International Journal of Grid Distribution Computing
Vol.7, no.3 (2014), pp.61-70
http://dx.doi.org/10.14257/ijgdc.2014.7.3.07

SH-Sim: A Flexible Simulation Platform for Hybrid Storage Systems

Puyuan Yang[1], Peiquan Jin[1,2] and Lihua Yue[1,2]

[1]*School of Computer Science and Technology,*
University of Science and Technology of China
[2]*Key Laboratory of Electromagnetic Space Information,*
Chinese Academy of Sciences
jpq@ustc.edu.cn

Abstract

Recently, it has been more and more popular to build hybrid storage systems on the basis of SSD (Solid State Drives) and HDD (Hard Disk Drive). In order to evaluate the performance of various hybrid storage based algorithms, this paper proposes a flexible simulation tool called SH-Sim. SH-Sim aims to provide a flexible virtual hybrid disk which is made up of a virtual SSD and a virtual HDD. SH-Sim can run on the various types of SSDs and HDDs with different parameter configurations. SH-Sim not only simulates the hardware features of HDD and SSD, but also provides a virtual management layer for the algorithms to support the switching between algorithms. The experiment proves SH-Sim can simulate the hardware performance effectively and can support the running of various algorithms.

Keywords: *Hybrid storage, DiskSim, SSD, Flash memory, HDD*

1. Introduction

As we know, although SSD has the performance significantly better than HDD, the high price makes it impossible to replace HDD totally. For the tradeoff between performance and price, the hybrid storage based on SSD and HDD has been the research issue. A lot of researchers have proposed a number of hybrid storage managements [1-3], but these works is restricted by 2 problems.

Firstly, the experimental environment is generally built by attaching SSD and HDD on PC. But it is impossible to test the effect of an algorithm on all devices because there are too many types of SSD and HDD. Especially, the SSD technology of various companies is very different and there is a great gap between the high-end and low-end SSD, which brings much trouble to test the algorithms.

Secondly, different researchers generally have various experimental environments with different devices and the hybrid storage research greatly relies on the hardware condition, so it is very hard to make the algorithm comparison which disturbs the algorithm evaluation.

Because of the two points as above, a hybrid storage simulation being flexible and recognized becomes the researchers' requirement. The ideal simulation should have some features as below.

(1) The simulator not only simulates flash and hard disk, but also supports to simulate the hardware of the storage subsystem, like bus, I/O scheduling and so on.
(2) The tool calculates the latency according to the work mode of device other than the static parameter.

ISSN: 2005-4262 IJGDC

(3) In order to support the hybrid storage on different types of flash and disk, the tool must be configured flexibly for kinds of flash and disk.
(4) The tool should be accepted widely by the researchers, so it is best to take a popular open source object as its base.

Based on the open source DiskSim [4], we propose a simulation named SH-Sim for the hybrid storage research. SH-Sim can support SSD and HDD. The contribution of this paper is below.

(1) We provide a flexible, accurate simulation SH-Sim. It is not only a tool for the hybrid storage research but also a platform to compare the algorithms.
(2) SH-Sim has a virtual flash disk (VFD) layer to support the SSDs having different types of flash chip. Besides, SH-Sim defines a structure and method to simulate the hardware feature of flash structure, flash operations and flash work.
(3) SH-Sim also has virtual hybrid storage (VHS) layer to load the hybrid storage algorithms having different architecture. VHS provides a set of interfaces for the hybrid storage management model.

The rest of this paper is organized as follow: the chapter 2 discusses the related work, the chapter 3 describes the SH-Sim in details, the chapter 4 shows the experiment on SH-Sim, the chapter 5 is the conclusion of this paper.

2. Related Work

There are some disk simulations for HDD or SSD before. The most famous one is DiskSim. DiskSim is an open source project maintained by CMU [4]. It is a simulation for HDD and it is version 4.0. It is very popular among academia and industry.

