A Virtual Storage Environment for SSDs and HDDs in Xen Hypervisor[*]

Yu-Jhang Cai
Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan
m10002113@mail.ntust.edu.tw

Chih-Kai Kang
Research Center for Information Technology Innovation
Academia Sinica
Taipei, Taiwan
akaikang@citi.sinica.edu.tw

Chin-Hsien Wu[†]
Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan
chwu@mail.ntust.edu.tw

ABSTRACT

Nowadays, virtualization is a popular technology to provide an abstract hardware emulation due to the increasing hardware speed and capabilities such as multi-core CPUs, large-size main memory, and high-bandwidth networking. Virtualization technology enables multiple virtual machines to run on a physical machine, where each virtual machine can run independently and own its operating system. In particular, I/O performance will be an important factor of virtualization technology. Current popular storage devices contain traditional hard-disk drives (HDDs) and solid-state drives (SSDs). Although HDDs can provide a more economical solution than SSDs, SSDs can provide high I/O performance and power saving, especially for random I/O accesses. In the paper, we will build a virtual storage environment for SSDs and HDDs in Xen hypervisor. With the proposed virtual storage environment in Xen hypervisor, we can receive and analyze I/O requests from multiple virtual machines, and perform I/O requests to any physical storage devices. According to the experimental results, the proposed method can include both SSDs (i.e., fast access) and HDDs (i.e, low cost) in virtualization environment. Overall, the proposed method can provide an adjustment mechanism in I/O performance for those (soft real-time) applications that require high I/O performance in virtualization environment.

Categories and Subject Descriptors

C.3 [**Special-Purpose And Application-Based Systems**]: Real-time and embedded systems; D.4.2 [**Operating Systems**]: Storage Management: Secondary storage; B.3.2 [**Memory Structures**]: Mass storage

General Terms

Design, Performance, Algorithm

Keywords

Virtualization, Solid-State Drive (SSD), Xen Hypervisor

[*]Copyright is held by the authors.
Workshop on Virtualization for Real-Time Embedded Systems (VtRES 2013).

[†]Corresponding Author: Chin-Hsien Wu. This work is also supported in part by a research grant from the National Science Council under Grant NSC 100-2628-E-011-014-MY3.

1. INTRODUCTION

Virtualization technology can allow multiple independent virtual machines to run their own operation systems in a single physical machine. With the virtualization technology, a software layer (called hypervisor) is used to manage hardware resources into virtual resources for virtual machines, and can allocate appropriate virtual resources according to each virtual machine's demand. For example, a hypervisor in Xen must handle physical CPU scheduling and main memory allocation, and enforce isolation for each virtual machine. In particular, I/O performance will become an important factor for the hypervisor in terms of resource adjustment. Current popular storage devices contain traditional hard-disk drives (HDDs) and solid-state drives (SSDs). Although HDDs can provide a more economical solution than SSDs, SSDs can provide high I/O performance and power saving, especially for random I/O accesses.

In the paper, we will build a virtual storage environment for SSDs and HDDs in Xen hypervisor. A revised Xen hypervisor will play an important role of receiving and analyzing I/O requests from virtual machines, and performing I/O requests to SSDs and HDDs. Therefore, we will create a monitor mechanism in Xen hypervisor to receive and analyze I/O requests from multiple virtual machines. Based on the monitor, we can determine which I/O requests will be transferred to SSDs or HDDs. We will also create a content mapping table to record which data are located in SSDs or HDDs. Therefore, the monitor and the content mapping table can help the Xen Hypervisor to receive and analyze I/O requests from multiple virtual machines, and perform I/O requests to any physical storage devices. We believe that the proposed virtual storage environment can provide users a platform to include various storages such as SSDs (i.e., fast access) and HDDs (i.e., low cost). According to the experimental results, we can find out that the proposed virtual storage environment using SSDs and HDDs can not only provide better performance than only HDDs but also provide a more economical solution than only SSDs. Overall, the proposed virtual storage environment can provide an adjustment mechanism in I/O performance for those (soft real-time) applications that require high I/O performance in virtualization environment.

