existence of hotspot, the load of each disk will be unbalanced.

Based on our analysis of the HP trace [6] such as cello 92, we observe that disk access has a high degree of space locality and very often less than 20% data objects serve more than 90% access. So, if replicas of hotspot, named hotspot replicas, are created and placed on the disk with light relative load, then this can not only improve

performance but also balance access load. The relative load is defined as the ratio of disk access frequency to disk bandwidth.

We update the replicas and the original data object synchronously to ensure the consistency of data. The number of hotspot replicas has a great influence. If the replicas number is too small, access load of disks can balance to some degree but there is still some space to go on. If the replicas number is too large, though the access load balances well, because every write should update many disks synchronously, the additional load can outweigh the advantage of load balance and the total performance of the system would degrade. So we calculate the number of hotspot replicas in the following way. Record the access frequency of each SG and calculate the average access frequency (AAF for short). Define a parameter ***threshold***, which is an adjustable number. If the access frequency of a SG is more than the product of average access frequency and the threshold, then this SG is a hotspot and replicas should be created. The number of replicas is determined by $\left\lfloor \log^{access\_frequency} \big/ _{(AAF * threshold)} \right\rfloor + 1$. The reason why logarithm is used is that logarithm can ensure that the number of hotspot replicas is moderate.

5.2 Reliable Replicas and Overhead

Besides balancing load, replicas can improve reliability [7], [8]. In an ideal data placement system, the crash of a physical disk can degrade the service of the whole storage system dramatically. Then replicas, named reliable replicas, can be created for each SG to improve reliability. These replicas can be placed by the algorithm in 4.3, and if C replicas are created for SG, the system can still work well when no more than C disks fail. The number of reliable replicas for a SG is determined by the importance of SG and how to get the importance is out of the scope of this paper.

Replication introduces two kinds of overhead. Firstly, excess space is needed to store replicas. In practice, the data stored at disks are much more valuable than the medium storing them. And with the fast increase of disk capacity and rapid decrease of disk price, we consider the additional disk space is worthy and an example is Google File System [14]. The second overhead is caused by synchronous update for writing replicas, which is unavoidable but can be alleviated. Firstly, not all hotspot are suitable for replication. When the majority of access to a hotspot is writing, then replication of that hotspot will degrade performance. Secondly, since hotspot replicas can also act as reliable replicas, if hotspot already has reliable replicas, then these reliable replicas can be changed to hotspot replicas.

6 Experiment Evaluation

6.1 Evaluating Data Placement

Figure 1 illustrates the ratio of data objects to be migrated when additional storage capacity is added to AXUM. Striping achieves scalability at a very high cost; almost every data object has to be moved. The number of data objects moved by WIA is very close to the

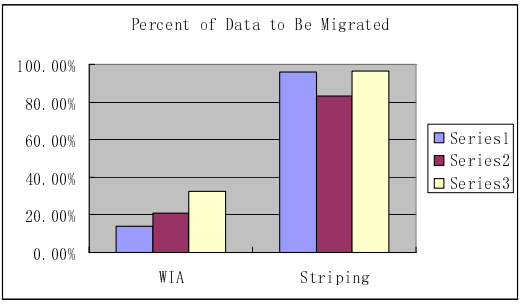

Fig. 1. Percent of data to be migrated when additional storage capacity is added to AXUM. Series 1 shows that 13.42% of storage capacity is added; series 2 shows 20.13% is added and series shows 33.55% is added.

optimal ratio. Our experiment also shows that WIA performs better for random I/O while striping is better for sequential I/O. This is consistent with the results in [4].

6.2 Evaluating Replication

If we define coefficient of relative load as the ratio of the difference between maximal relative load and minimal relative load to maximal relative load, then that value for AXUM without hotspot replicas is 78.09%, while the optimal value is 0. This shows the necessity of hotspot replicas.

Figure 2 is the result after hotspot replicas are created. The horizontal line in the figure is the average response time when there are no hotspot replicas. The result is the same as the discussion in section 5.1.

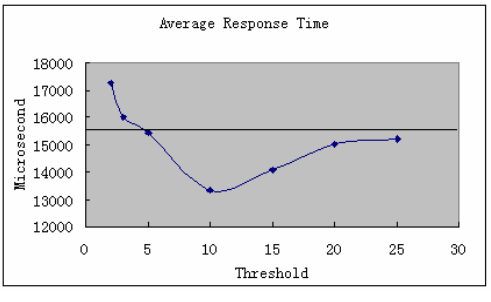

Fig. 2. Threshold and the corresponding average response time. The unit of time is microsecond. This experiment makes it clear that threshold has a great influence on the performance, and 10 is a good choice.

