HcDD: The Hybrid Combination of Disk Drives in Active Storage Systems

Shu Yin, Zhuo Tang, and Kenli Li
College of Computer Science and Electronic Engineering
Hunan University
Changsha, Hunan, China 410002
Email: {shuyin, ztang, lkl}@hnu.edu.cn

Zhiyang Ding
State Development and
Investment Corporation
Beijing, China
Email: ddzzyy@gmail.com

Xiaojun Ruan
Department of Computer Science
West Chester University of Pennsylvania
West Chester, PA 19383
Email: xruan@wcupa.edu

Xiaomin Zhu
College of Info. Systems and Management
National University of Defense Technology
Changsha, Hunan, China 410073
Email: xmzhu@nude.edu.cn

Xiao Qin
Department of Computer Science and
Software Engineering
Auburn University
Auburn, AL 36849

***Abstract*—Since large-scale and data-intensive applications have been widely deployed, there is a growing demand for high-performance storage systems to support data-intensive applications. Compared with traditional storage systems, next-generation systems will embrace dedicated processor to reduce computational load of host machines and will have hybrid combinations of different storage devices. The advent of flash-memory-based solid state disk has become a critical role in revolutionizing the storage world. However, instead of simply replacing the traditional magnetic hard disk with the solid state disk, it is believed that finding a complementary approach to corporate both of them is more challenging and attractive. This paper explores an idea of active storage, an emerging new storage configuration, in terms of the architecture and design, the parallel processing capability, the cooperation of other machines in cluster computing environment, and a disk configuration, the hybrid combination of different types of disk drives. Experimental results indicate that the proposed HcDD achieves better I/O performance and longer storage system lifespan.**

***Keywords*-active storage system, hybrid, SSD**

I. Introduction

Recently, the use of NAND-flash-based Solid State Devices (hereinafter referred to as SSDs) has evolved from specialized applications in mobile devices [7] and laptops to primary system storages in general-purpose computers and even data centers [23]. Evidence shows that tapes are used as archiving method, while disks 3serve as primary storage device for mainstream systems and flash based storage devices are taking place of disks in high-end systems [13]. Different from HDDs (a.k.a. magnetic hard disk drives) which rely on mechanical moving parts, SSDs are completely built on semiconductor chips resulting in high random access performance and lower power consumption. In addition, the cost of commodity NAND flash – often cited as the primary barrier to SSD deployment [24] – keeps decreasing, which increases the possibility for dynastic changes in the storage arena. Although SSDs begin to be deployed in advanced data service institutions, engineers still hesitate somehow to perform a large-scale deployment of SSDs in data centers due to their poor reliability and the limited lifespan issues [3].

As the cost of NAND flash has declined with increased density, the number of erase cycles that a flash cell can tolerate, on which the number of write operations depends (because of the erase-before-write characteristic), has suffered. Meanwhile, server applications, such as OLTP (Online Transaction Processing) [14], normally demand a high-performance and highly reliable storage system. Some researchers believe that the stressful workload and limited available erase cycles may further reduce lifetimes of SSDs, in some cases, even to less than one year [31]. From another perspective, when building terascale or petascale servers and data centers using only SSDs rather than HDDs, the cost – even though with the decreasing price of SSDs – is often beyond the acceptable budgets in most cases. Thus, HDDs are still regarded as indispensable components in the storage hierarchy because of their merits of low cost, huge capacity, above-average reliability, and fast sequential access speed. Instead of simply replacing HDDs with SSDs, researchers [32][21][16][26][8] are looking for complementary approaches that balance the performance, reliability and cost of storage.

In this paper, we propose a hybrid combination of disk arrays for active storage systems called HcDD. The hybrid-disk design involves an enhanced duo-buffer structure, which consists (1) a HDD-write buffer to serve and de-duplicate write requests and (2) an on-SSD buffer to provide parallel write processing. Read requests are all served by SSDs directly while writes to an active storage are firstly transferred to its HDD buffer and perform de-duplication before migrating to an SSD. The de-duplication computation and I/O operations from the buffer disk to the major storage disk are offloaded to the dedicated processor in the active storage. This paper also introduces an enhanced version of SSDs buffer, which supports internal parallelism processing. Together, the goal of

this paper is to minimize the number of writes sent to SSDs without significantly affecting the performance; by doing so, it reduces the number of erase cycles and thus extends SSDs lifetime.

