하이브리드 하드디스크를 위한 효율적인 선반입 기법

김정원*

Effecient Prefetching Scheme for Hybrid Hard Disk

Jeong-Won Kim*

요 약

하이브리드 하드디스크(Hybrid hard disk drive: H-HDD)가 SSD(Solid state drive)에 비해 경쟁력을 갖기 위해서는 저전력, 읽기 속도가 핵심 요소이다. 본 연구에서는 H-HDD에 장착되어 있는 비휘발성 메모리에 디스크 블록을 선반입하여 저전력과 응답시간을 향상시킬 수 있는 기법을 제안한다. 제안하는 기법의 핵심은 시스템파일이나 자주 사용되는 파일은 파일단위로 캐싱하고 나머지는 블록단위로 선반입한다. 선반입은 디스크 큐를 서비스하고 남은 여유 시간에 우선순위가 높은 블록부터 실행되며 이때 사용되는 우선순위는 시간적, 지역적 지역성을 동시에 고려하여 결정된다. 실험 결과 제안 기법은 기존 기법에 비해 전력소모가 낮고 응답시간이 향상되었음을 확인하였다.

ABSTRACT

The Competitiveness of Hybrid hard disk drive(H-HDD) for solid state disk(SSD) comes from both lower power consumption and higher reading speed. This paper suggests a prefetching scheme that can improve the performance of Non-Volatile cache(NVCache) memory installed on the H-HDD through prefetching disk blocks as well as files to the NVCache. The proposed scheme makes the highly used data such as booting files copy to the NVCache as an unit of file and the frequently accessed blocks copy to the NVCache. This prefetching is done on the idle time of disk queue and the priorities of prefetched target blocks are based on both time and spatial locality of blocks. Experiments results show that the suggested method can improve response time of H-HDD and also lower the power consumption.

키워드

hybrid hard disk, prefetching, cache, operating system
하이브리드 하드디스크, 선반입, 캐시, 운영체제

Ⅰ. 서 론

최근의 낸드 플래시 메모리는 전력소모가 적고, 이동성과 랜덤 액세스 성능이 우수하여 개인용 컴퓨터를 비롯한 다양한 가전제품의 보조 기억장치로 사용되고 있다. 특히 대용량의 낸드 플래시 메모리를 내장하고 있는 SSD는 하드디스크를 대체할 보조기억장치로 급부상하고 있다. 그러나 SSD는 HDD에 비해 가격 및 쓰기 측면에서 경쟁력이 낮기 때문에 하드디스크를 대체하기에는 아직 시간이 소요될 것으로 보인다.

이 SSD와 HDD의 장점을 상호 보완하는 기술들이

* 신라대학교 컴퓨터공학과(jwkim@silla.ac.kr)
접수일자 : 2011. 08. 19 심사(수정)일자 : 2011. 09. 21 게재확정일자 : 2011. 10. 12

등장하고 있는데 윈도우의 레디부스트(Ready Boost), 인텔의 터보 메모리(Turbo Memory), 그리고 하이브리드 하드디스크(H-HDD: Hybrid HDD)이다[1].

이중 H-HDD는 하드디스크 내에 플래시 메모리를 장착하여 메인메모리와 물리디스크 간의 캐시역할을 수행하여 입출력 기능을 향상시키는 기술로 삼성, 시게이트 등 제조사들이 관련 제품을 출시하고 있다. H-HDD는 플래시 메모리를 장착하고 있으므로 호스트의 요구 페이지가 플래시 메모리에서 히트되면 하드디스크의 스핀들 모터를 동작시킬 필요가 없으므로 배터리를 획기적으로 감소시킬 수 있다. 이것은 컴퓨터시스템에서 하드디스크가 전력의 20%를 차지하고 이중 스핀들 모터에서 90% 정도 발생한다는 점을 감안한다면 SSD가 HDD를 완전히 대체하기 전까지는 H-HDD의 유용성이 높을 것으로 보인다[2].

