Hybrid Solid-State Disks: Combining Heterogeneous NAND Flash in Large SSDs*

Li-Pin Chang
Department of Computer Science
National Chiao-Tung University, Hsin-Chu, Taiwan
lpchang@cs.nctu.edu.tw

Abstract— NAND-flash-based SSDs (solid-state disks) are recently used in embedded computers to replace power-hungry disks. This paper presents a hybrid approach to large SSDs, which combines MLC flash and SLC flash. The idea is to complement the drawbacks of the two kinds of NAND flash with each other's advantages. The technical issues of the design of a hybrid SSD pertain to data placement and wear leveling over heterogeneous NAND flash. Our experimental results show that, by adding a 256 MB SLC flash to a 20 GB MLC-flash array, the hybrid SSD improves over a conventional SSD by 4.85 times in terms of average response. The average throughput and energy consumption are improved by 17 % and 14%, respectively. The hybrid SSD is only 2% more expensive than a purely MLC-flash-based SSD, for which the extra cost is limited and very rewarded.

Type	SLC	MLC
Product No	K9F2G08UXA	K9GAG08B0M
Capacity	256M*8 bits	2G*8 bits
Page Size / Block Size	2K+64 / 128K+4K bytes	4K+128 / 512K+16K bytes
Page Read	25 us	60 us
Page Write	200 us	800 us
Block Erase	1.5 ms	1.5 ms
Block Endurance	100K cycles	10K cycles
Operating Characteristics of Read, Write, and Erase	3.3V / 15mA	3.3V / 15mA
Standby Power Dissipation	3.3V / 10uA	3.3V / 10uA
Cost per GB (mid 2007)	$10.37	$6.81

TABLE I
COMPARISON BETWEEN THE SPECIFICATIONS OF A TYPICAL SLC FLASH AND A TYPICAL MLC FLASH [5, 6].

I. INTRODUCTION

One among the design challenges of mobile computers is that disk-based mass storage systems are power hungry and fragile to shock. Recently, mobile-computer vendors start replacing hard drives with NAND-flash-based solid-state disks. because flash memory is non-volatile, power-economic, shock-resistant, and free from positioning penalty. An SSD comprises a micro-controller and many NAND-flash chips [7]. It interacts with the host computer (e.g., a mobile PC) via standard interface such as ATA or SATA as if it were a hard drive.

Even though SSDs better match the natures of embedded computers, one concern of the SSD technology is the cost of flash memory. Ordinarily, one flash-memory cell represents binary value. This is referred to as single-level-cell flash (i.e., SLC flash). If multiple voltage thresholds are defined and a memory cell can be reliably charged, then more than one bit can be represented. It is referred to as multi-level-cell flash (i.e., MLC flash). MLC flash is cheaper and larger than SLC flash. However, SLC flash is faster, more durable, and more power-economic than MLC flash. Table I is a comparison of an SLC flash and an MLC flash [5, 6].

In this paper, a hybrid approach to large SSDs is proposed. A hybrid SSD comprises one single small SLC-flash chip and a number of large MLC-flash chips. Our basic idea is to complement the drawbacks of SLC flash and MLC flash with each other's advantages. With proper management, a hybrid SSD would response as fast as a purely SLC-based SSD and cost as low as a purely MLC-based SSD. The management of heterogeneous NAND flash introduces several challenges. First, data placement over SLC flash and MLC flash should be aware of the physical characteristics of the two kinds of NAND flash. The SLC flash better handles small and frequently arriving data, while MLC flash is suitable to store bulk and cold data. Second, as SLC flash is relatively expensive, the SLC flash in a hybrid SSD is intended to be small. A strategy is needed to phased out data from the SLC flash to maximize its accommodation of newly arriving data. Lastly, wear leveling is conducted over blocks of different endurance. The placement of data has to consider not only efficiency but also lifetime of heterogeneous flash-memory blocks. A strategy is needed to resolve the conflict between the two.

The rest of this paper is organized as follows: Section II includes prior work related to the management of heterogeneous memory. Section III introduces the proposed hybrid SSDs. Section IV provides experimental results, and this paper is concluded in Section V.

II. RELATED WORK

In consideration of cost, power consumption, and design form-factor, embedded computers are usually equipped with heterogeneous memory including SRAM, DRAM, and non-volatile RAM. The management of heterogeneous memory has been an increasingly important issue. The technical issues mainly pertain to data placement and migration, as different kinds of memory show the ability of handling different data access patterns.