The rest of the paper is organized as follows: Section 2 provides an overview of background knowledge. Section 3 is the motivation. Section 4 is the related work. Section 5 presents a virtual storage environment for SSDs and HDDs in Xen hypervisor. Section 6 provides the experimental evaluation. Finally, Section 7 is the conclusion.

2. BACKGROUND KNOWLEDGE

Virtualization technology provides the hardware simulation which other softwares (e.g. operating systems) can run on. In other words, virtualization technology enables multiple virtual machines to run on a physical machine, where each virtual machine can run independently and own its operating system in a logically distinct environment. Virtualization is not a new concept and has been in use for decades. However, virtualization is more popular now than ever because it is convenient and flexible for IT administrators. Today, many hypervisors or virtual machine monitors (such as VMware[1], VirtualBox[2], Windows Hyper-V[3], Xen[4], and KVM[5]) can run on top of physical machines and schedule the execution of virtual machines. Multiple instances of a lot of operating systems may share the virtualized hardware resources.

Xen [4] is developed by the University of Cambridge Computer Laboratory and uses para-virtualization in its virtual machines because of the performance and administrative advantages. Since para-virtualization needs to modify guest OS kernel, Xen adopts Hypercalls as system calls for guest OSs to avoid simulating complex instruction sets. In the aspect of I/O virtualization, Xen uses front-end driver and back-end driver to improve I/O performance by sharing pages in main memory. When processor technology (such as Intel Virtualization Technology and AMD Secure Virtual Machine) for virtualization is added, Xen starts to support full virtualization after version of 3.0. Xen's virtualization architecture is divided into three parts: Xen hypervisor, Domain 0, and Domain U.

2.1 Xen Hypervisor

Xen hypervisor is an abstraction layer between the guest domains and the physical hardware and is responsible for allocating and controlling resources (such as CPU scheduling and main memory allocating), and enforcing protection and isolation. Xen hypervisor defines the communication interface for virtual machines but it does not contain the drivers of hardware devices (such as network card and graphics card). Xen hypervisor could not cause too much overhead, because it can directly access hardware resources and avoid simulating complex instruction sets.

2.2 Domain 0

Domain 0 is a modified Linux kernel and is only one privileged virtual machine. Domain 0 can directly access the hardware resources, and contain drivers for the hardware resources. Domain 0 also provides back-end driver that is responsible for receiving and handling all I/O requests of virtual machines. In fact, Domain 0 is the only one that can issue control commands to hardware, and other virtual machines need Domain 0 to access the hardware resources.

2.3 Domain U

Domain U is a general virtual machine. Since Domain U is a unprivileged virtual machine, it can not directly access the hardware resources. Domain U can send I/O requests through front-end driver, and back-end driver in Domain 0 will handle and perform the I/O requests.

3. MOTIVATION

We use a SSD (Intel 320 series 160GB) and a HDD (Hitachi HDS721010CLA332 7200RPM 1TB) as the storage devices and use Iometer [6] to test I/O performance under various sizes of request accesses (i.e., random accesses and sequential accesses). Overall, the performance of Intel SSD is better than Hitachi HDD. Intel SSD for random accesses has better performance than Hitachi HDD because random accesses could cause long mechanical latency time for Hitachi HDD. However, sequential accesses could favor Hitachi HDD, especially when the request size of sequential writes is increased. This is because large write requests could cause unexpected activities of garbage collection in Intel SSD and reduce its performance. Intel SSD for sequential reads and writes is about 4 times faster than Hitachi HDD with 4KB request size. When request size is increased to 512KB, Intel SSD is about 2 times faster than Hitachi HDD in sequential accesses. However, we observe that Intel SSD for random accesses has better performance than Hitachi HDD when request size is small. Intel SSD for random reads and writes is about 35 times and 40 times faster than Hitachi HDD with 4KB request size, respectively. When request size is increased to 512KB, Intel SSD is about 4 times faster than Hitachi HDD in random accesses. Moreover, when request size is above 64KB, the performance of Intel SSD increases slowly. According to the above experimental results, when some (soft real-time) applications in virtual machines require high I/O performance, SSDs (solid-state drives) can be used because of its better performance than traditional HDDs (hard-disk drives) in terms of read and write accesses. Although HDDs can provide a more economical solution than SSDs, SSDs can provide better performance than HDDs, especially for (soft real-time) applications with a lot of random accesses. Therefore, we will build a virtual storage environment for SSDs and HDDs in Xen hypervisor. We believe that the proposed method can provide users a platform to include various storages such as SSDs (i.e., fast access) and HDDs (i.e., low cost).

