Information Engineering Express
International Institute of Applied Informatics
2017, Vol.3, No.4, P.91 - 104

A Dynamic Configuration Scheme of Storage Tiers for an Autonomous Distributed Storage System

Shingo Shimano [*] , Atsushi Nunome [†] , Yuta Yokoi [‡] ,
Kiyoshi Shibayama [§] , Hiroaki Hirata [¶]

Abstract

We have proposed a distributed storage system which dynamically makes storage tiers and optimizes location of data blocks autonomously. This aims to enhance the I/O performance of the storage system without remarkable network overhead. Our system dynamically organizes storage tiers considering device characteristics. And the data blocks will be placed in a suitable storage tier according to their access pattern.

In this paper, we propose a method to select the destination storage node for migration using an access characteristic of a data block to be migrated. This method dynamically ranks the storage nodes, and autonomously configures storage tiers based on relative performance to the migration initiator. Simulation results show that our scheme can shorten the execution time of a program with file I/O by 80% at maximum, in comparison with the static migration without considering the access characteristics of the migration data.

Keywords: block relocation, distributed storage system, storage tiering

1 Introduction

With the advance of storage technology, many of recent computers, even client PCs, have been equipped with an adequate amount of storage area. However, not all of the storage area are not completely used up. We have proposed a distributed storage system which aims to make the most of such unused storage area by sharing them between the computers[1][2][3].

Our system is designed for distributed environment which consists of heterogeneous computers. In this system, we assume that each computer has various types of storage devices, which have diverse performance characteristics. This divergence might be not original system's intention, but a result of storage components' partial replacement. We consider that such heterogeneity of the storage device is brought from not only the difference of the device type, but also the disparity of access frequency between the storage nodes.

[*] Kyoto Institute of Technology, Kyoto, Japan
[†] Kyoto Institute of Technology, Kyoto, Japan
[‡] Kyoto Institute of Technology, Kyoto, Japan
[§] Kyoto College of Graduate Studies for Informatics, Kyoto, Japan
[¶] Kyoto Institute of Technology, Kyoto, Japan

Therefore, data blocks should be dynamically relocated on the suitable storage device according to their access patterns in order to utilize storage devices effectively. This aims to bring out the best I/O performance over the whole system. To achieve this intention, each network node gathers the information on storage utilization in the storage system, and migrates data blocks into the proper storage node. Our autonomous control mitigates the administrators' annoying tasks.

Our previous work in [4] presented an autonomous configuration scheme of storage tiers. However, it has a problem that the parameter using for defining the storage tiers is hard to tune for various running environments. And its decision procedure has been remained as a future work. In this paper, we enhance the storage tiering method by introducing a relative threshold value which is easy to define for various environments and also show the effectiveness of our scheme by simulation.

2 Autonomous Distributed Storage System

In this paper, we classify network nodes into the following two types, *Client node* and *Storage node*, in our target environment. A client node is the node which mounts a remote storage to use. A storage node has a public storage area, and provide it for remote client nodes.

In a distributed storage system, we assume that various types of storage devices are attached. For example, HDD and SSD are general classification for the storage, and also they are further categorized by various technologies into a number of performance grade[5]. However, heterogeneity might be brought by not only the difference of such storage technology but also the performance drop from access concentration. The information about such heterogeneity occurred during operation should be shared among the network nodes at run-time.

In order to reduce network overhead, we have also proposed the scheme to merge the storage information into an usual iSCSI packet. In our scheme, the storage information is appended to an Ethernet frame including iSCSI packet up to the size of the maximum transfer unit (MTU). Thus, we can reduce the number of dedicated Ethernet frames for the propagation of the storage information. Our previous investigations[2] show that there is potentially sufficient available space in the Ethernet jumbo frames which contain iSCSI packets.

In our previous scheme, plural performance indicators (e.g. read and write performance) have been dealt without any weighting. However, because of the ratio of read/write varies from application to application, it is difficult to decide the best location of data blocks without consideration of the difference of the storage characteristics. Hence, the data blocks should be located on the suitable storage device according to their access properties. For instance, a data block which is read frequently should be located on the storage with high read performance.

3 Dynamic Configuration Scheme of Storage Tiers

In this section, we describe the details of a scheme to configure the storage tiers autonomously.

