BEST: Best-effort Energy Saving Techniques for NAND Flash-based Hybrid Storage

Hyotaek Shim, Jin-Soo Kim, *Member*, IEEE, and Seungryoul Maeng

Abstract — *The NAND flash-based hybrid storage that consists of a hard disk and flash cache has been widely used in mobile PCs, such as laptop/desktop computers, that require large storage capacity, energy efficiency, and better performance. In particular, in such consumer devices, energy saving is a major concern under limited battery capacity. Although a disk can be spun down during idle time for energy saving, it has been observed that there are frequent read and write requests generated by background applications, even while there is no user activity.*

In the hybrid storage, the flash cache is used for reducing disk activities and prolonging disk spin-down time. During the spin-down time, write requests can be temporarily stored in the flash cache, whereas read requests should be carefully handled to avoid frequent disk spin-ups. This paper presents Best-effort Energy Saving Techniques (BEST) that make the best use of the hybrid storage to provide low energy consumption, even under many background applications. In addition, the proposed techniques ensure the lifetime of the hybrid storage by considering the limited block erase and load/unload cycles of the flash memory and hard disk, respectively. Evaluation results demonstrate that the proposed techniques achieve considerable energy saving under various workloads, compared with previous studies[1].

Index Terms —Power consumption, disk spin-down, hybrid storage, NAND flash memory

I. INTRODUCTION

Hard disks have been widely-used storage media in consumer devices that require large storage capacity, such as laptops/notebooks and portable media players [1], [2]. In such consumer devices, it has been considered as essential to extend operating time under limited battery capacity. In particular, hard disks are the main source of power dissipation due to their mechanical movements [3], [4]. To reduce the energy consumption, a hard disk can be spun down when it is not in use. However, the misprediction of idle periods makes the disk spin up frequently, rather consuming much power. Therefore, determining when to make a disk spin down to conserve energy remains a challenging issue.

To make the spin-down time longer, various spin-down techniques have been developed for a few decades in different ways. In the existing spin-down techniques, a static or dynamic time-out value is used to determine when to stop spinning [5]-[7]. If the idle time, during which there is no request to a disk, is longer than the time-out value, the disk is spun down. Many studies in the early stage have focused on predicting idle periods by dynamically adjusting the time-out value according to workload changes. However, the existing disk-based spin-down techniques bring limited energy saving under frequent I/O requests.

Recently, hybrid storage employing a flash memory-based non-volatile cache (called *flash cache*) has been adopted in various mobile PCs, such as laptop/notebook computers, for better performance and energy saving [8]-[11]. One of the main purposes of using the flash cache is to reduce the power consumption of a hard disk by eliminating disk activities. While a disk is spun down, the flash cache can absorb I/O requests and thus extend the spin-down time. In this hybrid storage architecture, there are several major research issues for better energy efficiency.

First, it is required to cache data to be read while a disk is spun down in order to extend spin-down periods. In the hybrid storage, write requests can be simply buffered in the flash cache while a disk is spun down. However, read misses that occur in the middle of spin-down periods are the main cause of disk spin-ups. In the existing studies, it has not received much interest what to cache for reducing and delaying disk spin-ups [9], [12]. This paper investigates what data should be sorted out to keep a disk spun down longer.

Second, a new disk spin-down algorithm that considers the hybrid storage should be developed. Most of the previous studies adopted conservative spin-down algorithms, even with the flash cache, since frequent spin-ups consume much power [9]-[13]. However, during spin-down periods, read requests can also be sufficiently redirected if the flash cache can provide the high read hit ratio. In such a case, traditional spin-down algorithms cannot trigger the spin-down at an appropriate time, because they reset the idle time whenever host requests arrive [5]-[7]. For this reason, a disk should keep spinning even while the requests can be handled by the flash

[1] This research was supported by the SW Computing R&D Program of KEIT (2011-10041313, UX-oriented Mobile SW Platform) funded by the Ministry of Knowledge Economy.

Hyotaek Shim and Seungryoul Maeng are with Computer Science Department, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea (e-mail: {htshim, maeng}@calab.kaist.ac.kr).

Jin-Soo Kim is with School of Information and Communication Engineering, Sungkyunkwan University (SKKU), 300 Cheoncheon-dong, Jangan-gu, Suwon, Republic of Korea (e-mail: {jinsookim@skku.edu).

Contributed Paper
Manuscript received 07/15/12
Current version published 09/25/12
Electronic version published 09/25/12.

cache. This paper presents a new disk spin-down algorithm, which initiates a disk spin-down efficiently, for the hybrid storage.

Third, there are two hardware limitations in the flash-based hybrid storage, such as the limited erase cycle of the NAND flash-based cache and the limited load/unload cycle of the hard disk [13], [14]. Many writes into the flash cache and frequent attempts of spin-downs may shorten the expected lifetime of the flash cache and hard disk, respectively. Accordingly, to ensure the lifetime of the hybrid storage, these two restrictions should be carefully considered in designing the energy-efficient hybrid storage system.

This paper presents *Best-effort Energy Saving Techniques* (BEST) that make the best use of the hybrid storage to reduce energy consumption without invading the warranted hardware limits. The proposed techniques consist of two major policies, which are based on the prior work of the authors [15]. First, *active write caching* is proposed to reduce or to delay spin-ups caused by read misses in the flash cache. Second, *a read miss-based spin-down algorithm* is devised for making a disk spin down efficiently, considering the read hit ratio of the flash cache. The energy-efficiency of the proposed techniques is compared with that of the preceding studies [9], [12], [14] by using trace-based simulation.