본 논문과 관련된 연구로 플래시 메모리를 캐시메모리로 사용하는 기법을 소개한다. [3]에서는 SLC와 MLC를 동시에 고려한 플래시 메모리를 위한 FTL을 제시하였는데 자주 접근되는 데이터는 SLC영역에 자주 접근되지 않는 데이터는 MLC영역에 저장하는 기법이다. SLC의 속도가 MLC에 비하여 빠르고, MLC의 생명이 SLC에 비해 짧은 점을 이용한 캐싱전략이라고 볼 수 있다. [4]는 로그와 데이터를 구분하여 저장하는 방법으로 로그는 SLC에 데이터는 MLC에 저장하여 로그가 신속하게 처리되도록 하는 FTL 기법을 제안하였다. [5]는 데이터 고정 기법을 사용하는 하이브리드 하드디스크 구조를 제안하였다. 소비전력과 입출력 성능을 향상시키기 위해서 자주 사용되는 데이터 블록은 고정 구역, 즉 낸드 플래시 메모리에 더 오래 머물게 하는 기법이다.

[6]에서는 SLC와 MLC를 동시에 장착한 하이브리드 하드디스크 구조를 제안하였는데 SLC를 MLC의 상위 캐시로 하는 구조로 SLC와 MLC의 용량 변화로 전력과 수명을 향상시키고자 하는 기법이다. [7]에서는 H-HDD의 디스크에서 플래시메모리로 n-블록을 선반입하여 하드디스크의 스핀업 횟수를 줄이고 호스트로부터 읽기 요청에 효율적으로 서비스하는 기법을 제안하였다. 이 n-블록이란 하드디스크의 스핀업을 요구하지 않을 정도의 충분한 블록을 선반입한다는 것이다. [8]에서는 H-HDD에 장착된 비휘발성 캐시의 경우 쓰기 요청이 집중되면 플래시메모리에 병목현상이 생기므로 이를 해결하기 위하여 읽기의 참조 빈도는 낮고 쓰기의 갱신 빈도가 높은 데이터 블록들을 교체하는 LFU-Hot 기법을 제시하였고 교체될 데이터 블록을 재배치하여 자기디스크로 플러싱하는 기법을 제안하였다.

이러한 H-HDD를 위한 연구들이 다수 제안되었는데 기존의 연구들이 주로 하드디스크의 스핀업 시간을 줄이기 위해 선반입을 하거나 다양한 캐싱 전략을 도입하였다. 그러나 H-HDD 내의 플래시메모리는 용량이 제한적이므로 읽기 요구 중에 스핀들 모터를 동작시키는 경우가 많이 발생하고 이것은 결국 사용자로 하여금 번거로움을 초래할 수 있다. 스핀들 모터가 동작 중에는 추가의 전력 소모가 필요 없으나 스핀다운에서 스핀업될 때는 모터가 다시 동작하므로 상당한 전력소모를 필요로 한다. 이 스핀다운/업 회수를 최소화하기 위해서는 호스트에서 요구하는 페이지가 플래시에서 히트되는 확률을 높이면 되는데 본 연구에서는 운영체제가 요구하는 시스템 데이터, 그리고 사용자가 특정 기간 동안 자주 사용하는 데이터는 파일 단위로 NVCache에 선반입하고 그렇지 않은 데이터는 페이지 단위로 선반입시켜서 스핀들 모터의 스핀다운/업 횟수를 감소시키고자 한다.

II. 접근 패턴에 기반한 선반입 기법

일반적으로 하드디스크는 전원이 인가된 후 스핀업(spinup), 활성상태(Active), 유휴상태(Idle), 대기상태(Standby), 휴면상태(Sleep)를 반복한다. 이 대기상태와 휴면상태는 스핀들(spindle)이 회전하지 않는 상태로 다시 스핀업하기 위해서는 상당량의 전력을 필요로 한다. Seagate의 Momentus® XT의 경우 활성상태는 평균 2.3W의 전력이 요구되나 스핀업하기 위해서는 최대 5W의 전력과 3초의 대기시간이 요구된다[9]. 따라서 이 스핀업의 횟수를 감소시키는 것이 배터리 운용시간을 증가시킬 수 있다.