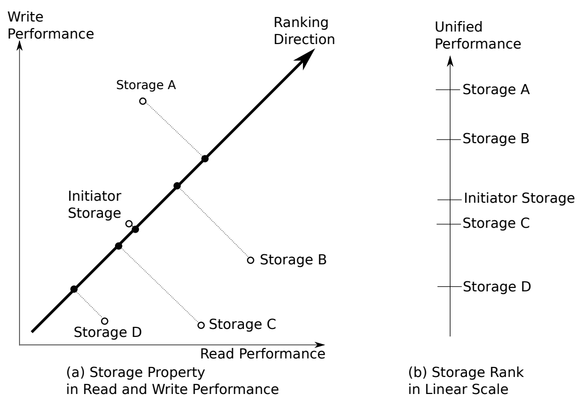

Figure 1: An Example of Storage Ranking.

3.1 Overview

In the autonomous distributed storage system, the storage tiers are dynamically constructed based on the value of storage performance collected in run time. This scheme mainly consists of following two steps, *storage ranking* and *storage tiering*. The former ranks the storage nodes in the order of the effective performance of them. In this operation, we refer the both of read and write performance value in order to respect the application's behavior. The latter is a method for tiering the storage nodes, which aims to avoid a useless migration.

3.2 Storage Ranking Method

Our proposed scheme ranks the storage nodes in order the one-dimensional performance value that is produced as the summary of the read and write performance. This unified performance value P can be calculated by Equation (1):

$$P = r \cdot p_r + (1-r) \cdot p_w \tag{1}$$

where p_r and p_w are read and write performance respectively, $r \in [0,1]$ denotes the ratio of read operation, i.e. $(1-r)$ represents the ratio of write operation.

For example, we show an example of ranking five storage nodes in Figure 1. Five storage nodes, A, B, C, D and initiator, are plotted in Figure 1(a). The initiator storage attempts to migrate a data block to one of the other storage nodes. In Figure 1(a), the vertical axis shows write performance, and the horizontal axis shows read performance. If the frequency of the read and write operation to a data block is identical, slope of the ranking direction will be one. The foot of a perpendicular from a point of the storage node to the ranking direction shows the unified performance of the storage.

A foot of a perpendicular (P_r, P_w) from the point of the storage node (p_r, p_w) will be calculated by Equations (2) and (3):

$$P_r = \frac{a \cdot p_w + p_r}{a^2+1} \tag{2}$$

$$P_w = \frac{a(a \cdot p_w + p_r)}{a^2+1} \tag{3}$$

Here a denotes the sloop of the ranking direction. Therefore, the unified performance

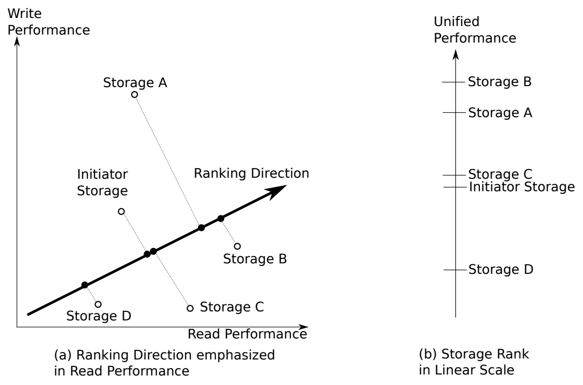

Figure 2: An Example of Storage Ranking Emphasized in Read Performance.

value P will be Equation (4):

$$P = \sqrt{P_r^2 + P_w^2} \tag{4}$$

Figure 1(b) shows the ranks of the storage nodes in a one-dimensional scale, which is made in the aforesaid manner. It also shows that the storage nodes that have synthetically higher performance in both read and write operation are located at upper position. In Figure 1(b), the two storage nodes, A and B, are considered to have higher performance than the initiator storage node.

Next, we investigate a situation which the initiator storage node attempts to migrate a data block which has intensive read operations. Figure 2 shows that the initiator storage node plans to migrate a data block which has double read frequency than write. In this case, we should choose a storage node which has superior read performance as the destination of the migration. In Figure 2(a), slope of the ranking direction is set at 0.5. Therefore, the rank of the storage node shown in Figure 2(b) differs from the rank in Figure 1(b). Especially, in Figure 2(b), the storage C that has higher read performance and is ranked under the initiator storage node in Figure 1(b) is ranked upper than the initiator. And the storage B instead of the storage A is ranked at the top of the unified performance scale. Thus, by adjusting the slope of the ranking direction, the destination of the migration can be selected appropriately.

3.3 Storage Tiering Method

In this section, we describe a method to tier the storage nodes. First, we discuss a threshold value which is used for tiering the storage nodes. This threshold can be made as an *absolute* value or a *relative* value.

Absolute Threshold We can construct static storage tiers by using common absolute threshold in system wide. However, a useless data migration may occur between storage nodes which have an insignificant performance difference. For example, in Figure 2(b), if the threshold is set at between the storage C and the initiator storage, the initiator storage recognizes the storage C as the storage in the upper storage tier. We should prevent frequent data migrations between the storage nodes that have such almost identical performance.