The rest of this paper is organized as follows. Section II and III contain a brief survey of background and related work, respectively. Section IV describes the proposed techniques in detail. Section V explains experimental methodology and presents evaluation results that demonstrate the reduced energy consumption with the proposed techniques. Finally, Section VI concludes the paper.

II. BACKGROUND

A. NAND Flash Memory

NAND flash memory [17] logically consists of an array of erase blocks, each of which contains a fixed number of pages. The unit of read and write operations is a page, and the unit of erase is a block. Write operations can be performed only on previously-erased pages. This restriction brings *garbage collection* that copies valid pages contained in old blocks to previously-erased blocks before erasing the old blocks. In large-block NAND flash memory, there is an additional restriction where all pages within a block should be written in a sequential manner from the first to the last page. When the flash memory is used for cache storage, this restriction can be simply preserved through sequentially writing within a block in the case of cache insertion.

B. Hybrid Storage System Architecture

Fig. 1 illustrates the overall system architecture that includes the flash-based hybrid storage. For the flash cache, NAND flash memory is widely used since it provides large capacity, non-volatility, and low power consumption that is considerably smaller than that of hard disks [18]-[20]. The *hybrid storage manager* controls the hard disk and the flash cache at the block layer, and thus the existence of the flash cache is transparent to the file systems. The hybrid storage manager is responsible for making a decision about what to cache, when to spin down the disk, and who should handle read and write requests. In addition, the manager maintains data mapping for the flash cache, and also performs garbage collection.

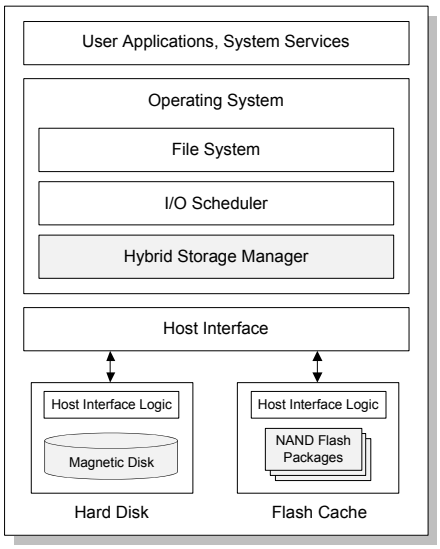

Fig. 1. Overall system architecture based on the hybrid storage

C. Flash Cache-based Disk Spin-Down

In the previous studies, it has been demonstrated that the flash cache is effective in reducing power consumption by eliminating disk activities and prolonging spin-down time [8]-[16], [19], [20]. Typically, while a disk is active, the flash cache stores some of read requests that are likely to be read again in case of following spin-down periods. While a disk is spun down, the flash cache is used for write buffering.

On a read request, if the data exist in the flash cache, the request is serviced by the flash cache. The missed data should be handled by the disk. If the read miss occurs during spin-down periods, the disk must be spun up. On a write request during spin-down periods, it can be simply written in the flash cache; due to the non-volatility of the flash memory, the flash cache can also ensure the reliability of buffered data, even under unexpected power failures. The disk must be spun up only when there is no more space for write buffering. If the spin-down time is shorter than the *spin-down cost* [4], [5], the disk rather consumes more power than keeping it spinning. The spin-down cost means the amount of time it should be spinning to consume as much as the power consumed by a spin-up.

III. RELATED WORK

There have been several studies for energy saving in a hard disk by using the flash cache. *NVCache* [12] strictly partitioned the flash cache into *read* and *write caches*, and investigated some read caching policies, such as LRU and

LFU, and their combination. The write cache accommodates write requests during spin-down periods. *SmartSaver* [9] divides the flash cache into three areas for read caching, read prefetching, and write buffering. The size of three areas is dynamically adjusted to optimize the overall energy saving efficiency.

In particular, read prefetching can be effective in reducing power consumption in certain activities such as video playback. An energy-efficient data prefetching scheme was proposed for portable media players [1]. This study showed that flash-based prefetching can provide longer spin-down time than the existing double buffering during continuous video playback.

A hybrid disk-aware spin-down algorithm [14] was proposed to further utilize the flash cache by computing the recent idle time, not from the last read/write request, but from the last read request. This is because write requests are not an obstacle to make a disk spin down any more due to the flash cache. However, such frequent disk spin-downs can shorten the lifetime of the flash memory and hard disk due to their limited erase and load/unload cycles. Restricting the spin-down rate was proposed to ensure that the number of spin-downs over the expected lifetime does not exceed the warranted limit of a hard disk [13].

IV. Best-effort Energy Saving Techniques

A. Active Write Caching

Fig. 2 depicts disk accesses for 12 hours while there is no user activity on a commercial desktop system. As shown in Fig. 2(a), there are few read requests without user running applications. In this case, the spin-down performance can be guaranteed by only write buffering with the flash cache. As a more practical example, Fig. 2(b) depicts disk accesses while four kinds of widely-used applications are running, such as a web browser, e-mail client, messenger, and anti-virus program. Unlike the above case, it was observed that read requests were frequently generated by the background operations of those applications, even without user controls. In this case, unless the flash cache accommodates read requests during spin-down periods, frequent spin-ups are inevitable and consume much power. Accordingly, it becomes more important to store the flash cache with the data that are likely to be read in the following spin-down periods to reduce or to delay the spin-ups.