The major contributions are as follows:

- We design and simulate a hybrid combination of storage devices for active storage systems.
- An HDD in our design is first assigned as a write buffer to SSDs, thereby supporting de-duplication service.
- We design and simulate the internal-parallelism supporting cache on SSDs.
- A system simulator is built to evaluate the hybrid drives combination in active storage systems.

The discussion of HcDD is followed by experimental results and the performance evaluation in Section III. Section IV briefly introduces the background and related work. Last but no the least, Section V is the conclusion.

II. The Design of HcDD – a Hybrid Combination of Disk Devices

A. System Architecture

In this section, we present the design of a hybrid disk system model called the Hybrid combination of Disk Drives in active storage systems (hereinafter referred as HcDD). The major goals of the HcDD are to (1) extend the lifespan and improve the reliability of SSDs, (2) make the use of the storage space and (3) improve the write performance of SSDs by eliminating duplications and redundant data.

Below are descriptions of modules that an active storage system consists.

- **Disk Drive** The disk drive is a major component in any storage system, where data are permanently stored.
- **Controller** The controller is a processing unit (i.e. data management unit) for disks in a storage system. The controller communicates between disks and host machines, manages disk drives, and distributes data among disks.
- **De-duplication Engine** It is in charge of computing fingerprints for incoming requests, looking the requests up in a fingerprint table, and deciding whether requests should be written to SSDs.

Figure 1 depicts the architecture of a HcDD in an active storage system. The active storage system has its own computation facilities (*i.e.,* a dedicated processor), the memory and storage facilities, including a disk controller, a de-duplication engine as well as some disk drives. And there is a hybrid combination of two kinds of storage devices – SSDs serving as main data disks, and HDDs serving as the write buffers – in each HcDD. Further, we enhance on-SSD buffer to support of internal parallelism processing.

B. Hybrid Combination of Storage Drives

Figure 2 is the configuration of the simulated hybrid combination of storage drives. The HcDD system mainly contains three types of modules: a controller, a de-duplication engine, and storage devices. The controller module is in charge of

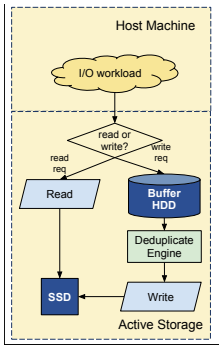

Fig. 1: The System Configuration of HcDD: A Hybrid Active Storage System

managing and distributing input I/O requests. The duplication engine compares the fingerprint of each write request with existing ones in a fingerprint table. To simulate the storage system, we integrated DiskSim in our system. DiskSim, developed by Ganger *et al* [12][11], is a well-known disk simulator. DiskSim has been validated against several disk drives using the the published disk specifications and I/O workload traces. Since DiskSim only support the storage-level simulation, the developers also provide programming interfaces to integrate DiskSim with any system-level simulator.

The proposed hybrid storage model integrates both cost-effective HDDs and high-speed SSDs as a hybrid combination storage component in the active storage system. The controller handles data distribution, which avoid undesirable significant changes to existing file systems and applications. The controller directs write requests to a buffer disk (i.e., HDD) and sends the read requests to the SSDs. Thus, in terms of read operations, the overhead caused by the controller can be ignored; there is no performance interference since the controller simply redirects requests without any computation overhead. Once a write request is issued by a host machine, the data will be written on the buffer disk of the active storage node. As soon as the data is buffed on the HDD, the data will be processed by the dedicated de-duplication engine, which calculates the hash value and looks the data up in the hash table – before sending the data to the next step. Then, the de-duplicated data will be written to SSDs. The duplicates are mapped to existing ones and removed from the buffer disk. We consider the de-duplication process hybrid in nature because of the following reasons. From the perspective of the host machine, all the request are handled inline; meaning that there is no calculation workload and thus no waiting time required. Meanwhile, within the active node, the de-duplication is handled more similar to the post-process pattern, which stores new data on the storage media and then the de-duplication engine will analyze the data looking for redundancy as soon as possible.

