An Improvement of Migration Efficiency in a Distributed Storage System with Dynamic Tiering

Atsushi Nunome
nunome@kit.ac.jp

Hiroaki Hirata
hrt@kit.ac.jp

Faculty of Information and Human Sciences, Kyoto Institute of Technology, Japan

***Abstract*—We have proposed a distributed storage system which autonomously migrates a data block into a suitable storage node. In our storage system, heavily accessing data blocks will be migrated autonomously to a higher performance storage node (i.e. an upper storage tier), and rarely accessed data blocks will be migrated to a lower performance storage node (i.e. a lower storage tier). Our investigation shows that it may need several migrations in order to locate a data block onto the most suitable storage node when the destination storage tier has a lot of storage nodes. In this paper, we propose a method to enhance data migration efficiency to improve I/O performance of the whole system. Our simulation results show that the proposed method can greatly reduce I/O execution time than the previous one. The execution time was reduced about one-fifth at the maximum in a system with several hundred scale nodes.**

***Keywords*—distributed storage system; storage tiering; block relocation.**

I. Introduction

Recent years, even client nodes can easily have enough storage area owing to the spread of inexpensive storage devices. In many cases, the large amount of an unused storage area is left on storage devices over the network. It is important to utilize such fragmented unused storage area. Hence, we have for our object to build an environment which not only file servers but also client nodes can provide their surplus storage area as public storage. However, such computing environment has essentially heterogeneity. We consider that system heterogeneity is brought by even a dynamic aspect. It means that both a static aspect (e.g. differences of hardware configuration) and a dynamic aspect (e.g. imbalance of I/O workload) make such heterogeneity. Therefore, data blocks should be located on a suitable storage device according to their access patterns. For example, frequently accessed data blocks are better to locate on high-performance storage, and it may cause no inconvenience when rarely accessed data blocks are located on low-performance storage. However, such location optimization in accordance with access patterns of data blocks makes the burden too heavy for system administrators.

We have proposed a distributed storage system which migrates a data block autonomously to a suitable storage device [1]. In our storage system, each storage node which provides public storage area monitors other storage nodes and migrates a data block to more proper storage node if necessary.

In our system, because data migration is performed over the network, the cost of migration is unignorable. The inefficient migration should be avoided to the utmost in order to prevent to disturb activity of the other network nodes or to consume extra processing power for the migration. Our investigation clarifies that the migration effect is excessively limited because of avoiding overconcentration to a specific storage node. It might degrade I/O performance, especially in a large system. Therefore, we propose an improved method of data migration aims to enhance migration efficiency in a large distributed storage system.

The rest of this paper is organized as follows. First, we present related work in section two. Section three describes a summary of the autonomous distributed storage system which has been proposed in our previous works. Section four provides details of enhanced migration policies to improve migration efficiency. In section five, we show the results of experiments and the considerations. Finally, we conclude this paper in section six.

II. Related Work

Easy Tier [2] and *FAST* [3] (Fully Automated Storage Tiering) are commercial storage tiering systems implemented for storage server products. They support up to three storage tiers which are statically classified by device technology. *Btier* [4] is a block device with automatic migration for Linux kernel. In Btier, though the arbitrary numbers of storage tiers can be made, most systems use only two tiers such as an SSD tier and an HDD tier to save administrators' effort. In those practical systems, they mainly focus that which data should be migrated to which storage tier.

Hystor [5] and *OTF-AST* [6] (On-The-Fly Automated Storage Tiering) provide automated data migration function. They perform data migration between statically defined two storage tiers, SSD and HDD. *Cost Effective Storage* [7] also proposed storage with dynamic tiering. Lipetz et al. [8] proposed an automated tiering using three storage tiers, SSD, performance-oriented HDD, and inexpensive HDD. They are designed for a stand-alone storage server, and data migration is concluded only inside the server. In those systems, because there is no need to consider the influence on the network and the other