As explained in Section II-C, most of the previous studies are based on read caching during active periods and write buffering during spin-down periods [9], [12], [14], [16]. In this policy, during active periods, only read requests are cached, while write requests are directly written into a disk, invalidating cached data if overlapped. In this paper, read and write requests during active periods are called *active read* and *active writes*, respectively. In real workloads, however, it is observed that there are many *write after read* patterns on the same sectors, and many data written by active writes are read again in the following spin-down periods. As a result, the obsolete data by active writes in the flash cache inevitably become the cause of disk spin-ups.

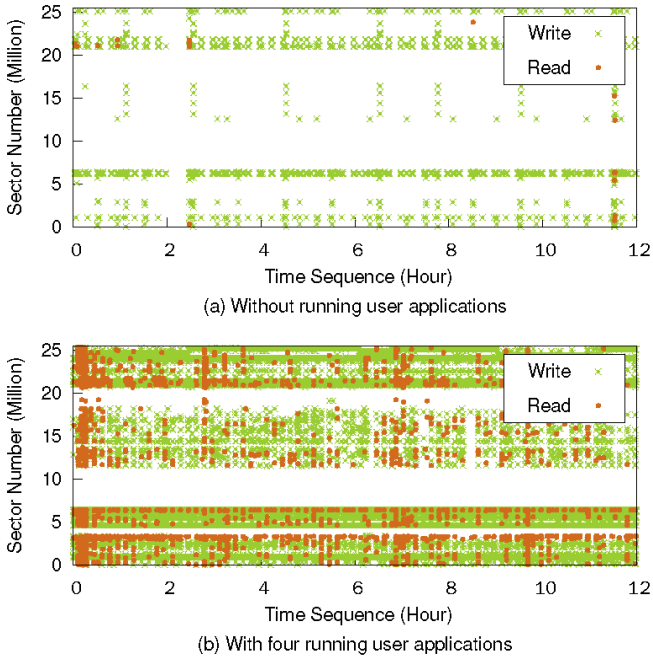

Fig. 2. Disk accesses while there is no user activity for 12 hours

TABLE I
Summary of Block-Level Trances used for Evaluation

Name	Duration (Day)	Read/Write Ratio (%)	Working Set (GB)
Search	7	23.7 / 76.3	8.7
Game	7	43.7 / 56.3	13.9
Document	7	23.7 / 76.3	10.6
Web	7	27.6 / 72.4	15.4

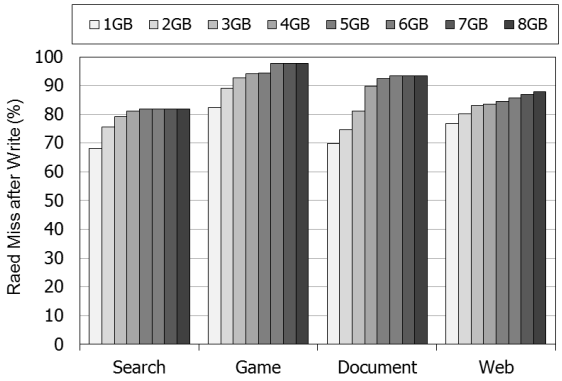

Fig. 3. Ratio of *read miss after write* to all read misses during spin-down periods

How many spin-ups are caused by the discarded active writes was investigated through trace-driven simulation based on the existing studies [9], [12], [14] with four block-level traces in Table I. The detailed implementation and simulation configurations are explained in Section V-A. Whenever a read miss occurs in the middle of spin-down periods, the number of read misses was accumulated. A read miss is classified into two types, such as *read miss after read* and *read miss after write* on the missed data. Read miss after read means that the

missed data were read-requested just before the read miss, but not cached. Read miss after write means that some or all of the missed data were directly written to the disk or invalidated in the flash cache. Fig. 3 shows the ratio of read miss after write to all read misses during spin-down periods according to the size of the flash cache for each trace. From the results of the Game and Document traces when the size of flash cache is over 4GB, the ratio is over 90%. As the flash cache size increases, the ratio of read miss after write more increases. These results demonstrate that a large portion of read misses in the middle of spin-down time were caused by the data that were written directly to the disk or cached-and-invalidated due to overwriting. Therefore, it is required to cache active writes into the flash cache to extend disk spin-down time, which is *Active Write Caching* (AWC).

B. Flash Cache Management

The flash cache is divided into *clean* and *dirty* caches that contain active reads and writes, and write-buffered data, respectively. The space of the flash cache is shared by those two caches, but erase blocks are separately used for better hot/cold separation [19]. The clean cache adopts the LFU policy in order to reduce the amount of insertion into the flash cache [12].

The missed data of active reads and the whole data of active writes are insertion candidates for the clean cache. When a read miss occurs or when an active write arrives, the frequency of the request data to be inserted is compared with the average frequency of victim blocks that contain clean data. If the frequency of the LFU block is lower than that of the request, the block is erased and is written by the request. Otherwise, the request is discarded. In the case of caching an active write, the cached data are written to the disk in a *write-through* manner, and the old data in the flash cache are invalidated if exist.

During spin-down periods, write requests are written in the dirty cache as evicting the clean cache. After a spin-up, some of dirty data in the dirty cache are flushed to the disk to make space for next write buffering, and the data are incorporated in the clean cache. To prevent the dirty cache from occupying the capacity, the maximum size of the dirty cache is limited to 50% of the total capacity. When there is no available space in the flash cache, if there are erase blocks, of which more than 50% of the pages are invalidated, garbage collection is performed on those blocks instead of evicting in order to utilize the flash cache capacity.

