IADIS International Journal on Computer Science and Information Systems
Vol. 13, No. 2, pp. 109-123
ISSN: 1646-3692

# EFFICIENT SNAPSHOT METHOD FOR ALL-FLASH ARRAY

Miho Imazaki[1], Norio Shimozono[2], and Norihisa Komoda[3]
*[1]Research & Development Division, Hitachi America, Ltd., California, the United States of America*
*[2]Research & Development Group, Hitachi, Ltd., Yokohama, Japan*
*[3]Osaka University, Osaka, Japan*

**ABSTRACT**

An all-flash array (AFA) equipped with solid state drives (SSDs) is the high-reliability, high-performance and large-capacity data storing system. Snapshot is used as a data backup function and one of existing methods of snapshot is Copy on Write (CoW). However, CoW consumes the memory bandwidth of storage controller (SC) because of chunk copies and isn't suitable for a AFA. Therefore, we propose an efficient snapshot method named Mapping Switching in Flash Translation Layer (MSiF). Instead of physical data transfers by a SC, MSiF rewrites the SSD's logical-physical address translation table using Flash Translation Layer in SSDs. A difference occurs between current data and backup data due to an updating data, and a SC can overwrite the current data by switching a mapping of the logical-physical translation table in a SSD without physical data transfer. Therefore, the amount of data transfer to SSDs and paths in SCs decreases. We estimate the performance right after taking snapshots using an evaluation model; compared to CoW, MSiF reduce the time, which it takes to recover the temporary throughput degradation due to taking a snapshot, by up to 76.9 %, and reduce the response time by up to 56.5 %, in the case of chunk size 128 kBytes.

**KEYWORDS**

Storage System, All-flash Array, Snapshot, SSD, Flash Translation Layer

## 1. INTRODUCTION

For enterprise IT infrastructures, such as large enterprise data centers, improvement of application performance and total cost reduction of system maintenance are required. Therefore, there is a need for a high-performance storage system with high capacity, low latency, high throughput, and easy performance design and performance tuning. As a storage system satisfying these requirements, there is an All-flash Array (AFA) composed of all Solid-State Disks (SSDs) instead of Hard Disk Drives (HDDs) (Yi, 2015). An AFA consists of Storage

Controllers (SCs) and drive boxes that store many SSDs. Enough Central Processing Units (CPUs) in the SC are mounted for the SSD performance to maximize the flash chip performance of SSDs. However, SSDs have the problem that the endurance of flash devices is rapidly decreasing and directly related to the number of program/erase cycles allowed to memory cells (Kim & Ahn, 2008), therefore it's important not to do useless write access to SSDs. Furthermore, metadata to be accessed at the time of data processing are made to reside in a Dynamic Random Access Memory (DRAM) of the SC, and this realizes further higher performance.

Storage systems including AFAs have various functions for protecting data. One of them is snapshot which stores data as a backup copy at an arbitrary time (Xiao et al, 2006). In AFA, it's important to reduce backup data copy, and there are Copy on Write (CoW) and Redirect on Write (RoW) as existing snapshot technologies to reduce the data copy amount (Xiao et al, 2006). CoW and RoW manage the difference between the current data and the backup data as metadata, this is because the current data and the backup data are the same if the data is not updated from a host machine after taking a snapshot.

In RoW, after taking snapshot, the update data from a host machine is written to the pool, which is the exclusive area for snapshot, in the order of receiving from a host machine. A DRAM area managed by the address in the SC is allocated to the pool. Since small size update data may be written from the host machine, it is necessary to manage the memory address in SC allocated to the pool in small size units. Therefore, the large quantity of metadata for pool address management in SC's memory is needed. For this reason, RoW is not suitable for AFA where metadata resides in SCs' DRAM.

On the other hand, in CoW, the current data before update is copied to a pool after taking a snapshot, and then the update data is overwritten on the current data. Current data is copied to a pool in units of chunks which is data aggregates. Thus, the metadata capacity for chunk address management can be saved by a large chunk size, and large capacity data can be stored without significantly increasing DRAM capacity. However, the chunk copy is occurred in a unit of chunk which is large size when update data is written from a host machine, and the chunk copy to a pool may include non-updated data. Hence unnecessary writing occurs in SSD.

In this work, we propose the snapshot method that is optimal for AFA to reduce data copy, based on CoW which can backup large capacity data. We focus on the point that consumption of memory bands of SCs causes performance degradation, and discuss whether it is possible to reduce the amount of data transfer in SCs by taking snapshots. Thus, in the proposed method, instead of CPUs of SCs, CPUs of SSDs performs processing equivalent to data copy by Flash Translation Layer (FTL) (Kim & Ahn, 2008).

The rest of this paper is organized as follows. Section 2 describes AFA's architecture and CoW problem in AFA. We present our proposal in Section 3, and Section 4 shows that the throughput and response time of the proposed method are evaluated and the results are considered. Section 5 shows related work, and Section 6 concludes the paper.