DiskSim has some its own features. First, it not only simulates the I/O latency of HDD, but also simulates the whole circuit of a storage sub system including bus, driver, adapter, etc, which help DiskSim calculate the I/O latency close to real device. Second, DiskSim adopts modular design to build its architecture, which is good for the redevelopment. Third, DiskSim can be configured easily by setting the configuration file for different types of HDD. But DiskSim has some features not suitable to SSD. First, DiskSim has its disk model designed as mechanical structure for HDD, but SSD is an electronic device. So DiskSim cannot support flash disk. Second, because DiskSim is designed for only one media device, its storage management cannot support the devices made by different media. Third is the most important that DiskSim cannot support different types of devices meanwhile due to it has no flexibility for the devices having different storage architectures. DiskSim must be modified for hybrid storage design.

Flash-DBSim is another open source object which is a simulation for only flash disk [5, 6]. Flash-DBSim is developed to support not only the algorithms above SSD like B+ tree, but also the technology inside SSD, such as FTL. It realizes FTL and buffer management inside, which can be changed for research. There have been some work running on this simulation [7, 8], but it has some disadvantages. First, Flash-DBSim has no simulation of the whole circuit, such as bus, driver, *etc.*, So Flash-DBSim is not accurate. Second, Flash-DBSim calculates the I/O latency simply with the static parameter of flash chip. It has nothing with the work mode of flash chip, which means inaccurate. Third, Flash-DBSim does not support hybrid storage.

As discussed above, the both tools have some limit. We combine the advantages of the two simulations and improve their disadvantages. We use DiskSim to simulate the hardware of the hybrid storage system for the latency of bus, controller, adapter, and so on. Then Flash-DBSim is an example for us to simulate the flash device, but our flash simulation is on DiskSim and its latency is not static parameters. So this paper has a

new flash simulation embedded into DiskSim and proposes a hybrid storage management modular in the controller to build SH-Sim.

3. Design of SH-Sim

3.1. SH-Sim Architecture

DiskSim has a complete circuit simulation of a storage subsystem, which can help SH-Sim simulate the latency of every module inside SSD and HDD. Besides, DiskSim has provided a good HDD simulation which is named disk model inside. So there is no need to reinvent the wheel for us. This paper focuses on the flash model and hybrid storage manager. As mentioned before, SH-Sim takes DiskSim as its base, as shown in Figure 1.

The white part is from DiskSim and the gray part is our work. The disk model simulates magnetic disk of HDD. The mem model simulates the buffer inside HDD. The controller simulates the kernel of the subsystem and controls all the operation of simulation. The adapter simulates the scheduler on disk. The bus model is the simulation of the bus on the subsystem.

SH-Sim adds the hybrid storage management above the controller through VHS and loads a flash model on the bus for the flash disk. Hybrid storage manager uses VHS to connect to DiskSim, which makes sure that the different hybrid storage policies can be realized without considering the compatible problem. The flash device in hybrid storage is simulated by the flash model, which has VFD to support different types of flash. The hard disk is simulated by the disk model of the original DiskSim.

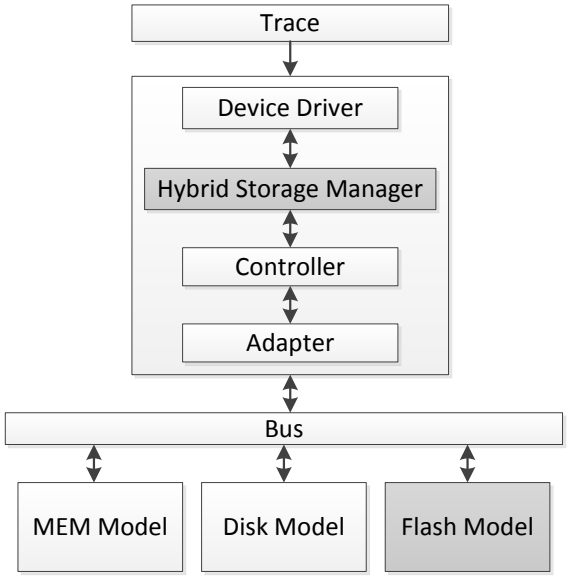

Figure 1. SH-Sim Architecture

3.2. Flash Model

Flash model is made up of 3 parts including access interface, FTL module and flash hardware, as shown in Figure 2.

