RMA: A Read Miss-Based Spin-Down Algorithm using an NV Cache

Hyotaek Shim, Jaegeuk Kim, Dawoon Jung, Jin-Soo Kim, and Seungryoul Maeng
Computer Science Department at KAIST
{htsim, jgkim, dwjung}@camars.kaist.ac.kr {jinsoo, maeng}@cs.kaist.ac.kr

***Abstract*—It is an important issue to reduce the power consumption of a hard disk that takes a large amount of computer system's power. As a new trend, an NV cache is used to make a disk spin down longer by servicing read/write requests instead of the disk. During the spin-down periods, write requests can be simply handled by write buffering, but read requests are still the main cause of initiating spin-ups because of a low hit ratio in the NV cache. Even when there is no user activity, read requests can be frequently generated by running applications and system services, hindering the spin-down. In this paper, we propose new NV cache policies: *Active Write Caching* to reduce or to delay spin-ups caused by read misses during spin-down periods and *a Read Miss-Based Spin-Down Algorithm* to extend the spin-down periods, exploiting the NV cache effectively. Our policies reduce the power consumption of a hard disk by up to 50.1% with a 512MB NV cache, compared with preceding approaches.**

I. INTRODUCTION

Green computing has surfaced recently as a new paradigm for the study of using computing resources efficiently. The low power consumption of computer systems has been a key issue on green computing. Especially, in mobile computer systems, it has emerged more seriously as an essential part to extend working time under the limited lifetime of a battery. Among various devices in a computer system, hard disks are the main source of power dissipation. Previous researches have shown that a hard disk can occupy 30% or more of the power consumed by typical computer systems [1], [2].

To reduce the power consumption, a disk can be spun down when it is not in use. However, it is well-known that spinning a disk up consumes much power. If the spin-down time is shorter than *spin-down cost* [3], a disk 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. Therefore, it is a challenging problem to determine when a disk should be spun down. In the existing spin-down techniques [3], [4], static or dynamic time-out value is used to determine when to spin it down. When an idle time where there is no request to a disk is longer than the time-out value, a disk is spun down.

Recently, a new technology employing a non-volatile storage cache called NV cache in computer systems has been introduced [5], [6]. One of the main purposes of the NV cache is to reduce the power consumption of a hard disk by eliminating disk activity, ensuring reliability after an unexpected power failure. In several previous researches [7], [8], [9], [10], [11], [12], [13] it has been demonstrated that the NV cache is effective in reducing power consumption by eliminating disk activity.

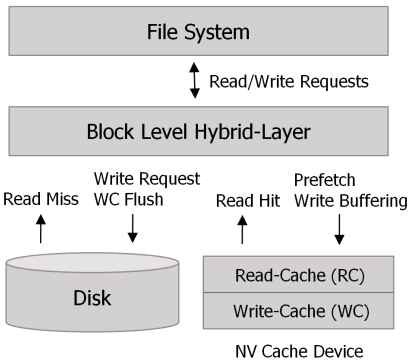


Fig. 1. The architecture of a hard disk with an NV cache

The NV cache is located at the block level as shown in Fig. 1, and so existing file systems can be compatibly used [7]. The NV cache consists of a read-cache and a write-cache. The read/write requests from the file system pass through the block level hybrid-layer, which manages where the requests are serviced, what requests are prefetched into the NV cache, and what cached data are flushed to the disk.

On a read request, if the requested data is in the NV cache, the read request is serviced by the NV cache. Otherwise, the request is serviced 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 redirected to the NV cache. The disk must be spun up only when there is no space for write buffering.

In this paper, our approach is predicated on the observation that read requests can be frequently generated by running applications without user activity. In this case, it is difficult to maintain and to initiate the spin-down. To address these problems, we present two new techniques. First, we propose *Active Write Caching* (AWC) to reduce or to delay spin-ups caused by read misses through investigating read requests that become the cause of the spin-ups. Second, we propose *a Read Miss-based spin-down Algorithm* (RMA) that makes the disk spin down even when read requests frequently arrive, which is supported by AWC. The proposed techniques are compared with the preceding studies [7], [8] by using trace-based simulation.

