# 論文 / 著書情報
# Article / Book Information

| | |
|---|---|
| Title | An Experimental Evaluation of the Adaptive Replica-assisted Migration forParallel Storage Systems |
| Author | Dai Kobayashi, Ryo Taguchi, Haruo Yokota |
| Journal/Book name | Proc. of International Special Workshop on Databases For Next Generation Researchers (SWOD 2007), Vol. , No. , pp. |
| Issue date | 2007, 4 |
| DOI | http://dx.doi.org/10.1109/SWOD.2007.353198 |
| URL | http://www.ieee.org/index.html |
| Copyright | (c)2007 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。<br>This file is author (final) version. |

# An Experimental Evaluation of the Adaptive Replica-assisted Migration for Parallel Storage Systems

Dai Kobayashi
*Tokyo Institute of Technology*
*Japan Society for the Promotion of Science*
*daik@de.cs.titech.ac.jp*

Ryo Taguchi
*Japan Broadcasting Corporation*
*taguchi.r-cs@nhk.or.jp*

Haruo Yokota
*Tokyo Institute of Technology*
*yokota@cs.titech.ac.jp*

## Abstract

*We have proposed a method replica-assisted migration that use temporally replica data in other storage nodes during executing data migration to keep both qualities of service accesses and data migration duration. However, we have not evaluated the method with real workload including update operations. In this paper, we first improve the method and evaluate an efficiency of the method under file server workloads. Base on the result of it, we also propose Adaptive replica-assisted migration for reduce migration duration.*

## 1. Introduction

Data migration is a method to balance workloads for remove hot spots in which heavy load is concentrate on parallel storage systems and distributed databases [2, 10]. Transferring data from hot spot to unused storage node keep performance of storage systems for changing workloads.

However, data migration causes performance degradation temporarily because data transfer consumes system resources such as I/O bandwidth and network bandwidth [1, 7, 11]. The storage nodes with heavy accesses need to process accesses for both services and data migration, though the storage node degredate its performance extremely with excessive load. Saturated I/O such as disks or networks of partial storage nodes makes a system difficult to guarantee its performance.

For the problem, we have proposed Replica-assisted migration that reduce and distribute load caused by migration, using replica data for data redundancy. The method reduces the amount of data transferred and provides a source node selectability by using replica data location. The method also achieves to keep performance of each storage node by forwarding moderate ratio of accesses to replicas.

However, our past studies don't include enough evaluations. We have indicated the efficiency of our method by only experiments under synthetic workload with only read operations, while update operations and real workload have an influence on behavior of the method because of the consistency control and real change of access trend. The evaluation and improvement of the method for real workload with update operations is required in order to apply our method to actual systems.

A purpose of the paper is to show the efficiency and applicability of the Replica-assisted Migration on real workloads with update operations. At first, In this paper, we use file server workload generated by trace logs on actual shared file server for evaluate our method. Current file servers become larger and larger storing enlarged personal information. So they are one of most significant application for large-scale storage systems The workloads include extremely skew distribution and a half of the loads consists of update operations.

First, we improve our method for file server workload. Then we observe a behavior of our method on simulation program with workload generated by real file server trace logs. Results of the simulation show that providing migration source node selectability can work well, while access forwarding cannot indicate efficiency.

Based on the results, we also propose Adaptive replica-assisted migration that improve the proposed method to control the migration source node adaptively with the load of the node for reduce migration duration. The experimental results with simulation and real workload indicate that adaptive method can reduce migration duration in many situations.

The rest of this paper is organized as follows. We de-

scribes assumed environment, technology and system organization in 2. In section 3, we summarize Replica-assisted migration we have proposed. Section 4 describes file server I/O and improvement of the proposed method. Section 5 indicate exeprimental results and discussions using simulation program. In section 6, we proposed adaptive replica-assisted migration and show its efficiency. In last section we include our concluding remarks.

## 2 Data migration

Performance of hard disk drives extremely decline when excessive accesses concentrated on them [8]. Current large-scale storage systems consist of a lot of storage nodes with connecting over high-speed networks. Saturated I/O such as disks or networks of partial storage nodes make a system difficult to guarantee its performance. Therefore access load skew should be balanced over all storage nodes.