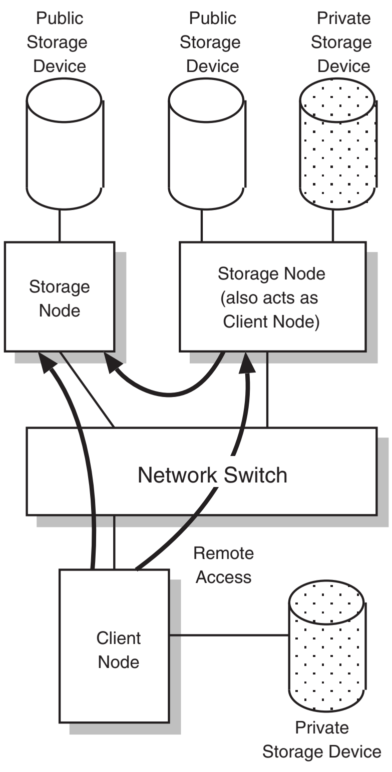

Fig. 1. A classification of network nodes.

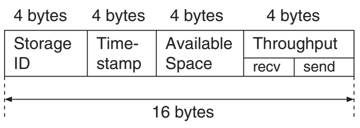

Fig. 2. Structure of storage information segment.

network nodes, the impact of overhead for data migration is less than a distributed storage system.

III. Autonomous Distributed Storage System

A. *Classification of Network Nodes*

We assume that all the network nodes are under the control of one administration group.

In our storage system, network nodes are classified as the following two types, *the storage node*, and *the client node*. Fig. 1 shows a classification of network nodes in the assumed environment. We refer the network node which has a public storage area as the storage node, and the node which remotely mounts that storage area as the client node. Some storage nodes which access to remote storage also act as client nodes.

Each storage node periodically exchanges its current status by using a small data structure named *storage information* [9]. Fig. 2 shows the structure of storage information segment [10]. In Fig. 2, the third field represents the available storage space in megabytes. The two throughput fields represent the effective throughput and are expressed in megabytes per second. The storage information has total of 16 bytes length.

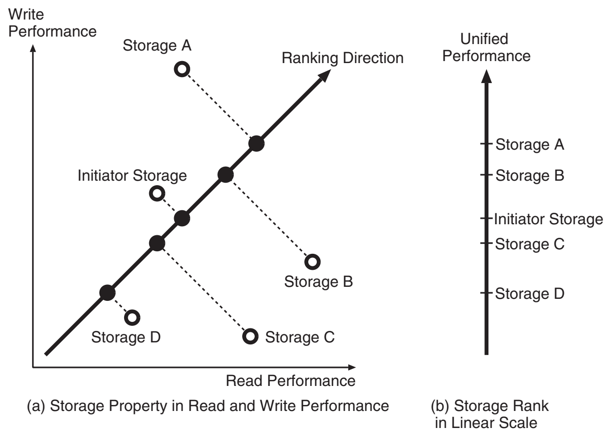

Fig. 3. An example of ranking storage nodes.

We also showed a scheme to reduce network traffic for exchanging the storage information in the SAN (Storage Area Network) environment which is using the iSCSI [11] protocol. Our observations [10] indicated that short packets about 100 bytes length are frequently transmitted in the iSCSI environment. It is because the iSCSI protocol is based on the SCSI protocol designed for a non-packet environment. In other words, there is a large gap between the data size using for iSCSI protocol and transmittable data size in the IP network. Accordingly, we proposed a scheme that plural segments of the storage information are appended to an Ethernet jumbo frame [12] including iSCSI packet up to the size of the maximum transfer unit (MTU). Our previous investigations [9] show that sufficient capacity in such an Ethernet frame can be used for transmitting the storage information. Thus because of reducing the number of dedicated frames for the propagation of the storage information, we can share the status of storage nodes with low network overhead.

B. *Storage Tiering*

To locate data blocks into the proper storage node, storage nodes are first ranked according to their effective performance [13]. The effective performance of a storage node is calculated by unifying its reading and writing performance according to reading and writing ratio of accessing by a client node and is also reflected the influence of the performance drop caused by access concentration. Next, the ranked storage nodes are stratified as *storage tiers*. The storage nodes which have a near level of performance of the migration source node are recognized as belonging in the same storage tier.

