T2R2 東京工業大学リサーチリポジトリ
Tokyo Tech Research Repository

論文 / 著書情報
Article / Book Information

Title	Evaluation of Placement and Access Asignment for Replicated Object Striping
Author	Makoto Kataigi, Dai Kobayashi, Tomohiro Yoshihara, Takashi Kobayashi, Ryo Taguchi, Haruo Yokota
Journal/Book name	Proc. of International Special Workshop on Databases For Next Generation Researchers (SWOD 2006), Vol. , No. , pp. 100-105
Issue date	2006, 4
DOI	10.1109/ICDEW.2006.57
URL	http://www.ieee.org/index.html
Copyright	(c)2006 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

Powered by T2R2 (Tokyo Institute Research Repository)

Evaluation of Placement and Access Asignment for Replicated Object Striping

Makoto Kataigi[1], Dai Kobayashi[2], Tomohiro Yoshihara[2], Takashi Kobayashi[3],
Ryo Taguchi[4], and Haruo Yokota[3,2]

[1] Faculty of Engineering, Tokyo Institute of Technology

[2] Grad. School of Info. Sci. and Eng., Tokyo Institute of Technology
[3] Global Scientific Info. and Comp. Center, Tokyo Institute of Technology
[4] Science and Technical Research Laboratories, Japan Broadcasting Corporation

mkataigi@de.cs.titech.ac.jp, daik@de.cs.titech.ac.jp, yoshihara@de.cs.titech.ac.jp,
tkobaya@gsic.titech.ac.jp, taguchi.r-cs@nhk.or.jp, yokota@cs.titech.ac.jp

Abstract

The number of stored objects that should be targets of high throughput retrieval, such as multimedia stream objects, is increasing recently. To implement a high throughput storage system, striping technique using multiple disk drives are commonly used. However, the ordinary disk striping methods implemented in RAID 0, 3-5 have problems of the flexibility, extensibility, and quality of services (QoS). We have proposed the autonomous disk cluster to realize the flexible and extensible storage system by treating each target object as a unit for management. It also adopts the primary-backup technique for the object to satisfy required QoS. We have shown that the autonomous management for the object unit is effective. However, there is room for improving its throughput. In this paper, we consider an approach for importing a flexible striping technique into the autonomous disk system to provide high throughput. It is important to locate fragments of an object into a part of disk cluster appropriately with considering workload skews to derive the required throughput. We use access history to decide their location. We also propose a method to divide access load into fragmented primary and backup adaptively. The experimental results using PC cluster indicate that the proposed methods are effective.

1 Introduction

The amount of data stored in a computer system has been very large recently. To implement a reliable large-capacity storage system at reasonable price, the storage system is commonly composed a number of hard disk devices. However, the enlargement of storage size makes the storage management very complex. The management of the large storage system should be enough flexible with satisfying the required quality of services (QoS).

To tackle the problem, a number of approaches to utilize the computation resources in each storage devices for managing the large amount of data [4, 5, 7]. We have also proposed the autonomous disks with a number of methods of realizing the flexible data management functions to balance amount and workload among storage devices, tolerate disk failures, recover from the failures, change the cluster size online, and so on [6, 8, 11]. In autonomous disks, we have assumed that a target object is stored in a storage device to simplify the strategy and processes for these functions, and adopts the primary-backup method for an object to guarantee the QoS.

On the other hand, the number of stored objects, such as multimedia stream objects, which should be targets of high throughput retrieval, is increasing. At the same time, variety of the stored objects requires variety of retrieval throughput. If the required throughput is higher than a storage device limit, it cannot be obtained by storing the object into the storage device only.

To implement a high throughput storage system, striping technique using multiple disk drives are commonly used [3, 9, 10]. However, the number of disks for striping groups is fixed and the placement manner of fragments is static in the ordinary disk striping methods used for RAID 0, 3-5. Because it is difficult for them to provide different throughput for each objects and to reconfigure the cluster online, they have less flexibility and extensibility. Moreover, the parity calculation technique in RAID 3-5 cannot guarantee the QoS under failures and during system reconfiguration.

