韓國 컴퓨터情報學會 論文誌
第12卷 第6號, 2007. 12.

2007-12-6-1-16

디스크 입출력 서브시스템을 위한 개선된 디스크 블록 캐싱 알고리즘

정 수 목*, 노 경 택**

Advanced Disk Block Caching Algorithm for Disk I/O sub-system

Soo-Mok Jung*, Kyung-Taeg Rho**

요 약

A hard disk, which can be classified as an external storage is usually capacious and economical. In spite of the attractive characteristics and efforts on the performance improvement, however, the operation of the hard disk is apparently slower than a processor and the advancement has also been slowly conducted since it is based on mechanical process. On the other hand, the advancement of the processor has been drastically performed as semiconductor technology does. So, disk I/O sub-system becomes bottleneck of computer systems' performance. For this reason, the research on disk I/O sub-system is in progress to improve computer systems' performance. In this paper, we proposed multi-level LRU scheme and then apply it to the computer systems with buffer cache and disk cache. By applying the proposed scheme to computer systems, the average access time to disk blocks can be decreased. The efficiency of the proposed algorithm was verified by simulation results.

Abstract

컴퓨터시스템에서 메모리시스템은 계층적인 구조를 갖는다. 외부기억장치에 해당하는 디스크는 용량이 크고 가격이 저렴하지만 동작은 기계적인 특성에 기반을 두고 있어 주기억장치에 비하여 매우 느리고 디스크의 성능 향상도 매우 느리게 이루어지고 있지만 처리기는 반도체기술의 발전으로 속도향상이 매우 빠르게 이루어지고 있다. 따라서 저속의 디스크 입출력서브시스템은 컴퓨터시스템의 전체 성능에 병목(bottle neck)을 일으키고 있다. 컴퓨터시스템내의 디스크 입출력 서브시스템의 성능을 개선함으로 컴퓨터시스템의 전체 성능개선을 실현하는 연구가 이루어지고 있다. 본 논문에서는 처리기가 필요로 할 가능성이 높은 디스크블록을 버퍼캐시와 디스크 캐시에 효율적으로 유지하여 디스크블록 평균접근시간을 줄임으로 컴퓨터시스템의 성능을 향상시키는 개선된 알고리즘인 multi-level LRU 기법을 제안하였고 이를 버퍼캐시와 디스크 캐시를 가지는 시스템에 적용하였다. 시뮬레이션을 통하여 제안된 방안의 성능을 평가하였다.

▸ Keyword : Disk I/O sub-system, Buffer cache, Disk cache, LRU, sLRU, multi-level LRU

• 제1저자 : 정수목 • 교신저자 : 노경택
• 접수일 : 2007.11.2, 심사일 : 2007.12.3, 심사완료일 : 2007.12.15.
* Sahmyook University, Professor, ** Eulji University, Professor

Ⅰ. Introduction

The speed improvement of a processor in computer systems is achieved very quickly in consequence of semiconductor technology improvement. On the other hand, the speed improvement of a hard disk is not fast because the operation of a hard disk is based on mechanical process. The speed difference of a processor and a hard disk is more serious. So, disk I/O sub-system becomes bottleneck of computer systems' performance. For this reason, the research on disk I/O sub-system is in progress to improve computer systems' performance.

The access time to read data form disk is composed of three items. One is seek time for disk head to move on the track in which the specified sector resides. Another one is rotational delay time for disk head to move the beginning point of the specified sector. The other is transfer time for disk head to pass the specified sector to read the data form the sector. These processes are based on mechanical process such as disk head moving or disk plate rotation. So, the performance improvement of a disk is achieved restrictively. For this limitation, disk block caching schemes were studied to improve the performance of disk sub-system. If a disk block which will be used by processor is stored in cache, processor can read the disk block directly from cache instead of disk which has very large access time when processor needs a disk block. So, the problem of performance degradation cased by the speed difference between processor and disk can be solved efficiently. As an example in UNIX, a part of main memory is used as buffer cache in which disk blocks are stored.[1, 2] Disk blocks used by processor are stored in buffer cache. When processor needs the disk block again, processor reads the disk block directly from buffer cache. When there is not the disk block requested by processor in buffer cache, disk read operation happen. At this time, the disk block which is read from disk is stored in buffer cache, after that the disk block is serviced to processor. So, if buffer cache is used, disk read operation decreased and computer systems' performance improvement can be achieved.

