DESIGN AND IMPLEMENTATION OF CACHE MANAGER FOR HYBRID STORAGE

Seongjin Lee, Youjip Won
Department of Electronics and Computer Engineering
Hanyang University, Seoul, Korea
E-mail: {insight|yjwon}@hanyang.ac.kr

ABSTRACT

The technology and cost barrier of the SSD motivated many research groups and companies to merge the SSD with low cost and large capacity storage device, Hard Disk Drive (HDD). As the NAND flash memory technology is getting more advanced, price of NAND flash memory dropped dramatically. As number of bits per cell in NAND flash memory is increasing, the price of $/GByte is reduced. This trend will continue until the market sought for another media as a stable and fast storage media. NAND flash based Solid State Drive (SSD) is considered as a choice for replacing Hard Disk Drive (HDD) to reduce and minimize the gap between IO latency DRAM of a storage device. In this work, we first examine three existing cache manager solutions and study if advertised measurements hold in in-house benchmark environment. We describe what has to be considered in implementing cache managers, and also describe design and implementation of a cache manager and measure its caching performance.

KEYWORDS

Hybrid Storage, Cache Manager, SSD, Performance Evaluation, Machine Learning

1 INTRODUCTION

As the NAND flash memory technology is getting more advanced, price of NAND flash memory dropped dramatically [1]. As number of bits per cell in NAND flash memory is increasing, the price of $/GByte is reduced. This trend will continue until the market sought for another media as a stable and fast storage media. NAND flash based Solid State Drive (SSD) is considered as a choice for replacing Hard Disk Drive (HDD) to reduce and minimize the gap between IO latency DRAM of a storage device. Making decision on changing a main storage device from HDD to SSD is not easy because capacity of a SSD is not as large as HDD and as cheap as HDD. Recently number of works has proposed to use SSD as a cache device for a HDD.

One most attractiveness of using a SSD as a cache device in a storage system is that it provides faster storage performance and better life time of SSD. When a SSD is used as a cache device on a storage system and used a read cache, users can enjoy faster random IO performance. As it is known, benefit of SSD over a HDD is IO performance, and power. So, SSD in a hybrid storage system not only allows a HDD to stay in spin down state but also reduce power consumption significantly [2]. Another benefit of having SSD as a cache in a storage system is shortened application launch time. A research [3] shows that by allocating files accessed in launch time to SSD speeds up the application launch process.

It seems like market has interest in hybrid storage solutions as well. There are a few manufacturers including Intel that tries to provide hybrid storage solutions. Intel provides Rapid Storage Technology which includes hybrid storage enabler called Smart Response

Technology [4]. Their product significantly reduces IO bottleneck by using SSD as cache device. Marvell [5] and SilverStone [6] are also interested in providing hybrid storage solutions with host controllers at a low price. Recently Samsung announced that they acquired storage caching software company Nvelo [7] which shows how serious the industry takes caching business in storage system. Nvelo is a software company specialized in cache managers. One of their products is DataPlex [8], a nonvolatile cache management solution. There are number of SSD form-factors that are different in size and shape to meet various criteria of vendors and users. The fact there are various form-factors makes difficult for Hardware solutions like HyperDuo [5] or HDDBoost [6] to cover all the needs. Software solutions like DataPlex [8] are certainly has meaningful position in the market.

In this work, we first examine existing cache manager solutions and study if advertised measurements hold in in-house benchmark environment. Next, we describe what has to be considered in implementing cache managers. We also provide design and implementation of a cache manager and measure its caching performance.

In Section 2, we describe the works that are done in field of hybrid storage. Section 3 compares three storage systems: HDD, SSD, hybrid disk. Section 4 describes benchmark environment and workload we used in the benchmark. Section 5 explores performance of existing cache manager solutions and compares its advertised solution against actual measurements. Section 6 describes design and implementation of cache manager, and Section 7 describes the result of using the cache manager. Section 8 concludes the paper.