Because SH-Sim is built on DiskSim, the access interface set is designed as the interface of disk model inside DiskSim for compatibility. Bus can access the flash mod-el without any modify. The FTL module is the storage management on flash. It loads the flash translation layer (FTL), garbage collection and wear leveling to simulate the work of SSD. SH-Sim

realizes NFTL [9] in the code. The FTL code can be replaced by any other algorithms with the FTL interface. So it supports to test FTL algorithm.

Flash hardware is the part to simulate the flash chip. It uses VFD to support various types of flash. VFD is an interface defining the access operations of flash including the page read, page writes, block erase, etc. So the upper module can access the flash simulation by the VFD's interfaces. Because of VFD, the flash hardware can support the simulation for the various types of flash.

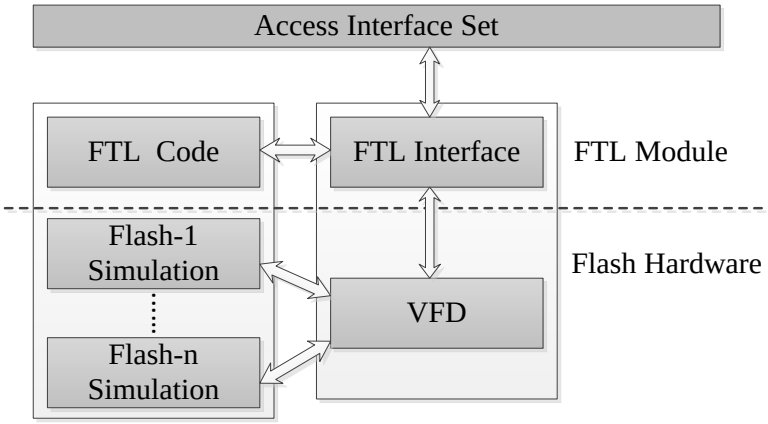

Figure 2. Flash Model Overview

The flash simulation is built based on the real flash. The flash has a multi-layer structure. A flash chip is made up of lots of blocks which are an erase unit. A block is made up of 64 or 128 or more pages which is the read/write unit. A page is made up of lots of cell and a page register. The flash simulation is Figure 2 is designed as the block-page structure.

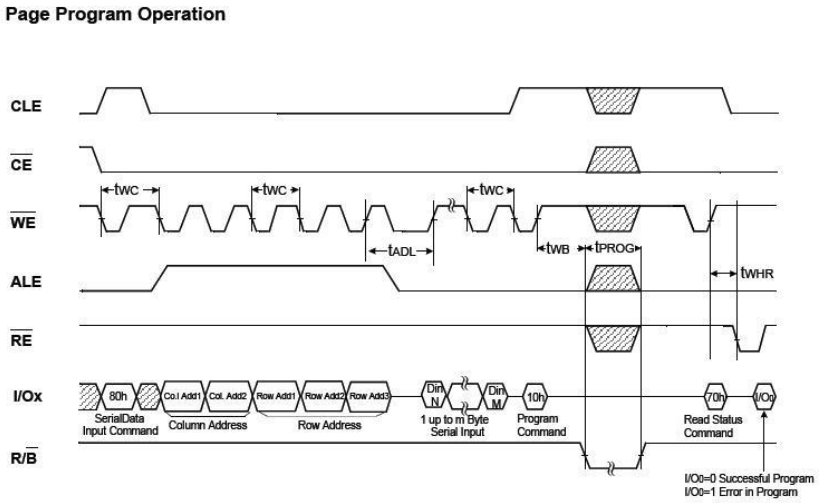

Figure 3. Radom Write Operation of K9HBG08U1M

The flash I/O latency is decided by its work mode. The flash chips of different types have different work modes. SH-Sim has its flash simulation from a Samsung flash [10] in the code. The work mode of a real flash contains 3 steps: command, address and data. Each flash has a command set including more than 10 basic operations. The flash works as follow steps: first step, flash receives the command to ensure the request, which takes up 2 cycles. Second step,