Data General Corporation developed disk controller with disk cache and I/O processor. In this system, I/O operation can be done without processor's interference and can be done side by side processor.[4] Processor, main memory, and disk controller in the system with buffer cache and disk cache are connected by system bus as shown in figure 1.

When processor requests a specific disk block, UNIX operating system investigates the disk block in buffer cache. If there is the disk block in buffer cache, processor reads the disk block from buffer cache.

If there is not the disk block in buffer cache, UNIX operating system requests the disk block to disk controller. Disk controller investigates disk cache to check whether the disk block exists in buffer cache or not. When the disk block exists in disk cache, the requested disk block is stored in buffer cache and then the disk block is serviced to processor. At this time, disk access is unnecessary.

So, computer systems' performance is improved. When the disk block does not exist in disk cache, disk controller executes disk I/O. In this kind of systems, disk blocks are stored concurrently in buffer cache and disk cache. So, computer systems' performance degenerates.[5] Also cache management scheme affects the number of disk access and computer systems' performance. Segmented Least Recently Used (sLRU) scheme [6] was proposed to manage cache efficiently, and many cache management schemes were proposed to limit duplicated storage of disk block[7-12].

DCD(disk caching disk)[13] was proposed. Buffer cache replacement algorithm such as LIRS(Low Inter-reference Recency Set)[14] was proposed, and ULC(Unified and Level-aware Caching)[15] was proposed for a large client/server cluster system, in which file blocks are cached in a multi-level storage hierarchy. uCache algorithm[16] and Counter-based

L2 Cache Replacement Algorithms[17] were proposed to improve performance of L2 caches in multi-level cache hierarchies for multiple clients.

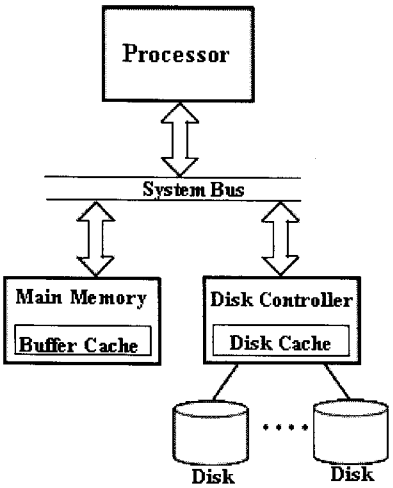

Figure 1. System configuration with buffer cache and disk
그림 1. 버퍼캐시와 디스크캐시를 가지는 시스템 구성도

To reduce the speed gap between RAM and disks, a new disk organization called

In this paper, we proposed multi-level LRU scheme which is the improved version of sLRU and we applied the proposed scheme to the system with buffer cache and disk cache to increase disk I/O sub-systems' performance by removing duplicated disk blocks between buffer cache and disk cache. The efficiency of the proposed algorithm was verified by simulation results.

This correspondence is organized as follows: In section 2, segmented Least Recently Used algorithm is reviewed briefly. In section 3, the proposed algorithm is presented. In section 4, experimental results are shown. Finally, we conclude this correspondence in section 5.

II. Segmented Least Recently Used Algorithm(sLRU)

A Disk block which is referenced by process is stored in cache and then it is provided to processor. If a disk block is referenced again within a certain time, there is a tendency of the disk block is used repeatedly. If a disk block is not referenced within a certain time, there is a tendency of the disk block is not used repeatedly.[9] Therefore, if a disk block which is reused within a certain time is stored continually in cache, computer systems' performance improvement can be achieved by supplying the disk block to processor from cache without disk access. By using this characteristic, sLRU cache management scheme was proposed. sLRU removes disk blocks form cache if the disk blocks are not reused within a certain time. Applying sLRU cache management scheme, disk blocks which were used only one time as in the case of sequential reference can be removed quickly form cache. So, the disk blocks which have high possibility of reusing can be maintained in cache for a long time, cache hit ratio will be increased, and computer systems' performance improvement can be achieved.