Relative Threshold We can dynamically construct storage tiers by using relative threshold every data migration. In this method, a storage node which is on an almost same level of performance is considered as a storage in the same storage tier. On the other hand, extra task for tiering the storage nodes will be needed every data migration.

To reduce the number of migrations, the proposed scheme tiers the storage nodes by using the relative threshold. The relative threshold parameter t is defined statically in consideration of migration overhead. Because the performance value of storage nodes could be widely varied according to the running environment, we define the value of $t(0 \leq t < 1)$ as the ratio to the performance of the migration source node rather than the differences of the performance value between storage nodes. Therefore, two threshold values T_h and T_l can be dynamically defined by Equations (5) and (6).

$$T_h = P(1+t) \tag{5}$$

$$T_l = P(1-t) \tag{6}$$

Here P is the unified performance value of the initiator storage node, which is calculated by Equation (4). The initiator storage regards the storage node having the performance exceeding the threshold value T_h as belonging to the upper storage tier and regards the storage node having the performance less than the threshold T_l as belonging to the lower storage tier. In this manner, at the time of block migration, the initiator storage recognizes the three storage tiers, i.e. the upper, lower, and the same tier relative to the initiator. The number of the storage tiers created by the initiator node is fixed and it is always three. If t is zero, both of the parameters T_h and T_l are equal to P. So, the initiator storage begins a migration to the destination node even if it has almost the same performance.

3.4 Data Migration Procedure

In this section, we describe the details of a procedure of data migration.

3.4.1 Data Migration to Upper Storage Tier

A storage node which has a heavy workload acts as an initiator storage and migrates some data blocks to a storage node in the upper storage tier according to the following steps.

i. On the initiator storage S_i, a data migration will be initiated when queue length for read and write access exceeds a threshold.

ii. From the table recording the access frequency on the node S_i, data block with high access frequency is selected as the migration data.

iii. The node S_i ranks other storage nodes with the unified performance value calculated by the method described section 3.2.

iv. A node group having a performance value higher than the threshold value T_h from the initiator storage is regarded as an upper storage tier. In order to moderate excessive concentration of the data migration to a few higher performance node, the lowermost node S_d in the upper storage tier is set as the migration destination.

v. The node S_i initiates migrating a data to the node S_d.

Table 1: Simulation Parameters.

Read Write Ratio (R:W)	A ratio of read and write operation on a client node.
Operation Interval (T_i)	Interval time from the time of read or write finish to the time of the next command issue in microseconds.
Read Latency (L_r)	A time delay from receiving of read command to sending a reply of finish in microseconds.
Write Latency (L_w)	A time delay from receiving of write command to sending a reply of finish in microseconds.
Network Latency (L_n)	A time delay from receiving a packet to sending the packet in a network switch.
Selection Time (T_d)	Time to select the migration destination.
Merging Time (T_m)	Time to merge storage information into a packet.

3.4.2 Data Migration to Lower Storage Tier

When the amount of storage used is nearing its maximum capacity in a storage node, the node will migrate some data blocks in order to keep certain free space.

i. On the initiator storage S_i, a data migration will be initiated when the used storage capacity exceeds statically defined maximum storage usage.

ii. From the table recording the access frequency on the node S_i, a data block with least recently accessed is selected as the migration data.

iii. The node S_i ranks other storage nodes with the unified performance value calculated by the method described section 3.2.

iv. A node group having a performance value lower than the threshold value T_l from the initiator storage is regarded as a lower storage tier. For the purpose of gradually migrating data to a lower performance storage, the uppermost node S_d in the lower storage tier is set as the migration destination.

v. The node S_i initiates migrating a data to the node S_d.

4 Experimental Evaluation

4.1 Condition of the Simulation

To evaluate the effectiveness of our proposed scheme, we make experiments in a simulation environment. The environment is designed to simulate a distributed storage system consists of ten storage nodes and m client nodes. Each storage node has a public storage area which can be accessed from all client nodes.

Simulation parameters are listed in Table 1. Though the parameter T_d will vary based on the number of storage nodes, we use fixed value because the number of storage nodes is invariant in the simulation environment. We measured the latency time on real system and used them as the parameters in the simulation. First, we measured the parameter L_n on Corega CG-SSW08GTR network switch which has wire-speed forwarding capability. We also measured latency for packet transmission and in the network switch by a TCP/IP packet exchange program written in C language.

Table 2: Node Configuration.

Node Type	OS	Processor	Memory
Client Node	FreeBSD 8.3 (amd64)	AMD Athlon X2 4450e	2GB DDR2
Storage Node	FreeBSD 10.3 (amd64)	Intel Xeon E5-2640v3	8GB DDR4

Table 3: Actual Measured Throughput for Sequential Read and Write.