C. Read Miss-based Spin-Down Algorithm

In addition to increasing the read hit ratio during spin-down periods, making a disk spin down at an appropriate time is required for more energy saving. In traditional spin-down algorithms for conventional hard disks [3]-[5], the idle time is computed as *current time – last access time,* regardless of the type of a request that arrives, because any request is likely to cause a spin-up in conventional hard disks. In the hybrid storage, write requests can be simply handled by write buffering without initiating spin-ups. For this reason, a *Hybrid Disk-aware spin-down Algorithm* (HDA) [14] was proposed,

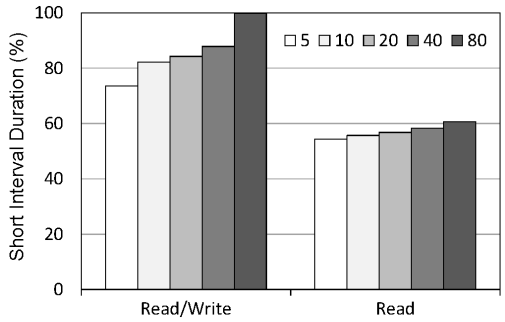

Fig. 4. Ratio of the accumulated *short-interval duration* to the overall duration according to the time-out value with the *Search* trace

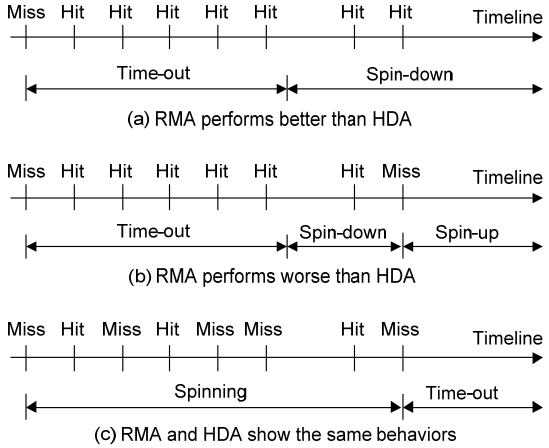

Fig. 5. Disk behaviors based on RMA under frequent read requests

where the idle time is computed as *current time – last read time* and reset only on a read request. While few read requests and frequent write requests arrive as shown in Fig. 2(a), HDA can perform better than traditional algorithms. However, if read requests frequently arrive as shown in Fig. 2(b), it can be difficult even to initiate the spin-down since the idle time is reset on each read request.

The duration while the interval time between requests is shorter than the time-out value is defined as the *short-interval duration*. The short-interval duration was measured with the Search trace in Table I. Fig. 4 presents the ratio of the short-interval duration to the total duration according to the time-out value. The bars on the left present the accumulated short-interval time between read/write requests, and the results on the right were measured between read requests. Although the ratio of the short-interval duration is reduced with HDA, the ratio is still more than 50% in all the results, even with the small time-out value. If read requests can be sufficiently handled in the flash cache for the short-interval duration, such a way of computing the idle time is inefficient in prolonging spin-down time and utilizing the flash cache.

In this paper, a *Read Miss-based spin-down Algorithm* (RMA) is proposed to make a disk spin down, even under frequent read requests, considering the read hit ratio. In RMA, the idle time is computed as *current time – last read miss time*. This algorithm makes a disk spin down earlier than HDA, and its profit and loss are determined by whether the following

read requests are hits or not within the spin-down cost. For example, Fig. 5(a) illustrates the case that RMA performs better than HDA under frequent read requests. When using RMA, a disk stops spinning after the time-out from the first read miss, and the spin-down is continuously maintained since the following read requests are all hits. Therefore, the early-initiated spin-down period is truly profitable. In contrast, as shown in Fig. 5(b), since a read request in the middle of the spin-down is a miss, a disk must be spun up shortly, thereby consuming more power than keeping the disk spinning. (In these examples of Fig. 5, it is assumed that the spin-down cost is the same as the time-out value.) If there are many spin-ups caused by this case, it would be better not to stop spinning with respect to the spin-down cost. However, if the read requests can be sufficiently handled by the flash cache with the assistance of the proposed AWC, there is still room for enhancing the performance of the spin-down algorithm.

In addition, although there may be energy loss caused by read misses with RMA, numerous read misses, which occur before the time-out due to the low hit ratio, reset the idle time and rather prevent a disk from stopping inadequately as shown in Fig. 5(c). In other words, the proposed spin-down algorithm can avoid inadequate spin-downs under workloads that exhibit quite low temporal locality. In Section V-B, the profit and loss of using RMA are investigated in detail.

D. Ensuring Expected Lifetime

In the hybrid storage, there are two hardware limitations that should be carefully considered in the proposed energy saving techniques. First, hard disk manufactures warrant only a limited number of spin-down cycles for a hard disk, e.g., 600,000 for 2.5-inch hard disks [21]. If a disk is spun down and up many times, its warranted spin-down cycle can be exhausted in a few months. Accordingly, the proposed techniques restrict the spin-down rate (the number of spin-downs per second) according to the expected lifetime. For example, if the expected lifetime is 5 years, the maximum spin-down rate becomes 600,000 / 5 years. After the time-out, a disk is spun down unless the current spin-down rate is over the maximum.