4. RELATED WORK

Many previous studies about hybrid storage systems often use SSDs as a secondary-level cache [12, 13, 14, 15]. [12] considers that SSDs are suitable to be placed between main memory and HDDs as a second-level cache rather than to store data. [13] also uses SSDs as a second-level cache to buffer data which stored in HDDs, but thinks not all data should be buffered in SSDs. Therefore, [13] sets a threshold value to filter out inappropriate data and only buffers data with smaller request size. [14] uses a LRU (Least Recently Used) method to filter out data which are least recently used, and removes data according to the access characteristics of flash memory. [15] tracks the count of block accesses, and identifies frequently used parts, and then caches the frequently used data in SSDs. When the capacity of SSDs grow larger and cheaper, SSDs are suitable to be a storage device rather than cache and some research [16, 17, 18, 19, 20, 21] tries to integrate SSDs and HDDs into one storage system. Different from the previous work, the objective of the paper will focus on the implementation and discussion on how to revise Xen hypervisor to provide a platform using SSDs and HDDs in virtualization environment. Note that our current work [22] is also based on the virtual storage environment to implement a hybrid storage access framework and evaluate its performance for virtual machines. To strengthen the paper's contributions, we write the virtual storage environment in the paper.

5. A VIRTUAL STORAGE ENVIRONMENT FOR SSDS AND HDDS

5.1 Overview

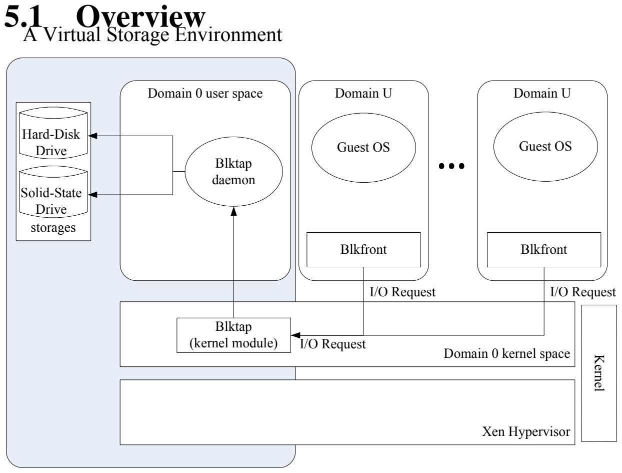

Figure 1: System Architecture

Figure 1 is the system architecture, and we build a virtual storage environment for SSDs and HDDs in Xen hypervisor by modifying the Blktap driver. With the virtual storage environment, even if multiple virtual machines run different guest operating systems, we do not need to modify the guest operating systems for the management of SSDs and HDDs. Some previous method is using upper-level applications to figure out which data should be transferred to SSDs or HDDs. For example, multimedia files (e.g., video and pictures) could result in more sequential accesses and can be transferred to HDDs. Some files are randomly accessed and can be transferred to SSDs. However, any virtual machine could run any operating system and applications, and specific upper-level applications are not suitable for the management of SSDs and HDDs. In order to avoid modifying (guest) operating systems, we propose a virtual storage environment in Xen hypervisor by revising related components (e.g., tap-disk and block-aio). With the virtual storage environment, we can receive and analyze I/O requests from multiple virtual machines and determine which I/O requests should be transferred to SSDs or HDDs without any modifications of (guest) operating systems.