Avissar et al. [12] considered a compile-time technique to partition data among scratch-pad SRAM, local DRAM, shared DRAM, and ROM. They showed that the technique can signif-

*This work is in part supported by research grant NSC-96-2218-E-009-017 from National Science Council, Taiwan, ROC.

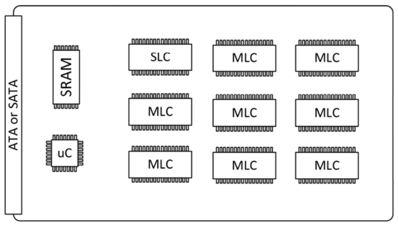

Fig. 1. The architecture of a hybrid SSD.

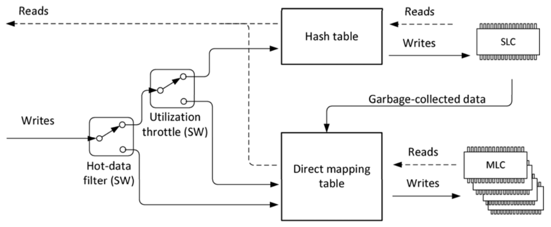

Fig. 2. Data/control flows of the handling of reads and writes inside a hybrid SSD.

icantly boost program runtime. Lee et al. [10] investigated energy-aware memory allocation and migration over different non-volatile RAM, including batter-backed SDRAM, NOR flash, and NAND flash. Heterogeneous memory management for storage systems is also a blooming research topic. In industry, at 2006 Samsung and Segate showed a prototype of hybrid disks that combine a hard drive and NAND flash. Hybrid disks provide host OS special commands to "pin" necessary data in NAND flash. It accelerates system boot because NAND flash reads much faster than hard drives. Related technologies including ReadyBoost™ by Microsoft and Turbo Memory by Intel. In academia, Kim et al. [9] proposed a file-placement method over NAND flash and hard drive for energy saving. Bisson et al. [14] and Chen et al. [15] proposed to use a USB flash drive as a buffer cache of a hard drive for energy saving at block level.

III. Hybrid Solid-State Disks

A. Overview

The layout inside a typical SSD is illustrated in Figure 1. An array of NAND-flash chips are managed by a micro controller, and an optional SDRAM is attached to the controller. In conventional SSDs, the NAND-flash chips are all of the same type (i.e., SLC flash or MLC flash). The SSD firmware implements Flash-memory Translation Layer (i.e., FTL) to emulate disk geometry. An SSD accepts reads and writes as if it were a hard drive, translates disk geometry into flash-memory physical locations, handles garbage collection and wear leveling, and responses to the host upon the completion of data transfer. A piece of data is addressed by *logical addresses* and is accessed by *physical addresses*. The two stand for disk-sector numbers and flash-memory locations, respectively. An SSD never buffers data in volatile memory to guarantee the write-through semantic for data integrity. The architecture of a hybrid SSD is exactly the same as that of a conventional SSD, except that it combines an SLC-flash chip with many MLC-flash chips, as Figure 1 shows. Note that the pin definition and the package of the two kinds of flash memory are compatible (i.e., TSOP-1 48 pin)[5, 6].

Figure 2 shows data flow and control transfer inside hybrid SSDs. Let a piece of data be hot if writes to the data arrive frequently, otherwise it is non-hot or cold. Upon the arrival of a write request, if it does not go to hot data, it is rejected by the *hot-data filter* and is directed to the MLC flash. A write passing through the hot-data filter then arrives the *utilization throttle*. The utilization throttle is to regulate the wearing of SLC flash and MLC flash. As Table I shows, typical SLC-flash blocks and MLC-flash blocks endure 100K and 10K erasure cycles, respectively. So if the SLC flash is worn 10 times faster than the MLC flash, the utilization throttle start rejecting some of the writes to quench the wearing of the SLC flash.

The SLC flash and the MLC flash are separately managed. Different schemes are adopted by the two kinds of flash memory to deal with fundamental issues including address translation, garbage collection, and wear leveling. Also, to maximize the accommodation of hot data, a piece of data will be phased out from the SLC flash if it becomes non-hot.

B. The Hot-Data Filter

The hot-data filter decides whether a newly arriving write should be accepted by the SLC flash. The decision cannot consider none of the differences between SLC flash and MLC flash. The advantages of SLC flash over MLC flash are: 1) SLC flash is much faster than MLC flash in terms of writes, 2) SLC-flash blocks are more durable than MLC-flash blocks, and 3) the geometry of SLC flash is finer than that of MLC flash. However the disadvantage is that SLC flash is more expensive than MLC flash. On the other hand, by analyzing realistic access patterns, we found that the majority of requests are small writes. Most of the small writes repeatedly go to a small amount of hot data, and non-hot data are touched by bulk writes with very low frequencies. It is also found that any piece of data is rarely read multiple times because the host has a data cache.