The rest of this paper proceeds as follows. Section II contains a brief survey of related work. Section III and IV describe the two main techniques in detail. Section V explains our experimental methodology. Section VI presents on simulation results that demonstrate the improvement caused

978-1-4244-2658-4/08/$25.00 ©2008 IEEE

by the proposed techniques, and Section VII concludes the paper.

II. Related Work

In NVCache [7], NV cache management policies were proposed to extend spin-down periods. In the policies, the read-cache is filled with read requests while a disk is active. They investigated some read-cache policies, such as Least Recently Used (LRU), Least Frequently Used (LFU), and their combination. The write-cache accommodates write requests during spin-down periods. The data in the write-cache are ordered by insertion time and are flushed in a FIFO manner.

Flushing Policies for an NV cache, which decrease the overhead of data synchronization, were proposed in [9]. In these policies, all the data of a write-cache are flushed to a disk only when the disk is spun up by write-cache fullness because flushing performance can be more improved by sorting and merging techniques. The sorting technique makes an array of requests, and the merging technique converts multiple requests into a single large request, to reduce disk access time.

III. Active Write Caching

A. Background and Motivation

Fig. 2 shows disk accesses for 12 hours without user activity on a Windows XP system. As shown in Fig. 2(a), there are few read requests during the period without running application. In this case, the spin-down performance can be guaranteed by write buffering with an NV cache. Fig. 2(b) illustrates the disk activity with four running user applications including web browser, e-mail client, messenger, and anti-virus program. We observed that read requests are frequently generated by the user applications. Unless the NV cache accommodates the read requests during spin-down periods, spin-ups are inevitable because most spin-ups are generated not by the write-cache fullness but by read misses [8], [9]. Accordingly, we need to populate the NV cache with data likely to be read during spin-down periods to reduce or to delay the spin-ups.

In the previous studies [7], [8], [9] write requests are redirected to an NV cache only while a disk is spun down. The write request during active periods is defined as *active write*. If the active writes overlap with cached data, the overlapped data are removed from the NV cache. When the removed data are read during spin-down periods, we cannot avoid a read miss and consequently a spin-up. If such a case frequently occurs, the performance of a spin-down algorithm gets worse.

We investigate the problem with four block level traces shown in Table 1. They are recorded from the desktops using Windows XP/Vista, with filter drivers. The Desk trace is collected from the desktop used mostly for desktop search, audio applications, and file downloads. The Soft trace is from the desktop used mostly for online games, electronic mail,

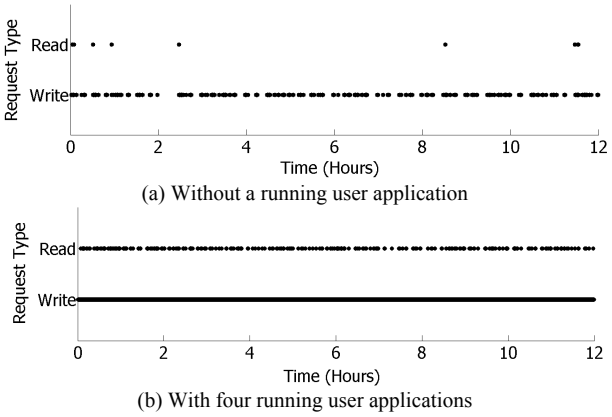


Fig. 2. The disk accesses of Windows XP system without user activity

Name	OS / File System	Duration	Read / Write (GB)
Desk	Win. XP / NTFS	7 days	12.1 / 47
Soft	Win. Vista / NTFS	7 days	53.2 / 45.5
Doc	Win. XP / NTFS	7 days	15.6 / 31.6
Web	Win. XP / NTFS	7 days	15.5 / 14.4

Table 1. The summary of the four block level traces

and web browsing. The Doc trace is from the desktop used mostly for document work. The Web trace is from the desktop used mostly for web-based applications without user activity.