따라서 본 논문에서는 자주 접근되는 파일을 파일시스템이 인식하여 플래시 메모리로 복사하여 두고, 자주 사용되는 블록은 플래시 메모리로 선반입을 시켜서 시스템의 응답성을 향상시키고 모바일 기기의 전력소모를 감소시키고자 한다.

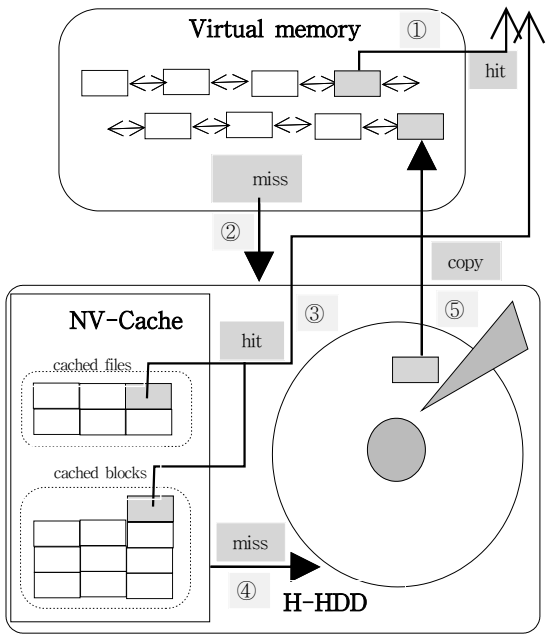

그림 1. 시스템 구조
Fig. 1 System structure

2.1. 입출력 시스템의 구조

그림 1은 H-HDD를 사용하는 시스템의 입출력 구조를 나타내고 있다. 전체적인 흐름은 현대 운영체제의 캐싱 구조와 유사하다. H-HDD는 하드디스크내의 NVCache가 있으므로 물리 디스크에서 페이지를 찾기 전에 NVCache에서 먼저 검색을 하게 된다. 이 NVCache에 페이지가 히트되면 하드디스크의 헤드를 이동시키지 않고 즉각 페이지를 전송하게 된다. 제안 구조의 특이점은 NVCache에 블록 단위 뿐만 아니라 파일단위의 캐싱도 이루어진다는 것이다.

2.2. 선반입 알고리즘

선반입을 위하여 본 연구에서는 장기와 단기로 구분하여 알고리즘을 적용한다. 장기 선반입은 시스템이 동작과 정지가 반복되는 동안 파일의 접근횟수를 파일시스템이 유지하여 접근 빈도가 높은 파일을 NVCache에 선반입시키는데 NVCache의 용량이 상당히 제한적이므로 주로 시스템 파일이나 사용자가 최근 접근 빈도가 높은 파일이 선반입의 대상이 된다.

```
LongTerm_Prefetching()
{
    sort Files_Priorities by Files_Reference_Count;
    if (NVCache_Available > 0) {
        while (MAX_LT_Prefetch > 0) {
            if (MAX_LT_Prefetch - File_Size > 0) {
                copy file to NVCache;
                Max_LT_Prefetch_Size -= File_Size;
            }
        }
        decrease all files' priorities by 1;
    }
    else {
        find a file which has lowest priority;
        remove the file;
        call NVCache_File_Replacement();
    }
}
```

그림 2. 장기 선반입 알고리즘
Fig. 2 Long-term Prefetching Algorithm

그림 2는 장기 선반입 알고리즘이다. 이 LongTerm_Prefetching() 함수는 파일을 우선순위에 의해 정렬하여 접근 빈도가 높은 파일을 NVCache에 선반입한다. NVCache에 여유 공간(NVCache_Available)이 있으면 우선순위가 높은 파일부터 여유 공간이 있을 때까지 복사한다. 만약 여유 공간이 없고 우선순위가 낮은 파일이 NVCache에 존재하면 삭제하여 공간을 확보한 후 우선순위가 높은 파일을 복사한다. 사용은 되지 않고 우선순위를 유지하는 파일을 대체하기 위해서 복사가 일어나면 기존 파일들의 우선순위를 1씩 감소시켜 최근 접근 빈도가 높은 파일이 NVCache에 더 오래 남도록 한다.