In this paper, we consider an approach for importing a

flexible striping technique into the autonomous disk system to provide high throughput. In other words, we try to apply the storage management functions of balancing workload and data amount, failure recovery and online cluster reconfiguration to striped objects fragmented into an appropriate number disks with dynamic placement.

However, it is difficult to enhance all functions of the autonomous disk system to be able to handle the striping objects at a time. We first try to locate fragments of an object into a part of disk cluster appropriately with considering workload skews to derive the required throughput. We use access history to decide their placement to balance the workload. We also propose a method to divide access requests to fragmented primary and backup adaptively. The experimental results using PC cluster indicate that the proposed methods are effective.

The rest of this paper is organized as follows. In Section 2, we describe terminology, the autonomous disk system and disk striping as preliminaries. We then explain how to apply flexible striping to autonomous disks in Section 3. The method to adjust access load between the primary and backup is described in Section 4. Section 5 reports experiments and discuss the results. Finally we sum up this paper and show future work in Section 6.

2 Preliminaries

2.1 Terminology

At first, we define the following terms:

Object An object is a semantical chunk of data such as a file or a record.

Access An access is a series of following operations. First, a client sends a single read or write request to the storage system. Next, system returns a preservation place for the object. Finally, the client transmits or receives a file to/from the preservation place.

Fragment A fragment is data that divides the object into the fixed length, and is a minimum unit of striping.

Subobject A subobject is a set of fragments of a same object placed in the same storage device.

2.2 Autonomous Disks

The autonomous disk system is a high functional parallel storage system we proposed [11]. It configures a disk cluster by connecting intelligent disk devices to network directly, and automatically balances the workload and data amount in the cluster. Failures in the cluster are also automatically handled by recovering data in the damaged disks. The cluster is easily reconfigured by adding or removing disks with keeping the QoS.

To realize these functions, the the autonomous disk system adopts a fat-Btree [12] and the primary-backup technique. The fat-Btree is a parallel Btree structure we proposed to provide update-conscious balanced access path for each distributed object. It is also useful for adjusting each object location to balance the workload and data amount. On the other hand, the primary-backup technique makes the system tolerate the disk failures with guaranteeing the QoS.

To simplify the strategy and processes for these functions, we have assumed that a whole target object is stored into a storage device in the autonomous disk system. We have demonstrated that the autonomous management for the object unit in each storage device is effective. However, there is room for improving its throughput to meet the requirement of high-throughput applications. In this paper, we try to import a disk striping technique into the autonomous disk system.

2.3 Disk Striping

The disk striping is the common technique to derive high throughput, such as in many types of RAID systems [3, 10]. In the ordinary disk striping technique, the number of disks for a striping group is fixed and the placement manner of the fragments is static in a storage system. The static placement manner within the fixed group size is not suitable for unifying the disk striping with the functions of autonomous disks.

When an arbitrary number of disks are added to the storage system or removed from it, many fragments need to be replaced in the system to satisfy the manner. If the system adopt the parity calculation technique in RAID 3-5 to tolerate disk failures, the system reconfiguration requires heavy re-calculation of the parity. Moreover, the parity calculation technique cannot guarantee the QoS under failures and during the reconfiguration.

There are many types of multimedia objects requiring individual throughput. It is also useful to balance the workload considering the different throughput requirement. If the system allow a dynamic placement manner within variable striping group size, the flexibility in managing different throughput for each object type increases. Moreover, the storage system can use disks having different performance properties while the ordinary striping assume a uniform performance property in the system.

A number of related striping methods has been proposed, such as the staggered striping [2] and striping with a pseudo-random hash function [1]. The staggered striping enable high-speed stable read access with different throughput for each object. However, it requires complete data reconstruction for handling failures and reconfigure system size because the data placement is fixed. The striping with pseudo-random hash function can reallocate the fragmented data, but the cost of the re-calculation of the hash function becomes high when the number of fragmentation is varied.