5.2 I/O Requests from Multiple Virtual Machines

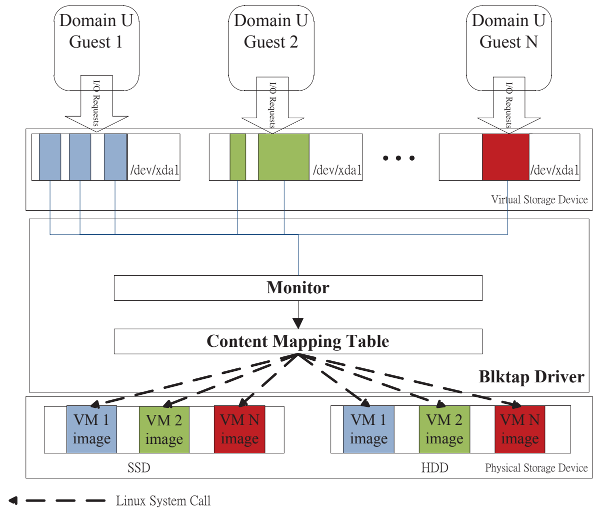

Figure 2: I/O Requests from Virtual Machines to Physical Storage Devices

For handling I/O requests from multiple virtual machines, we modify the Blktap driver in Xen hypervisor. The Blktap driver consists of two parts: one is located at kernel space of Domain 0 that acts similarly to the existing Xen/Linux blkback driver, and another one is at user space of Domain 0. Blktap [23] in the user space can provide Xen hypervisor an interface to access storage devices. Blktap can also mount virtual device images. As shown in Figure 1, the Blktap in the kernel space does not directly access physical storage devices, but it receives all I/O requests of Domain U and transfers the I/O requests to the Blktap in the user space of Domain 0 to perform I/O requests to physical storage devices. This kind of design is beneficial to develop convenient functions in the user space.

As shown in Figure 2, the revised Blktap driver in Domain 0 can receive all I/O requests from multiple virtual machines (i.e, Domain U) and then perform I/O requests to any physical storage devices (i.e., SSDs and HDDs). Therefore, we create a monitor mechanism in Xen hypervisor to analyze I/O requests by read/write operations, access times, priorities, and request size, etc. Based on the monitor, we can determine which I/O requests will be transferred to SSDs or HDDs. For those (soft real-time) applications in virtual machines, the monitor can be used to adjust and increase I/O performance for the applications in the virtual machines. We think that the monitor can play an important role for dynamic adjustment in I/O performance[1]. We also create a content mapping table to record which data are transferred to SSDs or HDDs. According to the content mapping table in Figure 2, we can know which data (e.g., virtual machine images) are located in SSDs or HDDs. Therefore,

[1]Note that a policy in the monitor to determine which data should be placed in SSDs or HDDs for dynamic adjustment in I/O performance can be found in [22]. The paper just focuses on the implementation and discussion on how to revise Xen hypervisor to include SSDs and HDDs for virtual machines.

Table 1: Related files and functions that we revised

Component	File	Function	Functionality

Tap-disk	Tapdisk-vbd.c	tapdisk_vbd_create()
		tapdisk_vbd_issue_request()
		counter content mapping table()
		monitor()
			1. Create and maintain a content mapping table
			2. Create a monitor mechanism
			3. Analyze I/O requests
			4. Deliver I/O requests and its locations to Block-aio component

Block-aio	Block-aio.c	tdaio_open()
		tdaio_complete()
		tdaio_queue_read()
		tdaio_queue_write()
			1. Maintain virtual machine images in SSDs and HDDs
			2. Perform I/O requests to SSDs and HDDs
			3. Migrate data between SSDs and HDDs

the monitor and the content mapping table can help Xen Hypervisor to receive and analyze I/O requests from multiple virtual machines and perform I/O requests to physical storage devices. The management and design of the content mapping table should consider not only search efficiency but also main memory usage. It is like the design of page table but could face different constraints and situations, especially when multiple virtual machines are executed simultaneously.