The workflow of de-duplication engine is introduced in Figure 3. When a write request of the input workload trace is received at the buffer disk, the processing of de-duplication can be described in four steps:

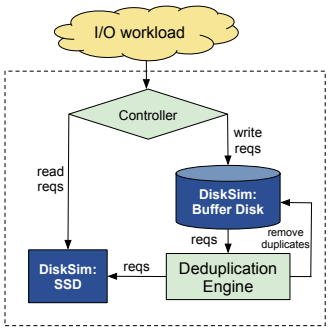

Fig. 2: The System Configuration of HcDD: A Hybrid Active Storage System

1) Before data are written to the buffer disk, the incoming request triggers the de-duplication engine in the active storage node.
2) Each updated page in the buffer is computed a hash value (*i.e.* hash fingerprint) by the dedicated processor of the active storage node.
3) Then each hash value is looked up against a hash table, which maintains the fingerprints of data already stored in the SSD.
4) a) If the hash value is fresh (*i.e.,* it is not found in the table) the write is performed as a regular operation in the SSD.
   b) Otherwise, if the fingerprint is found, the mapping tables are updated by mapping the duplicate request to the physical location of the residing data. Then the write operation to flash is canceled. The goal of this step is to minimize the number of writes sent to the SSD without significantly impacting the performance; in doing so, HcDD reduces the number of erase cycles, which is the performance bottleneck to the NAND flash. In addition to improved performance, extending SSDs lifetime is a second benefit gained from the de-duplication service.

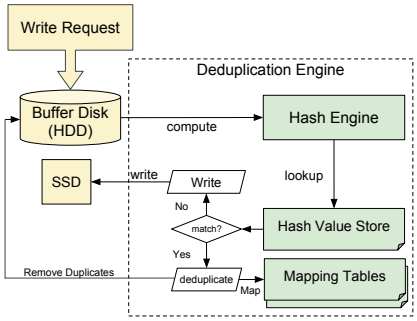

Fig. 3: The Workflow of Hybrid Active Storage System

C. Intra-parallelism buffer on SSD

An SSD is built by arrays of NAND flash memory, which is a semiconductor storage module [2][5]. Agrawals [2] describes that an SSD has one or multiple identical elements (*i.e.,* packages) and all of them can work in parallel. The challenge is which part of the multiple elements should be in charge of handling data distribution and parallel processing.

Unlike traditional hard drives, the flash-based storage has a Flash Translation Layer (FTL) which maps Logic Block Number (LBN) to Physical Block Number (PBN). Also, a remapping algorithm is designed in FTL for wear leveling. Thus, the block number in file system level is not the block number in flash memory level. Based on different remapping algorithms, the same LBN may lead to different PBN at different time periods. Hence, a buffer must be designed in the lower level of FTL to keep data consistency. We chose to use Synchronous Dynamic Random Access Memory (or SDRAM) as an on-board buffer for its high performance. The on-board buffer is the lower-level buffer in our duo-buffer design. Since the performance of random writes, especially re-writes, is the Achilles heel in flash memory, the buffer is designed for buffering only write requests.

Figure 4 presents the software structure of an enhanced SSD with SDRAM buffer. In the buffer, the number of lists is the same as that of packages. Each list contains data for its corresponding package. Since the buffer is built under FTL, the granularity in the buffer is 8 KB page. The size of pages can be tuned. Recall that all requests buffering in SDRAM are writes. Once the buffer is full, our algorithm will assign the same number of pages to each package in parallel since all packages are able to work independently. Below we presents an algorithm to enhance parallelisms by buffering writes (Algorithm 1). Even though there are nested loops, the number of parallelism pages and the number of packages are both small and fixed values, the time complexity is still approximately $O(n)$, where n is the number of packages.