flash receives the basic address of the data transferred, which takes up 5 cycles. Third step, flash starts transferring the data and the clock cycles of this step is proportional to the requested data size. There are some intervals between the 3 steps. For example, Figure 3 shows the clock cycle of read operation for the flash Samsung K9HBG08U1M [10].

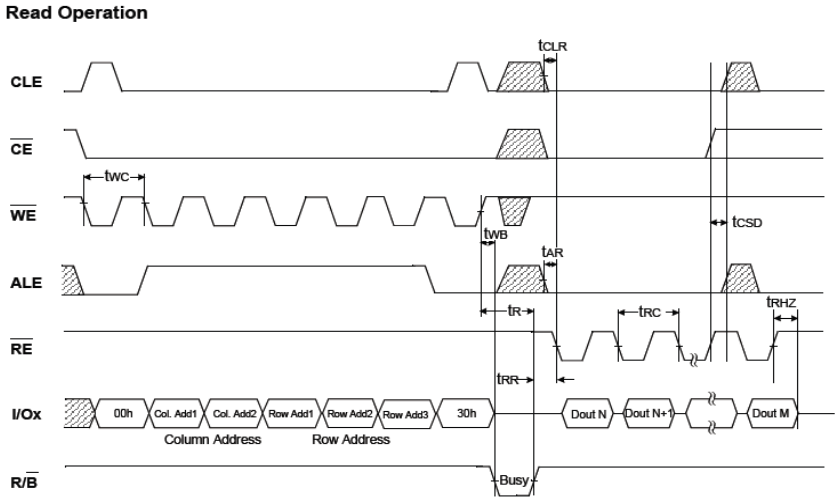

Figure 4. Radom Read Operation of K9HBG08U1M

According to Figure 3 and 4, the I/O latency of flash is made up of all the work cycles. t_{WC} is the latency of that one byte is transferred to flash's page register from outside. t_{PROG} is the latency of that one byte is written to the flash cell from page register. t_R is the latency of that data transfer from cell to register. Other parameters shown in Figure 3 and 4 can be found in the datasheet [10]. The Figure 3 and 4 proves that the I/O latency is decided by the work mode but static parameter. The I/O latency of read and write is calculated for the flash as formula (1) and (2).

$$t_{read} = t_{WC} \times 7 + t_R + t_{RR} + t_{RC} \times PageSize + t_{RHZ} \quad (1)$$

$$t_{write} = t_{WC} \times 6 + t_{ADL} + t_{WC} \times PageSize + t_{WC} + t_{WB} + t_{PROG} + t_{WC} + t_{WHR} \quad (2)$$

The flash simulation can get the latency of other operations by their work wave. Then different flash gets its simulation by redefining the interface of VFD according to their work wave.

3.3. Hybrid Storage Manager

DiskSim has its address management on one or more disks, but it cannot decide which disk the data should be on and move the data between the disks. So DiskSim storage manager does not support the hybrid storage. SH-Sim proposes hybrid storage manager for the algorithms, as shown in Figure 5. The main function of hybrid storage manager is to maintain the address mapping information and adjust the page placement by the hybrid policy.

VHS contains 2 components which are interface and access monitor. The interface provides the functions to get the address mapping as shown in Table 1. The access monitor maintains a data structure *pg_info* to record page's access information.

Mapping table is a structure to store the map information. It is read by the VHS and update by the page placement. Page placement is forked by the monitor and it decides the page

placement with the hybrid policy, so page placement can update the mapping table and execute the migration when a page should be migrated.