The above observations suggest that writes to hot data can better be handled by SLC flash. By quickly absorbing the frequently arriving writes to hot data, the overall response can largely be boosted. In addition, because the life cycles of hot data are very short, the SLC flash needs not be large. It would then be a question how writes to hot data can be identified. Past work regarding hot-data identification such as LRU (least-recently used), LRU-k, and hash-table-based approaches [4] have been proposed. However, they are complicated and may require large RAM space.

A new strategy dedicated to the behaviors of realistic SSD workloads is proposed. We have profiled a realistic SSD workload. According to our analysis (which is to be presented in Section IV.B), we found that the distribution of writes with respect to their sizes is bimodal: Most of the writes are either large or small. Writes are rarely of "medium" sizes. Small writes mostly go to hot data, while large writes seldom touch a piece of data more than once. Based on the observation, a threshold algorithm based over the distribution of writes respect to their sizes is proposed. Algorithm 1 shows a threshold

Algorithm 1 The threshold (2-means clustering) algorithm

```
Require: c[0...n]: $n$ counters to accumulate writes of sizes $2^0 \sim 2^n$.
Define $f(i,j,p)=\sum_{k=0}^{p} c[i] \cdot |i-p| + \sum_{k=p+1}^{n} c[j] \cdot |j-p|$
1: $i$=0; $j$=0; $p$=0;
2: while ($c'$ != $c$) do
3:     Fix $p$. Find $i$ and $j$ to minimize $c$=$f(i,j,p)$;
4:     Fix $i$ and $j$. Find $p$ to minimize $c'$=$f(i,j,p)$;
5: end while
6: return $i$; /* $i$ is the threshold of small writes */
```

algorithm that finds the two majorities among write sizes. The algorithm is based on the k-means clustering algorithm [16]. In the threshold algorithm, $c[]$ is an array of counters, and $c[i]$ is the accumulated number of writes of 2^i sectors. The largest acceptable write is 1024 sectors, and therefore only 11 counters (i.e., $2^0 \sim 2^{10}$) are kept in RAM. The algorithm iteratively finds the two majorities of write sizes (i.e., i and j in Step 3) and the best partition of write sizes (i.e., p in Step 4).

In our current design, the algorithm is periodically invoked (e.g., every 1,000 writes) to redefine the threshold of small writes (i.e., i). On the arrival of a write, if it is no larger than the threshold, it is accepted by the SLC flash. Otherwise it is rejected. In our experiments, most of the time the threshold remains 4 KB (i.e., 8 sectors).

C. Managing the SLC flash

Not surprisingly, the management of the SLC flash involves address translation [8], garbage collection [11], and wear leveling [13]. Garbage collection erases blocks to reclaim free space, and wear leveling tries to evenly distribute erasures over blocks. In addition, because the SLC flash is small, free space should also be reclaimed by moving non-hot data away to accommodate newly arriving hot data.

We propose to treat physically contiguous SLC-flash blocks as a circular log space, as shown in Figure 3. A head pointer refers to a head block to which newly arriving data are written to, and a tail pointer indicates a tail block from which free space should be reclaimed. The number of blocks between the two pointers is no larger than a pre-defined parameter k. Initially the two pointers refer to the same block. Blocks that are not between the two pointers must be all of free space. The pointers move toward large block addresses, and they are rewound when they grow beyond the last block.

On the arrival of a write, the data are written to free space in the head block. The head pointer is advanced if there is no free space left. The newly arriving data invalidate old versions of the data on the SLC flash. Whenever the head pointer is ahead of the tail pointer by more than k blocks, the tail block is erased for garbage collection, and then the tail pointer advances. The up-to-date version of data always appear close to the head pointer. The closer to the tail block a piece of valid data is, the more infrequent the update to the data is. Therefore any valid data found in the tail block are non-hot and should be phased out from the SLC flash. The management method has two major benefits: First, because most of the data on the SLC flash are hot, valid data are rarely found in the tail block. Thus garbage collection involves few copy operations. Second, as blocks are sequentially erased, wear leveling over blocks is perfect.