We classify a read request into two types of *read after read* and *read after write* on the same sectors. Fig. 3 presents the ratio of read after write to all reads. Without storing the preceding write requests, the following read requests can bring read misses during spin-down periods. Accordingly, we investigated the spin-ups caused by the discarded active writes through the simulation based on the existing studies [7], [8], of which the implementation details will be explained in Section V. We also classify a read miss into two types of *read miss after read* and *read miss after write* on the same sectors. Whenever a read miss occurs during spin-down periods, we accumulate the number of read miss after write. As a simulation result, Fig. 4 shows the ratio of read miss after write to all read misses during spin-down periods according to the size of the NV cache. From the results of the Soft and Web traces with an NV cache of over 1GB, the ratio is over 70%. Hence, to reduce read misses caused by the data written directly to a disk, or removed from the NV cache due to overlap, we need to store the active writes into the NV cache.

B. Write-Cache Replacement Policy

In our approach, all active writes are insertion candidates for a write-cache. The write-cache is operated as an LFU write-back cache because it reduces the amount of insertion into the NV cache and performs better than FIFO or LRU [7]. In addition, the number of write requests to the disk is significantly decreased compared with write-through cache. The frequency used in the LFU policy is the read access count collected from the history of all read requests.

When an active write arrives, the overlapped data in the NV cache is first removed. Then the frequency of the active

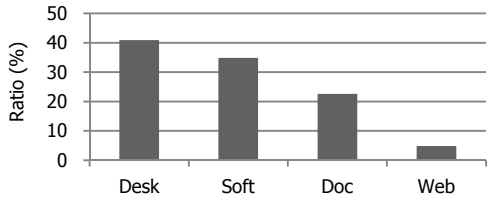


Fig. 3. The ratio of read after write to all reads

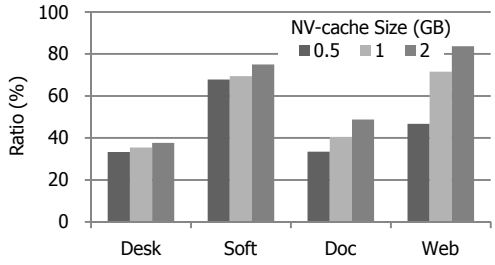


Fig. 4. The ratio of read miss after write to all read misses during spin-down periods

write is compared with the average frequency of the requests on victim blocks, which means erase blocks, in the write-cache. If its frequency is bigger than the victim, the victim blocks are flushed to the disk, and then the active write is stored into the write-cache. Otherwise, the active write is directly stored into the disk.

C. Spin-Ups caused by Write-Cache Fullness

In NVCache [7], when a disk is spun up, small empty space as much as the average amount of insertion during the recent N spin-down periods occupies the write-cache to avoid future spin-ups caused by write-cache fullness. To prevent the empty space of the write-cache from gradually getting smaller, the amount of the data flushed is increased by 25% when a spin-up is caused by write-cache fullness. Moreover, the empty space can become a bit larger during active periods because the write-cache is removed by overlap, decreasing the possibility of write-cache fullness.

However, if active writes are stored into the write-cache, its capacity always becomes full up to the limit of the empty space, thereby increasing spin-ups caused by write-cache fullness. If whole read-cache space is allowed for additional write buffering during spin-down periods, the spin-ups will be significantly decreased. But the removed data of the read-cache can be the cause of read misses during the future spin-down periods. Therefore, in our approach, a maximum of 10% of the read-cache space is allowed to reduce spin-ups caused by write-cache fullness. In addition, to alleviate the loss of the removed read-cache, the read-cache data replaced by write buffering are selected by the read-cache replacement policy.