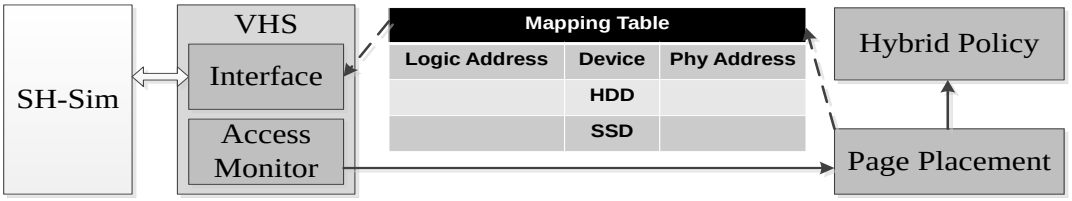

Figure 5. Hybrid Storage Manager Overview

As shown in Table 1, the function *pginitialize* is the initialization of hybrid storage manager including building the mapping table, applying VHS and registering the hybrid policy. The important function is *Tempchange* which is the migration of hybrid storage manager. It is forked by the model page placement and it calls the origin functions of DiskSim to generate write request on the target device. Except that, the functions *pgchange_for_physicalread* and *pgchange_for_physicalwrite* are used for the read and write request in hybrid storage manager. As mentioned in Table 1, hybrid type is the flag of hybrid policy. Each function of page placement is a switch-case structure on hybrid type. Every new hybrid policy must be registered a flag and then added into the switch-case structure.

Table 1. Main Functions of Interface

Function	Description
pginitialize(ioreq_event, hybridtype)	Hybrid stroage manager initialization, *hybridtype* is the registered flag of different algorithms.
pgchange_for_physicalread (ioreq_event,hybridalgo)	Read request for the page *hybridalgo*, *hybridalgo* is logical address and it is ruened by the mapping table
pgchange_for_physicalwrite (ioreq_event,hybridalgo)	Write request for the page *hybridalgo*, *hybridalgo* is logical address and it is ruened by the mapping table
Tempchange(ioreq_event,tempchangealgo,origin_pg_type)	Migration func, it calls the DiskSim origin function to generate write request on target device.

Table 2. Main Members of Pg_info

Member	Description
pid	The logical address of page
devno	The number of disk the page on
blkno	The number of block the page belonged to
pg_pw	The counter of the page physical writing form buffer or disk
pg_pr	The counter of the page physical reading from disk or buffer
pg_lw	The counter of the page writing hit in buffer
pg_lr	The counter of the page reading hit in buffer
pg_cost	The access cost of the page claculated by hybrid policy

The *pg_info* in Table 2 is the basic information of each page. It is initialized by the access monitor and it is transferred between VHS, page placement and hybrid policy models. The access history is recorded in this structure. The parameter *pg_cost* is the access cost calculated by the hybrid policy model according to the access history and it can impact the page placement. The parameter *blkno* is the mapping address. If the page is on HDD, *blkno* is its physical address. If the page is on SSD, *blkno* is another logical address which is mapped to physical address in FTL of flash model.

This paper selects 3 hybrid policies from the paper [11]. They are conservative, optimistic and hybrid. SH-Sim is test separately with the 3 policies in experiment.

4. Experiment

The experiment's main target is to test the accuracy and flexibility of SH-Sim. The HDD Simulation of SH-Sim is the disk model in DiskSim and DiskSim has been proved accuracy well, so the experiment focuses on the SSD simulation and hybrid storage.

The SSD Simulation test is executed on SH-Sim with disk model unattached, so the SH-Sim is just a SSD simulation. Then we choose 2 real SSD devices, as shown in Table 3. SH-Sim configuration parameters are set according to the datasheet [12] of the flash type inside the devices.