As to address translation, because a large logical-address space (i.e., disk sectors) is mapped a small physical-address space (i.e., blocks and pages), to use a direct mapping table would result in large RAM-space requirements. Instead, a small hash table is adopted, as indicated by Figure 2. Note that, due to hash collision, a write may not be accepted by the SLC flash.

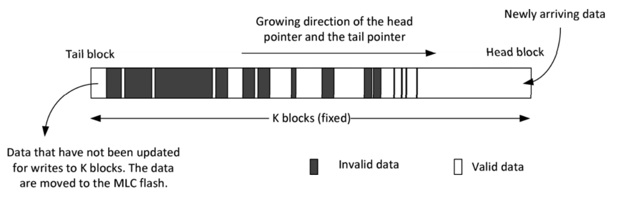
Growing direction of the head pointer and the tail pointer
Newly arriving data
Tail block
Head block
K blocks (fixed)
Data that have not been updated for writes to K blocks. The data are moved to the MLC flash.
Invalid data
Valid data

Fig. 3. How data are written onto and phased out from the SLC flash.

D. Managing the MLC flash

The management of the MLC flash also need to handle address translation, garbage collection, and wear leveling. Basically the MLC flash is managed as if the SLC flash were not there. There are three kinds of data that will be written to the MLC flash: Data rejected by the hot-data filter, data phased out from the SLC flash, and data rejected by the SLC flash because of wear leveling. The last case would be explained in the next section.

As the SLC flash intercepts hot data, most of the data reside in the MLC flash are non-hot. If the circular-logging approach is used to manage the MLC flash, there would be too many data found valid in the tail block. Garbage collection is very inefficient this way. Instead, free space should be reclaimed with the intention to reduce data-copy operations. We select the greedy policy [1] as the garbage-collection policy, which always chooses to erase a block of the smallest amount of valid data. Because the greedy policy is not aware of data attributes (i.e., hot or non-hot), it is reported that it sometimes suffers from performance issues [11]. However, as hot data won't arrive at the MLC flash, this concern is much relieved.

Unlike the circular-log-based approach, the wearing of MLC-flash blocks would gradually become uneven, because the greedy policy never prefers blocks having many static and valid data (e.g., read-only data). The dual-pool algorithm [13] is chosen as the wear-leveling algorithm because it better handles SSD workloads and it scales to very large SSDs.

For address translation, because the MLC flash is mapped to almost all the logical addresses, a conventional direct-mapping table is a good choice. Thus how large should the mapping-unit size be is a question. As reported in [8], large mapping units greatly shrink the size of the mapping table, while small mapping units benefit the handling of small writes. Because bulk writes mostly go to the MLC flash for non-hot data, coarse-grain mapping is a good choice for the MLC flash.

As our approach primarily aims at how a small SLC flash helps to improve SSD performance, we must point out that how the MLC flash is managed does not much concern the idea of hybrid SSDs. Many excellent management methods, such as the log-block-based FTL [17], can be adopted to manage the MLC flash.

Algorithm 2 The utilization-throttling algorithm

```
Require: $W$,$D$: Write request $W$ goes to data $D$; c: A static counter.
1: if ($rw_s \geq rw_m$) then
2:     $c \leftarrow c + 1$;
3:     if ($D$ is in the SLC flash) then
4:         return ACCEPT;
5:     end if
6: end if
7: if ($c \geq 1000$) then
8:     $k \leftarrow k - 100$; $c \leftarrow 0$; /* How fast it reacts can be tuned */
9: end if
10: return REJECT;
```

	SLC-flash specification				MLC-flash specification			
	Page size/ Block size	Mapping-unit size	Block endurance	Page read/ page write/ block erase	Page size/ Block size	Mapping-unit size	Block endurance	Page read/ page rite/ block erase
C1	2K B/128K B	2 KB	100 K cycles	25 us/200 us/ 1.5 ms	2K B/128K B	2 KB	10 K cycles	60 us/800 us/ 1.5 ms
C2	2K B/128K B	2 KB	100 K cycles	25 us/200 us/ 1.5 ms	4K B/256K B	16 KB	10 K cycles	60 us/800 us/ 1.5 ms
C3	2K B/128K B	2 KB	100 K cycles	25 us/200 us/ 1.5 ms	4K B/512K B	64 KB	10 K cycles	60 us/800 us/ 1.5 ms

TABLE II
CONFIGURATIONS OF EXPERIMENTAL HYBRID SSDS.