## 2. PERFORMANCE IMPACT DUE TO DATA UPDATE IN COW

### 2.1 AFA Configuration

Figure 1 shows the schematic view of AFA. There are frontend interface modules (FE I/Fs), which communicate with host machine, back-end interface modules (BE I/Fs), which communicate with host machine, CPUs and DRAM in SC. The DRAM is divided into the cache area for storing data before transfer to the SSD, and the management area for storing metadata necessary for volume and snapshot management. And SCs are duplicated in the AFA to prevent a data loss. Upon receiving an update data write instruction from the host machine, for example, after the update data is written in the cache area of SC 0, the update data is copied to the cache area of SC 1. Therefore, even if one SC fails before data is written to the SSD, data is not lost (Morishita et al, 2015). These components are wired with some cables to communicate with each other. Connection between components are performed in paths called front-end path (path FE), back-end path (path BE), path SC and path DRAM (path MEM).

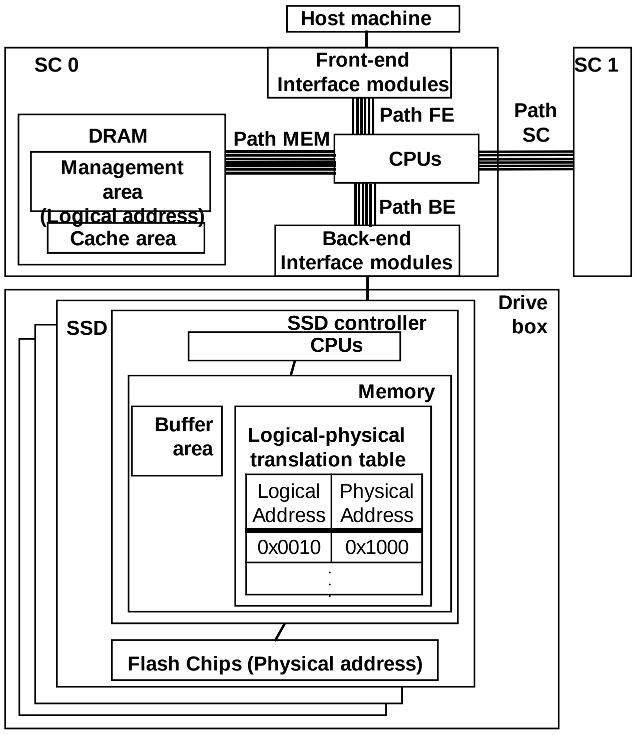


Figure 1. AFA Architecture

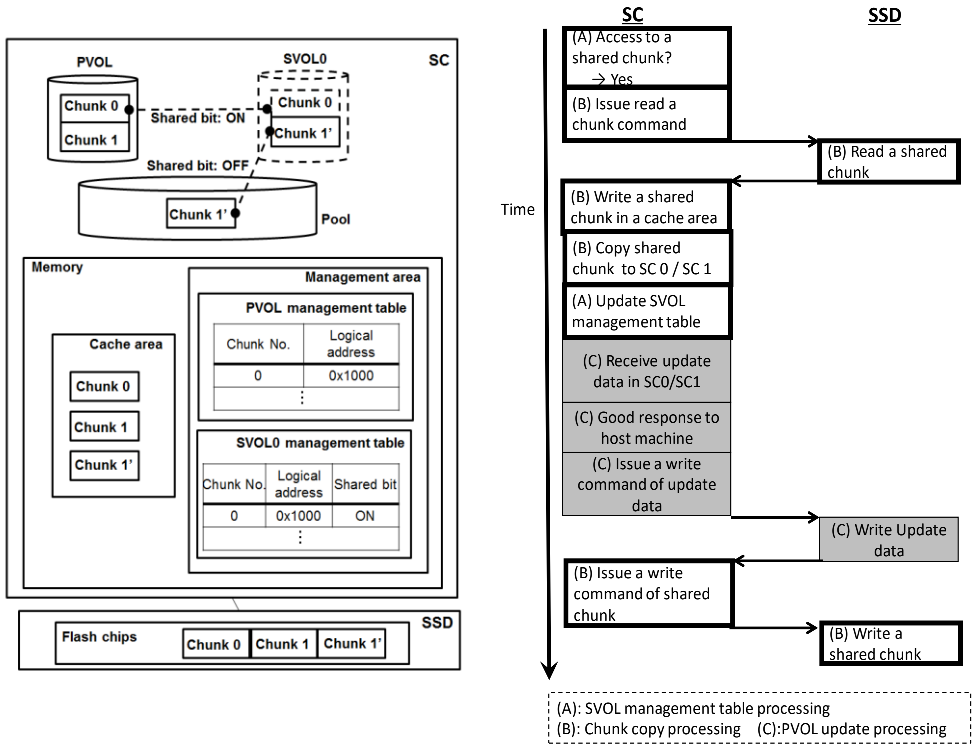
Many SSDs are stored in the drive boxes, and each SSD is connected to the back-end interface module of SC. A SSD is composed of a flash chip for storing data, CPUs, buffer area, in which read/write data stored temporarily, and an SSD controller for controlling reading and writing to flash chips. Furthermore, to recover data even if a part of SSDs fails, Redundant Arrays of Inexpensive Disks (RAID) are applied: a disk array technology that allows multiple

SSDs, called disk group, to be virtually seen as one SSD (Menon, 1994). In RAID, data are divided into blocks, which are called striping units, and stored in several disks in a block-based data striping. There are several types of RAID, called RAID levels, and RAID5 is one of them. RAID5 uses disk striping with a parity, and data and parity are striped across all the disks, and the traditional RAID implementation requires a SC to construct a parity with SC's CPUs for each write from a host machine.

