Significance of Write References on Nonvolatile Buffer Cache and its Implication on Hit Ratio and the Optimal MIN Replacement Algorithm

Sam H. Noh Inhwan Doh Jungkyu Park

School of Computer and Information Engineering
Hongik University, Seoul, Korea

Summary
Performance studies of buffer caches have generally been done with write references ignored. In this study, we show that once we consider write references in buffer caches that include nonvolatile memory, the traditional hit ratio performance measure is no longer an accurate representation, and that the disk access ratio should be used. We also show that in regards to this measure, the MIN replacement algorithm is either non-optimal or non-applicable.

Key words:
Operating Systems, Buffer Cache, Replacement Algorithm, Nonvolatile memory

1. Introduction

In considering performance of buffer cache management in traditional systems, the majority of previous works have not distinguished read and write references even though the two have distinct effects. As nonvolatile memory becomes prevalent, buffer caches that have nonvolatile features will be easily available. We present results that show that for systems where nonvolatile memory is used as the buffer cache, distinguishing read and write references can considerably influence the performance observed by the user. In this regard, we make two contributions in this paper. First, we show that when we consider caches with nonvolatile memory the hit ratio performance measure traditionally used to represent buffer management performance may be misleading; we advocate that the actual number of disk accesses should be accurately reflected in the performance measure of interest. Second, we show that once we accurately reflect the number of disk accesses, the MIN cache replacement algorithm [3] is no longer optimal and may not even be directly applicable to caches that employ nonvolatile memory.

1.1 Nonvolatile RAM

Battery supported nonvolatile RAM (NVRAM) has been in use for years, and numerous studies on making use of battery supported NVRAM and their effects have been reported [1,2]. Today, however, we are fast approaching an era where NVRAM will exist in a different form making it much more prevalent. NVRAM is being fabricated as semiconductors that do not require any external power source. FeRAM (Ferro-electro RAM), MRAM (Magnetic RAM), and PRAM (Phase-change RAM) are some of the more commonly talked about NVRAM [12].. They are being developed by major semiconductor companies such as Texas Instruments, IBM, Samsung, Fujitsu, Motorola, etc., and 1-2Mb chips are already being sold [13].

Table 1: Comparison of characteristics of volatile and nonvolatile RAM

	DRAM	SRAM	FeRAM	MRAM	PRAM
Non Volatility	No	No	Yes	Yes	Yes
Read speed	～100ns	～50ns	～100ns	～100ns	～100ns
Write speed	～100ns	～50ns	～100ns	～100ns	～500ns
Retention	Volatile	Volatile	10 years	10 years	10 years
Cost/bit	Low	High	High	High	Low

Characteristics of some of these new battery-less NVRAM compared with traditional volatile RAM are summarized in Table 1. Performance-wise they are similar to DRAM, while being able to retain information without power for approximately 10 years. The cost of these chips is still high. Some of the chips currently available in the market are fully compatible with SRAM, meaning that installing them to everyday computer systems should not be a difficult task (though not to be done by a layman as if installing a new program) [13]. As semiconductor technology continues to make progress we will soon see NVRAM become an everyday component of our commodity computers.

Manuscript received March 5, 2007
Manuscript revised March 25, 2007

1.2 Related Works

The key advantage of using NVRAM is in enhancing write performance as with NVRAM writes need not incur the hefty price of a disk access. In this regard, we can summarize previous research on enhancing system performance using NVRAM into two categories as follows.

One direction of research is in using NVRAM as an extension of storage and thus, maintaining metadata in this part of storage. Miller et al. introduces the HeRMES file system that makes use of MRAM to store metadata, while storing file data in disk [8]. Another file system, MRAMFS, uses a similar approach as HeRMES, but it utilizes compression on inodes in order to save NVRAM space as the authors assume that NVRAM is a scarce resource [5]. Conquest is another file system developed with NVRAM in mind [10]. Conquest considers storing not only metadata, but also small sized files while leaving large files on disk.

The other direction of research with NVRAM considers buffer caching. Since real systems generally tend to use the write-back policy for writes due to performance reasons, there is always a window of time in which consistency of the file system may be compromised. By making writes to NVRAM, this window of consistency loss can be removed, and consistency can be maintained in full without compromising performance. It was shown by Baker et al. that write traffic can be reduced significantly in a distributed file system with the help of NVRAM [2]. For management of the buffer space, they compare the LRU and random replacement algorithms and show that the two schemes show little difference in reducing write traffic to disk. Haining and Long propose and study algorithms for NVRAM write buffer management; specifically, LRU, shortest access time first (STF), and largest segment per track (LST) [7]. They not only consider the issue of replacement, but also consider the issue of staging, that is, when to clean the dirty blocks in cache. They report that in most cases LRU is most effective. Akyurek and Salem perform an extensive simulation study on managing NVRAM buffers [1].. They propose and categorize policies based on the actions taken upon a read miss and on write allocation. Recently, Gill and Modha studied the use of nonvolatile cache in storage devices, specifically a RAID-based storage device [6]. They propose a cache management scheme that effectively combines the temporal and spatial locality characteristics of the workload.