5.3 Implementation

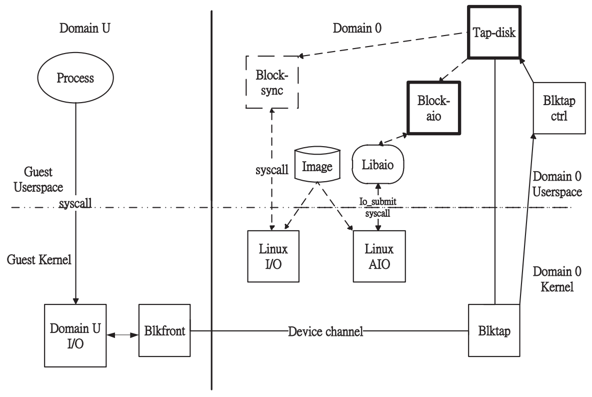

Figure 3: Execution Process of I/O requests

Figure 1 is the system architecture, and we build a virtual storage environment for SSDs and HDDs in Xen Hypervisor by modifying the Blktap driver. We mainly modify two components: Tap-disk and Block-aio in the Blktap driver which abstracts each virtual machine storage into a virtual block device. Figure 3 shows the execution process of I/O requests, where Blktap is in Domain 0 and Blkfront is in Domain U. When Domain U issues system calls (i.e., I/O requests), the calls would go from Blkfront to Blktap by device channel which is a connection between Domain U and Domain 0. Blktap will notify Tap-disk and wake up Tap-disk to receive and analyze I/O requests. Note that Table 1 shows related files and functions in Xen hypervisor that we revised. Related functionalities are also added to the Tap-disk and Block-aio components for the virtual storage environment. For example, in the Tap-disk component, it can deliver I/O requests and its locations to the Block-aio component. In the Block-aio component, it can perform the delivered I/O requests to SSDs and HDDs, and can migrate data between SSDs and HDDs.

5.3.1 *Receive and analyze I/O requests*

Tap-disk is a process which can handle I/O requests and provide an interface for virtual machine management. Because Tap-disk can receive every I/O request from each virtual machine and put the requests in I/O queue, we can modify Tap-disk to analyze the I/O requests and then determine their storage destinations (i.e., the monitor mechanism). Furthermore, we use a td_vbd_t structure to record related information of a virtual block device. td_vbd_t can contain the image name, the image location, and an access interface for a corresponding virtual block device. According to the analysis of I/O requests, we can transfer appropriate I/O requests to SSDs or HDDs. A content mapping table is used to record the address mapping for I/O requests.

5.3.2 *Perform I/O requests to Physical Storage Devices*

Because we adopt different physical storage devices such as SSDs and HDDs, we can perform I/O requests to any physical storage devices and maintain related virtual machine images. The Block-aio component is a process to handle each virtual machine image such as image open/close and I/O requests to the image. It is a bottom layer of Xen hypervisor to access physical storage devices. Therefore, we can modify Block-aio to perform I/O requests to the final storage devices (e.g., SSDs or HDDs). First, we classify I/O requests according to the content mapping table in Tap-disk. Then, Block-aio can issue system calls to perform I/O requests to physical storage devices.