To access the data in the AFA from a host machine, volumes obtained by dividing the disk group are allocated to a host machine. A host machine reads and writes data in units of blocks in a volume. In the management area of SC's memory, each block is given an address, which is called a logical address (Gupta et al, 2009). On the other hand, data is written to and read from flash chips in units of pages, which are given addresses called physical addresses (Gupta et al, 2009). The logical address to the physical address mapping information is stored in the logical-physical translation table of SSD controller. The logical-physical translation table is used for hiding the physical characteristics of the flash chip and providing a simple storage logical space SCs. This mechanism is called FTL (Imazakai et al, 2016). When a SSD receives write requests, FTL maps a logical address to a physical address in flash chip and the new data are stored in the new flash chip areas, where the old data aren't stored, before the old data are erased (Kim & Ahn, 2008).

## 2.2 Snapshot Processing Overview

The CoW mechanism describes in Figure 2 (a). The volume that stores data to be accessed from a host machine is called Primary Volume (PVOL). When a taking snapshot command is issued from a host machine, Secondary Volume (SVOL) is created as PVOL's backup data. It is possible to create multiple SVOLs for one PVOL. PVOL and SVOL are managed in units of chunks which are collections of blocks. A chunk No. assigned to the volume and the leading logical address of the chunk are managed in PVOL/SVOL management tables. In the SVOL management table, each chunk is managed with a share bit indicating whether it is shared with PVOL. A chunk whose shared bit is ON is called a shared chunk. When the PVOL is updated from a host machine, the shared bit of the chunk including the update target address is turned OFF. Only the difference data between PVOL and SVOL are physically stored in SSD, hence the SSDs' capacity efficiency is good in CoW. Figure 2(b) shows the PVOL update processing flow at the time of chunk copy occurrence. Metadata of snapshots are resident in the cache area of DRAM.


PVOL
SVOL0
SC
Chunk 0
Chunk 1
Shared bit: ON
Chunk 0
Chunk 1'
Shared bit: OFF
Chunk 1'
Pool
Memory
Management area
PVOL management table
Chunk No.
Logical address
0
0x1000
Cache area
Chunk 0
Chunk 1
Chunk 1'
SVOL0 management table
Chunk No.
Logical address
Shared bit
0
0x1000
ON
Flash chips
Chunk 0
Chunk 1
Chunk 1'
SSD
SC
SSD
Time
(A) Access to a shared chunk? → Yes
(B) Issue read a chunk command
(B) Read a shared chunk
(B) Write a shared chunk in a cache area
(B) Copy shared chunk to SC 0 / SC 1
(A) Update SVOL management table
(C) Receive update data in SC0/SC1
(C) Good response to host machine
(C) Issue a write command of update data
(C) Write Update data
(B) Issue a write command of shared chunk
(B) Write a shared chunk
(A): SVOL management table processing
(B): Chunk copy processing (C):PVOL update processing

(a) CoW mechanism. (b) PVOL update with chunk copy in CoW.

Figure 2. CoW

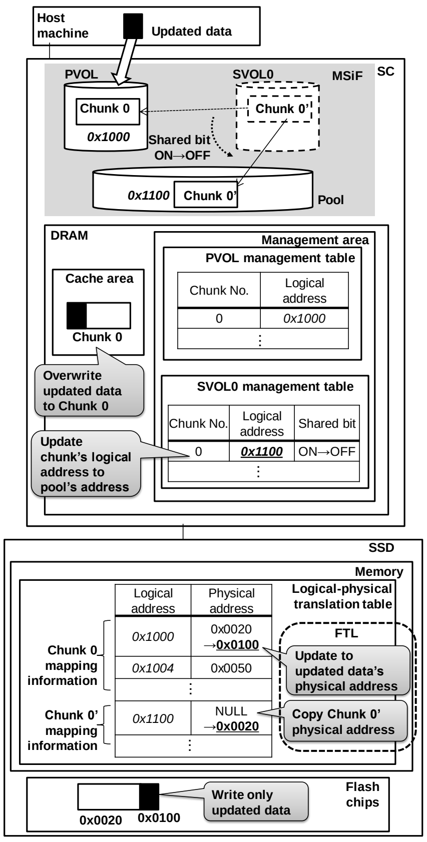
## 2.3 Performance Problem

For example, if the chunk size is set to 256 kBytes to save the DRAM capacity in a SC and the update data size from a host machine is 8 kBytes, the chunk copy processing occurs 32 times. Because of this, the path MEM bandwidth is consumed due to data transfer increases, and the neck point of the system limited performance becomes the Path MEM. When chunk copies occur, performance of the system is greatly and temporally reduced as compared with the case where chunk copies do not occur. Hence performance design and performance tuning are required in anticipation of performance degradation. Therefore, the advantage expected of AFA which is easy to perform performance design and tuning is lost. Especially immediately after taking a snapshot, since all chunks are in a shared state, chunk copies occur at most data update opportunities, and there is a problem that system performance greatly deteriorates.