Table 3. The SSD Devices Picked for Experiment

ID	SSD type	Flash type
SSD-1	OCZ AGT3-25Satellite-60G	Micron 29F64G08CBAAA-WP
SSD-2	INTEL SSDSA2CT040G3	Micron 29F64G08ACME1
Disk-h	ST500LM000	K9LCGY8S1B-HCK0

There are 10 trace files generated by DiskSim as shown in Table 4. The trace files are test on the real devices and SH-Sim separately, and then we can estimates the SSD simulation accuracy by comparing the result, as shown in Fig 6.

The real SSD is attached to PC with Linux 3.0 and open with flag O_DIRECT. We generate I/O requests to the device from trace and then record the real run time. SH-Sim takes trace files as input and calculates the I/O latency for every request and then gets the run time for whole trace. The result in Fig 6 tells the SSD simulation is very close to the real device. The accuracy of flash model in SH-Sim gets proved.

Table 4. Trace Files for SSD Simulation Test

Trace	Trace number	Read:Write
1	982294	4:1
2	100000	4:1
3	982294	1:0
4	300000	4:1
5	300000	3:2
6	300000	1:1
7	300000	2:3
8	300000	1:4
9	100000	0:1
10	300000	0:1

Table 5. Trace Files for Whole SH-Sim Test

Trace	Trace number	Read Proportion
1	300000	0
2	300000	20%
3	300000	40%
4	300000	60%
5	300000	80%
6	300000	100%

As shown in Figure 6, no matter the trace is read trend or write trend, the run time calculated by SH-Sim with flash model is very close the real device. This result proves that SH-Sim can simulate the flash device accurately. Considering SH-Sim utilizes DiskSim to simulate HDD, it means SH-Sim can support the hybrid storage simulation.

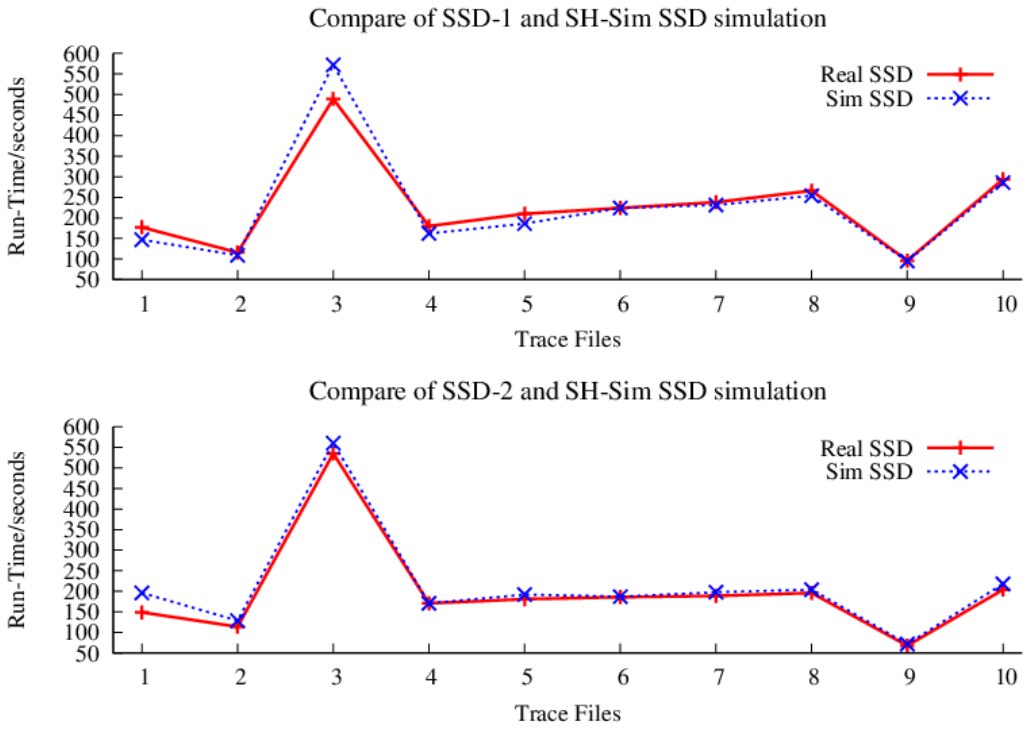

Figure 6. The Real SSD and SH-Sim SSD Simulation

The other experiment is to test SH-Sim with 3 hybrid algorithms mentioned before and compare SH-Sim with the real hybrid disk. The selected hybrid disk is ST500LM000 as shown in Table 3. The configuration file of SH-Sim is set according to the datasheet of the flash type inside ST500LM000. The trace files are generated by DiskSim as shown in Table 5. Besides, we choose 2 real devices including SSD-2 and HDD to compare.