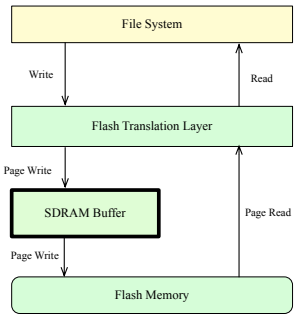

Fig. 4: The Workflow of On-board Buffer

III. Experimental Results and Evaluations

The HcDD storage system is implemented and evaluated based on a comprehensive trace-driven simulator. The reasons that we conduct the HcDD using the simulation are twofold: firstly the modification of both the disk controller and the FTL module of SSDs heavily rely on supports from hardware vendors. However, unfortunately, such supports are hard to obtain due to the commerce issues; and, secondly, it is

Algorithm 1 Enhancing Package-Level Parallelism Algorithm. r represents one request. L_i represents the List stores the request in the buffer for the ith package. P_n represents the nth package.

```
if r_current is a write request then
  find its corresponding package P_i
  for r_j ← all requests in L_i do
    if r_j = r_current then
      remove r_j
    end ifadd r_current to L_i^tail
  end for
  if buffer is full then
    for m ← number of parallelism pages do
      for n ← number of packages do
        issue L_n^head to P_n
      end for
    end for
  end if
end if
```

rational and economical to evaluate the performance of an emerging architecture under a low-lost simulation environment before pushing the architecture to implementation. This section presents the experiment environment along with results from the HcDD simulation studies under a variety of configurations.

A. The Experiment Configurations

The HcDD emulates the behaviors of a hybrid active storage systems with two types of disk drive modules. The HDD module, a 15,000 RPM Seagate Cheetah 15.5K SAS hard disk drive, is provided by DiskSim 4.0 [11], which emulates a hierarchy of storage components including buses, controllers, and disks. The enhanced SSD module supporting internal-parallelism processing is implemented in a sophisticated SSD simulator, from Microsoft Research SSD extension [22] for the DiskSim simulation environment [11]. Although the Microsoft extension implements the major components of FTL (i.e., indirect mapping, garbage collection and wear-leveling policies), it does not have a on-board buffer, which is an essential facility in newly released commodities. Thus, we designed and implemented an enhanced on-board buffer.

We evaluate the HcDD design by running simulations upon three real-world application traces (see Table I): traceKernel, tracePhoenix, and Finanical2 [1]. "traceKernel" was collected while a target machine compiling the Linux kernel. "tracePhoenix" was collected while the target machine running a MapReduce application, WordCount, provided by the Phoenix MapReduce System [28]. "Financial" is from OLTP (Online Transaction Processing) applications running at a large financial institution provided by the laboratory for Advanced System Software of University of Massachusetts Amherst. traceKernel and tracePhoenix are collected on a HP ProLiant ML110 G6 workstation with an Intel Xeon X3430 processor, a 2GB main memory, and a 500GB 7,200 RPM Seagate Barracuda hard disk drive. The operating system is Ubuntu 10.10 with the Ext3 file system. The logical address of all traces were evenly shrunk so that each requests address can be mapped to a physical address within the scope of the SSD configuration (32GB in this study). Table 5.2 presents the features of the three traces.

TABLE I: Num of Read/Write Requests

Trace	**Read Requests**	**Write Requests**
traceKernel	84,847	70,243
tracePhoenix	822,909	155,413
Financial	2,252,549	480,127

B. Internal Parallelism Supported Buffer for SSD

An SSD has one or multiple identical elements (i.e. packages) and all of them can work in parallel. In this section, we compare our internal-parallel SSD algorithm against the classic LRU cache management algorithm. The performance impact of parallelism levels and buffer size are presented and discussed as follows.

In this test, two general purpose traces provided by the Microsoft Research SSD extension [22], *iozone* and *postmark*, are evaluated. The size of write buffer scales from 1 MB to 64 MB. From Fig. 5, we observe that when the buffer size is small (1 MB or 2MB), internal-parallelism scheme surpasses the LRU one. However, when as the buffer size increases, the average response time of the internal-parallelism scheme is either similar to the competitor or worse. Thus, in Section 5.4.4, we choose a 1 MB buffer in the HcDD simulator.