D. Limited Lifetime of Flash Memory

As an NV cache, NAND flash memory [14] is widely used since it provides large capacity, non-volatility and low power requirement considerably smaller than hard disks. It has, however, the limited program/erase cycles from 10K to 100K according to the type of NAND flash memory, and the lifetime of flash memory varies according to the amount of data inserted into the NV cache. In our approach, as the amount of insertion into the NV cache increases by storing active writes, its lifetime is likely to get shorter than the user's expected lifetime. We need to, therefore, protect the lifetime of flash memory.

The preceding research [8] suggested one method to ensure the expected lifetime by keeping the insertion bandwidth lower than the maximum bandwidth (MB/sec), which is computed by the predefined expected lifetime, e.g., 10 years. In our approach, we use the maximum rate of an erase operation (erase/sec) because it precisely protects the expected lifetime. When the erase rate exceeds 90% of the maximum rate, we stop prefetching read/write requests. Over the maximum rate, we also stop permitting a disk to spin down.

IV. A Read Miss-Based Spin-Down Algorithm

A. Background and Motivation

In the existing spin-down algorithms for conventional disks [3], [4], an idle time is computed as *current time - last access time,* regardless of whether a request type is a read or a write, because they assume that any I/O is likely to cause a spin-up. In a hybrid-disk, through write buffering, write requests are not responsible for initiating spin-ups any more. Accordingly, in a Hybrid Disk-aware spin-down Algorithm (HDA) [8], it is proposed that the idle time should be computed as *current time – last read access time* and reset only on a read request.

With few read requests as shown in Fig. 2(a), the hybrid disk-aware spin-down algorithm can perform better than traditional algorithms. However, if read requests frequently arrive as shown in Fig. 2(b), it is difficult even to initiate the spin-down while the interval time between read requests is shorter than the time-out value, which is defined as *short interval duration*, considering that the idle time is reset at every read request.

We investigated the short interval duration with four block level traces shown in table 1. Fig. 5 represents the ratio of the short interval duration relative to total duration according to the time-out value. In all the traces, the rate is over 10% even when the time-out value is configured as small 5 seconds. When configured as 30 or 60 seconds, the ratio considerably grows up, and the problem gets worse.

B. Idle Time Computed by Read Miss

We propose a new spin-down policy, which is called RMA, to make a disk spin down even when read requests frequently arrive. In RMA, the idle time is computed as *current time - last read miss time*. This policy makes the disk spin down earlier than HDA, and its profit and loss is determined by whether the following read requests are hits or not within the spin-down cost. For example, Fig. 6(a) shows the case that RMA performs better than HDA. Using RMA, the disk is spun down after the time-out from the first read miss. In addition, the spin-down state is maintained since the

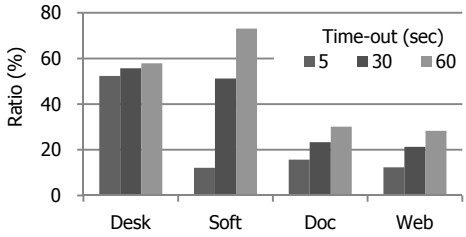


Fig. 5. The rate of short interval duration to total duration

following three read requests are all hits. Therefore, the earlier-initiated spin-down period is truly profitable. Conversely, as shown in Fig. 6(b), if the following read requests after spun down are misses, a disk must be spun up shortly, thereby consuming more power than keeping the disk spinning.

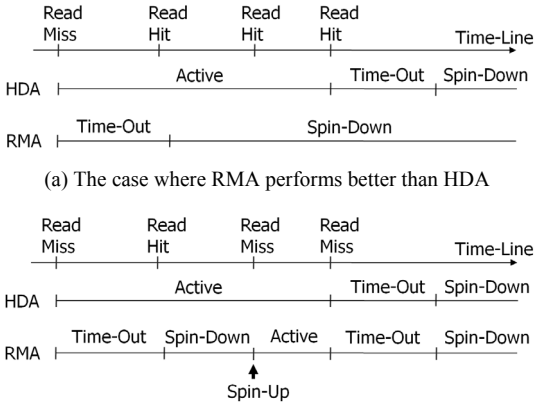
Thus, if numerous spin-ups are generated by the read misses, it is better not to spin it down with respect to the spin-down cost. But if the read requests are sufficiently satisfied by the NV cache, which is supported by AWC, we can point out that we still have opportunities to enhance the performance of the spin-down algorithm. In addition, although there may be loss caused by using RMA, the amount of the loss can be reduced gradually as the read hit ratio increases. In Section VI, the profit and the loss are investigated in detail.