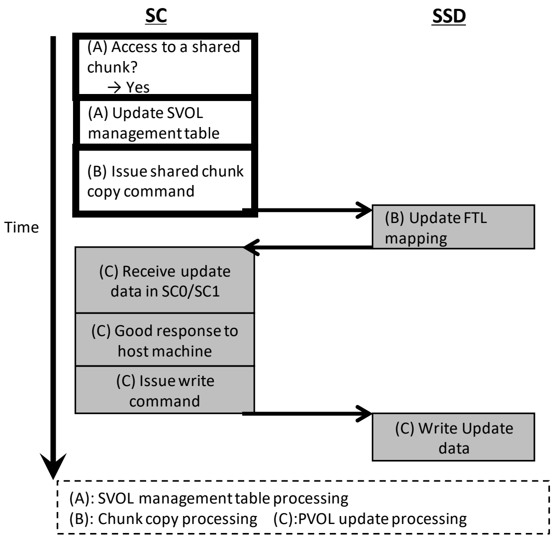
On the other hands, the data size of DRAM access at the time of updating SVOL management table is about several tens bytes, which is sufficiently smaller than the chunk size. Therefore, the amount of data transfer through the SC’s path MEM by accessing the DRAM from/to the CPU can be ignored.

## 3. PROPOSAL: MAPPING SWITCHING IN FTL

We propose Mapping Switching in FTL (MSiF) that uses SSD's CPUs to perform processing equivalent to chunk copies in SSDs. Figure 3. shows the details of MSiF. In MSiF, when the PVOL is updated from a host machine and the chunk copy processing is occurred, the copy-to physical address is set the original physical address and the copy-to physical address is set the updated data's physical address with FTL. Since no physical data copy occurs, the amount of data transfer in the SSD can be reduced. Even when taking snapshots, MSiF updates just the SVOL management table as CoW does, therefore no chunk copy occurs. The metadata in MSiF is the same as in CoW, and DRAM capacity efficiency equivalent to CoW can be realized.



(a) MSiF mechanism.


SC
SSD
Time
(A) Access to a shared chunk? → Yes
(A) Update SVOL management table
(B) Issue shared chunk copy command
(B) Update FTL mapping
(C) Receive update data in SC0/SC1
(C) Good response to host machine
(C) Issue write command
(C) Write Update data
(A): SVOL management table processing
(B): Chunk copy processing (C):PVOL update processing

(b) PVOL update with chunk copy in MSiF.

Figure 3. Proposal: MSiF

Figure 3 (a) shows the overview of PVOL update processing, and Figure 3 (b) shows the processing of PVOL update. When the SC receives the PVOL update command from the host machine, the SC's CPU checks whether the shared bit of the chunk containing the update area is ON or OFF. In the case of shared bit OFF, the SC overwrites the update data in the PVOL. While in the case of shared bit ON, the SC's CPU reserves the new logical address of the pool as the copy destination (chunk 0') for the current chunk (chunk 0), and updates the SVOL management table. The pool exists to manage only the logical address of the chunk copy destination. Therefore, it is unnecessary to allocate a cache area as an area for storing data like CoW. Next, the SC sends the chunk copy command to the SSD. This command is a unique command of Small Computer System Interface (SCSI) and is newly created for MSiF. The SSD, which receives this command, newly registers the logical address of the chunk 0 'in the logical-physical translation table and sets the physical address of chunk 0 to the physical address of chunk 0'. Then the SSD sends the complete response to the SC, and the SC implements PVOL update processing in the same way as CoW.

In the logical-physical translation table, for example, the information of chunk size 256 kBytes can be represented by 128 Bytes and this size is small enough for PVOL update data from a host machine. Thus, the DRAM access processing time by updating the logical-physical translation table at the time of a chunk copy is sufficiently short.

## 4. EVALUATION

### 4.1 Precondition

We evaluated the performance improvement in MSiF using an analytical model. In MSiF, a chunk copy is performed with SSD's CPU, and furthermore a chunk copy is replaced with address update using FTL. Therefore, to evaluate the effect on these two points, we compared MSiF to CoW and Copy in SSD (CiS). CiS is a method in which the SSD's CPU physically copies chunk in the SSD instead of the SC's CPU.

We described our evaluation precondition in Table 1. The SC internal network is based on the information of Hitachi Virtual Storage Platform F 600 (www.hitachivantara.com/en-us/pdfd/datasheet/vsp-f-series-datasheet.pdf). And SSDs' spec is based on the information of Hitachi Accelerated Flash FMD DC2 (www.hitachivantara.com/en-us/pdfd/datasheet/accele rated-flash-datasheet.pdf). Furthermore, since the average system capacity of entry level AFA is 30TByte, the system configuration capacity such as the number of SSDs is set (www.purestorage.com/content/dam/purestorage/pdf/datasheets/ps_flasharray_ds6p_02.pdf).