2 RELATED WORK

There a number of works that tries to improve energy consumption and IO performance by using hybrid storage system that exploits SSD as a cache device. Kgil et al. [9] proposed NAND flash based disk cache in server system. The disk cache is separated into read and write regions and serves different purposes depending on types of requested IO. The system improved power consumption up to 3 times compared to HDD only system. Hsieh et al. [10] shows another flash memory based cache system that reduces energy. Their trace driven simulation shows that 20% of energy can be saved while two thirds of response time is reduced by using the flash memory. Byun [11] proposed index management scheme in hybrid storage system, which enhances search and update performance by caching data objects. The proposed scheme tries to minimize the number of update operation in flash memory. Bisson et al. [12] proposes to use flash memory to reduce long write IO latency in HDD. Their result shows that 70% of write latency can be reduced by using flash memory as nonvolatile cache.

Some of other works tries to speed up the launch time of applications by using the hybrid storage system. Joo et al. [3] proposed to reduce the application launch time by exploiting hybrid disk with pinning only a small portion of an application launch sequence into flash memory. They have come up with latency model of hybrid storage, and solved integer linear programming problem to figure out optimal pinned set that goes to flash memory. Another work done by Joo et al [13] extends their previous work [3] and presents SSD-aware application prefetching scheme.

Many of works in the field of hybrid storage caching schemes use pattern mining algorithms to efficiently relocate frequently accessed objects in storage devices. Zaki proposed SPADE mining algorithm that discovers pattern sequences. SPADE algorithm discovers all sequences in only three database scans [14]. Ayres et al [15] improves SPADE by exploiting bitmap representation of the data for efficient counting. Yan et al. [16] proposed data mining technique called CloSpan to improve IO performance in storage systems by exploiting correlations among blocks in storage systems. CloSpan finds only closed frequent subsequences which are a subsequence whose support is not equal to support of its super-sequences. Li et al. [17] observed that CloSpan only produces the closed frequent subsequences rather than every frequent subsequence, which limits the number of the frequent subsequences. To resolve issue in

CloSpan, Li et al. proposed more scalable and space-efficient algorithm using non-overlapping and relatively large cutting window size.

3 BACKGROUND

3.1 Storage System

In general, storage devices are characterized by their capacity, bandwidth, throughput, and latency, which are measured in GB or TB, MB/sec, IOPS, and msec or μsec, respectively. This section describes difference in storage devices, and their characteristics.

HDD: HDD is composed of magnetic disk with mechanically actuated read/write recording heads. When the magnetic disk rotates, a concentric virtual circle called track is created with radius of defined by location of the recording head on a disk. The head picks up signals from magnetized track from the disk. A track is series of sectors of consecutive bits with same radius. IO latency of a HDD depends on how fast it can switch between tracks and find requested sector, which is the process called seek. Since essential parts of HDD are managed by mechanical components, it is inevitable that they are prone to sensitive to noise, vibration, and shock. Figure 1(a) shows a brief structure of HDD in logical IO hierarchy.

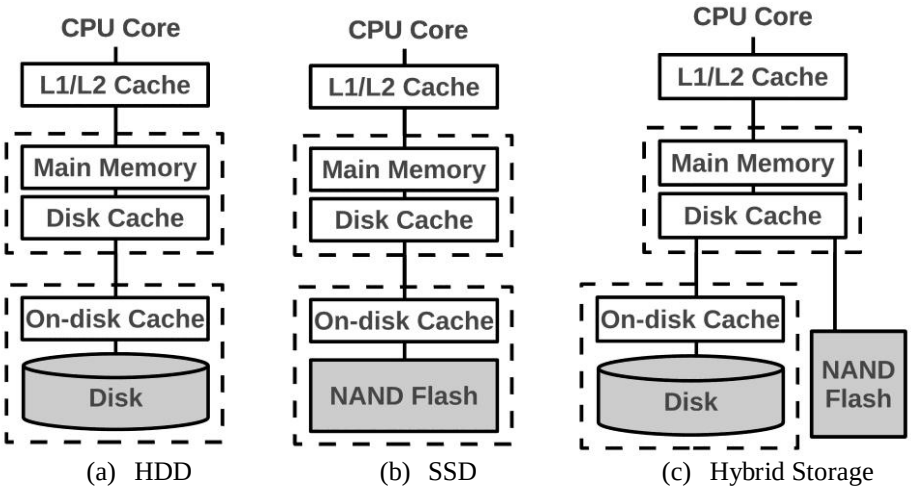
SSD: Figure 1(b) illustrates a brief structure of SSD in IO hierarchy. Since SSDs are composed of NAND flash memory, it has high bandwidth in random IOs, low power, low noise, and fast response time. SSDs use Serial-ATA interface which allows host system to recognize it as a block device, which allows maneuvering existing IO subsystem. However, there are significant distinction between SSD and HDD. What is special about SSD is that it cannot in-place update a page which is basic read and write unit of operation. Instead, updated page has to be out-of-place updated. For example, upon receiving update request on a page, the requested page is invalidated; a new page is allocated for the data. There is one additional unit of operation. Erase operation is performed on a block, which is group of pages. In general, the number of pages in a block is 128 or 256, depending on choice of manufacturers. These units of operations have different latency. Table 1 compares initial operating time, delay time, power, and noise.