E. The Utilization Throttle

As shown above, wear leveling over flash-memory blocks of the same type (i.e., SLC or MLC) is separately enforced. This does not balance the lifetime of the SLC flash and the MLC flash, however. As shown in Table I, the two kinds of flash-memory blocks endure differently. It is possible that the SLC flash reaches its lifetime before the MLC flash, and vice versa. Thus a question is how to regulate the wearing among heterogeneous flash-memory blocks.

A mechanism, named the *utilization throttle*, is proposed. The idea is to smartly reduce the write traffic to the SLC flash so that the wearing of the SLC flash can gradually be quenched. Let the total number of SLC-flash blocks and MLC-flash blocks be n_s and n_m, respectively. Let the erasure-cycle count of an SLC-flash block b_i^s be $e^{b_i^s}$, and its endurance be d^s. Let erasure-cycle count $e^{b_j^m}$ and endurance be d^m of an MLC-flash block b_j^m be accordingly defined. The relative wearing of the SLC flash rw_s and the MLC flash rw_m are defined as

$$rw_s = \frac{\sum_{i=0}^{n_s} e^{b_i^s}}{n_s} / \frac{d^s}{d^m}, \text{ and } rw_m = \frac{\sum_{j=0}^{n_m} e^{b_j^m}}{n_m} / 1$$

, respectively. Whenever $rw_s \geq rw_m$ is true, the utilization throttle is activated. Firstly the utilization throttle rejects writes to data that do not already exist in the SLC flash. Note that, if a piece of data is very hot, it repeatedly appears close to the head block and won't be phased out. Thus writes to very hot data would still arrive at the SLC flash. If $rw_s \geq rw_m$ remains after a period of time, the utilization throttle start decreasing parameter k. As k decreases, the tail pointer would quickly catch up with a piece of valid data. Only data updated with high frequencies won't be phased out by garbage collection, and thus the rejection of writes becomes aggressive. Still, the writes to very hot data are not affected because the data are rarely phased out from the tail block. The algorithm to throttle the traffic and to decrease parameter k is shown in Algorithm 2. Once the wearing of the SLC flash is properly controlled, the utilization throttle becomes inactive and parameter k is gradually enlarged.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup and Performance Metrics

A simulator is built to conduct experiments on the proposed hybrid SSDs and conventional SSDs. A realistic workload was gathered from a real-life mobile computer as the experimental data set. Three different hybrid-SSD configurations are considered in our experiments, as listed in Table II. The size of the SLC flash varies between experiments, and the total size of the MLC flash is fixed at 21 GB. The specifications of the SLC flash and the MLC flash are based on real products [5, 6]. Note that no parallelism among NAND-flash chips is considered.

Because an SSD interacts with the host as if it were a hard drive, disk I/O operations issued by the host are considered as the experimental workload. The disk traces were gathered from a real-life UMPC (Ultra Mobile PC) ASUS R2H by using a filtering driver inserted into the OS kernel. The UMPC is equipped with a Celeron-M ULV processor, 768 MB of RAM, and a 20GB disk. Thus in the experiments the space utilization of flash memory is *no higher* than 20/21=95%. Of the UMPC, the operating system is Windows XP, and the file system is NTFS. Applications ran on the UMPC are ordinary to people, including web browsers, email clients, movie players, FTP clients, office suites, and games. The time duration of trace collecting is one month long.

Experimental metrics on performance improvement, flash-memory lifetime, and hardware costs are considered. Regarding performance, response-speedup ratios (RS ratios) and throughput-speedup ratios (TS ratios) are defined. Let there be total n requests (reads and writes) in the experimental workload. Let R_i^S, R_i^H be the response time of request q_i in a conventional SSD and a hybrid SSD, respectively. Let D_i be the size of request q_i. RS ratios and TS ratios are defined as follows:

$$\text{RS ratio} = (\sum_{i=1}^{n} \frac{R_i^S}{R_i^H})/n, \text{ TS ratio}= (\frac{\sum_{i=1}^{n} D_i/R_i^H}{\sum_{i=1}^{n} D_i/R_i^S})/n$$

Intuitively, the RS ratios equally account the speedup to each individual request, while the TS ratio is more concerned about the overall speedups. Regarding the lifetime issue, it is concerned with two metrics: 1) The evenness of the wearing among blocks of the same type and 2) The ratio of the wearing of the SLC flash to that of the MLC flash. The second item needs to comply with the endurance of SLC-flash blocks and MLC-flash blocks. As to hardware costs, energy-saving ratios (ES ratios) and extra-cost ratios (EC ratios) are adopted. The ES ratio stands for the ratio of the total energy consumed by a hybrid SSD to that consumed by a conventional SSD. The EC ratio denotes the ratio of the total cost of flash-memory chips of a hybrid SSD to that of a conventional SSD. Readers may refer to Table I for the costs and the energy models of SLC flash and MLC flash.