Fig. 3 shows an example of ranking five storage nodes. In this figure, initiator storage denotes the node which intends to initiate a data block migration. In Fig. 3(a), the vertical and the horizontal axes are writing and reading performance of the storage nodes, respectively. The foot of a perpendicular from a plot of a storage node to a line of ranking direction shows the unified performance of the storage. In the case of the frequency of the reading and the writing operation to a data block is identical, the slope of ranking direction is set to one. If you prefer to migrate a data block which has read intensive access pattern, the slope of ranking direction should be set as lower than one. The unified performance of a storage node will be plotted on the straight line like Fig. 3(b), and it is used for ranking the storage nodes.

After ranking the storage nodes, the migration source node makes storage tiers. The migration source node regards the storage node having the performance exceeding a threshold as belonging to the upper storage tier. Similarly, the storage node having the performance less than a threshold is regarded as belonging to the lower storage tier. The threshold values, and , are led by using statically defined tiering parameter . The parameter p is set by a system administrator to prevent ineffectual data migration to a nearly same performance destination node considering migration overhead in the target system. and are defined by Equations (1) and (2).

(1)

(2)

Here is the unified performance value of the migration source node.

The migration source node set the lowermost node in the upper storage tier as the migration destination in order to moderate overconcentration of the data migration to a few storage nodes.

When the amount of storage used is nearing its maximum capacity of the storage node, it will initiate data migration to keep certain free space. The uppermost node in the lower storage tier is set as the migration destination in order to mitigate sudden performance drop.

IV. Improvement of Migration Efficiency

As stated in the previous section, the data block migration is performed in a conservative manner. This policy helps to prevent that frequently accessed data blocks concentrate to a specific storage node. However, it might defer a spreading speed of data blocks to be migrated. For example, in the environment which has a large number of storage nodes, there might be a lot of destination candidate nodes in the upper or lower storage tier. When a migration source node has an extremely frequently accessed data block, the block should be migrated to as higher performance node as it can. If the performance improvement by a data migration is limited by the tiering parameter p, the performance gain expected by a migration might be insufficient. This may cause a series of data block migration and increase network traffic.

We revise a part of migration strategy to enhance an effect of the data block migration. Fig. 4 shows three policies for migrating a data block to the upper storage tier.

Policy (1) is the original one presented by our previous work [13]. The lowermost storage node in the upper storage tier or the topmost storage node in the lower storage tier will be selected as the destination by this policy. A destination of data migration will be able to be highly dispersed in this policy.

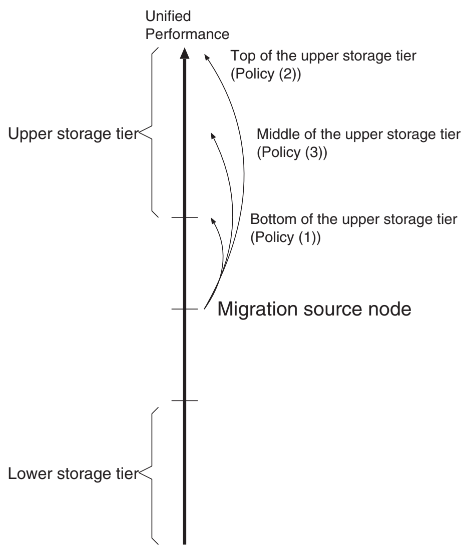
Unified Performance
Top of the upper storage tier (Policy (2))
Upper storage tier
Middle of the upper storage tier (Policy (3))
Bottom of the upper storage tier (Policy (1))
Migration source node
Lower storage tier

Fig. 4. Three migration policies to the upper storage tier.

However, when a storage node migrates a data block to the upper storage node, the estimated performance gain is limited by the threshold . Especially, when there is a number of storage nodes in the upper tier, it may lose an opportunity of much improvement in performance.

Policy (2) is the most aggressive data migration. If a storage node migrates a data block to the upper storage tier, the migration source node chooses the top storage node in the upper storage tier as the destination. In data migration to the upper storage tier, although the estimated improvement of performance reaches the maximum, the I/O workload may be concentrated on the specific storage node. If a storage node migrates a data block to the lower storage tier, the migration source node chooses the bottom storage node in the lower storage tier as the destination. Though performance degradation will be significant, this migration expects to minimize an impact for priority I/O requests issued from another client node.