Hybrid Disk: Figure 1(c) illustrates brief structure of a hybrid disk. The hybrid disk combines NAND flash media and magnetic disk as a single storage device. HDD uses the NAND flash media as a large buffer to store cache data. The cache device is not visible to the user; thus, the size of hybrid disk is dependent on the main storage device. However, physical size of the hybrid disk should include the size of cache device as well.

Although many researchers have worked on hybrid disks, there are still numerous issues that have to be solved and optimizations to be made. There are at least four issues in designing a hybrid disk. First, it is important to determine what attributes are stored in non-volatile memory in the hybrid disk. Attributes may be metadata, file, or logical block address. Length or size of the attribute also has to be determined. For example, non-volatile memory may store only partial of a file, or a whole file. Second issue is to identification of attributes to be stored. It is important to maximize the hit ratio and reduce IO latency via exploiting the cache media, which is non-volatile memory in the hybrid disk. It would be preferable to have a silver bullet for identifying the attributes in all sorts of workloads but in general, every application tends to create unique IO requests. Therefore, hybrid disk needs a good machine learning algorithms with low computational overhead that identifies the attributes on-the-fly. Third issue in hybrid disk is to determine when and how identified attributes are relocated to cache and flushed back to main storage device. This leads to fourth issue which is managing wear-level of non-volatile memory. NAND flash is a fast device but with limited erase count.

4 ENVIRONMENT

We conducted all our experiments on a machine with an Intel 3.3GHz i5-2500, 256KB of L1 cache, and 6MB of L2 cache. The motherboard was a



(a) HDD (b) SSD (c) Hybrid Storage

Figure 1 Disk models and their possible logical structures

Table 1 Device Specification

Spec.	SSD [18]	HDD [19]
Initial Operating Time	1.5s	8s
Delay, Search Time	Read Latency: 65us Write Latency: 85us	Avg. search: 12ms Avg. delay: 4.2ms
Power	Active: 150mW Idle: 75mW	Active: 2500mW Idle: 960mW
Noise	N/A	Idle Mode: 2.3 Bels Search Mode: 2.5 Bels

Gigabyte Z68X. The machine contained 4GB of DDR3 RAM. The system disk was a 7200 RPM Western Digital (SD10EALX) with 1TB capacity. The specification of computing environment is briefly listed on Table 2.

4.1 Workload

We captured three applications to test the performance of cache manager. Applications are (i) Boot up (Win7), (ii) Spreadsheet (Excel), and (iii) Presentation (Powerpoint). Description of tracing of each application is shown in Table 3. Figure 2 illustrates request length of read and write IO of each application.

5 EFFECT OF CACHE MANAGER ON HYBRID DISKS

In this section we review four different cache management solutions and compare with a HDD to illustrate importance of choosing a good cache