To store and manage separately disk blocks which are used only one time and several times, cache is divided into two segments in sLRU scheme, one is Protected segment and the other is Probationary segment. Protected segment and Probationary segment are linked as shown in figure 2. In sLRU scheme, when cache miss occurs, disk block is read from disk and it is stored in MRU end of Probationary segment.(1) When cache hit occurs, the specified disk block moves to MRU end of Protected segment.(2)(3) If the disk block is not reused during it is maintained in Probationary segment, it moves toward LRU end of Probationary segment and it is discarded at last.(5)

In sLRU, the disk blocks in Protected segment are reused by processor at least one time. And the disk blocks in Probationary segment are used only one time by processor or they are came from Protected segment because they are not reused for a long time. A boundary pointer is used in sLRU scheme to divide

cache into Protected segment and Probationary segment, and it is set to point the middle of the list. Each cache line has 1 bit flag to represent which segment involves the cache line.

When a disk block is inserted between MRU end of Probationary segment and LRU end of Protected segment due to cache miss, the pointer is adjusted to point the new disk block and the flag is set to show the disk block is in Probationary segment. So, the disk blocks which have high possibility of reusing can be maintained in cache for a long time.

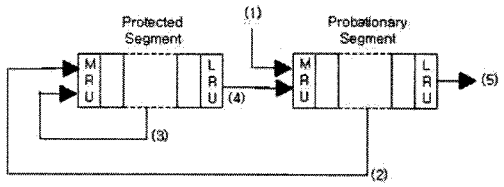

Figure 2. sLRU scheme
그림 2. sLRU 기법

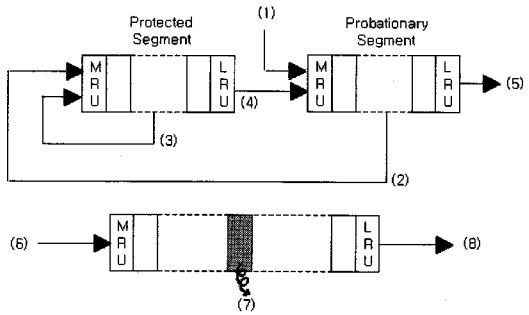

Figure 3. The operation of buffer cache and disk cache when sLRU scheme is applied to buffer cache
그림 3. sLRU 기법이 버퍼캐시에 적용된 경우, 버퍼캐시와 디스크캐시의 동작

In a system with buffer cache and disk cache, a disk management scheme in which sLRU scheme is applied to buffer cache and reused disk block is removed from disk cache was proposed as shown in figure 3.[12]

By using this algorithm, disk block duplication can be reduced and a disk block which has high possibility of reusing can be maintained in cache for a long time.

The operation of this algorithm is as follows. A disk block which is referenced at first is stored in MRU end of Probationary segment of buffer cache and MRU end of disk cache simultaneously.(1)(6) When a disk block is reused during it is stored in Probationary segment($\triangle t$), the disk block moves to MRU end of Protected segment(2) and it is removed simultaneously from disk cache.(7) When a disk block is not reused during it is stored in Probationary segment, the disk block in buffer cache moves from MRU end of Probationary segment to LRU end of it, and then it is discarded at the time of one disk block is read form disk owing to cache miss.(5) Similarly, a disk block in disk cache moves from MRU end toward LRU end, and after it arrives at LRU end it is discarded at the time of one disk block is read from disk owing to cache miss.(8) $\triangle t$ is determined by the size of Probationary segment and the ratio of disk block entry into cache.