Data migration is proposed for the problem. Data relocation by transferring stored data with considering performance of storage node and current workload trend can achieve load balancing on parallel storage systems.

### 2.1 Migration strategy

We assumed that all intelligent storage nodes record the degree of current load and planning task is executed on a node, aggregating current workload information. The node executing planning task, we call *coordinator*, planed not amount of migrating data but amount of migration load degree because aggregating workload information of each data is much complex.

The nodes execute following tasks at regular time intervals defined in advance. At first the nodes pick a node for coordinator and send the degree of current load on each node to coordinator. Then the coordinator node calculates average load degree and make migration amount list that include information that how much data or load each node should migrate to other node. At third, the coordinator node refine the list by multiply speed-factor parameter $\theta(0 \leq \theta \leq 1)$ to prevent overdo. The end of the strategy, each storage nodes transfer ordered amount of data to ordered nodes.

### 2.2 A Problem caused by load of data migration

It is significant problem that data migration causes performance degradation temporarily. Data migration consumes system resources such as I/O access ability and network transfer bandwidth that services for clients also uses. The migration causes serious performance degradation on systems because the migration tends to occur on nodes when the nodes already become hot spot. So when hot spots appear on systems, the migration cannot make a sense or cannot guarantee the quality of storage systems services.

Although there are many studies on achieve both storage QoS and data migration simultaneously [1, 7, 11], many of them consider not horizontal migration but vertical one. Thus, the horizontal migration with speed adjusting makes no sense because it cannot follow changing speed of workload trend.

## 3 Replica-assisted Migration

We have proposed Replica-assisted migration to achieve both storage QoS and data migration without slowing migration speed . The rest of the section describes a concept and summary of the method.

### 3.1 Concepts

We uses replica data for data redundancy described at section 2. We first reduce the amount of transferred data and make selectability on migration source node. We also use replicas for access services, forwarding partial accesses of primary data to replicas. Using replicas not only with migration but also with regular services is beyond the scope of this paper, although we discussed the usage in [5] An advantage of our method is that systems can migrate moderate amount of data even if the source node already becomes hot spot.

Following description of the method and algorithm assume that the systems adopt range-partitioning data placement policy and Chained Declustering replica placement policy [3] implicitly, although the concept of our method is applicable to other systems.

### 3.2 Migration source selectability using replicas

Replicas location make selectability of migration source node and the selectability make it available to generate migration load on less accessed nodes.

The algorithms implicitly assumed that Chained Declustering replica placement policy and range partitioning data placement policy. The stored data has unique identifier and sorted the identifier. Each node has range of identifier and equivalent data are stored the node. The range make two nodes as *neighbors*.

The migration destination node restricted to which *right* neighbor node that store data that has larger identifier or *left* neighbor node that store smaller one by range restriction. Therefore using replicas on right node, the migration can select data transfer source node from whether primary data or replica data. We call the former *S(Self)-pattern* and the latter N(Neighbor)-pattern. Figure 1 shows the S- and N-pattern of right migration.

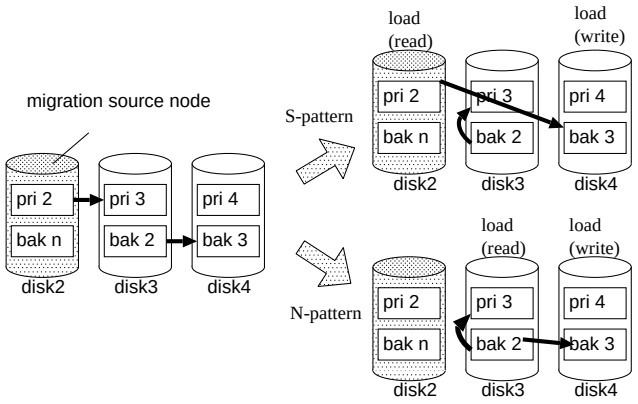


**Figure 1. Two data transfer source node. the coordinator node can select the source node from pri$_i$ on disk$_i$ or bak$_i$ on disk$_{i+1}$ when migrating data$_i$**

When data migration requires, the coordinator can select migration source node from S- or N- pattern with aggregated workload information.