B. A Profile of the Experimental Workload

To help to better understand our experimental results, this section provides a profile of the experimental workload. The

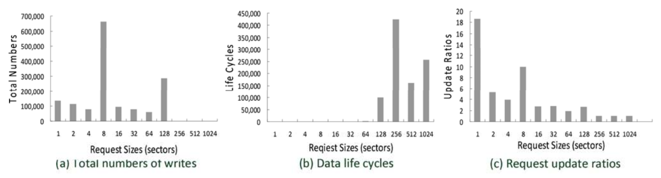

Fig. 4. The profile of a realistic SSD workload.

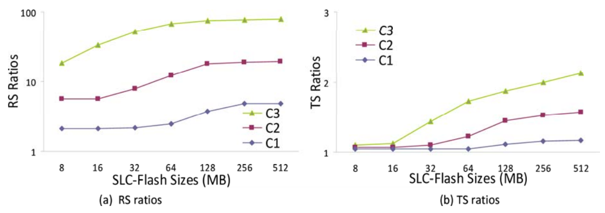

Fig. 5. (a) Response-speedup ratios (RS ratios) and (b) throughput-speedup ratios (TS ratios) achieved by different hybrid SSDs (The MLC flash is 21 GB).

following discussions are focused on writes because they concern flash-memory management most. Figure 4 shows the results of analyzing the gathered disk traces. The X-axes refer to the sizes of write requests in terms of disk sectors. Figure 4(a) is the distribution of writes with respect to different request sizes. The Y-axis stands for the total number of requests. The distribution seems bimodal: writes of 8 sectors and 128 sectors contribute the largest number and the second largest number of writes. It also shows that, in most of the time, the disk was accessed by small writes.

It is interesting whether there is any connection between sizes of write and attributes of written data (i.e., hot or non-hot). Let the *life cycle* of a sector be the average number of total requests between two successive writes to the sector. If the life cycle is related to a write size, then only consecutive writes of that size are considered. Figure 4(b) shows that the life cycles of data touched by small writes are very short, and data are touched by bulk writes with very low frequencies. To strengthen our observations, let the *update ratio* be denoted as the ratio of the total write traffic to the total amount of distinct data touch by the writes. If the ratio is related to a request size, only writes of that size are considered. Figure 4(c) shows that small writes contribute high update ratios. It means that small writes do update most of the time. In summary, if a write is small, then very possibly the write goes to hot data.

C. Performance Improvement over Conventional SSDs

The first part of experiments examines how hybrid SSDs improve over conventional SSDs in terms of response and throughput. How is the most effective combination of SLC flash and MLC flash is also investigated.

Figure 5(a) and Figure 5(b) shows the RS ratios and the TS ratios, respectively, of different hybrid-SSD configurations. The X-axes denote the size of the SLC flash, and the Y-axes stand for the RS ratios and the TS ratios. *Let us focus on configuration C_1 first.* When the SLC flash is no larger than 32 MB, there are no noticeable improvements. The 32 MB SLC

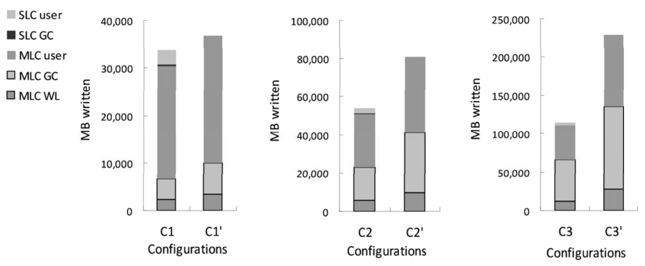

Fig. 6. Distributions of data written by the handling of user requests, garbage collection, and wear leveling. The SLC flash is 256 MB in C_1, C_2, and C_3. There is no SLC flash in C_1', C_2' and C_3'.

flash has only 256 blocks. The handling of all the small writes would distribute large erasure cycles over the 256 blocks, and the wearing of the small SLC flash would be too fast. As a result, many writes are rejected by the utilization throttle. When the SLC flash is larger than 32 MB, RS ratios and TS ratios are much improved. The larger the SLC flash is, the more writes the SLC flash can accept. The improvement saturates when the SLC flash is 256 MB. The RS ratio and the TS ratio are 4.85 and 1.17, respectively. Because the geometry of the SLC flash and the MLC flash are the same in configuration C_1, the speedups are solely contributed by fast operations of SLC flash.