Table 1. Analytical Model Definitions

| Items | Details |
|---|---|
| Storage system | Entry-level AFA |
| Connection to host machine | Fiber channel 16 Gbits/s, 16 cables |
| Bandwidth of path in SC | Maximum 2 GBytes/s per cable,<br>Maximum 16 GBytes/s per path (= 16 cables) |
| Drive | 16 SSDs, 1.6 Tbytes per SSD |
| Performance per SSD | Read: 2 GBytes/s, Write: 1 GBytes/s |
| RAID level | RAID5 (7D+1P) |
| Configuration of snapshot | PVOL: SVOL=1:2, 8 PVOL / 8 SVOL per system |
| Operation of snapshot | Create 1 SVOL for 1 PVOL and delete 1 SVOL at regular intervals |
| PVOL size | 1 TBytes per PVOL |
| Benchmark | SPC-1C$^{TM}$ |

We assumed the actual snapshot operation that reduces system performance degradation due to snapshot processing; it distributes business data to multiple PVOLs and taking snapshot is performed 1 PVOL at regular intervals. Moreover, this benchmark is based on a usual business situation that the volumes to be accessed from the host machine is only PVOLs.

SPC Benchmark 1C$^{TM}$ [1]is a sophisticated and practical performance measurement workload for storage systems (Storage Performance Council, 2013). It simulates the behavior when AFA's main use case, online transaction processing, is applied to entry level storage systems. Table. 2 shows the input/output (I/O) patterns which we used for this evaluation based on SPC Benchmark 1C$^{TM}$ public information (Storage Performance Council, 2013). Since the access locality exists in the data access range, the ratio of the access range to the system capacity is set based on the SPC - 1 C$^{TM}$ specification and it is described as "Access locality" in Table 2.

Table 2. I/O Patterns

| I/O pattern | Data size [kByte] | Event probability [%] | Access locality [%] |
|---|---|---|---|
| Random read | 4.0 | 28.9 | 13.9 |
| Random write | 4.0 | 32.5 | 13.9 |
| Sequential read | 14.4 | 10.5 | 18.0 |
| Sequential write | 14.4 | 28.1 | 7.0 |

The capacity allocated as the cache area is small in AFA, this is because the metadata is mainly stored in the SC's DRAM. Therefore, in this analytical model, the data which a host machine overwrite is not in a cache area.

## 4.2 Analytical Model

### 4.2.1 Parameter Definition

Table 3 shows the processing list in a file copy. Since there is SVOLs in each PVOL, processing type 1 is executed when there is no shared chunk, and processing type 2 or 3 or 4 is executed when there is a shared chunk.

The definition of parameters is shown in Table 4. path i (i = FE, BE, SC, MEM) refers to each internal network in the SC as shown in Fig 1. In path i, the maximum bandwidth of all cables is $B_i$, and the maximum bandwidth per cable is $b_i$. In the input / output processing k (k = r, w), read processing is defined as r and write processing is defined as w. The data size a (a = $D_f$, $D_c$) indicates the data size to be processed, the input / output data size from the host machine is $D_f$, and the chunk size is defined as $D_c$.

Table 3. Processing pattern in a file copy

| Processing type j | Detail |
|---|---|
| 1 | Access to PVOL |
| 2 | Access to PVOL, with a chunk copy of CoW |
| 3 | Access to PVOL, with a chunk copy of CiS |
| 4 | Access to PVOL, with a chunk copy of MSiF |

[1] SPC Benchmark-1/Energy, SPC-1/E, SPC-1, SPC1 IOPS, SPC-1 LRT and SPC-1 Price-Performance are trademarks of the Storage Performance Council.

Table 4. Parameter definitions

| Item | Definition |
|---|---|
| S | The number of SSDs |
| N | The number of chunks in a PVOL |
| g | Bandwidth per channel cable connected to the host machine [Byte / s] |
| $B_i$ | Maximum bandwidth of path i [Byte / s] |
| $b_i$ | Maximum bandwidth per cable of i path [Byte / s] |
| $W_k$ | Maximum performance of input / output processing k per SSD [Byte / s] |
| P(t) | Chunk sharing probability at time t |
| $A_i(t)$ | Average occupancy time of path i at time t [s] |
| $H_i(t)$ | Maximum throughput in path i at time t [IOPS] |
| $SA_k(t)$ | Average processing time of one input or output processing k per SSD at time t [s] |
| SH(t) | Maximum throughput in all SSDs at time t [IOPS] |
| H(t) | Maximum throughput of the system at time t [IOPS] |
| R | Response time to host machine in one processing [s] |
| R(I) | Average response time to a host machine in throughput I [s] |
| $D_f$ | Input / output data size from a host machine [Byte] |
| $D_c$ | Chunk size [Byte] |
| $d_{ija}$ | Access count of path i with data size a in processing type j [times] |
| $Sd_{jak}$ | Number of SSD accesses in the processing type j, input / output processing k and data size a [times] |
| m | The number of times of processing through the path MEM until returning a completion response to the host [times] |
| $\lambda_t$ | The average probability of occurrence of processing through the path MEM at time t |
| V(t) | Average time dispersion of processing through path MEM at time t |
| Ta | Average processing time of processing through path MEM [s] |

### 4.2.2 Throughput Evaluation

In the AFA environment where RAID and snapshot are applied, the sufficient numbers of CPUs are mounted and the amount of data transfer increases. Hence, the system performance is evaluated by the data transfer amount in this evaluation. We define the initial state (time t = 0, P (0) = 1) as the state where all chunks are not copied immediately after taking a snapshot.