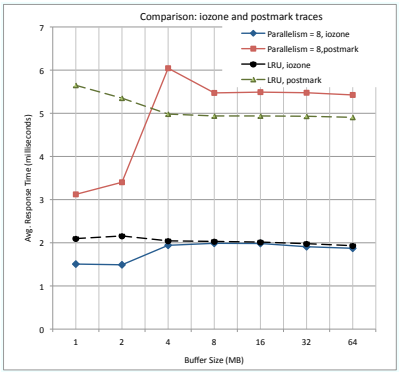

Fig. 5: Performance Comparison of *IOzone* and *Postmark*

C. System Performance Evaluation

In this section, we evaluate the the response time and the average response time of the HcDD by comparing then with three traditional storage architectures, $Trad.Hybrid$ (i.e., the traditional hybrid mechanisms that no buffers are involved), $SSDs$ (i.e., the storage systems that exclusively make use of SSD devices), and $HDDs$ (i.e., the storage systems that consist of hard disk drives only).

When compared with $Trad.Hybrid$ (see Fig. 6 and Fig. 7), the HcDD reduces both response time (averagely saved 12% of response time) and average response time (averagely saved 13% of average response time). And as mentioned in the

previous section, there are 29.15%, 46.93% and 28.9% request respectively removed from being issued to the SSD.

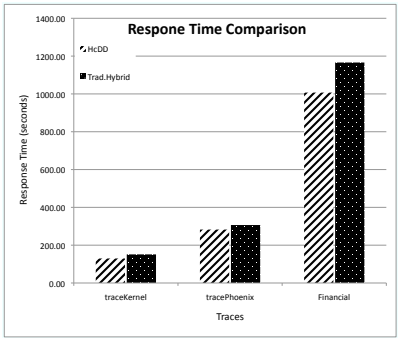

Fig. 6: Processing Time Comparison between HcDD and traditional hybrid storage

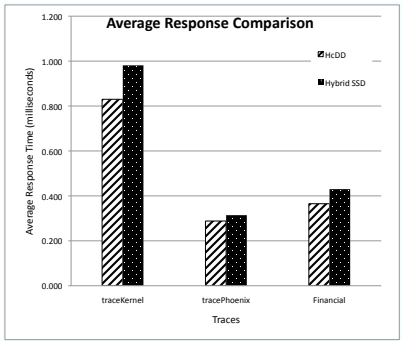

Fig. 7: Average Response Time Comparison between HcDD and traditional hybrid storage

In summary, based on the evaluation results, we can see that HcDD fulfills the previous expectations. It not only responds I/O requests with a very competitive speed (better than the traditional hybrid disk), but it also removes duplicate writes, which normally occupies 30% to 46% of the total requests, to the SSD. It saves the storage space and extend the lifespan. Thus, we can observe that HcDD is a promising hybrid storage model, which balances the performance, saves the storage space and extends the lifespan of the SSD, for data-intensive server applications.

IV. Related Work

A. SSD and Hybrid Storage

NAND-flash-memory based SSDs, which used to be storage methods in mobile devices and lap-tops, play a rising role in revolutionizing storage systems [9][17][18][19][20][27]. SSDs are completely built on semiconductor chips without moving units, giving rise to high random access performance and lower power consumption. And the cost of commodity NAND flash – often cited as the primary barrier to SSD deployment [24] – has dropped significantly, increasing the possibility for dynastic changes in the storage arena. In order to improve the storage performance, the San Diego Supercomputer Center (SDSC) has built a large flash-based cluster called Gordon, which adopts 256TB of flash memory for the research purpose[6]. This research project is funded by a $20 million grant from the U.S. National Science Foundation.

Different from the research scenario, however, business data centers still are unable to adopt large-scale deployments of SSDs due to its high Gigabyte per price (80/GB of an SSD vs. 5/GB of an HDD [15][29]) and limited lifespan of SSDs. When it comes to the limited lifespan, the challenge is that every single block on a flash-based storage media has limited erasure cycles and each block has to be erased before being written. Thus erase-before-write operations not only degrade an SSD write performance, but also shorten the SSD lifespan.