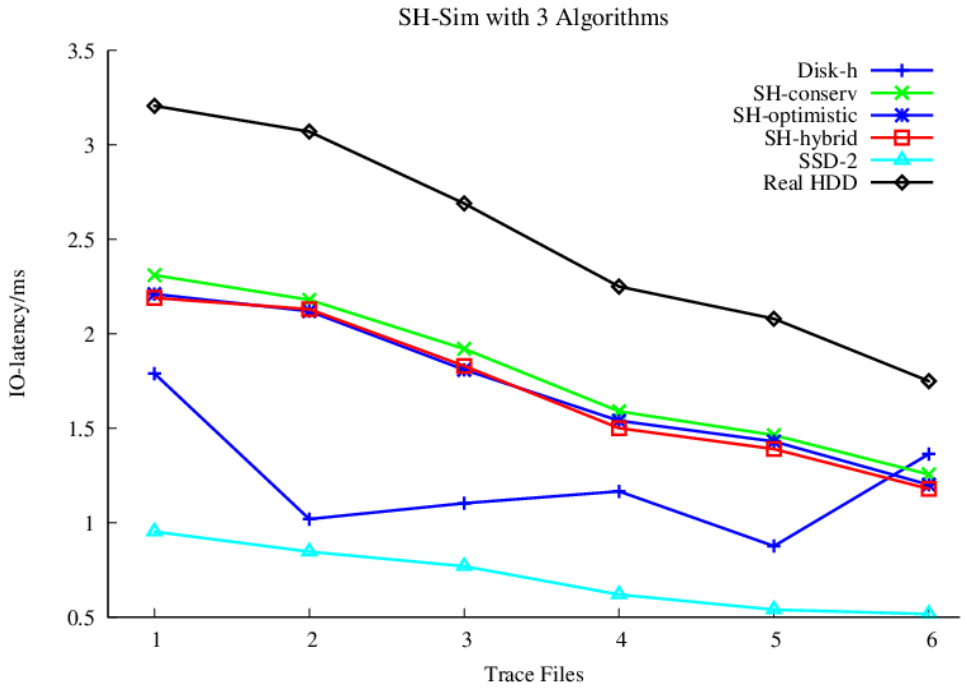

Figure 7. Average I/O Latency of Real Devices and SH-Sim

The real devices are attached to PC with Linux 3.0 and open with O_DIRECT, The trace files are executed on them and their latency is recorded. The I/O latency of SH-Sim is calculated with 3 hybrid policies. The result is shown in Figure 7.

Disk-h has the performance between SSD-2 and HDD. Because Disk-h is a black box to us, so we cannot Figure out the algorithms inside it, which means SH-Sim can-not get the same hybrid policy as Disk-h. The 3 policies of SH-Sim are very different from the algorithms inside Disk-h, so there is a gap between SH-Sim and Disk-h. Besides, the 3 hybrid policies are all proposed based on the access cost calculation, so their test results are very similar. The realization of 3 hybrid policies in SH-Sim shows SH-Sim can support different hybrid storage algorithms and this is a good example to compare different hybrid policies on SH-Sim. The 3 tests prove the flexibility of SH-Sim. The result of SH-Sim is between SSD-2 and HDD, which tells SH-Sim can support the hybrid storage research.

The two experiments above prove the accuracy and flexibility of SH-Sim. So SH-Sim can support the hybrid storage research and meet the requests of most researchers.