Seconds, a limited number of erase operations can be performed in NAND flash memory. For each erase block, the program/erase cycle is strictly limited from 3,000 to 100,000 [17]. If the erase cycle of a block goes beyond this limit, the block is likely to be error-prone and is not recommended to be used anymore. Accordingly, the lifetime of the flash cache varies according to the amount of data insertion. In the proposed techniques, since the amount of insertion into the flash cache is increased more by using AWC and RMA, the lifetime of the flash cache is likely to be diminished. Considering this problem, restricting the number of write and erase operations is strongly required to extend the lifetime of the flash cache.

In a similar way with restricting the spin-down rate, the expected lifetime of the flash cache is guaranteed by keeping the erase rate lower than the maximum erase rate (erase/sec), which is computed by the predefined expected lifetime, e.g., 5 years. When the erase rate exceeds the maximum rate, the proposed techniques stop caching active reads and writes, and also stop permitting a disk to spin down in order to prevent write buffering.

TABLE II
SPECIFICATIONS OF A HARD DISK AND FLASH STORAGE

	2.5-inch Hard Drive	NAND Flash Storage
Read/Write	2.3W	0.17 / 0.21 W
Seek	2.6W	-
Idle	2W	3.3 mW
Standby	0.25W	3.3 mW
Spin-up (Time)	5.5W (3 Sec)	-
Load/unload	600,000	-
Erase per block	-	100,000

V. EVALUATION

A. Simulation Environment

To evaluate the energy efficiency of the proposed techniques, a trace-based simulator was implemented, based on the specifications of a 2.5-inch hard disk [21] and flash storage [17], [22] as summarized in Table II. In power modeling of a hard disk, the power consumption for data transfer is computed with the maximum bandwidth. If read and write requests are non-sequential with the previous request, the rotational and seek power are computed by the average rotational delay and seek time, respectively.

The frequency history of the LFU policy is collected from all previous read requests. Assigning a frequency count to each request is likely to be incorrect since the sector address and length of each request are variable. Alternatively, the read frequency is assigned to each sector group that consists of 8 sectors, and the history is maintained in the form of a hash table for fast access in the main memory. The frequency of cached data is halved every 10,000 requests in order to prevent aged data from occupying the flash cache capacity.

The time-out value is computed by a dynamic disk spin-down technique [4], [5]. The algorithm is based on a machine learning technique that provides excellent performance in practice. In this algorithm, N experts are defined for computing their energy benefit according to different time-out values. (N was configured to 100.) Each expert has its own weight and fixed time-out value evenly spaced between 0 and the spin-down cost, 8.25, which was computed by the specification in Table II. On each idle time, all experts decide whether the disk should be spun down with their own fixed time-out value. In the hybrid storage, the idle time is computed at the end of every read miss. According to the energy benefits of each expert, the expert weight is updated. The algorithm uses the weighted average of the fixed time-out values of all the experts as the time-out value.

As shown in Table I, the traces used for evaluation were collected from commercial desktop operating systems by using filter drivers. The *Search* trace is workloads used mostly for desktop search, audio applications, and file downloads.