6. EXPERIMENTAL EVALUATION

Table 2: Environment Configuration

Hardware	CPU	Intel(R) Core(TM) i7 950 @ 3.07GHz
	Hard Disk	Hitachi 1TB
	Solid-State Drive	Kingston HyperX 120GB
	Memory	8 GB
System	Virtualization Software	Xen-4.1.2
	Linux Kernel	Linux 2.6.32
Domain 0	Operating system	Ubuntu 11.10
	Memory	2 GB
Domain U	Operating system	Ubuntu 11.10
	Memory	128 MB
	Storage	32 GB
	vCPU	1

In the hardware environment, CPU was Intel(R) Core(TM) i7 950 @ 3.07GHz, memory was 8GB, a HDD was Hitachi

HDS721010CLA332 1TB, and a SSD was Intel 320 series 160GB. We adopted Xen-4.1.2 as the hypervisor and linux kernel 2.6.32. Domain 0 deployed 2GB memory and Ubuntu 11.10 OS. Domain U deployed 128 MB memory, 32GB storage space, and Ubuntu 11.10 OS, as shown in Table 2.

In order to confirm that the virtual storage environment is workable and efficient, we used the logs of I/O requests, which are provided by Storage Performance Council (SPC) [24]. As shown in Table 3, the logs of web search are composed of random access requests. Financial On-Line Transaction Processing (OLTP) is mainly composed of sequential access requests. We use the logs to simulate real workloads.

Table 3: Workload Characteristics
(a) Web Search

Web search	
Read	99 %
Write	1 %
Request Size	8 KB
Workload Size	16.7 GB
Total Read	65.8 GB
Total Write	8 MB

(b) OLTP

OLTP	
Read	25 %
Write	75 %
Request Size	512 B ~ 2 KB
Workload Size	3.6 GB
Total Read	2.6 GB
Total Write	14 GB

We measured the normalized execution time under three configurations. Three configurations are Hitachi HDD, 50% HDD and 50% SSD, and Intel SSD. Hitachi HDD means that all I/O requests in virtual machines are handled by a hard-disk drive (Hitachi HDS721010CLA332 1TB) and Intel SSD means that all I/O requests in virtual machines are handled by a solid-state drive (Intel Series320 130GB). 50% HDD and 50% SSD means that all I/O requests in virtual machines are handled by a virtual storage device using a hard-disk drive and a solid-state drive, and the hard-disk drive and the solid-state drive will handle 50% workloads, respectively. One virtual machine which only use Intel SSD was used as a comparison baseline in terms of normalized execution time. We run different number of virtual machines (i.e., 1VM, 2VMs, 3VMs, and 4VMs). For example, 4VMs means that four virtual machines are executed at the same time to measure the normalized execution time.

Figure 4 is the experimental results of web search. Because the logs of web search have many random access requests (99% read and 1% wirte), the normalized execution time of Intel SSD was about 11 times faster than Hitachi HDD when running one virtual machine. When running two or more virtual machines, Intel SSD was about 11 to 20 times faster than Hitachi HDD. When compared to 50% HDD and 50% SSD, as shown in Figure 5, Intel SSD was about 2.2 and 1.7 times faster than it when running one/two and four virtual machines, respectively.

