http://dx.doi.org/10.14400/JDPM.2013.11.11.421

Minix3 마이크로커널 기반 디스크 캐쉬 관리자의 설계 및 구현

최욱진*, 강용호*, 김선종*, 권혁숭*, 김주만*
부산대학교 IT응용공학과*

Disk Cache Manager based on Minix3 Microkernel : Design and Implementation

Wookjin Choi*, Yongho Kang*, Seonjong Kim*, Hyeogsoong Kwon*, Jooman Kim*
Pusan National University*

요 약 마이크로커널 기반의 기능성 서버인 디스크 캐쉬 관리자(DCM)은 공유 디스크의 입출력 성능을 향상하는 목적으로 설계 및 구현되었다. DCM은 Misix3 마이크로커널의 멀티 쓰레드 모드로 동작하는 시스템 엑터간의 포트를 통하여 다른 서버들과 메시지 교환으로 정합한다. 본 논문에서 제안된 DCM은 병렬 입출력이 가능하도록 공유 디스크를 논리적으로 Seven 디스크와 Sodd 디스크로 나누어 사용한다. 본 논문에서 제안하는 DCM은 특정 디스크의 사용 빈도에 따라 캐쉬 사이즈를 증감함으로써 이용율이 높은 디스크의 캐시 사이즈 확대하여 캐시 적중률을 높여주므로서 효과적인 성능을 향상할 수 있게 한다. 경험적 결과를 통해서, 본 논문에서 제안한 DCM은 높은 이용율을 갖는 공유 디스크에서 아주 효과적임을 보였다.

주제어 : Disk Cache, Micro-Kernel, I/O Performance, Parallel Processing

Abstract Disk Cache Manager(DCM), a functional server of microkernel based, to improve the I/O power of shared disks is designed and implemented in this work. DCM interfaces other different servers with message passing through ports by serving as a system actor the multi-thread mode on the Minix3 micro-kernel. DCM proposed in this paper uses the shared disk logically as a Seven Disk and Sodd Disk to enable parallel I/O. DCM enables the efficient placement of disk data because it raises disk cache hit-ratio by increasing the cache size when the utilization of the particular disk is high. Through experimental results, we show that DCM is quite efficient for a shared disk with higher utilization.

Key Words : Disk Cache, Micro-Kernel, I/O Performance, Parallel Processing

1. Introduction

Parallel processing system architectures are designed to produce results much faster by dividing a large problem into subproblems with various applications[1]. However, if the applications have a data

* 본 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음.
Received 15 October 2013, Revised 10 November 2013
Accepted 20 November 2013
Corresponding Author: Joo Man Kim(Pusan National University)
Email: joomkim@pusan.ac.kr

ISSN: 1738-1916

dependency problem which reduces the concurrency or the parallelism, the special care needs.

The improvement of the I/O performance greatly depends not only on the performances of its components: such as disks, controllers, etc., but also on the I/O architecture. However there exists a limitation to the increase of processing power due to the mechanical characteristics of the components[2]. To overcome this limitation, It has been done to make lower access to the device by using buffer cache in the present operating system and to make the processor to increase the spatial and temporal locality of data[3]. But when the system becomes large, the buffer cache managed by the present OS increases the load between nodes due to the distribution of the function and resources and the capacity of the buffer cache is limited by the limited capacity of the memory on the node. Specially, a high-speed I/O system is necessary for the application with larger data than the capacity of the memory. So, by locating disk cache near the device where all nodes can access, not only the improvement of the performance but also efficient I/O management can be made. [1][4][5][6]

The DCM is focusing not only on the efficiency of disk cache but also on the management of shared disks. Most of bigger storage server are based on distributed shared disks. A consistency problem of the shared disk data will increase communication overload between the nodes and will not contribute to the improvement[7]. So, in this paper, a separation of the shared disk into two logical disks removes the disk data sharing and the I/O from the upper file system is designed to be requested to two logical disks on the parallel.