V. Implementation

We implemented a trace-based simulator to measure the power consumption of a hard disk and flash memory. As summarized in table 2, Hitachi Travelstar 7K200 hard drive [15] and Compact Flash Memory Card specifications [16] were used. On a read/write request, the power consumption was calculated with the disk's maximum I/O rate, and seek power was computed by average seek time when a request is not sequential. The expected lifetime of the flash memory was configured as 10 years.

A frequency history for LFU replacement policy is collected from all read requests composed of a sector address and length. But assigning a read count to each request [7] is likely to be incorrect, considering that the sector address and length of the request is variable. Hence, in our implementation, we assign read count to each sector group, which is composed of 8 sectors, and the history information is maintained as a hash table for fast access in main memories.

The time-out value is computed by a dynamic disk spin-down technique [3]. The algorithm is based on machine learning technique that has excellent performance in practice. The algorithm consists of N experts. Each expert has its own weight and fixed time-out value evenly spaced between 0 and the disk's spin-down cost, 8.25, which is computed by the specification in Table 2. On each idle time, all experts with own fixed time-out value decide whether the disk should be spun down or not. According to the energy benefits of each trial, the expert's weight is updated. The algorithm uses the weighted average of the expert's fixed time-out value as the time-out value.



(a) The case where RMA performs better than HDA

(b) The case where RMA performs worse than HDA

Fig. 6. RMA vs. HDA

	2.5 inch Hard drive	Compact Flash
Read / Write	2.3W	0.17 / 0.21W
Seek	2.6W	-
Idle	2W	3.3mW
Standby	0.25W	3.3mW
Spin-up (Time)	5.5W (3sec)	-
Capacity	80GB – 200GB	512MB – 16GB

Table 2. The characteristics of a hard disk (Hitachi Travelstar 7K200 [15]) and an NV cache (Compact Flash Memory Card [16])

VI. Evaluation

To evaluate the performance of the proposed ideas, AWC and RMA, our techniques are compared with the combination of NVCache [7] and HDA [8]. Basically, the NV cache is equally divided for a read-cache and a write-cache, of which the replacement policy is LFU.

Fig. 7 shows the average number of read hits satisfied by the NV cache per spin-down period with the four block level traces previously mentioned, and HDA is used as a spin-down algorithm. When AWC is applied, it increases the read hits during spin-down periods, and so reduces spin-ups. In our approach, the spin-ups caused by write-cache fullness are also reduced because up to 10% of the read-cache is available for additional write buffering.

In the results of all traces as shown in Fig.7, the improvement enlarges as the NV cache size increases, and there are two main reasons. First, the amount of AWC restricted for protecting the expected lifetime is reduced as the maximum erase rate becomes larger according to the NV cache size. Second, the available space for additional write buffering becomes larger according to the increment of NV cache size, thus considerably decreasing the number of spin-ups caused by write-cache fullness.