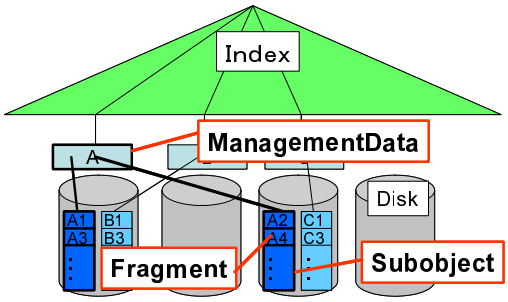


Figure 1. Subobjects and Management Data

3 Dynamic Placement

3.1 Management Data for Subobject

To import the flexible striping technique into the autonomous disk system, we first divide an object into subobjects, a set of fragments of the object placed in the same storage device. The number of subobjects is decided from the required throughput and performance properties of disks. We prepare a management data for each object to keep the information of the placement of the subobjects. The structure of subobjects and management data enables the dynamic placement within the striping group having variable number of disks. The parallel Btree structure is used as an index to provide an access path to the management data. Figure 1 illustrates the relationship between the indexed management data and subobjects consisting of fragments.

3.2 Placement of Subobjects

To balance access workload in the disk cluster, the placement of subobjects is critical. It is important to reduce the access concentration for subobjects store in the same disk drive.

The subobjects should be migrated between disks to reduce the concentration under the dynamic workload fluctuations. However, in this paper we assume the workload for each object is not dynamically change as the first step. We just decide the initial placement of each subobject using access frequency history. Figure 2 show the outline of the placement algorithm.

4 Backup Utilization

We inherit the primary-backup technique to the autonomous disks system with the flexible striping. The same structure of subobjects and management data is applied to the backup. It means that there is a restriction for the subobject placement. A backup subobject has not to be place the disk containing its primary. It guarantees the QoS under disk failures and reconfiguration of a striping group.

Moreover, the backup data can be use to improve the throughput for read access. In that case, we have to consider the workloads of both disks containing the primary subobject and its backup.

input: the number of subobjcet: n
the number of fragment: k
the number of disks: $M(> n)$
a list of access frequency for each storage device $AD[M]$
output: disk ID list to store fragments $f[k]$
Step1: select n disks from $AD[M]$, which have low access frequency and make a disk list $Subobjects[n]$
Step2: for $i \in [0, k]$, let $j = i \ mod \ n$ and $f[i] = Subobjects[j]$
Step3: Output $f[k]$ and terminate.

Figure 2. Subobject Placement Algorithm

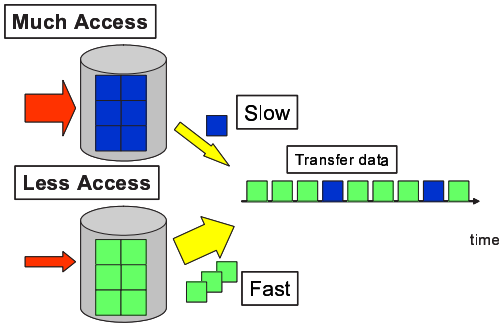


Figure 3. Primary-backup Read Ratio Adjustment

4.1 Read Time Minimization

If we adjust the ratio of dividing the read requests between the primary and backup, we can minimize the time for read time. Figure 3 illustrates the image of the adjustment. The read requests for a busy disk should decrease while these for a non-busy disk increase. Here, we assume that backups are up-to-date.

To minimize the read time, we first consider the following parameters:

- Read throughput of a disk containing primary: v_p
- Read throughput of a disk containing its backup: v_b
- Ratio for requiring primary: α
- Read time of primary subobject: T_P
- Read time of backup subobject: T_b
- Size of a subobject: S

The relationship between the size and time is as follows:

$$Primary \ : \ \alpha S = T_p \cdot v_p \quad (1)$$

$$Backup \ : \ (1-\alpha)S = T_b \cdot v_b \quad (2)$$

The whole reading time $T = max\{T_p, T_b\}$ becomes minimum when the primary read time is equal to the backup one.