5. Conclusion and Future Work

This paper proposes SH-Sim for hybrid storage research. SH-Sim is a hybrid stor-age simulation built based on DiskSim with new components flash model and hybrid storage manager. It has an accurate hardware simulation including HDD and SSD, which can flexibly support flash of various types. Besides, SH-Sim has a flexible hybrid storage manager which can support different hybrid policies, so SH-Sim is a good tool to test and compare the hybrid storage algorithms for researchers.

The future work will focus on two aspects. First, flash model of SH-Sim cannot support the parallelism of flash chips yet, which causes SH-Sim cannot simulate the high-end SSD, so we

will expand VFD and simulate SSD-inside control algorithms [13] to support that. Second, The VHS and access monitor must be expanded for more and more requirements of hybrid algorithms.

Acknowledgements

This work is supported by the National Science Foundation of China under the grant no. 61379037 and no. 61073039.

References

[1] G. Soundararajan, V. Prabhakaran, M. Balakrishnan and T. Wobber, "Extending SSD lifetimes with disk-based write caches", 8th USENIX conference on File and storage technologies, USENIX Association, **(2010),** pp. 8-8.
[2] H. Jo, Y. Kwon, *et al.*, "SSD-HDD-hybrid virtual disk in consolidated environments", Euro-Par 2009-Parallel Processing Workshops, Springer, Berlin Heidelberg, vol. 6043, **(2010),** pp. 375-384.
[3] P. Yang, P. Jin and L. Yue, "Hybrid storage with disk based write cache, "Database Systems for Advanced Applications, Springer Berlin Heidelberg, vol. 6637, **(2011)**, pp. 264-275.
[4] "Paralell Datalab", School of Computer Science Electrical & Computer Engineering, Carnegie Mellon University, http://www.pdl.cmu.edu/DiskSim/
[5] X. Su, P. Jin, X. Xiang, K. Cui and L. Yue, "Flash-DBSim: a simulation tool for evaluating flash-based database algorithms", In Computer Science and Information Technology, ICCSIT. 2nd IEEE International Conference on, IEEE, **(2009),** pp.185-189.
[6] P. Jin, X. Su, Z. Li and L. Yue, "A Flexible Simulation Environment for Flash-aware Algorithms", the 18th ACM Conference on Information and Knowledge Management (CIKM'09), ACM press, **(2009),**
[7] Z. Li, P. Jin, X. Su, K. Cui and L. Yue, "CCF-LRU: A New Buffer Replacement Algorithm for Flash Memory", IEEE Transaction on Consumer Electronics, vol. 55, no. 3, **(2009),** pp. 1351-1359.
[8] P. Jin, Y. Ou, T. Haerder and Z. Li, "ADLRU: An Efficient Buffer Replacement Algorithm for Flash-based Databases", Data and Knowledge Engineering (DKE), Elsevier, vol. 72, **(2012)**, pp.83-102.
[9] J. Kim, J. Min Kim, S. H. Noh, *et al.*, "A space efficient flash translation layer for compact flash systems", IEEE Trans on Consumer Electronics, IEEE, vol. 48, **(2002),** pp. 366-375.
[10] "Samsung flash chip K9HBG08U1M datasheet", http://www.datasheet.co.kr/datasheet-html/K/9/H/K9HBG08U1M_SamsungElectronics. pdf.html.
[11] I. Koltsidas and S. D. Viglas, "Flashing up the storage layer", Proceedings of the VLDB Endowment, ACM, vol. 1, no. 1, **(2008),** pp.514–525.
[12] "Micron flash chip 29F64G08CBA", https://www.micron.com/parts/nand-flash/mass-storage/mt29f64g08cbaaawp.
[13] H. Zhao, P Jin, P. Yang and L. Yue, "BPCLC: An Efficient Write Buffer Management Scheme for Flash-Based Solid State Disks", International Journal of Digital Contents and its Applications (JDCTA), vol. 4, no. 6, **(2010),** pp. 123-133.