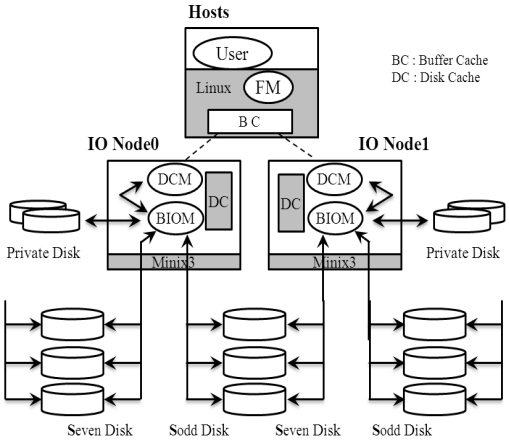
The rest of the paper is organized as follows. Section 2 looks at the system H/W and S/W models to which this study applies. At the section 3, we describe the design of the Disk Cache Manager. The measurement of the efficiency of the DCM through the experimental implementation is in section 4, and the conclusion and future research plan is presented in the last section.

2. System Model

Host mainly performs user threads on Linix and ION is a node which provides the environment where the thread with the data collection, transmission and storage related to the block I/O runs. DCM is a system server (Actor) which interacts with other servers through message communication of Minix3 and has his own address space and resources. Minix3 is a typical open source micro-kernel operating system. It is designed based on micro-kernel theory completely. Minix3 is divided into four layers. There are kernel, clock task and system task located in the first layer. The main job of kernel is to schedule the process and transform its state among ready, running and blocking. The function of the first layer is to offer a group of privilege kernel calling to higher layer, drivers and servers. The other layers upon kernel can be viewed as one layer because the kernel treats them same as one layer. Minix3 itself can be viewed as a set of processes. It makes Minix3 more flexible and portable[8].

DCM has port for communication and is divided into two parts where one receives and processes the message accessing to each port and the other manages disk cache.

The running environment of DCM proposed in this paper does not have a dependency on the shared & shared-nothing disk architecture. DCM defines two logical disks and a private disk to support shared & shared-nothing disk and manages each one cachable. If physical I/O is required due to buffer cache miss of file system buffer of OS, it is interfaced through IPC. It manages Caching Area by defining local disk mounted on IO Node as private disk and shared disk as two logical disks (S_{even} Disk and S_{odd} Disk).



[Fig. 1] Applied Model of DCM

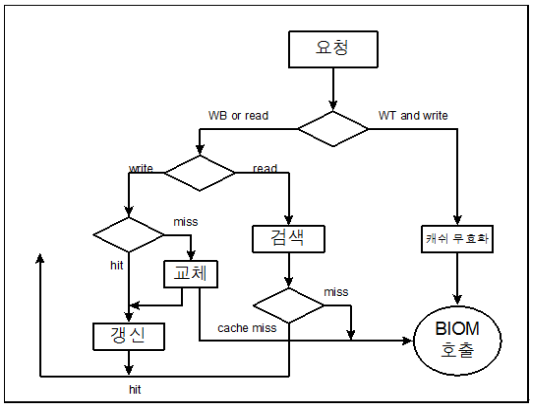
Fig. 1 is shown 2 step cache managing model. It represents the relationship among disks shared by two IO Nodes, disks shared by near IO Node and private disk and also the one between buffer cache of file system (first cache) which exists at host and disk cache (secondary cache).

3. Design and Implementation

3.1 Chache Management Policy

WB (Write-Back) is provided as a basic cache management policy. At read, WB reads from disk cache and at write, it writes on disk cache instead of writing on the disk immediately. Blocks renewed by writing data enlarges the ability of reading and writing disk by writing on the corresponding disk blocks at regular interval[2].

It also supports WT (Write-Through) as another cache management policy. At read, WT refers to disk cache but at write, it writes on disk immediately. The choice of these methods is indicated at configuration table when system is initialized.



[Fig. 2] DCM consistency maintenance algorithm

Fig. 2 is algorithm to maintain data consistency of disk cache. When WT writes something or cache miss, it transmits message to BIOM, disk driver server, for requesting the physical disk I/O.

3.2 Cache Management Architecture

As shown in Fig. 3, Cache management architecture provides framework for cache management policy and data consistency. Hash table is used to manage cache, and to hash cache segments, disk number and block number is used as tag.