Type	Model	Interface	Read [KB/s]	Write [KB/s]
SSD	Intel SSDSC2BW480H6	SATA3	451,304	500,600
HDD	Seagate ST3400620NS	SATA2	70,413	63,418

And we also measured execution time for determining the migration destination on a real client node. The configurations of the client and storage node are shown in Table 2. We measured the average time to process the determination of the destination by averaging the 10^6 executions of code fragments written in C. The processing time for merging storage information to a packet is measured by averaging the 10^6 executions of code which adds the information structure onto a packet. By conducting from the results of the preparatory experiment, we fix the parameter values as follows, $L_n = 4.0$ [μs], $T_d = 4.0$ [μs] and $T_m = 0.4$ [μs].

Next, we measured I/O performance of two types storage devices by Bonnie++[6] version 1.97.3 benchmark program. Table 3 shows results of sequential read and write throughput of them. Both of the storage devices are formatted in ZFS[7] file system.

Table 4 shows the configurations of latency for read and write operations and initial unused capacity in ten storage nodes. In Table 4, the latency values of the storage node S1 and S2 have been measured by Bonnie++ on the aforementioned storage node. The storage devices of the node S1 and S2 are HDD and SSD, respectively. The latency values of the other storage nodes have been configured as variants of performance based on the measured values. The node S3 is superior to the node S1 in writing. Although the nodes S4, S5, S6, S9 and S10 are inferior to the node S1 in reading, they have less latency as compared with the node S1 in writing. The nodes S7 and S8 are configured as intention of the cheap edition of the node S1. The latency values shown in Table 4 are statically defined performance indicator of the storage devices. The actual latency will change owing to the access to the migrated data block. Therefore, we cannot use static latency value for storage device evaluation, and it is necessary to exchange the storage information periodically and frequently. In this simulation, each storage node sends the latest latency to a client node by appending the latency information to the packet which is used for data access. Thus, the

Table 4: Latency Time and Initial Unused Capacity of Storage Nodes.

Storage Node	Read [μs]	Write [μs]	Capacity [MB]	Storage Node	Read [μs]	Write [μs]	Capacity [MB]
S1	16	1,136	4,070	S6	32	960	3,500
S2	12	88	350	S7	20	104	490
S3	16	200	750	S8	40	180	780
S4	40	400	1,550	S9	100	400	1,760
S5	80	320	1,410	S10	120	320	1,550

Table 5: Parameter Configurations of Client Node.

Client	R		W		E	
Pattern	R:W	$T_i[\mu s]$	R:W	$T_i[\mu s]$	R:W	$T_i[\mu s]$
1	95:5	4	5:95	4	45:55	4
2	1:0	4	0:1	4	55:45	4
3	1:0	8	0:1	8	55:45	8
4	1:0	12	0:1	12	55:45	12
5	95:5	8	5:95	8	45:55	8

storage nodes can share the effective latency with the other storage nodes through the client nodes. The values of the initial unused capacity shown in Table 4 are configured as small in order to confirm the effectiveness of data migration. Each storage node migrates data in units of ten megabytes. So, the time of single data migration is set at 200 milliseconds based on the average time measured in 1000BASE-T network.

Measurement conditions about the read and write ratio (R:W) and the number of client nodes (m) are set up as the following configurations.

First, the parameter R:W is configured as either the read frequency or the write frequency is higher than the other. The condition which has higher read frequency is called *R*, and the condition which has higher write frequency is called *W*. The condition which has almost even read and write frequency is called *E*.

The details of the parameter conditions are listed in Table 5. We evaluated the effectiveness of the dynamic migration by configuring the operation interval short to conflict the access to the distributed storage system.

Second, we also evaluated our scheme by giving variety the total number of client nodes to 50, 100 and 200 (hereinafter referred to as C50, C100 and C200, respectively). Five parameter patterns shown in Table 5 are evenly used for the all clients at each configuration.

Third, to evaluate the effect of the threshold parameter t, we also varied it between 0 and 0.5. For example, the case of $t = 0.5$ means that an initiator storage will perform a data migration to a target storage only when the target's performance differs over 50% from the initiator's.

We performed a simulation of the following two migration schemes using these parameters.

(a) Static Migration It statically ranks the storage nodes without any weighting the read and write performance. A data to be migrated is always assumed to be equal in the read and write frequencies.

(b) Dynamic Migration It dynamically ranks the storage nodes with weighted performance value, according to the read and write frequencies of a data block to be migrated.