Fig. 8 shows the ratio of the spin-down duration relative to the total duration according to the NV cache size. Using AWC, the extended spin-down periods are caused by the reduced number of read misses during the spin-down periods. If RMA is additionally applied, the spin-down periods are more extended by the earlier-initiated spin-down periods. In particular, the significant improvement of the result with the

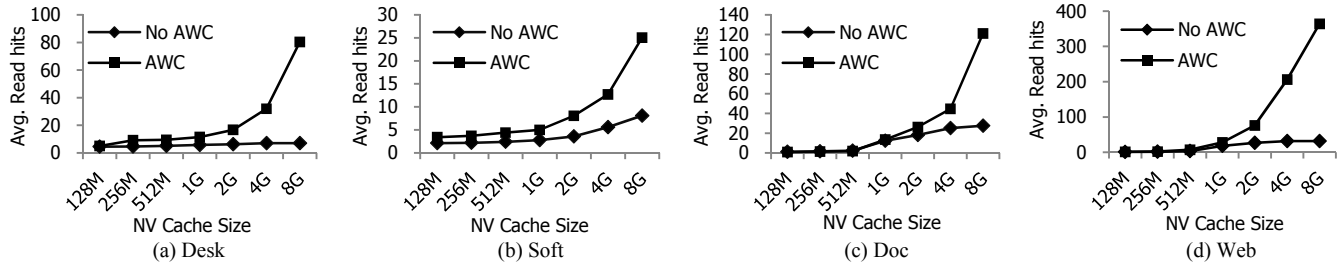


Fig. 7. The average number of read hits per spin-down period

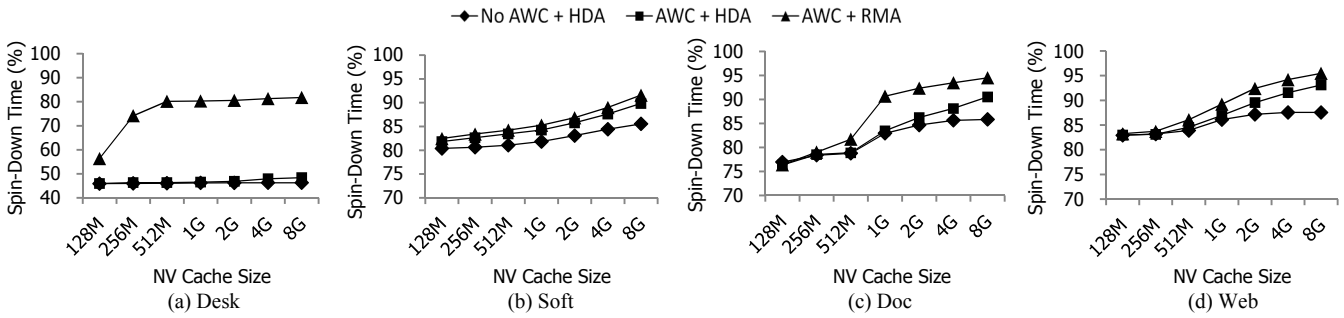


Fig. 8. The spin-down duration relative to the total duration

NVcache Size	Desk			Soft			Doc			Web		
	P-L (%)	P/L	Hit (%)	P-L (%)	P/L	Hit (%)	P-L (%)	P/L	Hit (%)	P-L (%)	P/L	Hit (%)
128MB	12.8	173.2	35.3	0.2	1.8	15.0	0.0	1.0	9.1	-0.2	0.7	34.2
256MB	31.8	267.0	67.0	0.2	1.6	18.1	0.3	1.8	14.9	-0.2	0.8	44.6
512MB	39.6	310.2	79.0	0.3	1.7	22.9	3.4	3.8	24.7	1.0	1.6	61.7
1GB	41.0	280.5	82.4	0.3	1.6	30.8	14.7	24.4	43.2	2.9	2.4	80.3
2GB	41.4	235.0	87.6	0.3	1.5	46.5	13.0	21.3	62.4	5.9	3.9	88.6
4GB	44.5	277.2	91.5	0.5	1.8	69.4	13.4	21.7	74.8	6.7	4.7	92.0
8GB	45.5	295.3	94.0	0.8	2.3	84.3	11.3	22.6	80.5	7.6	6.5	94.1

Table 3. The comparison of RMA and HDA with respect to the energy profit (P) and loss (L)

Desk trace is caused by two interesting characteristics. First, it has not only many *read after write* request patterns shown in Fig. 3 but also many *write after read* request patterns on the same sectors, compared with the other traces. Therefore, without AWC, the newly stored read requests are likely to be removed by the following write requests, thereby generating read misses on the sectors. By using AWC, the read hit ratio is significantly increased, e.g., from 9.32% to 67.51% with the only 256MB NV cache. Second, its short interval duration is longer than the other traces at a lower time-out value, as shown in Fig. 5. Hence, it has the largest improvement by using both RMA and AWC.