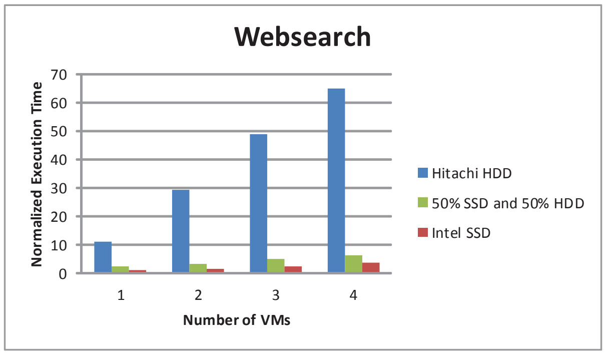

Figure 4: Web Search

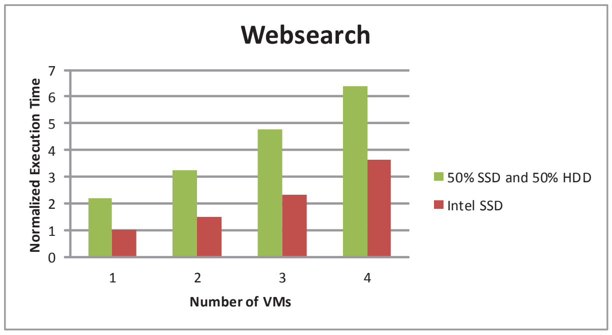

Figure 5: Web Search - Only Compare 50% HDD and 50% SSD whit Intel SSD

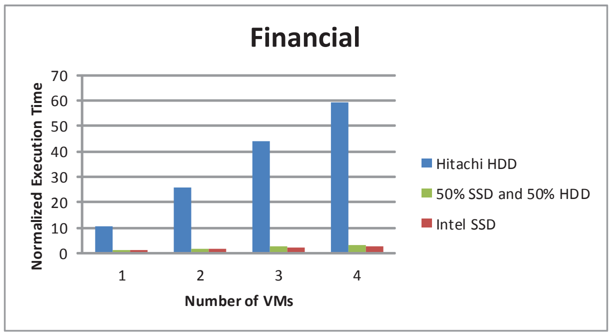

Figure 6: Financial OLTP

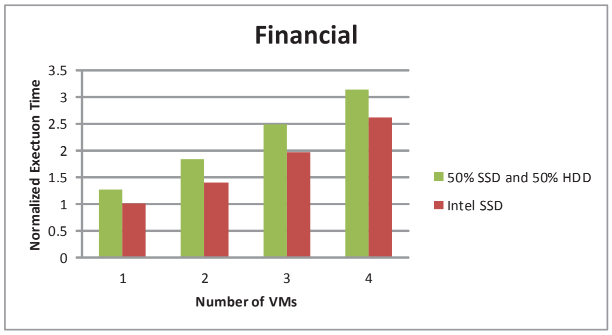

Figure 7: Financial OLTP - Only Compare 50% HDD and 50% SSD whit Intel SSD

Figure 6 is the experimental results of Financial OLTP. Most of OLTP workloads are sequential access requests, and its average request size was relatively small. The normalized execution time of Intel SSD was about 10 times faster than Hitachi HDD when running one virtual machine. Furthermore, when running two and three/four virtual machines, Intel SSD was about 18 and 22 times faster than Hitachi HDD, respectively. Although Financial OLTP workloads mainly consist of sequential access requests, Hitachi HDD cannot get benefit from small request size. On the other hand, Figure 7 shows that Intel SSD was only 1.3 times faster than 50% HDD and 50% SSD in all situations. Overall, we can find out that the proposed virtual storage environment using SSDs and HDDs can not only provide better performance than only HDDs but also provide a more economical solution than only SSDs.

7. CONCLUSION

In the paper, we build a virtual storage environment for SSDs and HDDs in Xen hypervisor. The objective of the paper focuses on the implementation and discussion on how to revise Xen hypervisor to provide a platform using SSDs and HDDs in virtualization environment. We revise the Xen hypervisor by modifying two components (i.e., Tap-disk and Block-aio) in the Blktap driver which abstracts each virtual machine storage into a virtual block device. The revised Xen hypervisor will play an important role of receiving and analyzing I/O requests from virtual machines, and performing I/O requests to SSDs and HDDs (i.e., the monitor mechanism and the content mapping table). According to the experimental results, we demonstrate that the proposed virtual storage environment can provide users a platform to include various storages such as SSDs (i.e., fast access) and HDDs (i.e., low cost). For those (soft real-time) applications in virtual machines, the proposed virtual storage environment can provide an adjustment mechanism in their I/O performance. Furthermore, the proposed virtual storage environment using SSDs and HDDs can not only provide better performance than only HDDs but also provide a more economical solution than only SSDs.