Each client node issues total 100,000 times requests for reading or writing data. We measured the time for completing all read/write operations in each parameter configuration.

The triggers for data migration differ among the target tier of migration. The migration to the upper tier is initiated for reducing latency of storage access. On the other hand, the migration to the lower tier is initiated to meet the demand for storage space. The migration to the upper storage tier will be initiated when the estimated waiting time which is calculated with the waiting queue length and averaged read/write latency exceeds one millisecond. This trigger is defined as the time of about quadruple of the average read/write latency time

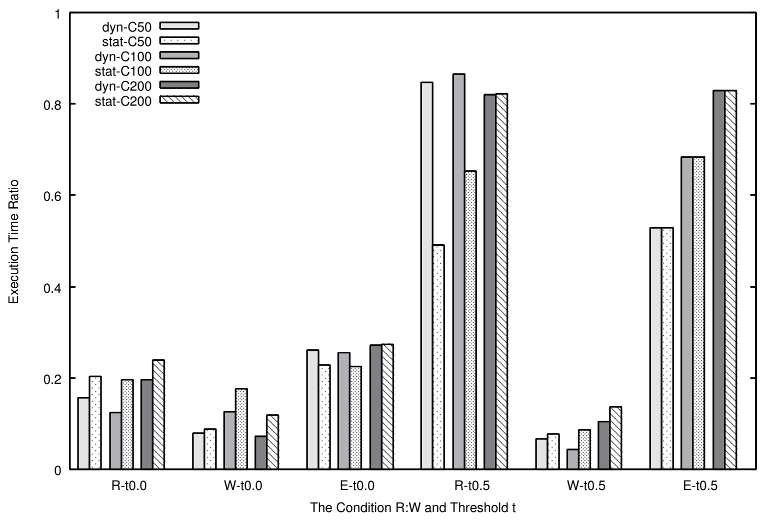

Figure 3: Execution Time Ratio of the Data Migration Schemes to the No Data Migration Scheme.

of ten storage nodes, from S1 to S10 shown in Table 4. We set this value as slightly short in order to occur enough number of times of migrations by attempting data migration actively. And the migration to the lower storage tier will be initiated when the size of free storage space is below 30% of the total storage space. Reference [8] reported that the mean space usage of 10,568 file systems of 4,801 Windows PC in a commercial environment is only 53%. And reference [9] also reported that 60% of the disks had more than 30% unused capacity and almost all the disks had at least 10% unused capacity, by investigating the 242 disks installed on the 22 public servers. Hence, we consider a storage node which has less than 30% unused capacity to be a relatively less space storage node, and data migration will be initiated in order to enlarge its free space.

4.2 Results and Considerations

Figure 3 shows the ratio of the execution time between the migration schemes (both of dynamic and static) and the scheme without data migration, in the conditions $t = 0$ and $t = 0.5$. The condition $t = 0$ is the most aggressive policy for migrating data, and contrary the condition $t = 0.5$ is the most conservative policy. In this figure, "dyn" and "stat" denote the type of data migration, namely, the dynamic migration scheme and the static migration scheme respectively. And C50, C100 and C200 denote the environment of 50, 100 and 200 client nodes respectively. The labels on the horizontal axis denote the combination of the parameter R:W and the threshold t. For example, "R-t0.0" means that the parameter R:W is set as R and the threshold t is set as 0. The execution time ratio is defined as T_{dyn}/T_{nomig} or T_{stat}/T_{nomig}. Here T_{dyn}, T_{stat} and T_{nomig} denote the average completion time of the read/write operations by the dynamic, the static migration and the no migration respectively. Since the average completion time of read/write operations by the no data migration scheme is set as one, a result less than one shows that the performance is improved by data migration. As shown in Figure 3, both the dynamic and the static migration scheme improve perfor-mance regardless the configuration of the threshold t. This shows the effectiveness of data migration.

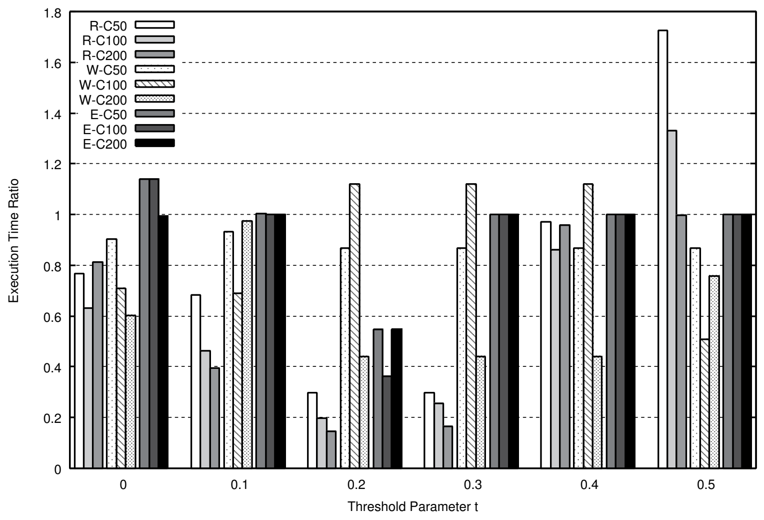

Figure 4: Execution Time Ratio of the Dynamic Migration Scheme to the Static Migration Scheme.