To compare RMA with HDA in terms of the energy benefit, we measure the decreased or increased amount of power consumption. The decreased power consumption means the profit of RMA relative to HDA. Conversely, the increased power consumption means the loss of RMA relative to HDA. Basically, the simulation is performed with RMA and AWC, and the profit/loss is computed and accumulated at every spin-up. Table 3 represents P-L, P/L, and the average ratio of a read hit. P-L means the profit minus the loss relative to the power consumed by using HDA and AWC, and P/L means the ratio of the profit to the loss. In most cases, the amount of the P-L enlarges as the read hit ratio increases. Although the profit is lower than the loss in the Web trace with a 128MB and a 256MB NV cache, its degradation is not serious.

Fig. 9 shows the power consumption relative to the power consumed by the disk always spinning. The reduced power consumption is up to 50.1% of the power consumed by the combination of NVCache [7] and HDA [8] with a 512MB NV cache using the Desk trace. The amount of the reduced power consumption enlarges as the NV cache size increases because both AWC and RMA are more effective with the larger size of the NV cache.

VII. Conclusions

Our goal was to reduce the power consumption of a hard disk by exploiting NV cache effectively. We investigated the two main problems of the existing policies when read requests are frequently generated by background system services and running applications without user activity. First, many spin-ups were caused by read misses on the data of discarded *active writes*. Second, the idle time is reset at every read request, and a disk cannot be spun down.

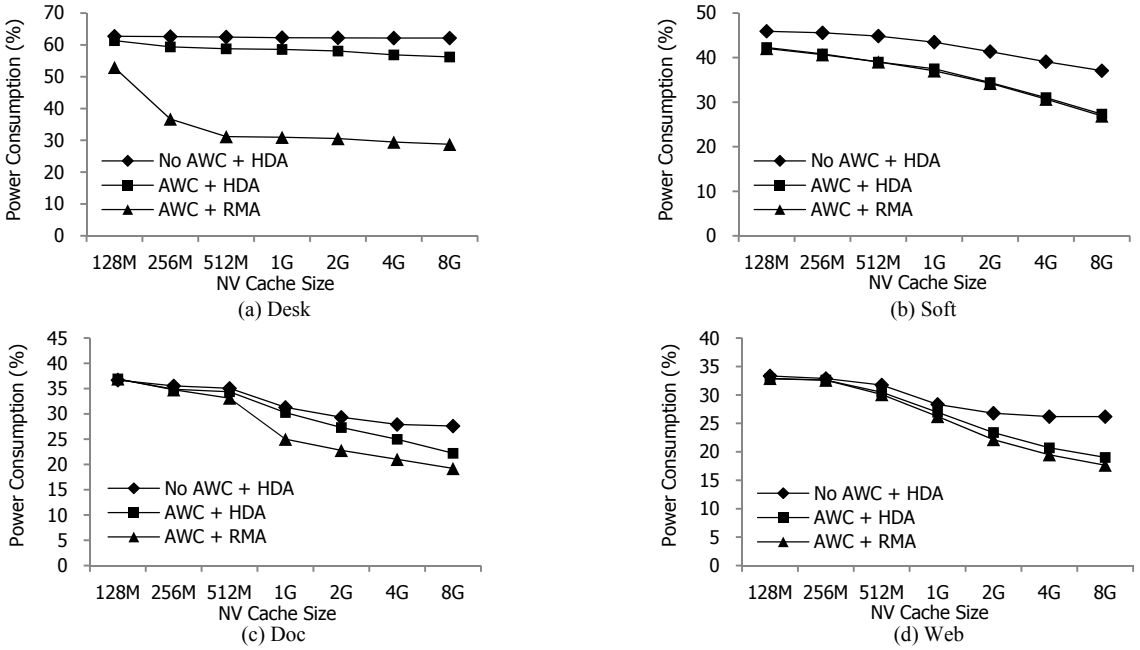


Fig. 9. Power consumption relative to the power consumed by a hard disk always spinning