• cache segment

Cache segment is a unit for managing cache and consists of Information and Data parts. There are a Tag which is made up of device number and block number, forward/backward pointer connected to hash table, forward/backward pointer connected to the usable area of freelist, the field of cache line and block condition and the pointer which indicates the line of disk cache area.

```
Struct CacheSeg{
    struct tagField      cs_tag;
    struct CacheSeg      cs_forw;
    struct CacheSeg      cs_back;
    struct CacheSeg      f_forw;
```

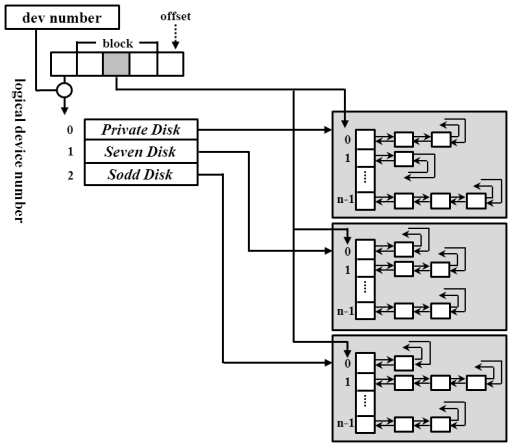
```
    struct CacheSeg      f_back;
    unsigned short       cs_lstat;
    unsigned char        cs_bstat[BLKSperLINE];
    struct cs_block      *cs_laddr;
};
```

• cache line

A unit of data which can be transmitted during disk access time most effectively and consists of contiguous 8 sectors of 512 Bytes

• cache block

A unit to maintain consistency and a minimum block size which is required by the operating system.



[Fig. 3] cache mapping architecture

• Hash Table

It is a table forming a cache architecture. In the cache segment tag field, it has the respective table for the device number and block number. Device number is necessary to distinguish devices shared by two IO Nodes. From the point of view of IO Node, three logical devices are defined as follow

Private Disk : local disk connected to the corresponding IO Node

S_{even} Disk : Even line area of shared disks.

S_{odd} Disk : Odd line area of shared disks.

〈Table 1〉 Table of status transformation according to cache I/O

I/O Rq	Line status	blk Status	disk I/O	Next Line status	Next Blk status
read	Valid	-	-	Valid	Valid
〃	Invalid	-	read line	Valid	Valid
〃	Dirty	Dirty	write line on syncio	Dirty or Valid on syncio	Dirty or Valid on syncio
〃	〃	Invalid	read blk	Dirty	Valid
〃	〃	Valid	-	Dirty	Valid
〃	Pvalid	Invalid	read line	Valid	Valid
〃	〃	Valid	-	Pvalid	Valid
write	-	-	write blks on syncio	Dirty or Pvalid on syncio	Dirty or Pvalid on syncio

3.3 Cache Consustency Management

To maintain cache consistency, cache entry is indicated by dividing it into three logical devices.

The following defines the values of the line and block status to maintain cache consistency.

• Status value of cache line

L_VALID : when all blocks are available.

L_INVALID : when all blocks are unavailable.

L_PVALID : when the part of the blocks are available.

L_DIRTY : when at least one block is dirty.

• Status value of cache block

B_VALID : available data block.

B_INVALID : not available or not cached.

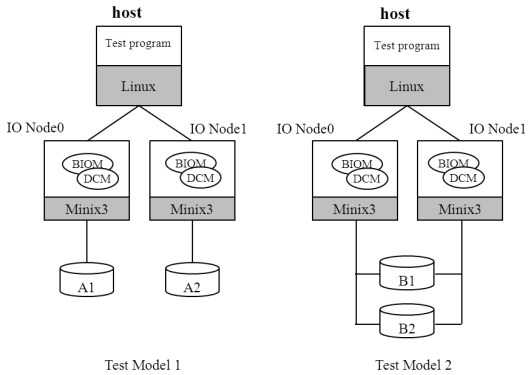
B_DIRTY : dirty block.

The transformation of cache status according to the I/O of data block is in Table 1.