1.3 Remainder of the Paper

The rest of the paper is organized as follows. The next section discusses the effects of NVRAM on the performance measure seen by users. We show that the generally accepted hit ratio measure is inadequate to accurately represent performance and that a new performance measure, the disk access ratio, should be used. In Section 3, we discuss the implications of the disk access ratio and caches with NVRAM on the optimal MIN replacement algorithm. Finally, we summarize and conclude in Section 4.

2. Hit Ratio, Miss Ratio, and Disk Access Ratio

Traditionally, the hit ratio has been used as a performance measure to compare various buffer cache management algorithms. This measure is based on the assumption that each miss will incur a single disk access. Correctly representing disk access is important as this is what directly affects the user perceived performance. In this section, we show that this measure is inadequate when considering systems that make use of buffer caches that incorporate NVRAM.

Hereafter, we assume that the buffer cache comprises a volatile part and a nonvolatile part. The volatile part uses traditional volatile RAM. In our discussions, we refer to this portion of the buffer cache as volatile space. The nonvolatile part uses NVRAM, and will be referred to as nonvolatile space. To take full advantage of NVRAM, we assume that all writes are made to nonvolatile space, that is, all dirty blocks reside in nonvolatile space to ensure that writes are permanent and safe. This does not preclude the fact that a clean block may reside in nonvolatile space as well.

Traditionally, in the field of cache management, the hit ratio, as introduced in conventional textbooks [4,9], is used as a measure to evaluate the effectiveness of memory access in the memory hierarchy. In traditional cache replacement algorithm discussions, one generally says that a reference is a hit in the cache, if the block it is referencing is found in the cache. A miss, on the other hand, is a situation where a block is not found in the cache. For a read request, a miss would mean that the disk will have to be accessed as the missing block needs to be brought into the cache. However, if the request is a write, a disk access may or may not happen. For the write to be safe and permanent, the write-through policy must be employed making disk access necessary. However, in real life, policies such as write-back or write-behind are used due to performance reasons, risking loss of data.

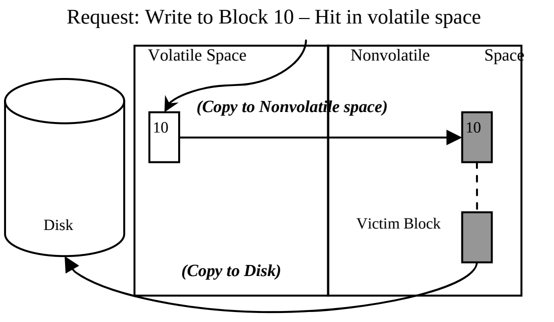
With the advent of nonvolatile space in the buffer cache, we need to look more carefully into whether the hit ratio is an accurate performance measure. For a read request, the notion of a hit remains unchanged; if the block is found in the cache it is a hit, otherwise it is a miss, and a miss incurs a disk access.

Now consider a write request. With the assumption that writes all happen on nonvolatile space, a write miss may not incur a disk access even with the write-through policy if the block to be replaced by the miss is a clean block in nonvolatile space. Thus, the negative connotation of a miss may be a misrepresentation as a miss may not incur a disk access.

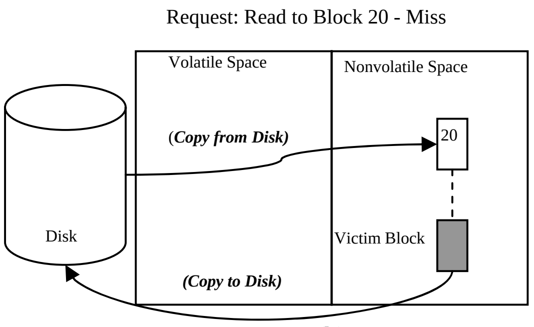
Let us now consider a write hit. Take the situation in Figure 1(a) where a write reference to Block 10 is made, and Block 10 is found in volatile space. This request is a hit in the traditional sense. However, since all dirty blocks must reside in nonvolatile space, we need to move the newly written block to nonvolatile space. Thus, we may have to evict a victim block from nonvolatile space, and if this block is dirty it must be written to disk, as shown in the example in Figure 1(a). Hence, a disk access may be incurred even on a write hit. This results in a hit misrepresenting a positive performance effect where, in fact, it actually incurs a negative performance effect, that is, a disk access.