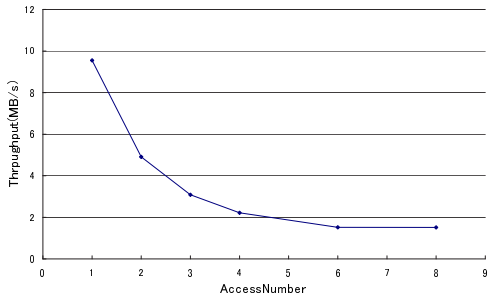


Figure 4. Connections vs. Read Throughput

Therefore, we can minimize the reading time by calculating

$$\alpha = \frac{v_p}{v_p + v_b}, \tag{3}$$

if v_p and v_b can be derived.

4.2 Connection based Adjustment

We expect that the number of connections for a disk system reflects its read throughput. We measured the throughput transition by changing the number of connections for a disk system in an ideal environment by inhibiting other accesses as a preliminary experiment. Figure 4 shows the result.

From the graph in Figure 4, we can suppose the inverse proportion between the number of connections and read throughput, which can be expressed by following equation:

$$v = \frac{k}{conneciton_number} \tag{4}$$

where k is a coefficient.

Based on the equation, we can derive the ratio using the number of connections for primary and backup.

- Connection number of primary storage device: L_p
- Connection number of backup storage device: L_b

From them, the primary read throughput v_p and the backup read throughput v_b are calculated as follows:

$$v_p = k/L_p \tag{5}$$
$$v_b = k/L_b \tag{6}$$

Under the assumption, the read request ratio for primary object R_p to minimize the read time is

$$R_p = \frac{L_b}{L_p + L_b}, \tag{7}$$

while it for backup object is $R_b = 1 - R_p$. They do not contain k.

input the number of connection for primary disk : L_p
the number of connection for backup disk : L_b
the number of fragment in a subobject: k
Disk ID which primary object exist : I_p
Disk ID which backup object exist : I_b
Read rate calculation function : $F(L_p, L_b)$
output List of reading fragments $List[k]$
Step1: The number of reading fragment of primary $S_p = F(L_p, L_b)$ and the number of reading fragment of backup $S_b = F(L_p, L_b)$ are calculated.
Step2: $pos = 0$
Step3: $List[i] = I_p$ for $i \in [pos, pos + S_p]$
Step4: $pos = pos + S_p$
if $pos > k$, go to Step7
Step5: $List[i] = I_b$ for $i \in [pos, pos + S_b]$
Step6: $pos = pos + S_b$
if $pos > k$, go to Step8
else if $pos \leq k$, go to Step3
Step7: output $List$

Figure 5. History Based Adjustment Algorithm

4.3 History based Adjustment

Because the preliminary experiment was done under the ideal environment, we cannot guarantee that the equation (4) is established in any environment. Here, we consider the other method using the access history.

We expect that the read throughput of storage devices does not change on the same condition. Therefore, the read throughput can be forecasted from the history of the combination of the number of connections and the read throughput for the previous read accesses. As the history, the following information is kept.

Beginning to Read The speed corresponding to the number of connections is acquired from the number of connections and the speed table.
End of Reading The connection and the average speed of reading at that time are written in the number of connections and the speed table.

The Figure 5 outline the algorithm using the history.

5 Experiment

We evaluate the dynamic placement within the striping group having variable number of disks and backup utilization with both the connection and history based adjustments by comparing the ordinary striping approach.

5.1 Environment

We implement these algorithms on a PC cluster. Table 1 shows the specification of the cluster.

Table 1. Performance of storage nodes

Number of Client nodes	4 nodes
Number of Disk nodes	8 nodes
CPU	AMD Athlon XP-M 1800+ (1.53GHz)
MEM	PC2100 DDR SDRAM 1GB
Network	1000BASE-T
HDD	TOSHIBA MK3019GAX (30GB, 5400rpm, 2.5inch)
OS	Linux 2.4.20
Local File System	reiser FS
Java VM	Sun J2SE 1.5.0_03

Table 2. Result of Positioning

	Average Throughput (MB/s)	Standard Deviation
Not striping	6.96	4.10
Random positioning	16.01	5.38
Access history	17.12	6.35

Each client reads or writes objects. At first, 100 objects is stored to make the initial set of objects. Then, the objects are read 200 times to derive the initial access history. After that, read and write access are randomly sent from the clients 2000 times in total under keeping the probability of write 0.02. The clients prepare all objects to be stored and randamly select objects to be sent from them based on Zipf distribution with the parameter setting $\theta = 0.9$. The size of each fragment is $8KB$ and the number of subobjects of an object is set to 3.