Table 2 Environment

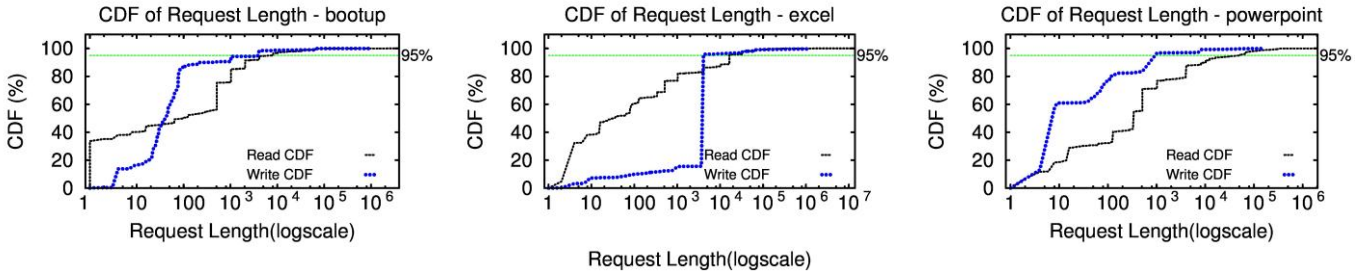
Device	Model
CPU	Intel i5-2500 3.3Ghz
RAM	DDR3 1333MHz 4G
M/B	Gigabyte Z68X-UD3H-B3
HDD	WD SD10EALX (1T, SATA3, 7200 RPM, 32M)
SSD	OCZ-Solid3 60G (SATA3, 20k IOPS)
O/S	Win7 32bit Home Premium English ed.
Benchmark	PCMark Vantage 1.0.2
HyperDuo [5]	Highpoint RocketRAID 62X (Safe Mode)
HDDBoost [6]	SilverStone HDDBoost Rev-1.02
SRT [4]	Rapid Storage Technology 10.6.0.1002 (Enhanced Mode)

manager. We evaluate three hybrid disk solutions available in the market.

5.1 Cache Manager Solutions

HyperDuo: Figure 3(a) illustrates structure of HyperDuo [5] with 88SE9130 host controller which supports two 6Gb/s SATA peripheral interfaces and a 5.0 Gb/s PCIe host interface. HyperDuo is connected to host computer with PCIe host interface and two heterogeneous storage devices, SSD and HDD, are connected to HyperDuo with SATA interface. Note that there are mismatch in maximum transmission speed of two interfaces. HyperDuo combines two heterogeneous storage devices in RAID 0 or 1 depending on the mode that user makes. There are two choices in using HyperDuo. First choice is to use it in capacity mode, which concatenates two heterogeneous storage devices, and second choice

ISBN: 978-0-9853483-3-5 ©2013 SDIWC



(a) Boot up (b) Spreadsheet (c) Presentation

Figure 2 CDF of request length of applications (Unit: sectors)

Table 3 User Workload Scenario

Workload	Scenario
Boot-up	Booting Win7 home. Log on and stay idle for 3 minutes
Spreadsheet	Executing Excel and maintain idle time for 1 minute
Presentation	Executing Powerpoint and maintain idle time for 1 minute

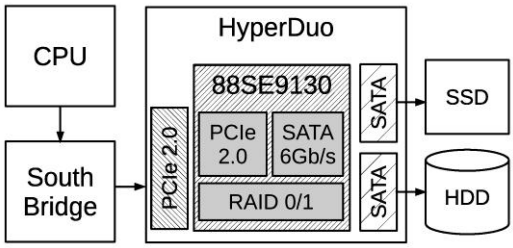
is to use safe mode, which mirrors the content on the storage. For example, when 60GB SSD is used with 500GB HDD, the total size of disk is 560GB on Capacity mode and 500GB on Safe mode. Purpose of SSD in this system is to store hot files in SSD [20]. HyperDuo determines Hot files to load to SSD. Some categories of files that it relocates are files from Office, Media player, Adobe Creative Suites, iTunes, Internet Browsers, and OS-related files.

HDDBOOST: Figure 3(b) illustrates structure of HDDBoost [6] with three SATA ports. HDDBoost is a RAID 1 controller with data caching ability. HDDBoost modifies the traditional RAID 1 controller to mirror the HDD storage area to the SSD. Unlike HyperDuo, HDDBoost uses SATA interface and provides mirror mode only. HDDBoost connects host system and two heterogeneous storage devices with SATA interface. At the initial phase, fixed area with size of SSD is mirrored to SSD. In a way to preserve program/erase cycle of SSD, HDDBoost reads from SSD first but writes to HDD first and accesses the other device. HDDBoost exploits the fact that random IO is fast in SSD, and seek time in HDD is time consuming.