Figure 4 shows the comparison results between the dynamic and the static migration scheme. In Figure 4, the execution time ratio is defined as T_{dyn}/T_{stat}. Each result is named as the concatenation of the R:W condition and the number of client nodes. For example, the result “R-C50” denotes the result which is evaluated under the condition R using with 50 client nodes.

Figure 5, 6 and 7 show the number of migrations under the condition R, W and E respectively. In these figures, “dyn” and “stat” denote the type of migration, namely, the dynamic migration scheme and the static migration scheme respectively.

In the condition R, we found that the dynamic migration is superior, especially at $t = 0.2$ or 0.3. The execution time is shortened about 80% in the condition R-C200. When the parameter $t = 0.5$, the execution time of the dynamic migration is inferior to the static migration in the conditions R-C50 and R-C100. Figure 5 shows that dyn-t0.5 is considerably shorter than stat-t0.5 in the conditions C50 and C100. This means that a storage node has not been able to find a proper destination which has 50% superior read performance. On the other hand, in the static migration scheme, not completely best migration may not be occurring because the storage performance is evaluated by blending the read and write performance equally. However, there is a possibility that some storage nodes have migrated data blocks to the node other than the best destination instead of cancellation of the migration.

In the condition W, the dynamic migration scheme has clearly shortened the execution time under only the condition W-C200. As shown in Table 4, there are large differences between the absolute values of read and write latency time. When both of the read and write performance are evaluated at equal weight in the static migration, the unified performance is quite influenced from the write performance. Hence, in the condition R, this prevents proper selection of the destination of data migration because an initiator node assesses the write performance which is not so important. Meanwhile, in the dynamic migration scheme, large write latency is evaluated overemphasized in the condition W. However, small write latency is evaluated as almost equal to the write latency by the static migration scheme. In the conditions which have a small number of client nodes, such as W-C50 and W-C100,

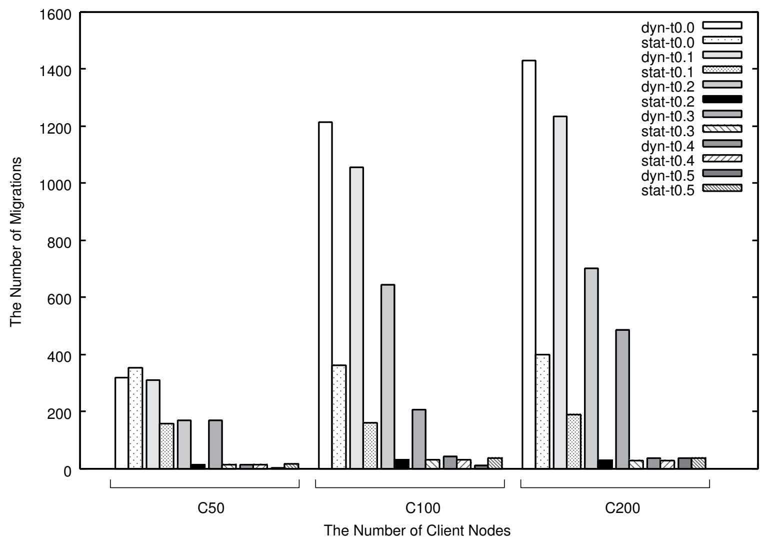

Figure 5: The Number of Migrations (Condition R)