The total number of occurrences of chunk copies is l (t) at time t, the chunk sharing probability P (t) is expressed by Equation (1).

$$\mathrm{P(t)} = \frac{N - l(t)}{N} \qquad (1)$$

Differentiating the chunk sharing probability P(t) with respect to time t is $\mathbf{dl(t)/dt}$, which is the number of chunk copies occurred per second at time t. Therefore, since $\mathbf{dl(t)/dt}$ is the same as the maximum throughput Hi(t) multiplied by the chunk sharing probability P (t) at time t, it can be expressed as Equation (2).

$$\frac{dP(t)}{dt} = -\frac{1}{N}\frac{dl(t)}{dt} = -\frac{P(t)H(t)}{N} \tag{2}$$

$A_i(t)$, the average occupancy time of path i in the SC at time t, is the ratio of the amount of data flowing in the i path to the maximum bandwidth of the path i, and the average occupancy time $A_i(t)$ is expressed by Expression (3).

$$A_i(t) = \frac{D_f d_{i1D_f}(1 - P(t))}{B_i} + \frac{(D_f d_{ijD_f} + D_c d_{ijD_c})P(t)}{B_i} \tag{3}$$

Therefore, $H_i(t)$, the maximum throughput in path i of the i path in the SC at the time t, is the reciprocal of the occupation time in path i, and this becomes Expression (4).

$$H_i(t) = \frac{1}{A_i(t)} \tag{4}$$

About SSDs, $SA_k(t)$, the average processing unit price per SSD at time t, is the ratio of the amount of data flowing in the SSD to the maximum SSD performance, and this is expressed by Equation (5).

$$SA_k(t) = \frac{D_f Sd_{1D_f k}(1 - P(t))}{W_k} + \frac{(D_f Sd_{jD_f k} + D_c Sd_{jD_c k})P(t)}{W_k} \tag{5}$$

SH(t), the maximum throughput, in all SSD at time t, is obtained by multiplying the reciprocal of the processing time per one input or output by the number of SSDs, and this is shown in Expression (6).

$$SH = \frac{S}{SA_r(t) + SA_w(t)} \tag{6}$$

Therefore, H(t), the maximum throughput of the storage system at the time t, is the minimum throughput value of all paths, and this is shown in the equation (7).

$$H(t) = \min(H_{FE}(t), H_{BE}(t), H_{SC}(t), H_{MEM}, SH(t)) \tag{7}$$

### 4.2.3 Response Time Evaluation

As shown in Figure 2 and Figure 3, it is assumed that the storage system transmits a completion response to the host machine at the timing when the update data is duplicated in the cache area of SCs. In the case where the request processing from the host machine is executed once, the response time R is the total amount of data flowing through one cable of each path i divided by the maximum bandwidth of the cable. And this is shown in Expression (8). The time to create the copy destination data on the memory in a host machine is sufficiently small as 0.5 μs or less, and it is the time which is equivalent in the proposed method and the comparison target method. Therefore, in this evaluation, the average response time does not include copy creation time in the host machine.

$$R = \sum_k SA_k(t) + \sum_i \left( \frac{D_f d_{i1D_f}(1 - P(t))}{b_i} + \frac{\left(D_f d_{ijD_f} + D_c d_{ijD_c}\right) P(t)}{b_i} \right) + \frac{2g}{D_f} \qquad (8)$$

In the assumed benchmark, SPC-1C$^{\text{TM}}$, the number of accesses to a cache area in a SC is large, and path MEM becomes a performance bottleneck. Therefore, when read / write processing is continuously issued from a host machine, it is necessary to consider the waiting time of a Path MEM. On the other hand,16 paths connecting a host machine and a front-end interface of a SC are sufficient, and the waiting time of other path i is not a performance bottleneck, therefore this is not considered in this evaluation. Among the plurality of cables constituting each path i, which cable is used is equally selected by round robin or the like. Since the processing request from a host machine is issued at random, the waiting time of path MEM is evaluated with the M/G/1 queue (Bhat, 1968), and the average response time R(I) is given by Expression (9).

$$R(I) = R + \frac{m\lambda_t(I)(V(t) + T_a^2)}{2(1 - \lambda_t(I)T_a)} \qquad (9)$$

To smoothly process the number of processes requested from a host machine in a path MEM, it is necessary that $\lambda_t$ (I)$T_a$, the average traffic intensity of MEM path, indicating the degree of processing inclusion should be 1 or less. As the average traffic intensity increases, the response time also increases. Therefore, it is assumed that the whole system is optimized and $\lambda_t$(I)$T_a$ is 0.9 or less in this evaluation.

## 4.3 Performance Evaluation after Taking Snapshot

We evaluated the changes in throughput and response time with elapsed time in the case where the snapshot processing was taken for 1 out of 8 PVOLs. The processing requests from a host machine were 2 patterns: the light load and the heavy load. A host machine had throughput requirements of 30% as the light load and 70% as the heavy load against maximum throughput of storage system in the assumed benchmark. We verified 2 patterns of chunk size $D_c$: 128 kBytes and 256 kBytes. The evaluation results in the heavy load are shown in Figure 4, and the ones in the light load are shown in Figure 5.