5.2 Effect of the Dynamic Placement

We verify the effect of the dynamic subobject plasement by comparing throughput for the following setting: 1) without striping, 2) striping with random subobject placement, and 3) striping with subobject placement based on access history. The results are shown in Table 2 and Figure 6. Figure 6 shows the transition of the average 100 times throughput that matches all clients.

Comparing the throughput without striping and that uses striping from Table 2, throughput has improved to more than twice by the effect of striping. Figure 6 also shows that the striping system achieves more than twice throughput than the system of no striping. This is because the eight storage devices in the maximum come to be used at the same time by doing striping while only four storage device in the maximum is used for reading at the same time in the method of not doing striping.

Comparing the subobject placement with the history and the use of random from Table 2 and Figure 6, we can observe that the former achieves higher throughput. Because the read performance of the storage device decreases due to the access concentration on a part of storage device by the

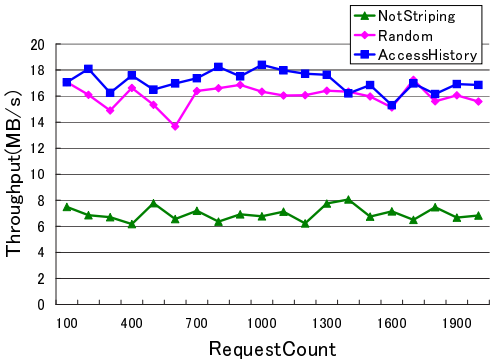


Figure 6. Throughput Transition by Placement

Table 3. Result of Backup Use

	Average Throughput (MB/s)	Standard Deviation
All Primary	17.12	6.35
P:B 1:1	21.87	8.02
Connection number	21.86	7.57
Access History	18.61	7.43

random subobjects placement. The reading performance of the average become low because these storage devices are often accessed. On the other hand, the read throughput of storage device using the access history does not decrease because the number of accesses across storage devices is equalized in this technique.

However, the difference of the read throughput increases as understood from the standard deviation of Table 2. This is because the throughput of each object is polarized to high throughput and low throughput in the technique that uses the access history. Therefore, it is necessary to specify the reason why throughput has been polarized, and to decrease the reading with low throughput for stabilization and further speed-up.

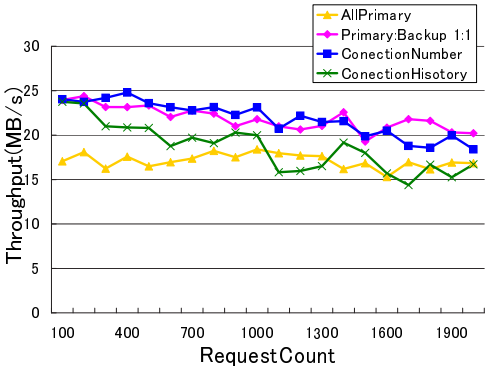


Figure 7. Throughput Transition by Backup Utilization

5.3 Effect of Backup Utilization

Next, we verify the effect of the backup utilization by comparing four setting: 1) reading fragments from only primary, 2) reading fragments from primary and backup at 1:1 ratio, 3) primary and backup at ratio settled based on the connection number, and 4) primary and backup at ratio settled by the access history.

A same request to the experiment with the positioning method is transmitted and the change of the average reading throughput for each object is measured. The result is as shown in Table 3 and Figure 7. Figure 7 shows the transition of the average 100 times throughput that matches all clients.

Throughput has been improved by reading backups. This is because the number of accesses to each client is equalized by the entire system by accessing the backup.