### 3.3 Access forwarding to replicas

Partial accesses towards data stored in a node which is hot spot and also be source node of data migration can be forwarded to nodes that store replicas of the primary data to avoid causing performance degradation by data migration. Thus you can execute data replacement even if hot spot has appeared.

Our method execute following tasks in addition to regular migration tasks described at 2.1 At first coordinator calculate access forwarding ratio $r$ as well as calculate the amount of migrating load, using load degree aggregated from all nodes. At that point, the coordinator issues orders to all node to switch replica consistency mode from asynchronous to synchronous one and apply uncommitted update logs, if the system adopt asynchronous replicas. Then the coordinator send the $r$ to all nodes and the nodes set each $r$ to itself. At third, required nodes migrate the stored data to ordered nodes. At that point, accesses to the nodes are partially forwarded to the nodes that stores replicas based on $r$. When the node finish to migrate ordered amount of data, the node set own forwarding ratio as zero.

#### 3.3.1 Calcuration of Forwarding Ratio

A coordinator node calculate access forwarding ratio $r$ as follows. In the discussion, $L_i(t)$ is load of primary data stored in node $i$ and $L_i^{\mathrm{TMP}}(t)$ is list of values using temporarily in the algorithm.

At first, the coordinator picks a node $i$ that total number of current load of the node $L_i^{\mathrm{TMP}}(t)$ plus migration load $L_{\mathrm{MIG}}$ is larger than max processable load per node $Lm_i$. Then, if forwardable load $h_i$ is smaller than the primary load $L_i(t)$, in other word, the node $i$ can forward its load to other one, $L_i^{\mathrm{TMP}}(t) - L_{\mathrm{MIG}}$ is subtracted from $L_i^{\mathrm{TMP}}(t)$ and added to $L_{i+1}^{\mathrm{TMP}}(t)$ and $h_i$ that is forwarding load of node $i$. The calculation involves *access forwarding*. At that point, if $L_{i+1}(t)$ that is load of nodes storing replicas of the node$i$, we call the node node $i+1$, exceeds the max load $Lm_{i+1}$ to keep agreed response time, node $(i+1)$ forwards the access load to node $(i+2)$, according similar to the way described as above. The algorithm terminate when a node $j$ that satisfy $h_j > L_j(t)$ exists or all node $i$ satisfy $L_i^{\mathrm{TMP}}(t) < Lm_i$, in other word, no update of $h_i$ occur. The output of the algorithm is access forwarding ratio $r_i$ that is calculated by $h_i/L_i(t)$

#### 3.3.2 Estimation of data migration load

This section describes the estimation value of data migration load $L_{\mathrm{MIG}}$ that is required at algorithm described section .

We estimate the migration load by performance curve of storage nodes that indicate response times of a storage node by access frequency. The migration load can be represented as $Q_{avg}/2R_{req}$ [Byte/s], where $Q_{avg}$ [ Byte ] is average size of data segment such as file and blocks, $R_{req}$ [s] is max response time that client can tolerate, and 2 on denominator represent sequential behavior between data read on source node and write on destination node.

You note that we improve the estimation method at section 5.1 because the estimation method assumed that behavior of storage nodes is linear to IO size, although workload with various IO size such as file server workload caused non-linear responses.

## 4 Improvement for file server workloads

In this paper, we uses file server workloads as real and update-including workloads. We describes summary of properties of file server workloads, improvement of premised migration strategy and Replica-assisted migration.

### 4.1 Properties and trends of file server workloads

Hsu et al. [4] analyzed trace logs on fifteen PCs and five fileservers and report their characteristics. In this section, we summarize the report as four properties:

**Property 1** *The workload includes update requests. The ratio of update requests is 30-60 percent by both request-size and number.*

**Property 2** *Request size is very small. The average request size for the original workloads is about 7-9KB, they said.*

**Property 3** *Only a small fraction of the data stored is in active use, they said. the average size of files accessed c times at a day $f(c)$ is approximately described as $a/c^b$, where a and b are positive constants. In addition, the daily working set for the various workloads ranges from just over for percent to about seven percent of the storage used.*

**Property 4** *There are few files tha t are both read and written. less than 25 percent of the total working set size are used both read and written.*

### 4.2 Improvement of premised migration strategy

Before improving our method, we first improve the premised migration strategy described at section 2.1.