B. Internal Parallelism Processing on SSD

The inter-disk parallelism technique has been well explored since decades ago. Data striping is the basic idea of inter-disk parallelism. However, the result of applying the parallel disks storage mechanism is that storage systems consume a significant amount of energy– as mush as 27% of the energy in a modern data center is consumed by storage devices [4][30]. Since (1) there is no mechanical movements in SSDs and (2) an SSD has one or multiple identical elements which can work in parallel [2], we have better chance to improve internal parallelisms in SSDs than HDDs. Park pointed out that intra-SSD parallelism is possible on die-level, package-level, and plane-level. Furthermore, parallelism-aware request processing is an effective solution to enhance intra-SSD parallelisms [25]. Chen and Zhang analyzed the essential roles of exploiting internal parallelism SSDs in high-speed data processing [10].

Unlike HDDs, SSDs have a Flash Translation Layer (or FTL) implemented to emulate a hard disk drive by exposing an array of logical block addresses (LBAs) to the host. FTL averagely spreads erase workloads on flash-based storage. A ill-designed FTL algorithm not only reduces the SSDs performance, but also wears out SSDs storage units rapidly.

V. Conclusion and Future Work

In this paper, we propose a hybrid combination of disk arrays designed for active storage system, using hard disk drives as a write buffer to cache write requests and de-duplicate the redundant write requests to the SSD. Read requests are all served from SSDs directly.

In order to evaluate the proposed architecture, we design and implement a trace-driven storage system simulator for the hybrid active storage system. We compare our internal-parallelism algorithm with the classic LRU scheme in terms of single-SSD drive performance. The results shows that our proposed algorithm outgoes LRU in most cases, when the interleaving level is 8 and buffer size is 1 MB or 2 MB. Then, we compare the overall performance of HcDD with the other three storage configurations, including the traditional hybrid storage scheme, the SSDs disk array and the HDDs disk array. The evaluation results are generated using several traces collected from daily usages and some applications in academic and financial areas.

ACKNOWLEDGMENT

The work reported in this paper was supported by the US National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0917137 (CSR), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS), as well as Auburn University under a startup grant and a gift (Number 2005-04-070) from the Intel Corporation. Shu Yin's research was supported by the National Natural Science Foundation of China under Grant 61402158, the Hunan Province Natural Science Foundation under Grant 14JJ4025, the Scientific Research Foundation for the Returned Overseas Chinese Scholars under Grant [2013]1792, Ministry of Education of the People's Republic of China, as well as Hunan University under a start-up grant and an educational reform grant.

REFERENCES

[1] "Umass trace repository," http://traces.cs.umass.edu/index.php/Storage, December 2009.

[2] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for ssd performance," in *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ser. ATC'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 57–70. Available: http://dl.acm.org/citation.cfm?id=1404014.1404019

[3] D. G. Andersen and S. Swanson, "Rethinking flash in the data center," *IEEE Micro*, vol. 30, no. 4, pp. 52–54, Jul. 2010. Available: http://dx.doi.org/10.1109/MM.2010.71

[4] B. Battles, C. Belleville, S. Grabau, and J. Maurier, "Reducing data center power consumption through efficient storage," *Netapp white paper*, February 2007.

[5] S. Boboila and P. Desnoyers, "Write endurance in flash drives: Measurements and analysis," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, ser. FAST'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 9–9. Available: http://dl.acm.org/citation.cfm?id=1855511.1855520

[6] A. M. Caulfield, L. M. Grupp, and S. Swanson, "Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications," *SIGARCH Comput. Archit. News*, vol. 37, no. 1, pp. 217–228, Mar. 2009.

[7] F. Chen, S. Jiang, and X. Zhang, "Smartsaver: Turning flash drive into a disk energy saver for mobile computers," in *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, Oct 2006, pp. 412–417.

[8] F. Chen, D. A. Koufaty, and X. Zhang, "Hystor: Making the best use of solid state drives in high performance storage systems," in *Proceedings of the International Conference on Supercomputing*, ser. ICS '11. New York, NY, USA: ACM, 2011, pp. 22–32.

[9] S. Chen, "Flashlogging: Exploiting flash devices for synchronous logging performance," pp. 1–3, 2009.

[10] R. L. Feng Chen and X. Zhang, "Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing," Berkeley, CA, USA, 2011.

[11] G. Ganger, "The disksim simulation environment," http://www.pdl.cmu.edu/DiskSim/, March 2011.

[12] G. R. Ganger, "System-oriented evaluation of i/o subsystem performance," Tech. Rep., 1995.