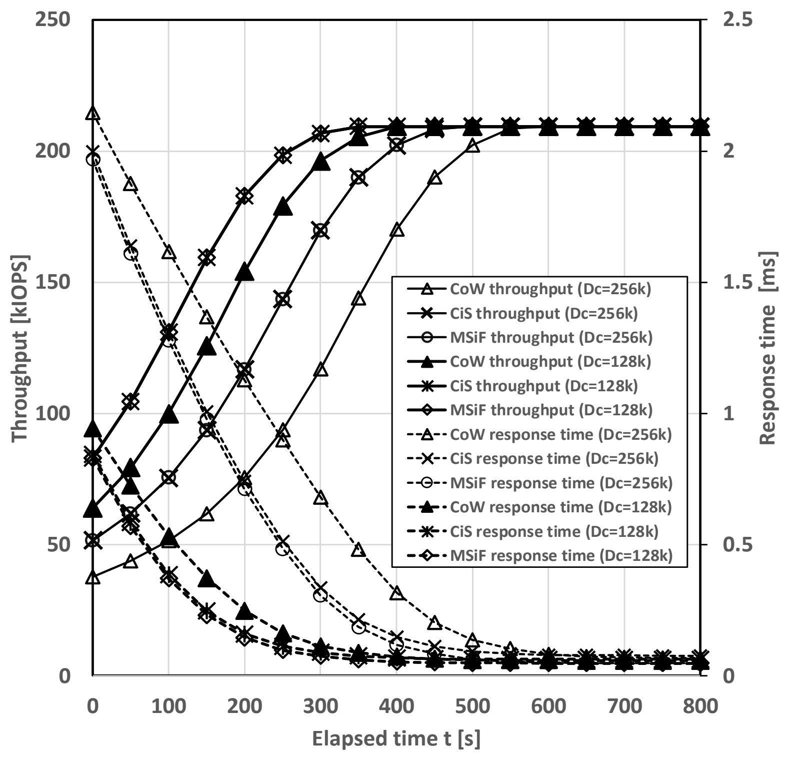


Figure 4. Performance stability in the heavy load after creating a snapshot

In CoW, the pass band in SC and the amount of data flowing in SSD increased and the throughput decreased, since chunk copies always occurred immediately after taking a snapshot. Therefore, as the chunk sharing rate decreased with elapsed time, the occurrence frequency of chunk copy decreased and the throughput gradually recovered. On the other hand, in CiS and MSiF, chunk copies don't occur in the SC. Therefore, compared with CoW, the time taken to recover the throughput of MSiF and CiS was shortened by 17.6% at the heavy load and 41.4% at the light load at chunk size $D_c$=256 kBytes, and shortened by 15.1% at the heavy load and 76.9% at the light load at chunk size $D_c$=128 kBytes. The throughput of MSiF and CiS is the same, since the number of accessed to path MEM in SC is large and it becomes a performance neck.

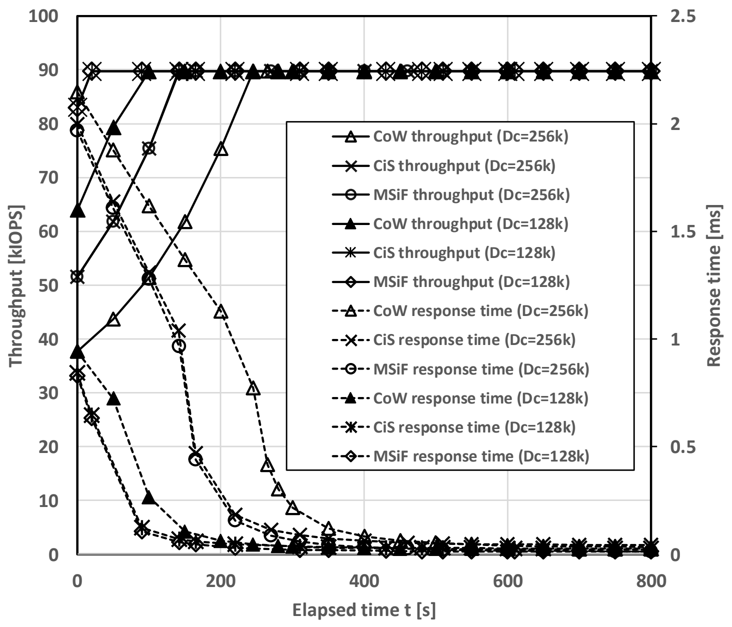


Figure 5. Performance stability in the light load after creating a snapshot

On the other hand, in response time, CiS and MSiF are superior to CoW at the beginning. However, the response time of CiS exceeds the one of CoW, this is because chunk copies in SSD increases response time with a lapse of time. Table 5 shows the result of comparing the response times (at the elapsed time t=0s, and t=800s) to confirm details of changes in response times. In case of chunk size $D_c$=256 kBytes at the heavy load, the response time at time t=0s decreased by 6.9% in CiS compared with CoW, and by 8.2% in MSiF compared with CoW. Furthermore, the response time at time t=800s increased by 13.3% in CiS compared with CoW, however it decreased by 29.4 % in MSiF compared with CoW. In the case of chunk size $D_c$=256 kBytes at the light load, the response time at time t=0s was the same as at the heavy load. On the other hand, the response time at time t=800s increased by 22.9 % in CiS compared with CoW, and it decreased by 40.0 % in MSiF compared with CoW.