4. Test and the Result

4.1 Test Environment

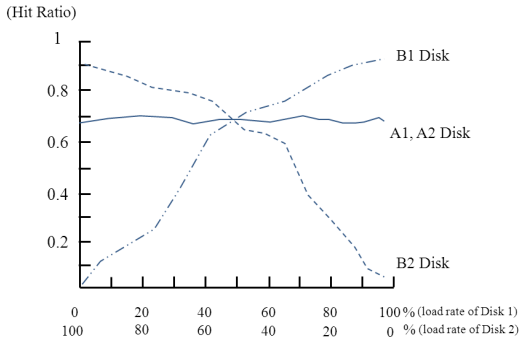
Test bed is organized with 3 1GHZ pentium PCs. Linux on one PC to play as a host and DCM and BIOM(Block IO Manager) are placed at 2 PCs to play as IO Node. Each disk has 100 Gbytes' capacity and each capacity of disk cache consists of 1 Gbytes. To measure and compare the results of DCM, the experiment was performed with two models as shown in Fig. 4.



[Fig. 4] Test bed for experimental test

4.2 Experimental Result

The hit-ratio according to the change by loading the I/O overhead to the disk 1 and 2 of each model was measured.



[Fig. 5] Cache hit ratio according to the utilization of disk.

At this time the sum of I/O load of two disks should not exceed 100% of total load. Global variables are about cache hit and miss inserted to DCM kernel and hit-ratios according to the percentages were obtained by storing them. The results of fig. 5 were obtained by running the same program on two models.

Because A1 and A2 disk are mutually independent with non-sharing mode, they showed almost the same hit-ratios regardless to the degree of the loads. Because B1 and B2 disks are divided into S_{even} Disk and S_{odd} Disk and are accessed respectively at IO node0 and IO node1, the disk with 100% load can use exclusively disk cache of 1 Gbytes.

The hit-ratio of disk cache is proportional to the capacity of cache[2]. When one disk is used in model B, hit ratio goes up by using all the disk caches of two IONs. This ultimately improves the performance by improving the throughput per hour of the whole system. If frequently used data is stored on the B1 disk and occasionally used data is stored on the B2 to run the system effectively, DCM is sure to show the efficiency.

5. Evaluation

When one physical disk is shared by two I/O nodes, Directory Scheme should be applied like the method of supporting cache consistency under the distributed computing environment, but because I/O node is not the final data consumer but the mediator, it is not necessary to take the communication loads caused by cache consistency of the shared data among the nodes. In other words, if data consumer(requiring server) requires allotment of the specified unit which does not allow disk data's share to ION, there is no shared data between two IONs and effective cache management is possible. The above results prove this fact. The unit which FM requires to DCM is the physical unit which DCM requires to disk. Disk Access Time can be

expressed as following equation.

(eq. 1) Taccess = Toverhead + Tseek + Tlatency + Txfer.

Without respect to the amount of required I/O by one request, Toverhead, Tseek and Tlatency is uniform. How many data can be transmitted during the given Taccess estimates the capability of disk drive and it can be expressed as following equation[5].

(eq. 2) Performance of Disk Drive = Byte transferred/Taccess

In other words, the performance of the recent disk drive is greatly improved, so when the larger amount of transmission according to the characteristics of the disk is requested, it will be more effective. By finding this most effective disk fetch or de-stage size and using it as a disk cache line,[8] cache hit-ratio can become higher by increasing the spatial locality in the application where a requested amount of data transmission is large.

6. Conclusion

In this paper, the design and implementation of Disk Cache Manager is presented as a functional server in the parallel operating system operating on the Misix3 micro-kernel. To maximize cache efficiency, first of all, the optimized unit of the physical I/O is defined as a cache line and 2 step cache management method, which makes one line as the minimum block unit required by the operating system, proposed. Also, by separating the device mounted on IO node into 2 logical and private devices, parallel I/O to the Even/Odd cache line is enabled through distributor of file systems.

By providing an alternative path against an IO node breakdown, I/O performance is improved and becomes fault-tolerable. On the basis of this design, the effect of DCM is verified by implementing the test-bed system.

In the future, this research will go on the mod(n) logical disks by connecting n IO nodes and m disks by expanding the problem of the shared disk connected to two IO nodes.