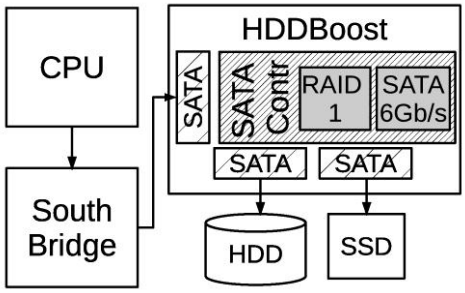
SRT: Smart response Technology, SRT [4] tries to provide SSD-like system responsiveness by directly embedding its core technology in Z68 chipset. The motherboard chipset includes RAID controller. Two heterogeneous devices are attached to motherboard without any other extra peripherals because all are integrated to the motherboard. There are two modes of operation available in SRT enabled motherboard: Enhance mode and Maximized mode. Enhanced mode uses Write-thru and Maximized mode uses write-back mode. There might be performance gain in using Maximizing mode that uses write-back. Unlike storage system with volatile cache, SRT uses SSD as a cache storage, which means it has tolerance against power failures and corruption of data.

5.2 Performance of the Cache Solutions

We used benchmarking environment illustrated in Table 2. PCMark Vantage 1.0.2 is used to benchmark different cache manager solutions. Table 4 illustrates advertised performance of given cache manager solution and measured performance. The in-house benchmark result shows that HyperDuo and HDDBoost show about half the advertised performance, whereas SRT shows about 20% better performances than the advertised. Note that models used in the benchmark are not the same; however, SSD used in in-house benchmark is a higher throughput device and yet the measurements read lower scores on PCMark Vantage. Only SRT seems to reflect the performance gap in using higher throughput device.



(a) Hyper Duo



(b) HDDBoost

Figure 3 Structure of Cache Manger Solutions

Table 4 User Workload Scenario

	Factory	Adv.	In-house	Measured
HyperDuo	X25-M	25,000	OCZ-Solid3	13.676
HDDBoost	OCZ800EX	10,600	OCZ-Solid3	6,136
SRT	Intel SSD 311	16,548	OCZ-Solid3	19,852

6 CACHE MANAGER

In this section, we explain our implementation of cache manager for hybrid storage system. Purpose of cache manager is to read in an arbitrary IO trace file to test and optimize various cache algorithms and to provide insights into how a cache must be designed. As an output of the cache manager, we acquire hit ratio and response time of read and write lOs. There are three parts to the cache manager. First is IO tracer, which acquires IOs from a running system. We traced IO traces from number of applications. Second is mining algorithm, which finds a frequently accessed sector address and makes sequences of frequently accessed sectors. Third is address checker which combines the result of the tracer and the mining algorithm to measure hit ratio and response time of IO requests.

6.1 Implementation of Tracer

Trace used in this paper is acquired using DiskMon

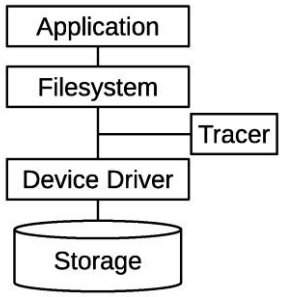


Figure 4 Trace Capturing Tool

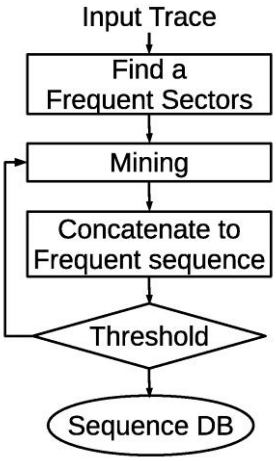


Figure 5 Mining Algorithm

[21] from the same environment described in Table 2. Trace file acquired from DiskMon is not directly applicable to mining algorithm or to cache manager because output of the program has many features that are not required in our application. DiskMon captures all of IO activities under file system as shown in Figure 4. Table 5 shows data format that DiskMon produces. Raw output of DiskMon is parsed and converted to a format that the mining algorithm can read. Note that raw output of DiskMon produces offset of a file not actual sector address of the file. Our application consults Master File Table in NTFS to locate the sector address of the file.