The coefficient of relative load is 47.46% after hotspot replicas are created with the threshold of 10. Of course we can create more hotspot replicas by decreasing threshold to make the coefficient of relative load drop to 0, but this will increase overhead of synchronous update and degrade the performance.

As to reliable replicas, in our experiment, we create 2 reliable replicas for each SG and pull out one and two disks from disk array and test the function and measure the average response time of AXUM. The result is that every request gets a perfect response which means the integrity of data objects is guaranteed and the average response time increases by 12.5% when a disk fails.

7 Conclusion and Future Work

This paper proposes Weight Interval Algorithm for data placement, which is simple, flexible and can be applied to any case. Through it is not optimal in theory, the experiment data shows that in practice it can obtain a satisfying result. Based on this algorithm, we build a data-placement system. And to balance disk access and improve reliability, reliable replicas and hotspot replicas are introduced into this system. By experiment, we prove that this data-placement system is of high scalability, reliability and performance.

Data placement is about the issue of how to distribute data objects to different disks to improve performance and scalability. And the issue of data layout, which is about how to place data objects in a disk to reduce seek time and rotational delay, is one direction for further research. On the other hand, data placement can be easily extended to information lifecycle management, which places data not only according to the characteristic of storage medium but also the importance of data.

References

1. R. J. Honicky and E. L. Miller. A fast algorithm for online placement and reorganization of replicated data. In Proceedings of the 17th International Parallel & Distributed Processing Symposium, Nice, France, Apr. 2003.
2. Andre Brinkmann, Kay Salzwedel, and Christian Scheideler. Compact, Adaptive Placement Schemes for Non-Union Capacities In Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pages 53-62, Winnipeg, Manitoba, Canada, Aug. 2002.
3. Beomjoo Seo. Survey on Data Placement and Migration Algorithms in Distributed Disk Systems, In Proceedings of 2004 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04), Las Vegas, Nevada, USA, June 21-24, 2004.
4. Jose, Richard, Berthier. Comparing Random Data Allocation and Data Striping in Multimedia Servers In Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems , Santa Clara, California, United States, Pages: 44 - 55 .
5. Roger Zimmermann Shahram Ghandeharizadeh. Continuous Display Using Heterogeneous Disk-Subsystems In Proceedings of the fifth ACM international conference on Multimedia Seattle, Washington, United States Pages: 227 – 238.
6. Chris Ruemmler and John Wilkes. UNIX disk access patterns In Proceedings of the Winter'93 USENIX Conference, January 1993, Pages 405-420.

7. John Wilkes, Richard Golding, Carl Staelin, and Tim Sullivan. The HP AutoRAID hierarchical storage system In ACM Transactions on Computer Systems, Pages: 14 (1):108-136.
8. Akitsugu WATANABE Haruo YOKOTA. Adaptive Overlapped Declustering: A Highly Available Data-Placement Method Balancing Access Load and Space Utilization 21st International Conference on Data Engineering (ICDE 2005).
9. W. Litwin, J. Menon, and T. Risch. LH* schemes with scalable availability. Technical Report RJ 10121 (91937), IBM Research, Almaden Center, May 1998.
10. W. Litwin, M. Neimat, G. Levy, S. Ndiaye, T. Seek and T. Schwarz. LH*s: a high-availability and high-security scalable distributed data structure. In proceeding of the 7th International Workshop on Research Issue in Data Engineering. 1997, Birmingham, UK, Apr. 1997. IEEE, Pages 141-150.
11. W. Litwin and M.-A Neimat. High-availability LH* schemes with mirroring. In proceeding of the Conference on Cooperative Information Systems, 1996, Pages 196-205.
12. W. Litwin, M-A. Neimat, and D. A. Schneider. LH*-a scalable, distributed data structure. ACM Transactions on Database Systems, 1996, Pages: 21(4):480-525.
13. Yong-Sook Park, Jeong-Won Kim, Ki-Dong Chung. A Continuous Media Placement using B-ZBSR on Heterogeneous MZR Disk Array In Proceedings of International Workshops on Parallel Processing, 1999,Pages: 482-487.
14. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System, In Proceedings of the nineteenth ACM symposium on Operating systems principles Bolton Landing, NY, USA, Pages: 29 – 43.