Policy (3) is a moderate one. The migration source node chooses the middle storage node in the upper or lower storage tier as the destination. In data migration to the upper storage tier, this policy expects to obtain a higher performance gain by one migration than Policy (1) though the gain will not reach the gain by Policy (2). Moreover, in data migration to the lower storage tier, the migrated data block might disturb priority I/O requests at the destination node. Policy (2) and Policy (3) hereinafter are referred to as *enhanced migration policies*.

In Policy (1), as stated as Equations (1) and (2), the threshold values and are defined as the relative from the performance of the migration source node. If the absolute performance value of the migration source node is low, both of the two threshold values become nearly equal to the performance of the source node. Therefore, the effect of performance improvement by data block migration may be limitative. When

TABLE I
SIMULATION PARAMETERS.

	Total number of client nodes	60, 120, 180
	Total number of storage nodes	30, 60, 120, 180
	Tiering parameter	0.1, 0.2, 0.3, 0.4
	Number of I/Os per a client node	10,000
	Network latency	
	Destination selection time	
	Merging Time	
	Upper limit of waiting time	
	Lower limit of unused storage ratio	

the unified performance value of the migration source node is situated at low, the performance improvement using the enhanced migration policies is expected to be large.

First, the migration source node sorts all the other storage nodes in descending order by the unified performance. Next, the migration source node picks an appropriate storage node as the destination from the upper or lower storage tier. In the case of data migration to the upper storage tier by Policy (3), if the top storage nodes have superior performance than the migration source node, the th node will be chosen as the destination node.

V. EVALUATION

A. Simulation Environment

We evaluated the execution times of the three policies described in the previous section.

The evaluations were made on a simulator coded in C language. The static parameters used for the simulation are shown in Table I. The number of storage nodes is set smaller than the number of client nodes to make a concentration of I/O operations. Because previous work [13] shows that relative better performance is obtained in the tiering parameter or 0.3, we defined the parameter as a range of 0.1-0.4. The parameter is latency for switching a packet on an Ethernet switch. It was measured on a real switch which has wire-speed forwarding capability. In this evaluation, we use an ideal configuration that network latency among all nodes is uniform in order to focus upon observing the behavior of the proposed method. Two timing parameters, and , have been defined by the results of preparatory experiments same as the previous work [13]. The destination selection time is constant regardless of the number of storage nodes because each node manages other nodes by using an array structure and picks a destination directly from the array. The parameter is the time of merging the storage information to an iSCSI packet. The parameter is used for a threshold to decide whether to initiate data block migration to the upper storage tier. When the estimated waiting time which is calculated with the waiting queue length and averaged read/write latency exceeds , the migration to the upper storage tier will be initiated. The parameter is used for a threshold to decide whether to initiate data block migration to the lower storage tier. When the size of free storage space is below of the total storage space, the migration to the lower storage tier will be initiated. This threshold is the same as the value used for our previous simulation [13]. We decided that the definition of is reasonable from the result of large investigation in the real environment reported by reference [14]. It reported that the mean space usage of 10,568 file systems of 4,801 Windows PC in a commercial environment is only 53%.

TABLE II
ASSUMED STORAGE DEVICES AS STORAGE TEMPLATES.

Storage Template	Major Specifications	Vendor Name and Model Name
	SATA2 7200rpm HDD	Seagate ST3400620NS
	SATA3 7200rpm HDD	Seagate ST6000NM0235
	SAS3 15000rpm HDD	Seagate ST900MP0146
	PCIe 3.0x4 MLC SSD	Intel SSD 750
	SATA3 TLC SSD	Intel SSD D3-S4610
	SATA2 MLC SSD	Intel X25-M

TABLE III
CONFIGURATIONS OF STORAGE TEMPLATES.