그림 3은 단기 선반입 알고리즘이다. 선반입의 목적은 디스크 큐의 서비스 시간과 NVCache의 공간이 여유가 있을 때 블록을 미리 복사하여 시스템의 응답성을 향상시키고 스핀들이 회전하지 않는 상황에서 페이지 요구가 발생했을 때 가능한 스핀업의 횟수를 감소시키고 불가피하게 스핀업을 하더라도 사용자의 대기시간을 감소시키는 것이다. 선반입을 위한 고려요소는 시기, 대상 블록, 그리고 선반입을 위한 최소 개수이다. 선반입의 시기는 디스크큐와 NVCache에 여유가 있을 때이다. 대상 블록은 시간적, 지역적 지역성에 의하며 선반입 최소 개수는 디스크 스핀업 지연시간 동안 페이지 블록이 NVCache에서 가상메모리로 전송할 수 있는 수에 기반한다.

```
ST_Prefetching()
{
    if (disk_Queue_Size-disk_Queue_Request <
            minPrefetchBlks)
        return;
    Blk_Priority =|ceil( Σ_{t=W}^{1} (Weight_t * Acc_t)/MaxPriority)
                        − MaxPriority − 1|
    if (NVCache_Available > 0) {
        while (MAX_Blk_Prefetch > 0) {
            if (MAX_Blk_Prefetch-minPrefetchBlks>0) {
                copy Blocks to NVCache;
                Max_Blk_Prefetch_Size -= minPrefetchBlks;
            }
        }
        decrease all blocks' priorities by 1;
    }
    else {
        select blocks which have lowest priorities;
        flush the blocks;
        call Block_Prefetching();
    }
}
```

그림 3. 단기 선반입 알고리즘
Fig. 3 Short-term Prefetching Algorithm

선반입의 시기를 결정하기 위해서 디스크의 성능을 참고하여 디스크큐의 크기를 결정할 수 있다. 디스크 스케줄링 알고리즘이 SCAN일 경우 디스크 헤드의 full-strock 시간동안 읽을 수 있는 블록의 개수를 디스크 큐의 크기로 정한다. 예를 들면 Seagate의 Momentus® XT의 경우 full-strock은 22 ms가 소요된다[9]. 이 디스크는 평균 전송률이 SATA 인터페이스의 경우 58.4MB/s(4KB 블록)이므로 한 블록을 전송하기위해서 0.067ms가 소요된다. 따라서 디스크큐의 크기는 330블록(22ms/0.067ms ≒ 328.36)으로 지정할 수 있다.

선반입할 최소 블록의 개수는 스핀업 지연시간과 첫 번째 블록으로 디스크헤드의 탐색시간의 합에 의존적이다. 즉 이 스핀업 지연시간+탐색시간의 합 동안 NVCache에서 요구 블록을 읽으면 되는 것이다. 다음 식 (1)은 최소 블록을 구하는 공식이다.

$$\min PreBlks = \frac{SpinUpTime + AverageSeekTime}{TransferRatesPerOneBlk} \quad (1)$$

식 (1)에서 *SpinUpTime* 은 스핀업 지연시간이고 *AverageSeekTime*은 디스크 헤드의 평균탐색시간이다. 그리고 *TransferRatesPerOneBlk*는 한 블록을 메모리로 전송하는 데 걸리는 시간이다. Seagate Momentus® XT의 경우 스핀업 지연시간은 3sec이고, 평균 탐색시간은 11ms이며, *TransferRatesPerOneBlk*은 평균 0.067ms이므로 *minPreBlks*는 44,940개 이다.