III. Advanced Disk Block Caching Algorithm for Disk I/O sub-system

As shown in figure 3, a disk block which is entered into buffer cache moves to MRU end of Protected segment when it is reused and then it stays in Protected segment. And then the disk block moves toward LRU end of Probationary segment and then it is removed finally at LRU end of Probationary segment. In this case, a disk block which is reused only one time is treated as same with a disk block which is reused more than two times. So, sLRU scheme can not regulate the residing time of a disk block within cache according to the pattern of disk block reuse. Therefore, sLRU scheme does not reflect the pattern of disk block reuse. Also, a disk block is stored in buffer cache and disk cache in duplicate until it is removed from disk cache when it is reused by processor. In proposed multi-level LRU, disk blocks are managed in cache which is divided into Probationary segment, Transitional segment, and

Protected segment like as shown in figure 4 to solve this problem. A disk block is stored at MRU end of Transitional segment when it is referenced by processor at first.(1) When the disk block is reused by processor during it exists in Transitional segment, the disk block moves to MRU end of Protected segment.(2) At this time, all the blocks in Protected segment move toward Transitional segment.(4) If a disk block is reused by processor during it exists in Protected segment, it moves to MRU end of the Protected segment.(3) If a disk block is not reused by processor during it exists in Protected segment, it moves gradually toward Transitional segment.(4)

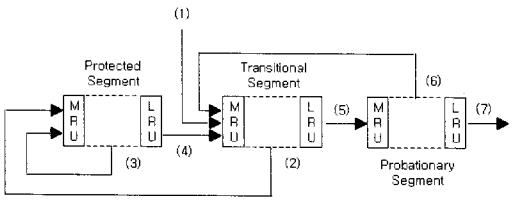

Figure 4. Multi-level LRU scheme
그림 4. 다단계 LRU 기법

The procedure of multi-level MRU as shown in figure 4 is as follows:

step 1: while (disk block is requested by processor?) {
step 2: if (the disk block is in cache?) execute disk_block_hit_routine;
step 3: else execute disk_block_miss_routine;
step 4: }

disk_block_hit_routine
step 1: if (the disk block is in Probationary segment?) execute Proba_routine;
step 2: else if (it is in Transitional segment?) execute Trans_routine;
step 3: else execute Prote_routine;

Proba_routine
step 1: shift right all the blocks in Transitional segment and the blocks from MRU end to the previous one of the specified disk block in Probationary segment.
step 2: move the specified disk block to MRU end of Transitional segment

Trans_routine
step 1: shift right all the blocks in Protected segment and the blocks from MRU end to the previous one of the specified disk block in Transitional segment
step 2: move the specified disk block to MRU end of Protected segment

Prote_routine
step 1: shift right the disk blocks from MRU end to the previous one of the specified disk block in Protected Segment
step 2: move the specified disk block to MRU end of Protected segment

The cache hit ratio of the proposed multi-level LRU scheme increases because the disk block is reused more frequently the disk block resides in cache longer. So, the requested disk block is serviced directly from cache to processor without disk access. Therefore, computer systems' performance can be improved.

The proposed multi-level LRU scheme was applied to the system with buffer cache and disk cache as shown in figure 5 to limit the disk block duplication and to enhance the cache hit ratio.

The procedure of figure 5 is as follows:

step 1: while(!disk block is requested by processor?) ;
step 2: if (is it in Protected segment?) execute Prote_routine;
step 3: else if (it is in Transitional segment?) execute Trans_routine;

step 4: else if (it is in Probationary segment?)
execute Prob_routine;
step 5: else if (it is in disk cache?)
execute Disk_cache_routine;
step 6: else execute Disk_block_insert_routine;
step 7: goto step 1

Prote_routine, Trans_routine, and Proba_routine describe the operation of buffer cache and Disk_cache_routine describe the operation of disk cache.

Disk_cache_routine
step 1: shift right from MRU end to the previous one of the specified disk block in disk cache
step 2: move the specified disk block to MRU end of Probationary segment

Disk_block_insert_routine
step 1: shift right all the blocks of Transitional segment, Probationary segment, and disk cache(the disk block at LRU end of disk cache is removed)
step 2: The requested disk block which is read from disk is stored at MRU end of Transitional segment

Cache hit ratio increases if multi-level LRU scheme is applied to the system with buffer cache and disk cache to eliminate disk block duplication and to reside a disk block longer in cache as it is referenced more frequently. So, the number of disk access is minimized and average disk block access time is reduced. Therefore, computer systems' performance is increased.