The RS ratios and the TS ratios of configurations C_2 and C_3 are larger than that of configuration C_1. It can better be explained by Figure 6, which depicts the total amount of data written by handling user requests, garbage collection, and wear leveling. The SLC flash is 256 MB in configurations C_1, C_2, and C_3. In configurations C_1', C_2', and C_3', there is no SLC flash. Without SLC flash, each small write rewrites a large mapping unit of MLC flash. The larger the mapping units are, the more costly the handling of small writes is [8]. On the other hand, the larger a flash-memory block is, the more data-copy operations would potentially be involved in garbage collection [11]. Configuration C_3' is most benefited by adding an SLC flash because its mapping-unit size and the block size of the MLC flash are the largest.

D. Wear-Leveling Issues

This part of experiments examines 1) the wearing of blocks of the same type and 2) the wearing of the SLC flash and the MLC flash. Configuration C_1 is used, and the SLC-flash size is 256 MB. In the experiments, the MLC-flash block endurance remains 10K cycles. Two SLC-flash block endurance 100K and 50K cycles are considered. So the ratio of SLC-block endurance to MLC-block endurance is 10:1 or 5:1. To amplify the accumulation of erasure cycles over blocks, the entire traces are replayed 10 times over our simulator.

Figure 7 shows the distributions of erasure cycles (i.e., ECs) over SLC-flash blocks and MLC-flash blocks. The X-axis and the Y-axis denote the physical block addresses and the erasure cycles, respectively. The wearing of SLC-flash blocks is shown perfect. It is because the blocks are erased in a round-robin fashion. The distributions of erasure cycles of MLC-flash blocks also seem very even, as the dual-pool algorithm [13] is used. When the ratio of block endurance is 10:1, the average block-erasure cycles of the SLC blocks and the MLC

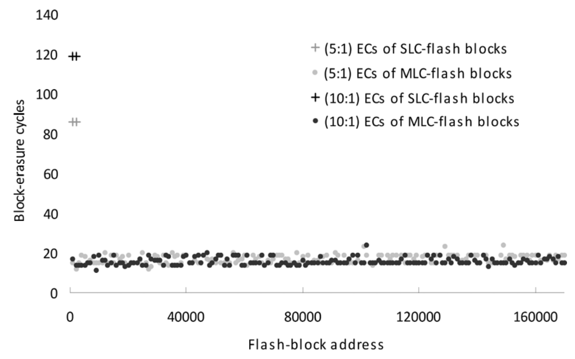

Fig. 7. Distributions of block-erasure cycles of the SLC flash and the MLC flash under different ratios of SLC-block endurance to MLC-block endurance.

Configuration	SLC size (MB)	MLC size (GB)	EC ratio	ES ratio
C1	256	21	1.02	0.86
C2	256	21	1.02	0.66
C3	256	21	1.02	0.48

TABLE III
THE EXTRA-COST RATIOS (EC RATIOS) AND ENERGY-SAVING RATIOS (ES RATIOS) OF DIFFERENT HYBRID-SSD CONFIGURATIONS.

blocks are 118 and 16, respectively. The actual wearing ratio is 118/16=7.375, which is smaller than 10/1. In this case the utilization throttle is inactive since the SLC flash has sufficiently many blocks to share the wearing introduced by the handling of all the small writes. When the ratio of block endurance is 5:1, the actual wearing ratio is 85.5/17 $\approx$ 5, which matches 5:1. In this case, the SLC flash can not sustain all the small writes and the utilization throttle start rejecting writes. The utilization throttle is proven effective.

E. Energy Consumption and Hardware Costs

This part of experiments are focused on energy consumption and hardware cost of hybrid SDDs. Table III shows the extra-cost ratios (EC ratios) and energy-saving ratios (ES ratios) of different hybrid-SSD configurations (please refer to Section IV.A for the definitions of ES ratios and EC ratios). Let the SLC flash be 256 MB in configurations C_1, C_2, and C_3. The total cost of hybrid SSDs and conventional SSDs are calculated based on Table I. The cost of a hybrid SSD is only 1.02 times the cost of a purely MLC-flash-based SSD. Regarding energy, Table III shows that hybrid SSDs consume less energy than conventional SSDs. In configuration C_1, the ES ratios are maintained at 86%. Surprisingly, if the geometry of MLC flash is coarser than that of SLC flash (i.e., configuration C_2 and C_3), energy saving is significant. To configuration C_3, a hybrid SSD reduces the total energy consumption of a conventional SSD by 52%. That is because the SLC flash efficiently handles small writes. Extra traffic due to large mapping units and inefficient garbage collection are greatly reduced, as mentioned in the previous section.