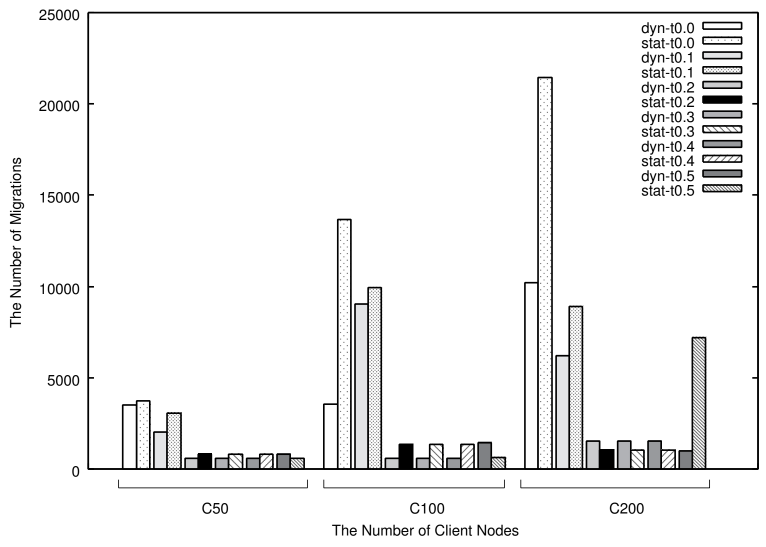

Figure 6: The Number of Migrations (Condition W)

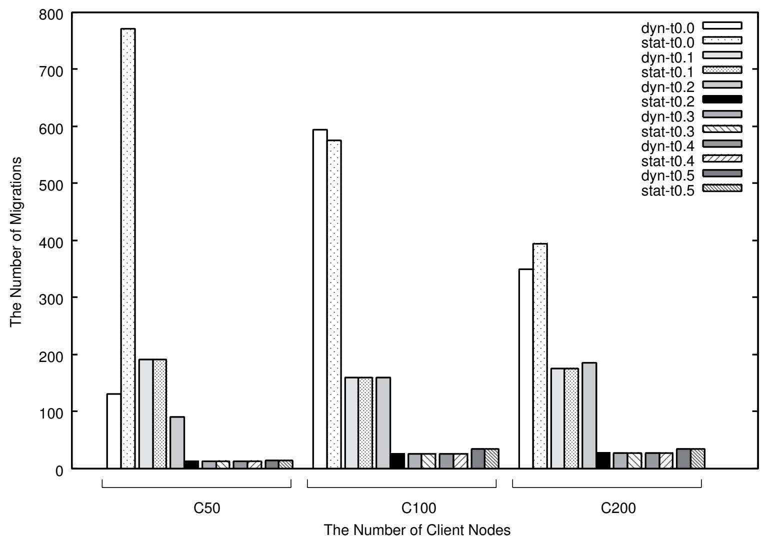

Figure 7: The Number of Migrations (Condition E)

the dynamic migration scheme will not be fully effective because the number of migrations will keeps low as the conditions C50 and C100 shown in Figure 6. On the contrary, data blocks will not be adequately distributed among the storage nodes on account of avoiding of the migration to the lower storage tier.

In the condition E, the execution time of the dynamic migration scheme are almost same as the static migration scheme, except at the condition $t = 0.2$. Figure 7 shows that the number of migrations in the conditions $t \geq 0.3$ are kept low. This means that many initiator nodes have few adequate destinations of data migration. In the condition $t = 0.2$, the dynamic migration scheme has performed over six times migrations as many as the static migration scheme. This shows that the dynamic migration scheme has been able to find the appropriate destination node by evaluating the actual performance of the other storage nodes precisely. Particularly, as shown in Figure 4, the result of the condition E-C100 is superior than those of E-C50 and E-C200. In the two conditions E-C50 and E-C200, by the dynamic migration scheme, we found that many data blocks converge rapidly on the storage node S2 or S7 which are relatively high performance. Because there are no nodes which have better performance than those nodes, further migrations have not executed even though there are many demands for migration to the upper storage tier, and the performance improvement has peaked. On the one hand, because there have been no such immoderate convergence of data blocks to the high performance storage node, the restriction of migration has not been observed in the condition E-C100. To avoid this, when the highest performance storage node acts as an initiator node in order to intend to migrate data to the upper storage tier, as the second best policy, it should migrate data to the same level of storage tier.

Our simulation results in Figure 4 show that the condition $t = 0.2$ is relatively better than the other configurations of the parameter t. If the conditions $t < 0.2$, an initiator node aggressively migrates data blocks to seek to gain a little performance improvement in spite of taking much migration cost. Especially in the dynamic migration scheme, an initiator node tends to excessively refrain or conduct data migrations in consequence of a measurement error of the storage performance. To prevent such unstable behavior, we should configure a certain amount of threshold t. On the other hand, when the threshold t is too large, it may miss the chance of data migration. Experiment results show that $t = 0.2$ is adequate for this simulation environment. However, in the runtime environment which has other configurations, we may need to make fine adjustments to the threshold t because the optimal value of the parameter might be different.