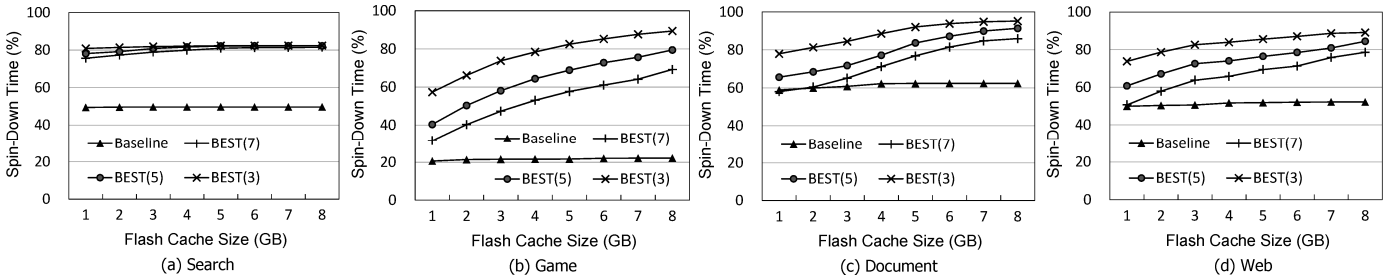

Fig. 6. Ratio of the accumulated spin-down time to the overall duration

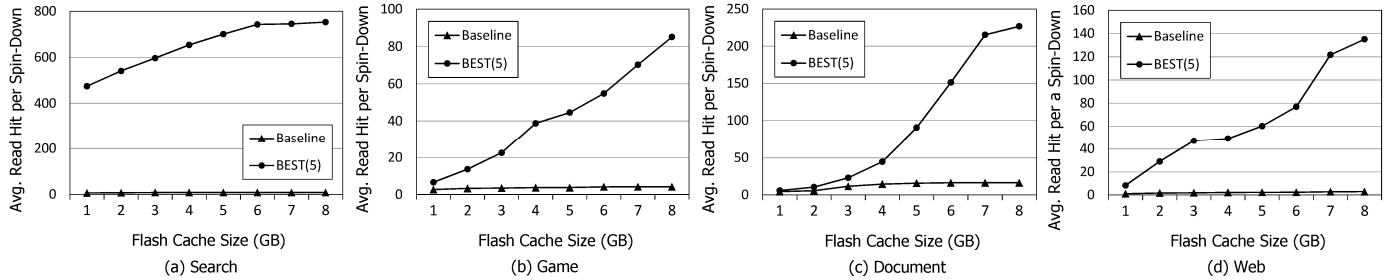

Fig. 7. Average number of read hits during the spin-down periods

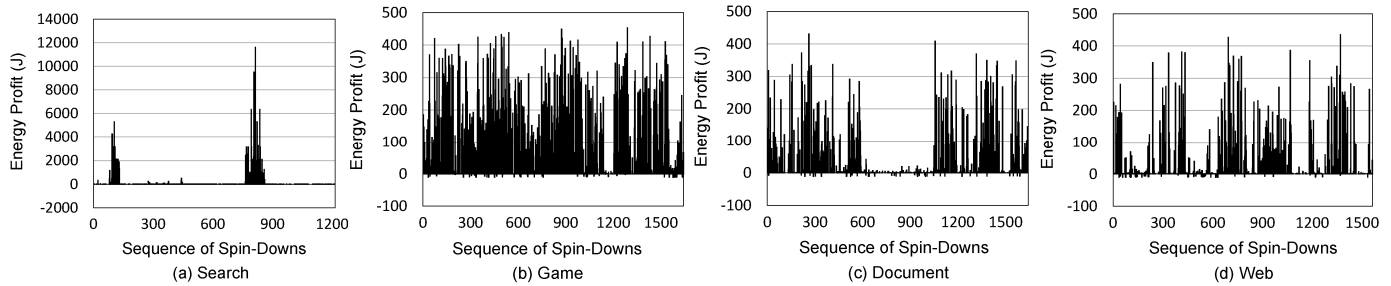

Fig. 8. Energy profit and loss with the read miss-based spin-down algorithm

The *Game* trace consists of workloads, such as online games, e-mail clients, and web browsing. The *Document* trace was produced by document works. The *Web* trace was collected by web-based applications. All of these traces are 7 days long, and the simulation runs for aging during the first two days and runs for evaluation during the remaining five days.

The energy consumption of the proposed techniques was compared with the *Baseline* configuration where HDA was used without AWC [9], [12], [14]. The proposed techniques, which are referred to *BEST(n)*, have three configurations according to the expected lifetime (n years), such as 3, 5, and 7 years. In Baseline, the expected lifetime was configured to 5 years.

B. Simulation Results

Fig. 6 shows the ratio of the accumulated spin-down time relative to the total duration according to the flash cache size from 1 to 8GB. In all the results of BEST, the spin-down time lengthens as the cache size and the expected lifetime increase. There are two main reasons. First, the read hit ratio is significantly increased due to the increased cache capacity. Fig. 7 shows the average number of read hits in the flash cache for each spin-down period with Baseline and BEST(5). When AWC is used, the average number of read hits is significantly increased. Second, more spin-downs are allowed due to the increased maximum spin-down rate. In addition, when RMA is applied, the spin-down time is further extended by the early-initiated spin-downs.

In particular, the significant improvement of the result with the Search trace is caused by two interesting characteristics. First, the trace has not only many *write after read* request patterns, but also many *read after write* request patterns on the same sectors, compared with the other traces. Therefore, without AWC, the newly-cached read requests are likely to be invalidated by the following write requests, so that many read misses occur on the invalidated data. When AWC is used, the read hit ratio is significantly increased, e.g., from 16.4% to 90.4% with the 4GB flash cache. Second, since the ratio of the short-interval duration is high as shown in Fig. 4, using RMA can be more effective under such a high read hit ratio in this trace. Consequently, in the result of the Search trace, the energy consumption is reduced much with the proposed techniques.

In contrast, Baseline provides similar energy saving regardless of the flash cache size due to the low read hit ratio caused by discarded active writes. The Baseline policy caches only active reads during active periods, and thus the flash cache is mostly filled with active reads. The active writes also