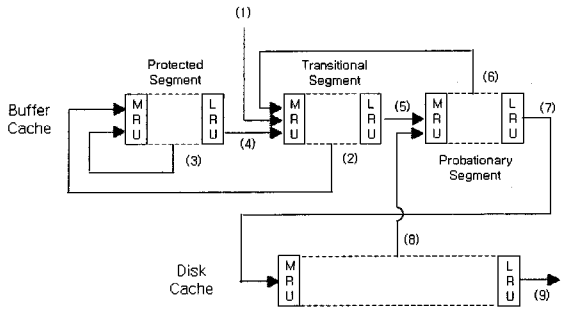

Figure 5. The operation of buffer cache and disk cache with multi-level LRU scheme
그림 5. 다단계 LRU 기법을 적용한 경우, 버퍼캐시와 디스크캐시의 동작

IV. Experimental Results

We executed simulation to evaluate the performance of the proposed algorithm which is multi-level LRU scheme is applied to the system with buffer cache and disk cache to improve disk I/O sub-systems' performance by removing duplicated blocks between buffer cache and disk cache.

The simulation environment is as follows. The size of buffer cache and disk cache is 8MB and 4MB respectively. The size of Protected segment and Probationary segment is 4MB respectively when sLRU scheme is applied to the system with buffer cache and disk cache. The size of Probationary segment, Transitional segment, and Protected segment is 2.5MB, 3MB, 2.5MB respectively. We executed simulation for 2000 disk blocks and the disk block size is 8KB. We assumed that disk blocks don't change. So, disk block update to disk is unnecessary when a disk block is replaced. To generate the sequence of requested disk blocks, the number of reference for each disk block is generated by random function as 1 to 20, the interval of reuse for each disk block is generated by weighted random function as 1 to 100. The total number of reference to disk block was 20.857 times.

The simulation results are shown in table 1 where multi-level LRU and sLRU scheme were applied to

the system with buffer cache and disk cache. Cache hit ratio is the percentage of the serviced number of disk blocks from cache to the number of requested disk blocks by processor.

Table 1. Cache hit ratio for the multi-level LRU and sLRU algorithm are applied to the system with buffer cache and disk cache
표 1. 버퍼캐시와 디스크캐시를 가지는 시스템에 다단계 LRU 기법과 sLRU기법이 적용된 경우의 캐시 적중율

Applied scheme	Classification		Cache hit ratio		
sLRU	buffer cache	Protected segment	27.5% (5743/20857)	60.4% (12598 / 20857)	75.7% (15794 / 20857)
		Probationary segment	32.9% (6855/20857)		
	disk cache		15.3%(3196/20857)		
multi-level LRU	buffer cache	Protected segment	18.0% (3752/20857)	71.7% (14951 / 20857)	91.9% (19164 / 20857)
		Transitional segment	30.1% (6279/20857)		
		Probationary segment	23.6% (4920/20857)		
	disk cache		20.2%(4213/20857)		

As shown in table 1, the proposed algorithm can increase the hit ratio of the previous algorithm in which sLRU scheme is used by 18.7% in buffer cache. This result comes from that a disk block resides longer in cache as it is referenced more frequently by using multi-level LRU scheme. In disk cache, the hit ratio is increased by 31.8%. This result comes from that many disk blocks are reside in disk cache owing to disk block duplication is eliminated between buffer cache and disk cache. If disk block can be serviced from cache to processor without disk access, the number of disk access can be reduced, and computer systems' performance improvement can be achieved. As shown in table 1, by using the proposed algorithm, disk access is minimized because more disk blocks can be serviced form cache without disk access. So, a process can be executed rapidly. Therefore, computer systems' performance can be improved by using the proposed algorithm.

V. Conclusions

We proposed multi-level LRU scheme which is improved version of sLRU and applied it to computer systems with buffer cache and disk cache. By using the proposed algorithm, disk block duplication is eliminated and the disk block exists longer in cache as it is referenced more frequently. So, cache hit ratio is increased. In our simulation, when applying the proposed algorithm, buffer cache hit ratio is increased by 18.7% and disk cache hit ratio is increased by 31.8% then sLRU scheme is used. As hit ratio is increased, the more disk blocks are serviced to processor from cache without disk access. So, average disk block access time is reduced and a process can be executed rapidly. Although disk I/O sub-system is the bottleneck of computer systems' performance, computer systems' performance can be improved by using the proposed algorithm.