Storage Template	Read Latency [s]	Write Latency [s]	Initial Capacity [MB]
	110	110	1,000
	36	36	2,000
	25	25	1,000
	3	4	250
	11	20	500
	29	116	1,000

Table II shows storage templates used for the simulation. We assumed six templates of device to use for storage nodes. These devices are chosen from various type of HDD and SSD to emphasize system heterogeneity. In the case of the total number of storage nodes is 60, ten instances per a storage template are created.

Table III shows the configurations of the template of the storage nodes. The latency values are calculated with performance information collected from their data sheet published by the manufacturer. The initial capacity shows the size of public storage shared for client nodes.

Table IV shows parameter configurations of six types of templates for the client node. Each client template is set to issue an I/O request at intervals of 2 s, 4 s, and 8 s. Reading and writing ratio of the I/O requests from each client template is shown in the I/O ratio column in Table IV. In this simulation, the frequency of reading and writing is set to nearly equal. In a similar manner as the storage node, the client nodes are created from these templates.

The unit size of data migration is set as ten megabytes. The

TABLE IV
CONFIGURATIONS OF CLIENT TEMPLATES.

Client Template	I/O Interval [s]	I/O Ratio (R:W)
	2	55:45
	4	55:45
	8	55:45
	2	45:55
	4	45:55
	8	45:55

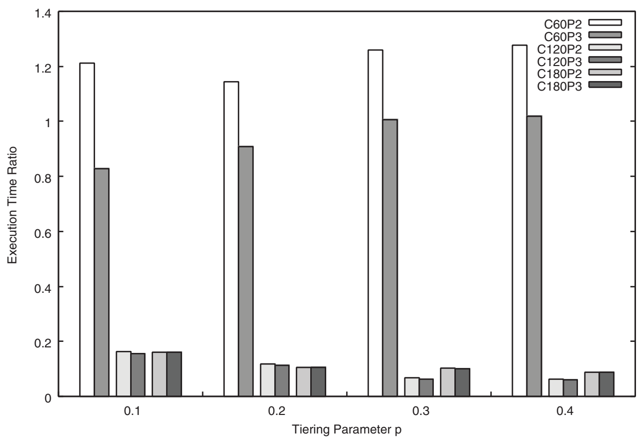

Fig. 5. Execution time ratio of Policy (2) and (3) to Policy (1) ( ).

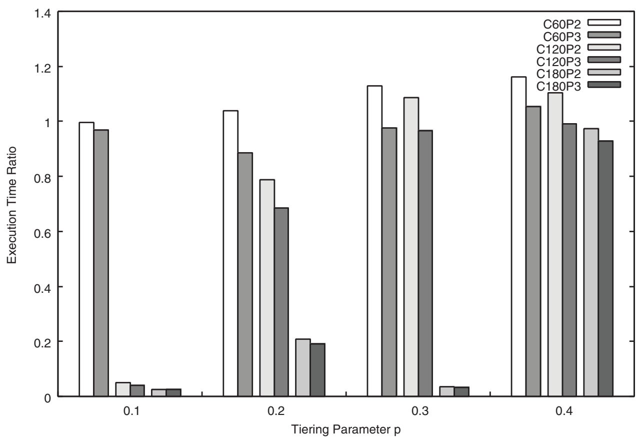

Fig. 6. Execution time ratio of Policy (2) and (3) to Policy (1) ( ).

latency time of one data migration is set as 200 milliseconds. It is the average time measured in the 1000BASE-T network.

Initially, all the client nodes access data blocks located on one of the storage nodes created from the storage template .

B. Results and Considerations

We measured the average execution time to complete times I/O requests by each client node. It is calculated by using Equation (3).

(3)

Here denotes the time to complete all I/O requests on the th client node. Next, we define as the average execution time measured using the migration policy (i.e. Policy ( )).

Figs. 5, 6, 7, and 8 show results of the simulation. Each result is labeled as the concatenation of the number of client nodes and the migration policy. For example, the result "C60P2" denotes the result which is measured in the condition under 60 client nodes and the Policy (2) migration policy. In these figures, the execution time ratio is calculated by using Equation (4).