ACKNOWLEDGMENTS

This work was supported for two years by Pusan National University Research Grant

REFERENCES

[1] P. Ramya, K. Mahmut, J. Myoungsoo, Disk-Cache and Parallelism Aware I/O Scheduling to Improve Storage System Performance., 2013 IEEE 27th International Symposium on Parallel & Distributed Processing, p357-368, 2013

[2] Shi, Xiaodong, Feng, Dan, PCAR: Parallelism Based Cache Replacement Scheme to Exploit Inter-disks Parallelism and Intra-disk Spatial Locality in Parallel Disk Array. 2011 Fourth International Symposium on PAAP); p218-222, 2011

[3] Xiaoning Ding, Song Jiang, Feng Chen, A Buffer Cache Management Scheme Exploiting Both Temporal and Spatial Localities. ACM Transactions on Storage, Vol. 3 Issue 2, p1-27, 2007

[4] L. Cai and Y. Lu. Power reduction of multiple disks using dynamic cache resizing and speed control. In Proc. Of ISLPED'06, 2006.

[5] Arfan, Abdul, Kim, Young-Jin, Kwon, Jin Baek, Access time-aware cache algorithm for SATA hard disks, IEICE Electronics Express. 9(21):1707-1713, 2012

[6] Deng, Yuhui, Zhou, Jipeng, Meng, Xiaohua, Deconstructing on-board disk cache by using block-level real traces, Simulation Modelling Practice & Theory. Vol. 20 Issue 1, p33-45. 2012

[7] Chakraborty, Abhirup, Singh, Ajit, Cost-aware caching schemes in heterogeneous storage systems. Journal of Supercomputing. Vol. 56 Issue 1, p56-78. 23p, 2011

[8] "MINIXReleases".wiki.minix3.org. Retrieved 29 February 2012.

최 욱 진(Choi Wook Jin)

- 2012년 2월 : 국민대학교 전자과(공학사)
- 2012 ~ 현재 : 부산대학교 IT응용공학과 석사과정
- 2012년 8월 ~ 2013년 2월 : ETRI 보바일보안 연구팀 위촉연구원nt in
- 관심분야 : 임베디드 시스템, 실시간운영체제 및 모바일보안)
- E-Mail : wtonic@pnu.edu

강 용 호(Kang Yong Ho)

- 1994년 2월: 충남대학교 컴퓨터공학(공학사)
- 1997년 2월 :충남대학교 컴퓨터공학(공학석사)
- 2000년 2월 :충남대학교 공학박사 수료
- 2012년 3월 ~ 현재 : 부산대 IT응용공학과 박사과정
- 관심분야 : 클러스터 컴퓨팅, 모바일 클라우드컴퓨팅, 모바일 가상화 및 보안
- E-Mail : kang@r2soft.co.kr,

김 선 종(Kim, Seong Jong)

- 1989년 2월 : 경북대학교 전자공학(공학사)
- 1991년 2월 :경북대학교 전자공학(공학석사)
- 1996년 2월 :경북대학교 전자공학(공학박사)
- 1997년 3월~ 현재 : 부산대학교 IT응용공학과 교수
- 관심분야 : 디지털 시스템 설계, 이미지 처리, 팬턴인식, 스마트장치 응용
- E-Mail : ksj329@pusan.ac.kr,

권 혁 숭(Kwon, Hyeog Soong)

- 1985년 2월 : 영남대학교 전자공학(공학사)
- 1987년 2월 :영남대학교 전자공학(공학석사)
- 1995년 2월 :영남대학교 전자공학(공학박사)
- 2011년 3월 ~ 2012년 2월 : 미국 조지아공대 전자및컴퓨터공학 객원교수
- 1996년 3월 ~ 현재 : 부산대학교 IT응용공학과 교수
- 관심분야 : 디지털 필터 설계, 바이오 의료기기, 바이오 신호처리 및 이동통신
- E-Mail : hskwon@pnu.edu,

김 주 만(Kim, Joo Man)

- 1984년 2월 : 숭실대학교 전자계산학(공학사)
- 1998년 8월 :충남대학교 컴퓨터공학(공학사)
- 2003년 2월 :충남대학교 전자공학(공학박사)
- 1985년 1월 ~ 2000년 2월 : ETRI OS팀장(책임연구원)
- 1995년 7월 ~ 1996년 6월 :미국 Novell사 객원연구원
- 2000년 3월 ~ 현재 : 부산대학교 IT응용공학과 교수
- 관심분야 : 임베디드 시스템, 실시간 시스템, 클러스터 컴퓨팅, 병렬분산 시스템
- E-Mail : joomkim@pnu.edu