6.2 Implementation of Mining Algorithm

Purpose of mining algorithm in the cache manager is to find frequently accessed sectors and relocate them to SSD. Every application has set of files or sectors that are frequently accessed. Some example of accessed files is various library files with extension of "dll." According to our experiments, about 50 files are accessed during a

ISBN: 978-0-9853483-3-5 ©2013 SDIWC

Table 5 Raw Data Structure of DiskMon

Field	Description
Relative Time	Relative time of observed IO request
Duration	Duration of processing IO request (unit: usec)
Process Name	Name of the process that called from the app.
Operation	IO type
Category	Indication of read or write operation
Path	Path of a file that requested IO
Detail	Information on offset and size of IO

launch time of a spreadsheet program and they are accessed 946 times in the launch time. Top five files accessed by the spreadsheet program issues about 230 IO requests, which is about one fourth of total IO requests. The mining algorithm computes to figure out these frequently accessed sectors and turn them into a sequence of frequently accessed sectors.

Parsed trace from the tracer is used as input trace file for mining sequence of frequently accessed sectors. A naïve way to finding frequently accessed sectors is to count frequency of each sector. Frequency counting is a simple and may serve the purpose of generating table of frequently accessed sectors; however, it not only time consuming and does not reflect IO access behavior of an application.

A better way to reflect IO access behavior of an application is to find frequently accessed sequence of sectors. Figure 5 illustrates process of finding frequently accessed sectors. There are four key parameters that the mining algorithm uses to generate the table of frequently accessed sequence of sectors: Window size notifies large sequence of sectors that are analyzed in a mining process. Cutting window size determines a sub-group of sectors within the window to generate sequences of sectors for finding frequent sectors. Minimum support determines if a sequence is qualified as a sequence with meaningful access count. And, Maximum gap determines distance of each sequence, which prevents from overlapping of sequences.

The algorithm first segments given trace file into a window. Then, it looks for most frequently accessed sector address. Upon finding frequently accessed sector from each cutting window size, the mining algorithm computes to find sequence of sectors that are most frequently accessed within

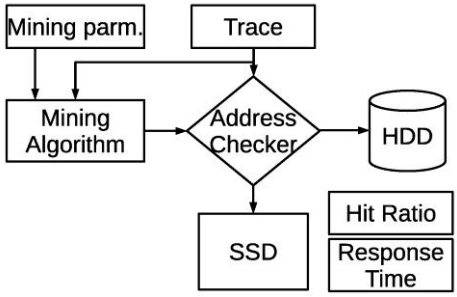


Figure 6 Structure of Cache Manager

a gap. Second sector in the sequence is determined by comparing frequent sector in all of the window groups. Second sector is concatenated to the first sector. The mining process continues until it cannot find a sector with frequency count larger than a threshold. When the process ends, we are left with a sequence DB, which holds series of frequent sequences.

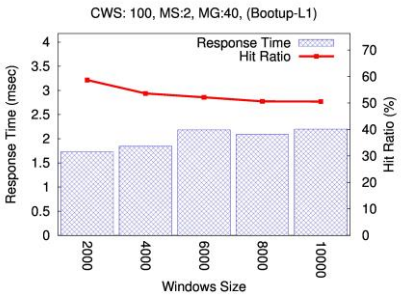
6.3 Implementation of Cache Manager

Figure 6 illustrates structure of the cache manager. A trace file generated from the tracer is input to mining algorithm and address checker module. As explained in previous section, mining algorithm generates series of frequently accessed sequence of sectors. The input trace file and result of mining algorithm is process in address checker module.

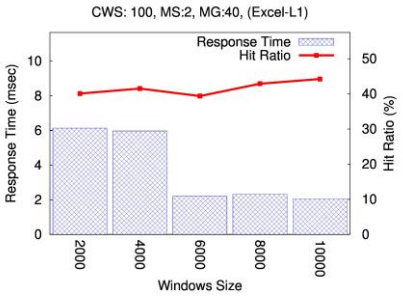
The address checker module has two components, cache loader and data allocator. Cache loader keeps a mapping table to identify which data in HDD is stored in which page in SSD. Since there are two physically independent storage devices, a mapping table is required to keep track of location of physical block address and page of HDD and SSD, respectively.