For the future work, we think the monitor mechanism should play different roles in different situations, such as different access patterns of virtual machines and different characteristics of SSDs. We think the monitor should be configured dynamically and adaptively according to current demands, available resources, and system workloads. We hope to find different deployment approaches to improve the virtual storage environment.

8. REFERENCES

[1] VMware Homepage. `http://www.vmware.com`.

[2] Virtual Box. `https://www.virtualbox.org/`.

[3] Windows Hyper-V server. `http://www.microsoft.com/en-us/server-cloud/hyper-v-server/default.aspx`.

[4] Xensource Homepage. `http://www.xen.org/`.

[5] KVM. `http://www.linux-kvm.org/page/Main_Page`.

[6] Iometer Filesystem Benchmark. `http://www.iometer.org/`.

[7] F. Chen, S. Jiang and X. Zhang, "SmartSaver: Turning flash drive into a disk energy saver for mobile computers," in *ACM ISLPED.*, 2006.

[8] L.-P. Chang, "Hybrid solid-state disks: combining heterogeneous NAND flash in large SSDs," in *ASP-DAC.*, 2008.

[9] G. Soundararajan, V. Prabhakaran, M. Balakrishnan and T. Wobber, "Extending SSD lifetimes with disk-based write caches," in *USENIX FAST.*, 2010.

[10] G. Sun, Y. Joo, Y. Chen, D. Niu, Y. Xie, Y. Chen and H. Li, "A Hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement," in *IEEE HPCA.*, 2010.

[11] B. Mao, H. Jiang, S. Wu, L. Tian, D. Feng, J. Chen and L. Zeng, "HPDA: A hybrid parity-based disk array for enhanced performance and reliability," in *ACM TOC.*, 2012.

[12] A. Leventha, "Flash storage memory," in *Communications of the ACM.*, 2008.

[13] J. Matthews, S. Trika, D. Hensgen, R. Coulson and K. Grimsrud, "Intel Turbo Memory: Nonvolatile disk caches in the storage hierarchy of mainstream computer systems," in *ACM Transactions on Storage.*, 2008.

[14] T. Kgil, D. Roberts and T. Mudge, "Improving NAND Flash Based Disk Caches," in *ACM ISCA.*, 2008.

[15] T. Pritchett and M. Thottethodi, "SieveStore: a highly-selective, ensemble-level disk cache for cost-performance," in *ACM ISCA.*, 2010.

[16] Youngjae Kim, Aayush Gupta, Bhuvan Urgaonkar, Piotr Berman, and Anand Sivasubramaniam, "HybridStore A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs," in *IEEE MASCOTS.*, 2011.

[17] S.-F. Hsiao, P.-C. Hsiu and T.-W. Kuo, "A Reconfigurable Virtual Storage Device," in *IEEE ISORC.*, 2009.

[18] M. Canim, G. A. Mihaila, B. Bhattacharjee, K. A. Ross and C. A. Lang, "An object placement advisor for DB2 using solid state storage," in *Proceedings of the VLDB Endowment.*, 2009.

[19] H. Payer, M. A. Sanvido, Z. Z. Bandic and C. M. Kirsch, "Combo Drive: Optimizing Cost and Performance in a Heterogeneous Storage Device," in *WISH.*, 2009.

[20] Q. Yang and J. Ren, "I-CASH: Intelligently Coupled Array of SSD and HDD," in *IEEE HPCA.*, 2011.

[21] F. Chen, D. A. Koufaty and X. Zhang, "Hystor: making the best use of solid state drives in high performance storage systems," in *ACM ICS.*, 2011.

[22] C. K. Kang, Y. J. Cai, C. H. Wu, and P. C. Hsiu, "A Hybrid Storage Access Framework for Virtual Machines," in *IEEE RTCSA.*, 2013.

[23] Xen blktap2 driver. `http://wiki.xen.org/wiki/Blktap2`.

[24] Storage Performance Council. `http://www.storageperformance.org/home/`.