#### 4.2.1 Various request size

Property 2 shows that the workload includes various size of access requests. Thus it is difficult to estimate max performance of storage nodes as units on MB/s or `iops` because behaviors of hard disk drives tend to be non-linear on size.

We refine our migration strategy not to use response performance on a request size but to use one on various request size by measuring with benchmarks.

#### 4.2.2 Extremely skew access distribution

Property 3 shows the extremely skew access distribution. Our premised migration strategy described at section 2.1 must migrate massive amount of data because the strategy is based on load equalization on each node with the workload. The massive amount migration causes longer migration duration than changing speed of workload trend.

We refine the strategy not to base on load equalization but keeping max load to keep agreed response time of each node. The strategy calculates amounts of migrating data by keeping performance of storage node as hot spot on levels on service level agreement (SLA).

The refine reduce amount of migrating data on practical one. Our simulation experiments show that the amount is reduced from about 100GB to about 1GB, while keeping performances of all storage nodes.

#### 4.2.3 skew distribution of update ratio

Property 4 shows that update ratio of each data segments are funnel towards zero or one. Thus the coordinator cannot determine the migrating data segments with abstract average information of update ratio on stored data.

We refine to re-obtain update ratio of each data on each stored node after the coordinator node issues migration orders for keeping both scalability and accuracy of update ratio information. The migration destination node also informs the update ratio of migrated data to the node that has replica of it.

### 4.3 Improvement of Replica assisted Migration

Replica-assisted Migration should be improved as follows for real workload such as file servers.

Consistency control between primary and replicas are as follows. We assume the synchronous update between primary and replicas. Request sequence is received the node containing primary data. The update requests and read requests are serialized by the primary node and sending to nodes containing replicas if forwarding is needed.

Forwarding ratio should be calculated by load of read requests. We change the strategy described at section **??** to calculate by load of the node and average update ratio of node $i$ $u_i (0 \leq u_i \leq 1)$. First refined point is to change definition of load of a node from only primary load to total of read and write to primary and write to replicas. Second one is forwardable load from total primary load to only read load of primary. Third one is calculation of forwarding ratio $h_i$ to $h_i/\{L_i(t)(1-u_i)\}$

## 5 Experiments

We evaluate and observe the behavior of our method under file server workload by simulation program we constructed.

### 5.1 Environments

We use a queuing network based storage system simulation program that we constructed in past study. Behavior of hard disk drives in the simulation is calculated with parameters on HGST deskstar T7k250 specification [9]. The simulation constructs of 16 storage nodes with 64MB cache connected by 800 Mbps network.

Write accesses are processed as below. Cache on each storage node is set to write through and replicas are updated synchronously. Then the write request arrived at disk drives on both primary node and replica node . Disk drives eliminate requests to same blocks. Command re-ordering method is not implemented.

As described at section , we need benchmark result of the storage nodes. We measured the max access frequency a node can keep practical response time with storing 1GB files by request size. Responses of larger size are dominated by networks, while one of smaller size are constricted by

**Table 1. Specifications of used workloads**

| workload parameter | a | b | c | d | e |
|---|---|---|---|---|---|
| Head Timestamp (hour) | 15 | 16 | 17 | 18 | 19 |
| Time span (hour) | 1.1 | 1.0 | 0.9 | 1.0 | 0.9 |
| Write Ratio on # (%) | 46.9 | 41.7 | 38.5 | 42.7 | 44.4 |
| Write Ratio on size (%) | 43.4 | 42.2 | 39.6 | 42.0 | 44.7 |
| Accessed size (GB) | 36 | 40 | 40 | 42 | 40 |
| # of accessed files ($10^3$) | 14.0 | 13.4 | 11.6 | 13.4 | 11.9 |
| # of requests | $1.2 \times 10^5$ | | | | |
| total file size(GB) | $450 \times 2$ (Primary & Backup) | | | | |
| # of total files | $1.0 \times 10^6$ | | | | |

disk drives. Applying least square method, we can calculate load of a request to $l(s)$ by following equation:

$$l(s) = \begin{cases} s \times 3.38 \times 10^{-9} + 5.49 \times 10^{-3} & (s \leq 800KB) \\ s/(96 \times 10^6) & (s > 800KB) \end{cases} \quad (1)$$

where the node performance is normalized as one. We use the equation on following experiments and discussion.