In this paper, two new NV cache policies were proposed to handle the problems. First, an *Active Write Caching* technique was presented to reduce or to delay spin-ups caused by read misses. In this technique, active writes are stored into the write-cache using the LFU replacement policy, protecting the expected lifetime of the NV cache. Second, *a Read Miss-Based Spin-Down Algorithm* that computes the idle time from a read miss was proposed to make the disk spin down even when read requests frequently arrive. When both techniques were applied, the power consumption of the hard disk was reduced by up to 50.1% with 512MB NV cache, compared with the preceding approaches.

ACKNOWLEDGMENT

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute for Information Technology Advancement) (IITA-2008-C1090-0801-0045).

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. R01-2006-000-10724-0).

REFERENCES

[1] F. Douglis, P.Krishnan, and B. Marsh, "Thwarting the power-hungry disk," in Proc. of the USENIX Winter Conference, pp. 292-306, San Francisco, CA, USA, Jan. 1994.

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

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

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

[5] HM080HHI / HM12HII / HM16HJI Hybrid HDD Product Data Sheet Rev0.1, Samsung Electronics, Available: http://www.cmsproducts.com/pdf/hybrid/MH80_Hybrid_Product_Spec_Sheet.pdf, 2006.

[6] Intel® Turbo Memory article, "Overcoming disk drive access bottlenecks with Intel® Turbo Memory," Available: http://download.intel.com/design/flash/nand/turbomemory/article.pdf, 2007.

[7] T. Bisson, S. Brandt, and D. D. E. Long, "NVCache: Increasing the effectiveness of disk spin-down algorithms with caching," in 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.

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

[9] T. Bisson and S. A. Brandt, "Flushing polices for NVCache enabled hard disks," in Proc. of the 15th NASA / 24th Conference on Mass Storage Systems and Technologies (IEEE MSST), pp. 299-304, San Diego, CA, USA, Sep. 2007.

[10] Y.-J. Kim, K.-T. Kwon, and J. Kim, "Energy-Efficient File Placement Techniques for Heterogeneous Mobile Storage Systems," in Proc. of the 6th International Conference on Embedded Software (ACM EMSOFT), pp 171-177, Seoul, Korea, Oct. 2006.

[11] J. Matthews, S. Trika, D. Hensgen, and R. Coulson, "Intel® Turbo Memory: Nonvolatile Disk Caches in the Storage Hierarchy of Mainstream Computer Systems," in ACM Transactions on Storage (ACM TOS), vol. 4, no. 2, article 4, May 2008.

[12] T. Kgil, D. Roberts, and T. Mudge, "Improving NAND Flash Based Disk Caches," in Proc. of the 35th International Symposium on Computer Architecture (ISCA), pp. 327-338, Beijing, China, Jun. 2008.

[13] N. Agrawal, V. Prabhakaran, and T. Wobber, "Design Tradeoffs for SSD Performance," in Proc. of the USENIX Annual Technical Conference, pp. 57-70, Boston, MA, USA, Jun. 2008.

[14] 1G x 8Bit / 2G x 8Bit / 4G x 8Bit NAND Flash Memory (K9WAG08U1M), Samsung Electronics, 2005.

[15] Hitachi Travelstar 7K200 2.5 inch SATA hard disk drive specification, Hitachi Global Strorage Technologies, May. 2007.

[16] Sandisk CompactFlash Memory Card OEM Product Manual Version 12.0, Sandisk Corporation, Feb. 2007.