Moreover, when we compare the method of adjusting reading ratio to 1:1 between primary and backup and the method of deciding reading ratio by using connection number, the average throughput of the two methods is the same as shown in Figure 7 and Table 3. This is because influence of change of connection number and system process cannot be considered.

The reading throughput is steady by changing the reading rate. This is because low throughput reading decreased, though high throughput reading decreased by using the ratio of the number of connections too.

Throughput has not been improved by the method using access history, because all of the access history is used in this experiment. So it becomes difficult to reflect a present performance. Moreover, the throughput begins to decrease when the number of request is more than 200 as show in Figure 7. Therefore, we consider that throughput decrease is suppressed by setting the maintenance period to the history and deleting the old history.

6 Conclusions and Future Work

In this paper, we import the flexible striping technique into the autonomous disk system to improve its throughput with the rich management functions. We prepare a management data for each object to keep the information of the placement of the subobjects. We use access history to decide the placement of the subobjects to balance access workload in the disk cluster. We then propose methods to utilize the backup subobjects by adjusting the access ratio using connection number and access history to improve the throughput with balancing workload. The experimental results using a PC cluster indicate that the proposed methods are effective.

As future work, we plan to do more experiment to evaluate the dynamic subobject placement with different parameters. The data amount balance with the dynamic subobject placement keeping the workload balance is also an important issue to be attacked. We have to consider the situation of dynamic workload change. For that situation, we have to consider the cost and effect of the subobject migration. Moreover, we have to consider the influence of buffering effect with the semiconductor memory.

Acknowledgements

This work is partially supported by the JST CREST, SRC, MEXT of Japanese Government via Grant-in-Aid for Scientific Research #16016232 and the MEXT 21st Century COE Program.

References

[1] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered striping in multimedia information systems. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 79–90, New York, NY, USA, 1994. ACM Press.

[2] A. Brinkmann, K. Salzwedel, and C. Scheideler. Efficient, distributed data placement strategies for storage area networks. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 119–128, 2000.

[3] P. M. Chen, E. L. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. Raid: High-performance, reliable secondary storage. *ACM Comput. Surv.*, 26(2):145–185, 1994.

[4] S. Frølund, A. Merchant, Y. Saito, S. Spence, and A. Veitch. FAB: enterprise storage systems on a shoestring. In *HOTOS 2003*, Kauai, HI, May 2003.

[5] G. R. Ganger, J. D. Strunk, and A. J. Klosterman. Self-* storage: Brick-based storage with automated administration. Technical Report CMU-CS-03-178, Carnegie Mellon University, Aug 2003.

[6] D. Ito and H. Yokota. Automatic reconfiguration of an autonomous disk cluster. In *Proc. of 2001 Pacific Rim International Symposium on Dependable Computing (PRDC 2001)*, pages 169–172. IEEE, Dec. 2001.

[7] J. MacCormick, N. Murphy, M. Najork, C. A. Thekkath, and L. Zhou. Boxwood: Abstractions as the foundation for storage infrastructure. In *OSDI*, pages 105–120, 2004.

[8] M. Nakano, D. Kobayashi, A. Watanabe, T. Uehara, R. Taguchi, and H. Yokota. The versioning system balancing data amount and access frequency on distributed storage system. In *ICDE Workshops*, page 1264, 2005.

[9] D. A. Patterson, G. A. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (raid). In H. Boral and P.-Å. Larson, editors, *SIGMOD Conference*, pages 109–116. ACM Press, 1988.

[10] K. Salem and H. Garcia-Molina. Disk striping. In *Proceedings of the Second International Conference on Data Engineering*, pages 336–342, Washington, DC, USA, 1986. IEEE Computer Society.

[11] H. Yokota. Autonomous disks for advanced database applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pages 441–448, Nov. 1999.

[12] H. Yokota, Y. Kanemasa, and J. Miyazaki. Fat-btree: An update-conscious parallel directory structure. In *Proc. of the 15th Int'l Conf. on Data Engineering*, pages 448–457, 1999.