(4)

1) Performance Comparison between Original Migration Policy and Enhanced Migration Policies: Fig. 5 shows that the execution time in the conditions and are reduced less than one-fifth. In this condition, because the number of server nodes is relatively small, accessing data tends to concentrate on a small number of storage nodes from a large number of client nodes. Hence, a need to migrate data block is high, and the enhanced migration policies which can obtain a large effect by one migration reduced the execution time. However, in the condition , the execution time in Policy (2) became worse than Policy (1). When the number of client nodes is low, a need for data migration is not so much. Especially in Policy (2), access concentration to a specific storage node had a more harmful effect upon

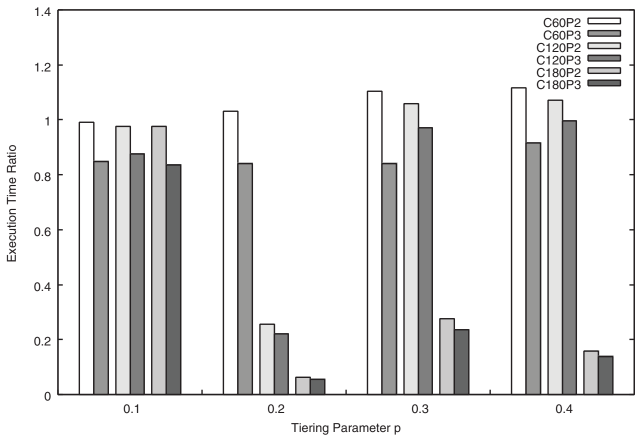

Fig. 7. Execution time ratio of Policy (2) and (3) to Policy (1) ( ).

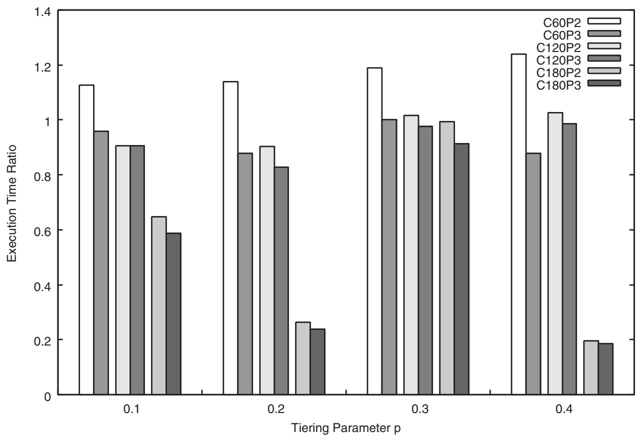

Fig. 8. Execution time ratio of Policy (2) and (3) to Policy (1) ( ).

the execution time rather than a good effect taken from an effective migration.

In Fig. 6, the execution time in the condition combined and are greatly reduced less than 20%. On the other hand, the improvement in the condition is less than the results in the conditions . It is because of that the tiering parameter is sufficiently large and the migration effect could be obtained even in Policy (1) at this condition.

In Fig. 7, the execution time by the Policy (2) in the condition were about the same as the Policy (1). The effect of improvement of data migration might have been canceled by the performance decline caused by a concentration of the destination of migration. In the conditions , when the number of client nodes is large as compared to the number of storage nodes, the enhanced migration policies reduced the execution time. Although the number of destination candidate for migration will be generally decreased when parameter is increased, the demand for data migration augmented because of data access from a large number of client nodes. In such a situation where there is a large number of client nodes in the system, the enhanced migration policies worked effectively.

Fig. 8 shows that the performance improvements by the enhanced migration policies are overall declined. Especially, in the condition combined and , the reduction of the execution time is only under 10%. When the parameter is varied from 0.1 to 0.4 in the condition combined and , the execution time by Policy (1) was the shortest at . This is because there is only a little margin to achieve further improvement because the absolute time of execution by Policy (1) is sufficiently small.

In Figs. 5, 6, 7, and 8, most of the execution time ratio exceed one in the condition and Policy (2). Because there is relatively little demand for data migration from a small number of client nodes, the defect of concentrate data blocks onto a specific storage node may have been marked more than the performance gain obtained by a few migrations. On the other hand, the execution time ratio of Policy (3) almost below the ratio one. The execution time of the worst case is just 5.5% longer than Policy (1). This shows that Policy (3) can achieve consistent performance improvement in most cases not only in a large system.