We evaluated the case of other chunk sizes. As the result, the response time became smaller as the chunk size $D_c$ became smaller regardless of the method. For example, in the case of chunk size $D_c$=128 kBytes at the light load, the response time at time t=800s increased by 17.4 % in CiS compared with CoW, and it decreased by 56.5 % in MSiF compared with CoW. However, the response time of MSiF, in which data transfer by chunk copy does not occur both at the heavy load and the light load, didn't exceed the response time of CoW / CiS.

Table 5. Response time stability (unit: ms)

| Method | | | The heavy load | | The light load | |
|---|---|---|---|---|---|---|
| | | | Elapsed time t | | Elapsed time t | |
| | | | 0s | 800s | 0s | 800s |
| CoW | Chunk size $D_c$ [Byte] | 128k | 0.943 | 0.058 | 0.943 | 0.023 |
| | | 256k | 2.146 | 0.068 | 2.146 | 0.035 |
| CiS | Chunk size $D_c$ [Byte] | 128k | 0.843 | 0.063 | 0.843 | 0.027 |
| | | 256k | 1.997 | 0.077 | 1.997 | 0.043 |
| MSiF | Chunk size $D_c$ [Byte] | 128k | 0.829 | 0.048 | 0.829 | 0.013 |
| | | 256k | 1.968 | 0.048 | 1.968 | 0.014 |

## 5. RELATED RESEARCH

There is ioSnap as one of the snapshot methods for AFA (Subramanian, 2014). A high-performance requirement is stricter in AFAs than in conventional disk-based storage systems, and the snapshot processing method adopted in the disk-based storage system cannot satisfy the requirements of AFA. ioSnap uses the FTL mechanism based on RoW processing method. In ioSnap, the generation number is assigned to the update data, and the snapshot tree where the data sharing relationship between the PVOL and the multiple SVOLs is managed is created. Thus, ioSnap reduces processing overhead at data access. However, ioSnap isn't appropriate for backup of large volume data since it is based on RoW.

## 6. CONCLUSION

We proposed snapshot method MSiF which is appropriate for AFA. We evaluated the throughput and response time immediately after taking a snapshot. The benchmark is assumed SPC-1C$^{TM}$ simulating online transaction processing. Resultantly, MSiF reduced throughput recovery time by 76.9% and response time by 56.5% compared to CoW, when chunk size was Dc=128 kBytes and the light load was required from a host machine. Furthermore, the performance superiority of MSiF was confirmed regardless of chunk size and the load demand from a host machines. Therefore, MSiF is the appropriated method for AFAs.

## REFERENCES

Bhat, U.N., 1968. *A study of the queueing systems M/G/1 and GI/M/1*. Springer, Germany

Gupta, A. et al, 2009. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings, *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*, Washington, D.C., USA, Vol. 44, No. 3, pp. 229-240.

Hitachi Vantara Corporation, 2018. Hitachi Vantara Storage Platform F Series, Retrieved July 13, 2018. https://www.hitachivantara.com/en-us/pdfd/datasheet/vsp-f-series-datasheet.pdf

Hitachi Vantara Corporation, 2018. Hitachi Accelerated Flash: Speed Application Performance and Data Center Efficiency, Retrieved July. 13, 2018. From https://www.hitachivantara.com/en-us/pdfd/datasheet/accelerated-flash-datasheet.pdf

Imazaki, M. et al, 2016. Restorable Update Write by SSD Controller to Improve Destage Throughput in Enterprise Storage System. *Proceedings of Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress*, pp. 953-958.

Kim, H. and Ahn, S., 2008. BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage. *Proceedings of USENIX Conference on File and Storage Technology*, USA, Vol. 8, pp. 1-14.

Menon, J., 1994. Performance of RAID5 disk arrays with read and write caching. *Distributed and Parallel Databases*, vol. *2*, no. 3, pp.261-293.

Morishita N. et al, 2015, Storage system and storage control method, US Patent, US 2015/030713 A1.

Pure Storage Inc, (2017). FLASHARRAY, Retrieved July 13, 2018, from https://www.purestorage.com/content/dam/purestorage/pdf/datasheets/ps_flasharray_ds6p_02.pdf

Storage Performance Council, 2013. SPC BENCHMARK 1C$^{TM}$ (SOC-1C$^{TM}$) SPC BENCHMARK 1C/ ENERGYTM EXTENSION (SPC-1C/E$^{TM}$) OFFICIAL SPECIFICATION, USA, version 1.5.0

Subramanian, S. et al, 2014. Snapshots in a flash with ioSnap, *Proceedings of EuroSys'14*, no. 23

Xiao, W. et al, 2006. Implementation and performance evaluation of two snapshot methods on iSCSI target storages. *Proceedings of NASA/IEEE Conference on Mass Storage Systems and Technologies*, USA, pp.101-110.

Yi, W. et al, 2015. A flash-aware intra-disk redundancy scheme for high reliable all flash array. *IEICE Electronics Express*, *12*(13), pp.1-11.