Then, a measure that better represents the real world for buffer cache management performance may be the miss ratio, which we define as the ratio of the number of references that incur a disk access to the total number of references. (Note that this definition is different from the conventional definition where the miss ratio is simply (1-hit ratio) [9].) Then, in the Figure 1(a) example, the reference that was considered a hit is still a hit in terms of its availability in the buffer cache, yet will be counted as a miss. Consequently, a disk access occurs when a miss occurs, and hence, the miss ratio represents the ratio of references that will actually incur a disk access. Hence, the miss ratio may be a more accurate measure to use to represent nonvolatile buffer cache management performance.

Even so, the miss ratio may still not be accurate enough. Consider Figure 1(b) where a read request to Block 20 incurs a miss. Normally, a read miss will incur one disk access to fetch the referenced block. However, if the victim block to be evicted is a dirty block in nonvolatile space an extra disk access occurs to write back the dirty block. That is, two block accesses result from a single miss. The accumulation of these extra disk accesses can have a significant effect on buffer management performance. Hence, we propose that the disk access ratio, which we define as the ratio of the number of disk accesses incurred by all references to the total number of references, should be used to accurately reflect the performance of nonvolatile buffer cache management.


Request: Write to Block 10 – Hit in volatile space
Volatile Space
Nonvolatile Space
(Copy to Nonvolatile space)
10
10
Disk
Victim Block
(Copy to Disk)

(a)



(b)

Fig. 1 Actions taken when (a) a write hit occurs in volatile space and (b) when a read miss incurs a dirty block eviction.

Table 2: Performance numbers showing the difference in value for various performance measures.

Cache Size	Nonvolatile Space	(1 – Hit Ratio)	Miss Ratio	Disk Access Ratio
32MB	10%	27.25%	23.89%	30.61%
	20%	27.04%	23.63%	30.45%
	30%	26.92%	23.45%	30.39%
	40%	26.82%	23.30%	30.35%
	50%	26.74%	23.14%	30.34%
64MB	10%	24.02%	21.42%	26.61%
	20%	23.73%	21.04%	26.42%
	30%	23.56%	20.78%	26.34%
	40%	23.43%	20.56%	26.30%
	50%	23.32%	20.35%	26.29%
128MB	10%	20.87%	18.95%	22.79%
	20%	20.54%	18.49%	22.60%
	30%	20.34%	18.15%	22.53%
	40%	20.20%	17.90%	22.51%
	50%	20.07%	17.63%	22.51%
256MB	10%	17.66%	16.37%	18.95%
	20%	17.23%	15.76%	18.70%
	30%	16.92%	15.25%	18.60%
	40%	16.71%	14.85%	18.56%
	50%	16.54%	14.52%	18.55%

Consider Table 2 that compares the three performance measures discussed above. The first column is (1-hit ratio), the traditional definition of miss ratio, the second column shows our definition of miss ratio, that is, the ratio of references that incurs a disk access, and finally, the last column shows the disk access ratio, that is, the ratio of the actual number of disk references to the total number of references. All measures were obtained running a replacement algorithm based on the optimal MIN algorithm [3]. The difference between this MIN-based algorithm and the MIN algorithm is that since all dirty blocks need to be in nonvolatile space, when a dirty block is moved to nonvolatile space, the victim is selected using the MIN algorithm from among only those blocks in nonvolatile space. For other references that do not move dirty blocks to nonvolatile space, the MIN algorithm is employed. The point here is not to introduce a new algorithm, but to show the significance of the measure used to represent performance even when the exact same algorithm is used.

The numbers shown in Table 2 were obtained by simulating a buffer cache running the TPC-C trace [11]. The total request size of this trace is 99.4GB with 84.65% of the requests being read requests and the remaining 15.35% of the requests being write requests. The size of the cache simulated range from 32MBs to 256MBs with a block size of 8KBs. For each of the buffer cache size, we assume that part of the cache is nonvolatile space. The specific size of the nonvolatile space is represented in percentage size of the total cache size. For example, for a 32MB cache with 50% nonvolatile space, 16MB of the cache is volatile space, while 16MB of the cache is nonvolatile space.

Observe from Table 2 that the miss ratio is lower than (1 – hit ratio). The reason for this is that many of the misses counted in (1- hit ratio) does not incur a disk access as a write miss may replace a clean block in nonvolatile space. In that sense, (1-hit ratio) is underestimating the performance of buffer management compared to the miss ratio. However, a comparison of the miss ratio with the disk access ratio shows that the miss ratio is overestimating performance by quite a large margin, specifically, by 2.5 to 7.2 percentage points, for the range of experiments we conducted.