[13] J. Gray, "Tape is dead, disk is tape, flash is disk. ram locality is king," http://research.microsoft.com/en-us/um/people/gray/JimGrayHomePageSummary.htm, 2007.

[14] S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker, "Oltp through the looking glass, and what we found there," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 981–992.

[15] T. Iyer, "Hdds return to pre-flood prices," http://www.tomshardware.com/news/Thailand-Flood-Storage-Price-SSD,22150.html, April 2013.

[16] Y. Joo, Y. Cho, K. Lee, and N. Chang, "Improving application launch times with hybrid disks," in *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '09. New York, NY, USA: ACM, 2009, pp. 373–382.

[17] W. K. Josephson, L. A. Bongo, K. Li, and D. Flynn, "Dfs: A file system for virtualized flash storage," *Trans. Storage*, vol. 6, no. 3, pp. 14:1–14:25, Sep. 2010.

[18] A. Kawaguchi, S. Nishioka, and H. Motoda, "A flash-memory based file system," in *Proceedings of the USENIX 1995 Technical Conference Proceedings*, ser. TCON'95. Berkeley, CA, USA: USENIX Association, 1995, pp. 13–13. Available: http://dl.acm.org/citation.cfm?id=1267411.1267424

[19] H. Kim and S. Ahn, "Bplru: A buffer management scheme for improving random writes in flash storage," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, ser. FAST'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 16:1–16:14. Available: http://dl.acm.org/citation.cfm?id=1364813.1364829

[20] T. Makatos, Y. Klonatos, M. Marazakis, M. D. Flouris, and A. Bilas, "Using transparent compression to improve ssd-based i/o caches," in *Proceedings of the 5th European Conference on Computer Systems*, ser. EuroSys '10. New York, NY, USA: ACM, 2010, pp. 1–14.

[21] B. Mao, H. Jiang, D. Feng, S. Wu, J. Chen, L. Zeng, and L. Tian, "Hpda: A hybrid parity-based disk array for enhanced performance and reliability," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010, pp. 1–12.

[22] Microsoft, "Ssd extension for disksim simulation environment," http://research.microsoft.com/en-us/downloads/b41019e2-1d2b-44d8-b512-ba35ab814cd4/, March 2009.

[23] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and R. A., "Migrating enterprise storage to ssds:analysis of tradeoffs," March 2009.

[24] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating server storage to ssds: Analysis of tradeoffs," in *Proceedings of the 4th ACM European Conference on Computer Systems*, ser. EuroSys '09. New York, NY, USA: ACM, 2009, pp. 145–158.

[25] C. Park, E. Seo, J.-Y. Shin, S. Maeng, and J. Lee, "Exploiting internal parallelism of flash-based ssds," *Computer Architecture Letters*, vol. 9, no. 1, pp. 9–12, Jan 2010.

[26] L. Prada, J. Garcia, J. Carretero, and F. Garcia, "Saving power in flash and disk hybrid storage system," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, Sept 2009, pp. 1–3.

[27] T. Pritchett and M. Thottethodi, "Sievestore: A highly-selective, ensemble-level disk cache for cost-performance," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 163–174.

[28] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multiprocessor systems," in *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, Feb 2007, pp. 13–24.

[29] A. Richi, "Ssd price crash:80 per gb!" http://h30565.www3.hp.com/t5/Mobility-Matters/SSD-price-crash-80-per-GB/ba-p/5032, February 2012.

[30] S. Sankar, S. Gurumurthi, and M. R. Stan, "Intra-disk parallelism: An idea whose time has come," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 303–314. Available: http://dx.doi.org/10.1109/ISCA.2008.11

[31] G. Soundararajan, V. Prabhakaran, M. Balakrishnan, and T. Wobber, "Extending ssd lifetimes with disk-based write caches," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, ser. FAST'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 8–8. Available: http://dl.acm.org/citation.cfm?id=1855511.1855519

[32] A.-I. A. Wang, G. Kuenning, P. Reiher, and G. Popek, "The conquest file system: Better performance through a disk/persistent-ram hybrid design," *Trans. Storage*, vol. 2, no. 3, pp. 309–348, Aug. 2006.