Another use of the mapping table is to exploit information written in the table for wear-leveling of SSD. Along with physical address and page number of devices, it also notifies access count, state of the sector, and sequence number on the DB. There are three states, empty, valid, and dirty.

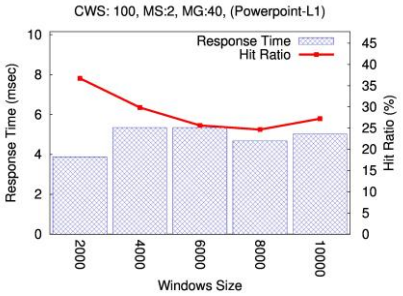
Data allocator copies data on HDD to SSD and updates mapping table on cache loader. When host requests data the address checker consults mapping table. If the requested data is in SSD, the data is fetched from SSD and returns to the host. If the data is in HDD, it serves from HDD. Upon



(a) Boot up



(b) Spreadsheet



(c) Presentation

Figure 7 Hit Ratio and Response Time

receiving IO request, address checker keeps track of time that each device has received and processed each IO.

7 RESULTS

Figure 7 illustrates the result of running mining algorithm in the cache manager. We examine boot up process and two office applications, Excel and Powerpoint. Two measurements are shown in the y-axis. Average response time is shown on the left hand side of the graph and hit ratio in percentile is shown on the right hand side. To distinguish the two results, average response time is given in bar and hit ratio is depicted in a line. Mining parameters used in the experiment is shown in heading and x-axis of the graph. Cutting window size is 100, minimum support is set to 2, and maximum gap is set to 40. We varied window size from 2,000 to 10,000 in multiples of two.

In boot-up workload (Figure 7(a)), as the window size increase response time increased and hit ratio is decreased with our mining algorithm. It is because sequences are not frequently accessed again in the boot-up process, even though there are more frequent sequences generated and stored in the SSD. Response time difference on window size of 2,000 and 10,000 is only 0.5msec, and about 8% difference on the hit ratio. On the other hand, spreadsheet (Figure 7(b)) has lot to gain from our cache manager because as window size increases response time has dropped about one thirds. Hit ratio improved only about 4% when window size is increased to 10,000. Considering that there are about 50 files accessing during the launch time, it seems we achieved reasonably high hit ratio. Response time of Presentation tool (Figure 7(c)) has increased about 2msec as the window size increased from 2,000 to 10,000, and hit ratio dropped about 10%. This result seems to be like the case with Boot-up workload. It does not benefit much from using our mining algorithm that searches frequently accessed sequence of sectors.

7 CONCLUSION

Although price of SSD is becoming cheaper, it is still hard to win over the GB/$ of HDD. Some manufacturers have turned their focus to use SSD as cache. Four related products, namely Dataplex from Nvelo, HyperDuo from Marvell, HDDBoost from SilverStone, and Smart Response Technology from Intel, are reviewed in this document. These products are some of highly acclaimed cache system for HDD using SSD. In this document, we have taken initiative to analyze the aforementioned products and provide benchmark results on our own test environment. Test results shows that Smart Response Technology gives highest and closest performance result as test configuration, and other two cache managers did not perform as expected. After explaining four points that has to be considered in designing hybrid storage, we discussed design and implementation of cache manager for hybrid storage. We implemented three core modules in

ISBN: 978-0-9853483-3-5 ©2013 SDIWC

cache manager to examine the performance of hybrid storage, and three modules are tracer, mining algorithm, and address checker. We have captured traces in Boot-up process, spreadsheet, and presentation and used them as an input to the cache manager. Result shows that mining algorithms finds frequent sequence of accessed sectors and improves storage performance by allocating them to SSD.

As a future work, we are going to refine the cache simulator and validate it more rigorously. First, mining algorithm can be improved by refining its sequence generating algorithm. Second, reading and writing to SSD and flushing back to HDD can be optimized to reduce cache manager overhead of finding the location in the mapping table. Third, although we believe it is a good caching algorithm for improving launch time of an application, this paper lacks in comparison with other caching algorithms. We are going to compare the algorithm with other caching algorithms to improve the algorithm described in this paper.