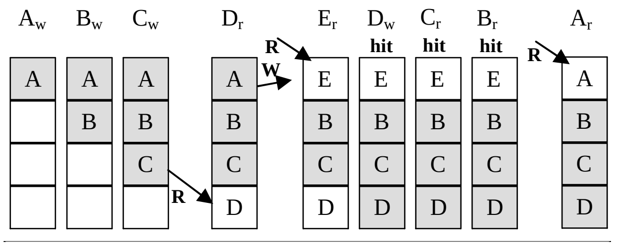
3. Implications of nonvolatile space on the MIN algorithm

In the previous section, we showed that the disk access ratio is a more accurate performance indicator for cache management. In traditional buffer cache management, the MIN replacement algorithm is considered the benchmark with which newly proposed algorithms are compared. This is because the MIN algorithm, even though infeasible in real life, is the proven optimal replacement algorithm in terms of the hit ratio. In this section, we discuss the implications of buffer caches that contain nonvolatile space on the optimal MIN algorithm. We consider two separate cases. In the first case where the buffer cache consists of only nonvolatile space, we show that MIN is not optimal in terms of the disk access ratio. Secondly, when nonvolatile space and volatile space coexist in the buffer cache, we show that MIN may not be applied directly. Though we do not prove the non-optimality of MIN in this situation, we conjecture that MIN is no longer optimal.

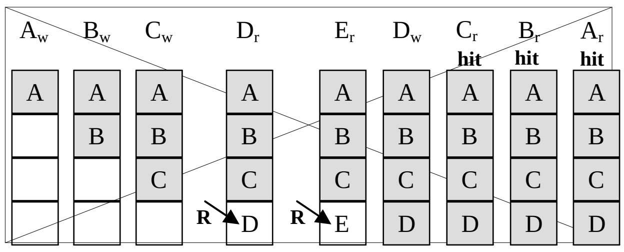
3.1 Non-Optimality of MIN in regards to the disk access ratio

Consider a buffer cache with only nonvolatile space. There are no alterations to the read and write operations. All operations are serviced just like in the traditional buffer cache. The only difference is that now, all writes are safe. Since the buffer cache is limited, eventually cache replacement will occur, and if the MIN algorithm is employed, the block that will be referenced furthest in the future will be selected and evicted. If this block happens to be dirty, it will have to be written to disk, incurring a disk access. In the following, we give a counterexample that shows that the MIN algorithm is no longer optimal in terms of the disk access ratio.

Reference String : A_w B_w C_w D_r E_r D_w C_r B_r



(a) MIN replacement algorithm (arrows indicate disk access)



(b) Replace most recently used block (arrows indicate disk access)

Fig. 2 Depiction of disk access as (a) the MIN replacement algorithm and (b) the most recently used block replacement algorithm are used.

Consider Figure 2. The reference string is as given with the 'r' and 'w' subscripts referring to read and write requests. Initially, all blocks in the buffer are clean and empty. After the first three references, blocks A, B, C are dirty, depicted by the shadings. Upon a read request on block D, a miss occurs and a corresponding disk access occurs. Figure 2(a) shows the rest of the changes and disk accesses (represented by the arrows) when employing the MIN replacement algorithm. Note that on a read request for block E, two disk accesses are invoked, one to write dirty block A, which is the block to be referenced furthest in the future, and another to read block E. Figure 2(b) shows the results when replacing the block that is most recently used. We see from the two figures that after the 9 references, the MIN algorithm incurs 4 disk accesses resulting in a disk access ratio of 0.44, while the disk access ratio is 0.22 in the later case. Hence, we conclude from this counterexample that the MIN algorithm is no longer the optimal algorithm in terms of the disk access ratio.

3.2 Non-Applicability of MIN

Let us now consider the situation where the buffer cache is composed of both volatile and nonvolatile space. We show that the MIN algorithm is no longer directly applicable given the assumption that all writes must be safe. This can be shown with a simple example as follows. Take the situation where a write miss has occurred and a block has to be replaced. With the MIN algorithm, the block that is to be referenced furthest down the future would be selected. Let us say that this block is a block in volatile space. As writing to this block will not be safe, we cannot replace this block with the incoming dirty block. A block in nonvolatile space, which is no longer the block that will be referenced furthest down the future, will have to be replaced. Hence, MIN is not applicable in this situation. We may choose to employ the MIN algorithm locally in each of the volatile and nonvolatile space. However, whether this is optimal or not is an open question.