V. CONCLUSION

As high-capacity NAND flash is becoming affordable, to replace power-hungry disks with large and cheap solid-state disks is an emerging application. However, there is always compromise between storage density and performance, as reflected by the physical characteristics of SLC flash and MLC flash. In this paper, it is investigated to combine heterogeneous NAND-flash chips in large SSDs. The idea is to complement the drawbacks of the two kinds of NAND flash with each other's advantages. We proposed data placement/migration and wear leveling strategies over heterogeneous NAND flash. The design and evaluation of the proposed hybrid SSD are based on realistic SSD workloads. Our main result is that a hybrid SSD greatly improves over a conventional SSD in terms of response, throughput, and even energy consumption. All the improvements are achieved by adding a small SLC flash to a large MLC-flash array. The extra cost paid for a hybrid SSD is limited and very rewarded.

REFERENCES

[1] A. Kawaguchi, S. Nishioka, and H. Motoda,"A flash-memory based File System," Proceedings of the USENIX Technical Conference, 1995.
[2] C. Reummler and J, Wilkes, "UNIX disk access patterns," Proceedings of USENIX Technical Conference, 1993.
[3] W. Vogels, "File system usage in Windows NT 4.0," Proceedings Of the 17-th ACM Symposium On Operating Systems Principles, 1999
[4] J. W. Hsieh, T. W. Kuo, and L. P. Chang "Efficient Identification of Hot Data for Flash Memory Storage Systems," ACM Transactions on Storage, Volume 2, Issue 1, 2006.
[5] Samsung Electronics Company, "K9NBG08U5M 4Gb * 8 Bit NAND Flash Memory Data Sheet".
[6] Samsung Electronics Company, "K9GAG08U0M 2G * 8 Bit NAND Flash Memory Data Sheet (Preliminary)".
[7] Samsung Electronics Company, "NAND Flash-based Solid State Disk Data Sheet," http://www.samsung.com/ Products/ Semiconductor/ FlashSSD/ download/ Standard_type.pdf.
[8] L. P. Chang and T. W. Kuo, "Efficient Management for Large-Scale Flash-Memory Stroage Systems with Resource Conservation", ACM Transactions on Storage, Vol. 1, Issue 4, 2005.
[9] Y. J. Kim, K. T. Kwon, and J. Kim, "Energy-efficient file placement techniques for heterogeneous mobile storage systems," Proceedings of the 6th ACM & IEEE international Conference on Embedded Software, 2006.
[10] H. G. Lee and N. Chang, "Energy-Aware Memory Allocation in Heterogeneous Non-volatile Memory Systems," Proceedings of International Symposium on Low Power Electronics and Design, 2007.
[11] M. L. Chiang, Paul C. H. Lee, and R. C. Chang, "Using Data Clustering To Improve Cleaning Performance For Flash Memory," Software - Practice and Experience, Vol. 29, No. 3, 1999.
[12] O. Avissar, R. Barua, and D. Stewart, "An Optimal Memory Allocation Scheme for Scratch-Pad-Based Embedded Systems," ACM Transactions on Embedded Computing Systems, Vol. 1, No. 1, 2002.
[13] L. P. Chang , "On Efficient Wear-Leveling for Large-Scale Flash-Memory Storage Systems," Proceedings of the 22st ACM Symposium on Applied Computing (ACM SAC), 2007.
[14] T. Bisson and S. Brandt, "Reducing energy consumption with a non-volatile storage cache," Proceedings of International Workshop on Software Support for Portable Storage (IWSSPS), 2005.
[15] F. Chen, S. Jiang, and X. Zhang, "SmartSaver: turning flash drive into a disk energy saver for mobile computers," in Proceedings of 11th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2006.
[16] E. Forgy, "Cluster Analysis of Multivariate Data: Efficiency versus Interpreability of Classifications.", Biomertrics 21, 768. 1965.
[17] S. W. Lee, D. J. Park, T. S. Chung, D. H. Lee, S. W. Park, and H. J. Song, "A log buffer-based flash translation layer using fully-associative sector translation," ACM Transactions on Embedded Computing Systems, Vol 6, Issue 3, 2007.