ACKNOWLEDGMENT

This work is sponsored by IT R&D program MKE/KEIT. [No.10035202, Large Scale hyper-MLC SSD Technology Development].

REFERENCES

[1] D. Reinsel and J. Janukowicz, "Datacenter SSDs: Solid footing for growth," ed: IDC Corporation, Tech. Rep., 2008.

[2] R. L. J. C. Dave B. Anderson, Walter Fry, "Flash Cache Form Factors & HDD Economics," in Flash Memory Summit 2010, 2010.

[3] Y. Joo, Y. Cho, K. Lee, and N. Chang, "Improving application launch times with hybrid disks," in Proc. of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis, Grenoble, France, Oct. 2009, pp. 373–382.

[4] Intel, "Smart Response Technology," 2012. [Online]. Available:http://www.intel.com/content/www/us/en/solid-state-drives/ssd-smart-response-video.html

[5] Marvell, "HyperDuo," 2012. [Online]. Available: http://www.marvell.com/storage/system-solutions/sata-controllers/hyperduo/

[6] SilverStone, "HDDBOOST," 2012. [Online]. Available: http://www.silverstonetek.com/product.php?pid=245

[7] Nvelo, "Samsung Electronics Acquires NVELO : Acquisition Adds Storage Software Expertise to Next-generation SSD Solutions," Dec. 2012. [Online]. Available: http://www.nvelo.com/company/press-release/2012-12-14

[8] ——, "Dataplex," 2012. [Online]. Available: http://www.nvelo.com/dataplex

[9] T. Kgil, D. Roberts, and T. Mudge, "Improving NAND flash based disk caches," in Proc. of the 35th International Symposium on Computer Architecture, Washington, DC, USA, June 2008, pp. 327–338.

[10] J. W. Hsieh, T. W. Kuo, P. L. Wu, and Y. C. Huang, "Energy efficient and performance-enhanced disks using flash-memory cache," in Proc. of the 2007 international symposium on Low power electronics and design, Portland, OR, USA, Aug. 2007, pp. 334–339.

[11] S. Byun, "Enhanced index management for accelerating hybrid storage systems," Journal of Convergence Information Technology, vol. 4, no. 2, pp. 164–169, 2009.

[12] T. Bisson and S. A. Brandt, "Reducing hybrid disk write latency with flash-backed i/o requests," in Proc. of the 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Washington, USA, Oct. 2007, pp. 402–409.

[13] Y. Joo, J. Ryu, S. Park, and K. Shin, "Fast: quick application launch on solid-state drives," in Proceedings of the 9th USENIX conference on File and Storage Technologies, 2011, pp. 19–19.

[14] M. Zaki, "Spade: An efficient algorithm for mining frequent sequences," Machine Learning, vol. 42, no. 1, pp. 31–60, 2001.

[15] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in Proc. of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, July 2002, pp. 429–435.

[16] X. Yan, J. Han, and R. Afshar, "Clospan: Mining closed sequential patterns in large datasets," in Proc. of the Third SIAM International Conference on Data Mining, 2003, pp. 166–177.

[17] Z. Li, Z. Chen, S. M. Srinivasan, and Y. Zhou, "C-miner: Mining block correlations in storage systems," in Proc. of the 3rd USENIX Conference on File and Storage Technologies, San Francisco, CA, April 2004, pp. 173–186.

[18] Intel, "Intel x25-m mainstream sata solid-state drives," 2009.

[19] Seagate, "Momentus laptop hard drives," 2010. [Online]. Available: http://www.seagate.com/www/en-us/products/laptops/laptop-hard-drives/#tTabContentSpecifications

[20] Marvell, "Marvell hyperduo for 6gb/s sata controllers: Automated ssd/hdd tiering: 80% SSD Performance at 1/3 the Cost http://www.marvell.com/storage/system-solutions/assets/Marvell-HyperDuo-Product-Brief.pdf

[21] Microsoft, "Diskmon for windows v2.01." [Online]. Available:http://technet.microsoft.com/en-us/sysinternals/bb89664