5 Related Work

Easy Tier[10] and *FAST* (Fully Automated Storage Tiering)[11] are commercial storage tiering systems implemented for storage server products. They support up to three storage tiers which are statically classified by device technologies, and the effective performance of the storage is not evaluated at the timing of storage tiering. *Btier*[12] is a block device with automatic migration for Linux kernel. The number of tiers can be controlled manually in accord with performance differences in the storage devices. Actually, most systems use only two tiers such as an SSD tier and an HDD tier in order to save time and effort of administrators. *OTF-AST* (On-The-Fly Automated Storage Tiering)[13] performs automated data migration between predefined two storage tiers, SSD and HDD. Lipetz et al.[14] proposed an automated tiering mechanism which maps the user data to three storage tiers, SSD, performance oriented HDD and inexpensive HDD. Though user data may be able to locate on an appropriate tier because of finer tiering, the design of storage tiers is dependent on the administrators' skill.

Because they are designed for a stand-alone storage server, they have not focused the dynamic heterogeneity of storage. Moreover, because they only give consideration to a few storage tiers which is defined statically, they are interested in picking up hot data blocks rather than choosing the destination of migration.

6 Conclusion

In this paper, we proposed a dynamic configuration scheme of storage tiers for an autonomous distributed storage system. Our scheme rates the performance of storage nodes by a dynamic scale according to the access characteristics of a data block to be migrated and selects an appropriate node as a destination of data migration. The simulation results show that the proposed scheme can short the execution time, especially in read intensive applications. The results also show that the scheme can raise performance by choosing a storage node which has an adequate difference of performance, as the destination of migration.

We plan to build some types of experimental systems in imitation of real environments, and evaluate our proposed scheme in more detail.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 15K00169 and 15K00076.

References

[1] A. Nunome, H. Hirata, and K. Shibayama, "A Distributed Storage System with Dynamic Tiering for iSCSI Environment," *Int'l Journal of Networked and Distributed Computing*, vol. 3, no. 1, pp. 42–50, Jan. 2015.

[2] S. Shimano, A. Nunome, H. Hirata, and K. Shibayama, "An Information Propagation Scheme for an Autonomous Distributed Storage System in iSCSI Environment," *Proc. the 3rd Int'l Conf. Applied Computing and Information Technology (ACIT 2015)*, Jul. 2015, pp. 149–154.

[3] A. Nunome, H. Hirata, and K. Shibayama, "An Interval Control Method for Status Propagation in an Autonomous Distributed Storage System," *Proc. the 15th IEEE/ACIS Int'l Conf. Computer and Information Science (ICIS 2016)*, Jun. 2016, pp. 723–728.

[4] S. Shimano, A. Nunome, Y. Yokoi, K. Shibayama, and H. Hirata, "An Autonomous Configuration Scheme of Storage Tiers for Distributed File System," *Proc. the 18th IEEE/ACIS Int'l Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2017)*, Jun. 2017, pp. 453–458.

[5] D. Anderson, J. Dykes, and E. Riedel, "More than an interface — SCSI vs. ATA," *Proc. the 2nd USENIX Conf. File and Storage Technology*, Mar. 2003, pp. 245–257.

[6] R. Coker, "Bonnie++," http://www.coker.com.au/bonnie++/experimental/.

[7] OpenZFS Project, "OpenZFS," http://www.open-zfs.org/.

[8] J.R. Douceur and W.J. Bolosky, "A Large-Scale Study of File-System Contents," *Proc. the 1999 ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems*, May 1999, pp. 59–70.

[9] H. Huang, W. Hung, and K.G. Shin, "FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption," *Proc. the 20th ACM Symp. Operating Systems Principles (SOSP '05)*, Oct. 2005, pp. 263–276.

[10] B. Dufrasne, B.A. Barbosa, P. Cronauer, D. Demarchi, H.-P. Drumm, R. Eliahu, X. Liu, and M. Stenson, *IBM System Storage DS8000 Easy Tier*. IBM Corp., 2013, available at ibm.com/redbooks.

[11] "EMC VNX FAST VP A Detailed Review," http://www.emc.com/collateral/software/white-papers/h8058-fast-vp-unified-storage-wp.pdf, EMC Corp., Dec. 2013.

[12] "btier," http://sourceforge.net/projects/tier/.

[13] K. Oe, T. Nanri, and K. Okamura, "On-The-Fly Automated Storage Tiering with Caching and both Proactive and Observational Migration," *Proc. the 3rd Int'l Symp. Computing and Networking*. IEEE, Dec. 2015, pp. 371–377.

[14] G. Lipetz, E. Hazan, A. Natanzon, and E. Bachmat, "Automated Tiering in a QoS Environment using Coarse Data," *Proc. 2013 IEEE 10th Int'l Conf. High Performance Computing and Communications & 2013 IEEE Int'l Conf. Embedded and Ubiquitous Computing*. IEEE, Nov. 2013, pp. 1022–1030.