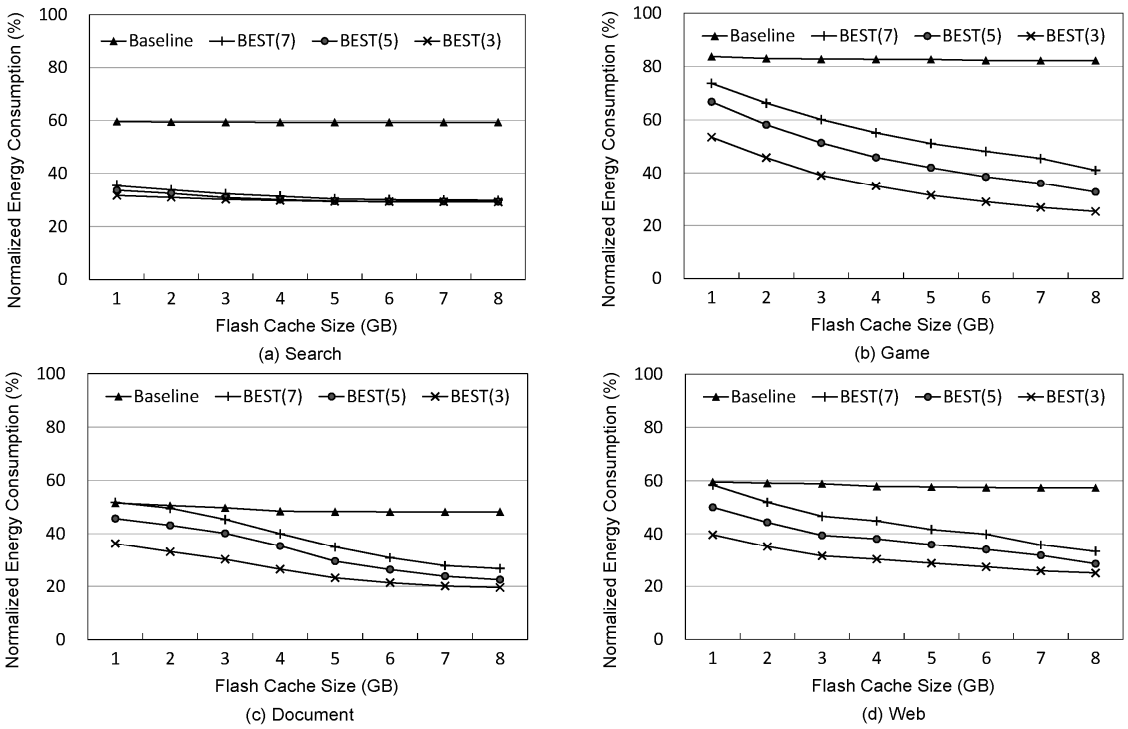

Fig. 9. Total energy consumption normalized to the energy consumed by a hard disk always spinning

invalidate the cached data when overlapped, and the invalidated or discarded data become the main cause of spin-ups as previously explained in Section IV-A.

To compare RMA with HDA with respect to the energy benefit, the amount of power consumption was measured for each spin-down. Fig. 8 illustrates the profit or loss of using RMA for each spin-down. Basically, the simulation was performed under the BEST(5) configuration, and the energy consumption was computed at each end of spin-ups according to the spin-down algorithms: RMA and HDA. The profit is computed as *the energy consumption of using HDA – that of using RMA*. The negative value means the energy loss. In most of the results, using RMA was beneficial in reducing power consumption. Although the loss occurs several times in the results of the Game, Document, and Web traces, the degradation is small enough to be offset by the profit.

Fig. 9 shows the total energy consumption normalized to the energy consumed by a disk always spinning without the flash cache. The average amount of reduced energy consumption was 38.8% of the energy consumed by the Baseline configuration with the 4GB flash cache. In particular, in the Game trace, the amount of the reduced power consumption enlarges without saturation as the flash cache size increases, and it is expected that more energy saving is achievable with the larger size of the flash cache. From these results, it can be demonstrated that the proposed techniques are quite efficient in reducing energy consumption in hybrid storage systems through making the most use of the flash cache within the warranted hardware limitations.

VI. CONCLUSION

In mobile PCs, such as laptop/notebook computers, it has been considered as essential to extend the operating time under limited battery capacity. Recently, in such consumer devices, hybrid storage employing the flash cache has been adopted to further extend disk spin-down time. However, two major problems were observed in the existing studies for hybrid storage under the background operations of running user applications. First, many disk spin-ups are caused by read misses on the invalidated or discarded data of *active writes*. Second, the idle time is reset before the time-out by frequent read requests, so that a disk cannot be spun down at an appropriate time, even with the high read hit ratio.

In this paper, two novel best-effort energy saving techniques were proposed to handle those problems. First, the *active write caching* technique was presented to reduce or to delay spin-ups caused by read misses that occur in the middle of spin-downs. Second, *a read miss-based spin-down algorithm* that computes the idle time from the last read miss time was proposed to make a disk spin down efficiently, even under frequent read requests, considering the read hit ratio. In those techniques, although more insertion into the flash cache and more spin-downs can occur, the expected lifetime of hybrid storage is guaranteed by restricting the erase and spin-down rate within the hardware limitations that the manufactures warrant. In this way, the proposed techniques make the best use of the hybrid storage within the given restriction in order to achieve lower energy consumption. In the evaluation results of trace-based simulation, when the proposed techniques were applied, the average amount of reduced energy consumption was 38.8% of that of the previous studies with the 4GB flash cache.

REFERENCES

[1] J. Kim, A. Yang, and M. Song, "Exploiting flash memory for reducing disk power consumption in portable media players," *IEEE Trans. on Consumer Electronics*, vol. 55, no. 4, pp. 1997-2004. Nov. 2009.

[2] Y.-J. Kim, S.-J. Lee, K. Zhang, and J. Kim, "I/O performance optimization techniques for hybrid disk-based mobile consumer devices," *IEEE Trans. on Consumer Electronics*, vol. 53, no. 4, pp. 1469-1476. Nov. 2007.

[3] F. Douglis, P. Krishnan, and B. Marsh, "Thwarting the power-hungry disk," *Proc. of the USENIX Winter Technical Conference*, pp. 293-306, San Francisco, CA, USA, Jan. 1994.

[4] P. M. Greenawalt, "Modeling power management for hard disks," *Proc. of the 2nd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (IEEE MASCOTS)*, pp. 62-66, Durham, NC, USA, Jan. 1994.

[5] P. Krishnan, P. M. Long, and J. S. Vitter, "Adaptive disk spindown via optimal rent-to-buy in probabilistic environments," *Proc. of the 12th International Conference on Machine Learning (ICML)*, pp. 322–330, Tahoe City, CA, USA, Jul. 1995.