참고문헌

[1] C. P. Grossman, "Cache-DASD storage design for improving system performance," IBM Systems Journal, vol. 24, no. 3/4, pp. 316-334. 1985.

[2] P. J. Jalics and D. R. McIntyre, "Caching and Other Disk Access Avoidance Techniques on Personal Computers," Communications of the ACM, vol. 32, no. 2, pp. 246-255, Feb. 1989.

[3] M. J Bach, "The Design of the UNIX Operating System," Prentice-Hall, Englewood Cliffs, NJ. 1986.

[4] Data General Corporation, "Configuring and Managing a CLARiiON Disk-Array Storage System," 1994.

[5] B. McNutt, "I/O subsystem configurations for ESA: New roles for processor storage," IBM Systems Journal, vol. 32, no. 2, pp. 252-264, 1993.

[6] R. Karedla, J. S. Love, and B. G. Wherry, "Caching Strategies to Improve Disk System Performance," Computer, vol. 27, no. 3, pp. 38-46, March 1994.

[7] D. M. Muntz and P. Honeyman, "Multi-level Caching in Distributed File Systems," In Proceedings of the 1992 Winter USENIX Conference, pp. 305-313, 1992.

[8] J. T. Robinson and N. V. Devarakonda, "Data Cache Management Using Frequency-Based Replacement," In Proceedings of the ACM SIGMETRICS Conference, pp. 134-142, 1990.

[9] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," In Proceedings of the 1993 ACM SIGMOD Conference, pp. 297-306, 1993.

[10] P. Cao, E. W. Felton, and K. Li, "Application-Controlled File Caching Policies." In Proceedings of the Summer 1994 USENIX Conference, pp. 171-182, Jun. 1994.

[11] M. D. Dahlin, R. Y. Wang. T. E. Anderson, and D. A. Patterson, "Cooperative Caching: Using Remote Client Memory to Improve File System Performance," First Symposium on Operating Systems Design and Implementation, pp. 267-280, Nov. 1994.

[12] D. H. Lee, S. H. Noh, S. L. Min, and Y. K. Cho, "Efficient Cache Management Schemes for Reducing Duplication Caching between Buffer and Disk Caches," Journal of KISS(A), vol. 22, no. 10, october 1995.

[13] Yiming Hu, Oing Yang, "A New Hierarchical Disk Architecture," IEEE Micro, Vol. 18, Issue 6, pp. 64-76, Nov. 1998,

[14] S. Jiang and X. Zhang. "LIRS: An efficient low interreference recency set replacement policy to improve buffer cache performance," In Proc. ACM SIGMETRICS, pp. 31-42, 2002.

[15] S. Jiang and X. Zhang, "ULC: A file block placement and replacement protocol to effectively exploit hierarchical locality in multi-level buffer caches," In Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04), Mar. 2004.

[16] Li Ou Xubin Ben He, Martha J. Kosa, Stephen L. Scott, "A Unified Multiple-level Cache for High Performance Storage Systems," In Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Vol. 00 (MASCOTS '05), Sep. 2005.

[17] Mazen Kharbutli, Yan Solihin, "Counter-Based Cache Replacement Algorithms," In Proceedings of the 2005 International Conference on Computer Design (ICCD '05), Oct. 2005.

저 자 소 개

Soo-Mok Jung

1984 Kyungpook National University (B.E.)
1986 Kyungpook National University (M.E)
2002 Korea University (Ph.D.)

Associate Professor
Department of Computer Science
Sahmyook University
jungsm@syu.ac.kr
〈Research interests〉: computer systems, multimedia, wireless network

Kyung-Taeg Rho

1986: Chung Ang University(B. E.)
1989: New Jersey Institute of Technology(M.E.)
2007: Korea University (Ph. D. Candidate)

Assistant Professor
Department of Medical Computer Science
Eulji University
rho@eulji.ac.kr
〈Research interests〉: mobile communication, wireless sensor network, computer systems