4. Conclusion

Performance studies of buffer caches, to date, have generally been done with write references ignored. In this study, we argue that with the introduction of new nonvolatile memory technology, this is not a wise choice. We show that when we consider write references in buffer caches with nonvolatile RAM (NVRAM), the traditional hit ratio performance measure is no longer an accurate representation. We advocate that representing the actual number of disk accesses is necessary to accurately represent performance, and hence, propose that the disk access ratio, defined as the ratio of the number of disk accesses incurred by all references to the total number of references, be used. Actual measurement numbers are used to show that considerable differences may exist among the representations. We also show that once buffer caches that contain NVRAM and the disk access ratio performance measure is used, the MIN replacement algorithm, which is optimal in terms of the hit ratio in traditional cache management, is no longer optimal and that the MIN algorithm may not be directly applicable in certain cache environments.

Though studies in using NVRAM have been conducted, this area of research is still quite young. Through this study, we have left some questions open. What is the optimal algorithm when the buffer cache has only nonvolatile space? What about when both volatile and nonvolatile space exists in the cache? What is the best ratio of volatile and nonvolatile space for maximum performance? We are in the process of examining some of these and many other new problems that arise with the addition of nonvolatile memory in our computer system.

Acknowledgments

This work was supported in part by grant No. R01-2004-000-10188-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

[1] S. Akyurek and K. Salem, "Management of Partially Safe Buffers," IEEE Transactions on Computers, Vol. 44 No. 3 pp. 394-407, 1995.
[2] M. Baker, S. Asami, E. Deprit, J. Ousterhout, and M. Seltzer, "Non-volatile memory for fast, reliable file systems," in Proc. of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 10--22, October 1992.
[3] L. Belady, "A study of replacement algorithms for a virtual-storage computer," IBM Systems Journal, 5(2):78–101, 1966.
[4] C. Crowley. *Operating Systems: A Design-Oriented Approach.*. IRWIN, 1997.
[5] N. K. Edel, D. Tuteja, E. L. Miller, and S. A. Brandt, "MRAMFS: a compressing file system for non-volatile RAM," in Proc. of the 12th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOT 2004), pp.596-603, October 2004.
[6] B. S. Gill and D. S. Modha, "WOW: Wise Ordering for Writes—Combining Spatial and Temporal Locality in Non-volatile Caches," in Proc. of the 4th USENIX Conference of File and Storage Technologies (FAST 2005), December 2005.

[7] T. R. Haining and D. D. E. Long, "Management policies for non-volatile write caches," in Proc. of the IEEE International Performance, Computing and Communications Conference, pp. 321–328, Feb. 1999.
[8] E. L. Miller, S. A. Brandt, and D. D. E. Long, "HeRMES: High Performance Reliable MRAM-Enabled Storage," in Proc. of the 8th IEEE Workshop on Hot Topics in Operating Systems, pp. 83-87, 2001.
[9] D. A. Patterson and J. L. Hennessey. *Computer Organization and Design*. Morgan Kaufmann, Third Edition, 2005.
[10] A. A. Wang, P. Reiher, G. J. Popek, and G. H. Kuenning, "Conquest: better performance through a disk/persistent-RAM hybrid file system," in Proc. of the 2002 USENIX Annual Conference, June 2002.
[11] Y. Zhou, Z. Chen, and K. Li. "Second-Level Buffer Cache Management," IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 7, pp.505-519, July, 2004.
[12] http://www.memorystrategies.com/report/EmergingMemories.htm, Emerging Memories: Applications, Device and Technology.
[13] http://www.ramtron.com/doc/Products/overview.asp

Sam H. Noh received the BS degree in computer engineering from the Seoul National University, Korea in 1986, and the PhD degree from the Department of Computer Science, University of Maryland at College Park in 1993. He held a visiting faculty position at the George Washington University from 1993 to 1994 before joining Hong-Ik University in Seoul Korea, where he is now a Professor in the School of Information and Computer Engineering. From August 2001 to August 2002, he was also a Visiting Associate Professor to UMIACS (University of Maryland Institute of Advanced Computer Studies). His current research interests include operating system issues on embedded/computer systems and robotics. Dr. Noh is a member of the IEEE Computer Society and the ACM.

Inhwan Doh received the MS degree in computer engineering from the Hongik University, Korea in 2006. He is currently a Ph.D student at Hongik University.. His research interests include operating system issues on embedded/computer systems and robotics.

Jungkyu Park received the MS degree in computer engineering from Hongik University, Korea in 2003. He is currently a Ph.D student at Hongik University. His research interests include mobile robot localization and mapping and embedded /computer system.