[6] D. P. Helmbold, D. D. E. Long, and B. Sherrod, "A dynamic disk spin-down technique for mobile computing," *Proc. of the 2th International Conference on Mobile Computing and Networking (ACM MobiCom)*, pp. 130-142, Rye, NY, USA, Nov. 1996.

[7] D. P. Helmbold, D. D. E. Long, and T. L. Sconyers, and B. Sherrod, "Adaptive disk spin-down for mobile computers," *Mobile Networks and Applications*, vol. 5, no. 4, pp. 285-297, 2000.

[8] J. Matthews, S. Trika, D. Hensgen, R. Coulson, and K. Grimsrud, "Intel® Turbo Memory: nonvolatile disk caches in the storage hierarchy of mainstream computer systems," *ACM Trans. on Storage (ACM TOS)*, vol. 4, no. 2, article 4, May 2008.

[9] F. Chen, S. Jiang, and X. Zhang, "SmartSaver: Turning flash drive into a disk energy saver for mobile computers," *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 412-417, Tegernsee, Germany, Oct. 2006.

[10] Y.-J. Kim, K.-T. Kwon, and J. Kim, "Energy-efficient file placement techniques for heterogeneous mobile storage systems," *Proc. of the 6th International Conference on Embedded Software (EMSOFT)*, pp 171-177, Seoul, Republic of Korea, Oct. 2006.

[11] S. Liu, X. Cheng, X. Guan, and D. Tong, "Energy efficient management scheme for heterogeneous secondary storage system in mobile computers," *Proc. of the 25th Symposium on Applied Computing (ACM SAC)*, pp. 251-257, Sierre, Switzerland, Mar. 2010.

[12] T. Bisson, S. A. Brandt, and D. D. E. Long, "NVCache: Increasing the effectiveness of disk spin-down algorithms with caching," *Proc. of the 14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (IEEE MASCOTS)*, pp. 422-432, Monterey, CA, USA, Sep. 2006.

[13] W. Felter, A. Hylick, and J. Carter, "Reliability-aware energy management for hybrid storage systems," *Proc. of the 27th Symposium on Massive Storage Systems and Technologies (IEEE MSST)*, pp. 1-13, Denver, CO, USA, May 2011.

[14] T. Bisson, S. A. Brandt, and D. D. E. Long, "A hybrid disk-aware spin-down algorithm with I/O subsystem support," *Proc. of the International Performance, Computing, and Communications Conference (IEEE IPCCC)*, pp. 236-245, New Orleans, LA, USA, Apr. 2007.

[15] H. Shim, J. Kim, D. Jung, J.-S. Kim, and S. Maeng, "RMA: A read miss-based spin-down algorithm using an NV cache," *Proc. of the 26th IEEE International Conference on Computer Design (IEEE ICCD)*, pp. 520-525, Lake Tahoe, CA, USA, Oct. 2008.

[16] T. Bisson and S. A. Brandt, "Flushing polices for NVCache enabled hard disks," *Proc. of the 24th Conference on Mass Storage Systems and Technologies (IEEE MSST)*, pp. 299-304, San Diego, CA, USA, Sep. 2007.

[17] "1G x 8Bit / 2G x 8Bit / 4G x 8Bit NAND Flash Memory (K9XXG08UXM)," *Samsung Electronics*, 2005.

[18] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," *Proc. of the USENIX Annual Technical Conference*, pp. 57-70, Boston, MA, USA, Jun. 2008.

[19] T. Kgil, D. Roberts, and T. Mudge, "Improving NAND flash based disk caches," *Proc. of the 35th International Symposium on Computer Architecture (ISCA)*, pp. 327-338, Beijing, China, Jun. 2008.

[20] J. Ren and Q. Yang, "I-CASH: Intelligently Coupled Array of SSD and HDD," *Proc. of the 17th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 278-289, San Antonio, TX, USA, Feb. 2011.

[21] "Hitachi Travelstar 7K200 2.5 inch SATA hard disk drive revision 1.0," *Hitachi Global Strorage Technologies*, May. 2007.

[22] "Sandisk CompactFlash memory card OEM product manual version 12.0," *Sandisk Corporation*, Feb. 2007.

BIOGRAPHIES

Hyotaek Shim received his B.S. degree in Computer Engineering from Inha University, Republic of Korea, in 2006. He has been enrolled in the M.S.-Ph.D. joint degree program in Computer Science at Korea Advanced Institute of Science and Technology (KAIST) since 2006. His research areas include flash memory-based storage systems, embedded systems, and operating systems.

Jin-Soo Kim received his B.S., M.S., and Ph.D. degrees in Computer Engineering from Seoul National University, Republic of Korea, in 1991, 1993, and 1999, respectively. He is currently an associate professor at Sungkyunkwan University. Before joining Sungkyunkwan University, he was an associate professor at Korea Advanced Institute of Science and Technology (KAIST) from 2002 to 2008. He was also with the Electronics and Telecommunications Research Institute (ETRI) from 1999 to 2002 as a senior member of the research staff, and with the IBM T. J. Watson Research Center as an academic visitor from 1998 to 1999. His research interests include embedded systems, storage systems, and operating systems.

Seungryoul Maeng received his B.S. degree in Electronics Engineering from Seoul National University (SNU), Republic of Korea, in 1977, and M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), in 1979 and 1984, respectively. Since 1984, he has been a faculty member of Computer Science Department at KAIST. From 1988 to 1989, he was with the University of Pennsylvania as a visiting scholar. His research interests include micro-architecture, parallel processing, cluster computing, and embedded systems.