To evaluate our method, we use trace logs on real file server [6]that Hewlette Packard published. The used workloads are generated by clipping five set of one hour traces form the trace log and be as sequences of stateless accesses. Table 1 shows the characteristics of clipped workloads. Stored file set is also generated by trace log and results of `ls` command.

We use range partitioning as file placement policy. Identifier of each files are generated as follows:

$$id = sid \oplus ((\overline{sid} \wedge \texttt{0xff}) \times 2^{12}) \quad (2)$$

where *sid* is preliminary numbered by trace sequence order. The transform distribute files on $2^{20}$ ranges.

### 5.2 Results

Figure 2 shows the average and max response time per blocks under five segments of workload. both of them are normalized by response time without our method. The figure indicates that average response time are reduced, while max response time has no differences.

The experimental results show that the proposed method make no sense from aspects of QoS. We analyze the result log on simulation and discuss two differences of behaviors between synthetic workload and real workload.

One is by property 3 that is high skew distribution. Extremely skew distribution causes isolated hot spot and the migration can transfer data from other idle nodes by using replicas. Therefore migration load don't occurs on hot spot.

The other is high update load. In spite of the similar ratio between read and write operation, it is observed that many heavy workload consist of much of write accesses.

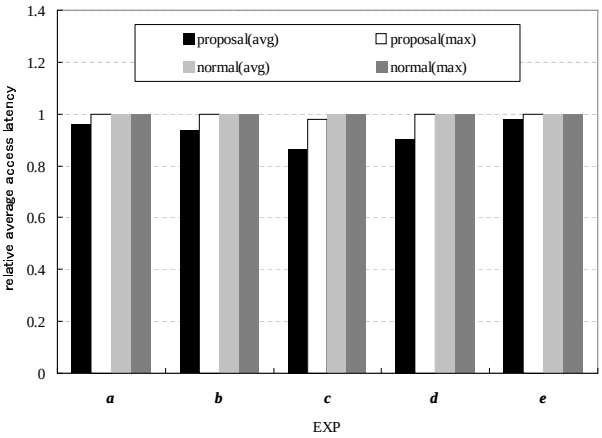


**Figure 2. Relative access latency with/without Replica-assisted Migration~iwithout proposed method ~j**

In addition, cache on storage nodes absorb almost all of read accesses with 90 percent hit ratio. Thus most accesses requests concentrated on hot spot are write operations that cannot forward to replicas.

We summaries the result as that while providing migration source selectability well works, access forwarding to replicas should be refined to work well with real and hard update operations.

## 6 Adaptive replica-assisted migration

The experiments and analysis shows that providing migration source selectability well work even under real workload.

In this section, we propose to enhance the method in order to reduce migration duration.

### 6.1 Adaptive selection of migration source nodes

When concentrated update operations make hot spot nodes, the node containing their replicas also become hot spot with synchronous update load. In order to reduce migration duration, the migration load should be distributed to colder hot spot.

We propose Adaptive selection of migration source nodes to solve the problem. The method dynamically select migration source node based on short-span evaluated load information such as I/O queue length.

The method enhances our past proposed method to add following tasks.

When migration of each file or moderate amount of files finishes, the migration source node, the replicas of the node and destination node communicate the short-span evaluated

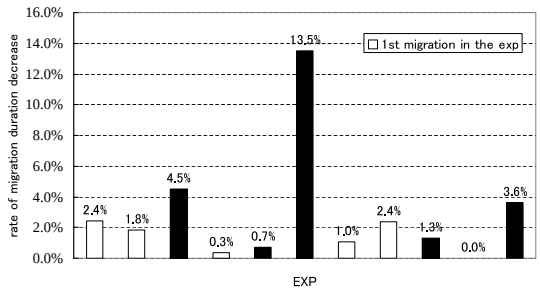


**Figure 3. reduction ratio of migration duration by Adaptive replica-assisted migration.**

load information. The information tend to transfer in piggyback payload. Then the primary of migration source node select which S-pattern or N-pattern, so that migration load occur at colder hot spot. At a result, the migration load is distributed to two nodes equally and migration duration is reduced.