2) Performance Comparison between Enhanced Migration Policies: In all cases, the execution time using Policy (3) is shorter than the execution time using Policy (2). Especially, the improvement of the execution time is remarkable in the condition . The maximum improvement reached 31.5% in the condition , , and combination. Because the performance improvement by Policy (3) is always superior than Policy (2), we can use only Policy (3) as the enhancement over the original migration policy.

VI. Conclusion

In this paper, we proposed two new migration policies as an improvement of data migration in a distributed storage system with dynamic tiering and evaluated the effect of the policies. The policy which aims to maximize the migration efficiency degraded system I/O performance under a certain combination of conditions. We confirmed that the migration policy which balances an avoidance of data concentration and improvement of migration efficiency consistently improved the performance.

As future work, we will develop a scheme which can be applied for an actual network which has not uniform network latency, such as a network with cascaded switches. We will also build an experimental system for more detailed evaluation.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 18K11214.

References

[1] A. Nunome, H. Hirata, and K. Shibayama, “A distributed storage system with dynamic tiering for iSCSI environment,” *International Journal of Networked and Distributed Computing*, vol. 3, no. 1, pp. 42–50, Jan. 2015.

[2] 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 Corporation, 2013, available at ibm.com/redbooks.

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

[4] “btier,” http://sourceforge.net/projects/tier/.

[5] F. Chen, D. Koufaty, and X. Zhang, “Hystor: Making the Best Use of Solid State Drives in High Performance Storage Systems,” in *Proceedings of the International Conference on Supercomputing (ICS ’11)*, May 2011, pp. 22–32.

[6] K. Oe, T. Nanri, and K. Okamura, “On-the-fly automated storage tiering with caching and both proactive and observational migration,” in *Proceedings of the 3rd International Symposium on Computing and Networking*. IEEE, Dec. 2015, pp. 371–377.

[7] J. Guerra, H. Pucha, J. Glider, W. Belluomini, and R. Rangaswami, “Cost Effective Storage using Extent Based Dynamic Tiering,” in *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST ’11)*, Feb. 2011, pp. 273–286.

[8] G. Lipetz, E. Hazan, A. Natanzon, and E. Bachmat, “Automated tiering in a QoS environment using coarse data,” in *Proceedings of 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*. IEEE, Nov. 2013, pp. 1022–1030.

[9] S. Shimano, A. Nunome, H. Hirata, and K. Shibayama, “An information propagation scheme for an autonomous distributed storage system in iSCSI environment,” in *Proceedings of 3rd International Conference on Applied Computing and Information Technology (ACIT 2015)*, Jul. 2015, pp. 149–154.

[10] A. Nunome, H. Hirata, and K. Shibayama, “An interval control method for status propagation in an autonomous distributed storage system,” in *Proceedings of the 15th IEEE/ACIS International Conference on Computer and Information Sciences (ICIS 2016)*, Jun. 2016, pp. 723–728.

[11] M. Chadalapaka, J. Satran, K. Meth, and D. Black, “Internet Small Computer System Interface (iSCSI) Protocol (Consolidated),” RFC 7143, RFC Editor, Fremont, CA, USA, pp. 1–295, Apr. 2014.

[12] “Ethernet Jumbo Frames version 0.1,” http://www.ethernetalliance.org/wp-content/uploads/2011/10/EA-Ethernet-Jumbo-Frames-v0-1.pdf, Ethernet Alliance, Nov. 2009.

[13] S. Shimano, A. Nunome, Y. Yokoi, K. Shibayama, and H. Hirata, “A dynamic configuration scheme of storage tiers for an autonomous distributed storage system,” *Information Engineering Express*, vol. 3, no. 4, pp. 91–104, Dec. 2017.

[14] J. R. Douceur and W. J. Bolosky, “A Large-Scale Study of File-System Contents,” in *Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, May 1999, pp. 59–70.