### 6.2 Efficiency of adaptive selection

We observe behaviors of the adaptive migration source selection by simulation and real workload. In the experiments we use queue length of disk module as short-span evaluated load information.

Figure 3 shows reduction ratios on migration duration of eleven periods of migration with six segments of workload. The first three segments of workload are same as $a, b, c$ in table 1 and the second three are heavier version of first three to reduce 30 percent of intervals between accesses. In the figure, while bar indicate first migration on the migration and black bar indicate after second migration.

The results show that all of the migration duration is reduced by adaptive version of proposed method. About 2.6 percent of migration duration is reduced. Although we have not analyze details yet, the figure also show that the method work more efficient after second migration than first migration.

## 7 Conclusions

In this paper, we improved, observed, analyzed and evaluated Replica-assisted migration we have proposed with file server workload as actual and update-including workload. Refining a few point in order to work as file server, one of our method to provide migration source node selectability worked well with the workload, while the other of our method to forward partial accesses to replicas made no sense because of high skew access distribution and high update load.

Based on the result of analyze, we also proposed Adaptive replica-assisted migration by enhancing our past proposed method to reduce migration duration. The adaptive method worked well with almost all time of real file server workload.

To solve current issue, we plan to enhance and improve the premised migration strategy to apply our method to actual systems. We also should refine replica consistency control and cache behavior in order to reduce update load.

Detailed evaluation and analysis are one of important future works.

## Acknowledgement

This work is partially supported by Japan Science and Technology agency via CREST program, JSPS via Grant-in-aid for Scientific Research, Storage Research Consortium, MEXT via Grant-in-Aid for Scientific Research #18049026 and the MEXT 21st Century COE Program.

## References

[1] K. Dasgupta, S. Ghosal, R. Jain, U. Sharma, and A. Verma. Qosmig: Adaptive rate-controlled migration of bulk data in storage systems. In *the 21st ICDE*, pages 816–827, 2005.

[2] H. Feelifl, M. Kitsuregawa, and B. Ooi. A fast convergence technique for online heat-balancing of btree indexed database over shared-nothing parallel systems. In *11th DEXA*, pages 846–858, Sep 2000.

[3] H.-I. Hsiao and D. J. DeWitt. Chained declustering: A new availability strategy for multiprocessor database machines. In *Proceedings of the 6th ICDE*, pages 456–465, 1990.

[4] W. W. Hsu and A. J. Smith. Characteristics of I/O traffic in personal computer and server workloads. *IBM Syst. J.*, 42(2):347–372, Feb. 2003.

[5] D. Kobayashi, A. Watanabe, R. Taguchi, T. Uehara, and H. Yokota. An efficient access forwarding method based on caches on storage nodes. In *SWOD 2005*, pages 188–191, 2005.

[6] S.-P. Labs. Publicly-available storage traces( "file system traces"). http://tesla.hpl.hp.com/public_software/.

[7] C. Lu, G. A. Alvarez, and J. Wilkes. *Aqueduct:* online data migration with performance guarantees. In *USENIX FAST*, pages 219–230, 2002.

[8] H. Simitci. *Storage Network Performance Analysis*. Wiley Technology Publishing, 2003.

[9] H. G. S. Technologies. *Deskstar T7K250 Hard Disk Drive Specification*. http://www.hitachigst.com, ver. 1.7 edition, 2006.

[10] G. Weikum, A. Mönkeberg, C. Hasse, and P. Zabback. Self-tuning database technology and information services: from wishful thinking to viable engineering. In *VLDB*, pages 20–31, 2002.

[11] J. Zhang, P. Sarkar, and A. Sivasubramaniam. Achieving completion time guarantees in an opportunistic data migration scheme. *SIGMETRICS Perform. Eval. Rev.*, 33(4):11–16, 2006.