만약 디스크가 대기상태나 휴면상태가 아니고 유휴상태인 경우는 디스크의 지연시간이 full-strock에 의존하므로 식 (2)를 적용한다. Seagate Momentus® XT의 경우 full-Strock이 22ms 이므로 *minPreBlks*는 약 328개의 블록이다.

$$\min PreBlks = \frac{full - StrockSeekTime}{TransferRatesPerOneBlk} \quad (2)$$

그림 3의 ShortTerm_Prefetching() 함수에서 보듯이 디스크큐의 여유 슬롯이 존재하면 선반입이 발생된다. 선반입 대상 블록을 결정하기 위하여 블록의 우선순위를 결정하는데 공간적 지역성과 시간적 지역성을 고려한다. 지역성은 최대 관찰 시간, 즉 윈도우(W)를 정하고 1에서 W까지의 시간($1 \le t \le W$)동안 참조회수(공간적 지역성 : Acc_t)를 누적한다. 윈도우의 각 슬롯별로 가중치를 두는데 가중치는 다음과 같이 최근 참조의 경우 가중치 값이 높은 방식으로 다음의 식 (3)과 같다.

$$Weight_t = 1 - \frac{t}{W} \quad (3)$$

이 식에서 W값은 블록 참조 패턴의 시간별 최대 이력을 정하는 것으로 클수록 보다 정확한 이력을 반영할 수 있으나 복잡도를 고려하여 적절한 값을 정한다. 참조회수에 이 가중치를 모든 윈도우에 대하여 곱하게 되면 참조 빈도가 높은 블록일수록 높은 값이 산출된다. 이 값을 식 (4)와 같이 최대 우선순위(*MaxPriority*) 값으로 나누어 참조 빈도가 높은 블록이 최대 우선순위의 정수 값이 부여된다.

$$|ceil(\sum_{t=W}^{1}(Weight_t * Acc_t)/MaxPriority) - MaxPriority - 1| \quad (4)$$

III. 실험 결과

제안하는 선반입 기법의 성능을 측정하기 위해 H-HDD 시뮬레이터를 작성하여 실험을 진행하였다. 실험의 목적은 시뮬레이터를 통하여 기존 하드디스크에 대한 성능 향상, 선반입을 통한 응답시간의 향상, 그리고 효율적인 전력의 사용을 확인하는 것이다.

실험 환경은 윈도우7에서 매트랩 및 시뮬링크를 사용하여 H-HDD 시뮬레이터를 구성하였다. 보다 정확한 실험을 위해서는 실제 H-HDD의 펌웨어를 재구성해야 하는데 이것은 현실적으로 곤란하므로 시게이트사의 H-HDD인 Momentus® XT(ST93205620AS)의 사양을 사용하여 프로그래밍하였다. 아래의 표 1은 실험에서 사용된 Seagate Momentus® XT(ST93205620AS)의 사양이다.

표 1. H-HDD 디스크 사양
Table 1. Disk Specification

Specification	ST93205620AS
Capacity	320GB
Cache	32MB
NVCache	4GB
Stanby to Ready	3 Sec
Average seek	11ms(read), 13ms(write)
Transfer rates	1.23Gb/s(internal), 3.0Gb/s(I/O)

3.1. 선반입 크기에 따른 응답 시간

그림 4는 선반입의 크기에 따른 각 주기의 평균 응답시간을 나타낸다.

그림에서 선반입을 하지 않는 경우인 'Noprefetch'는 각 주기의 응답시간이 선반입을 하는 경우보다 평균 15% 이상의 응답시간을 보이고 있다. 그림에서 'prefetch'는 논문에서 제안하는 수식에 의하여 선반입을 하는 경우 이고 'prefetch+n'은 선반입의 크기에 따른 성능 비교를 위해 n 블록을 추가하는 경우이다. 그림에서 보듯이 선반입 블록의 수를 증가하더라도 응답시간이 개선되지 않는데 이것은 선반입을 하기위해 추가의 디스크 헤드 이동이 소요되고 이것이 전체 응답시간을 증가시키기 때문이다. 그림의 전체적 패턴을 보면 초기의 응답 시간이 다소 높은데 이것은 NVCache의 히트율이 낮기 때문이고 주기가 지날수록 응답 시간이 수렴해지고 선반입의 개수에 관계없이 응답시간이 수렴하는 것을 볼 수 있다. 따라서 논문에서 제안하는 수식에 의해 선반입 수를 최적화하면 H-HDD의 응답시간을 최대로 향상시킬 수 있음을 알 수 있다.

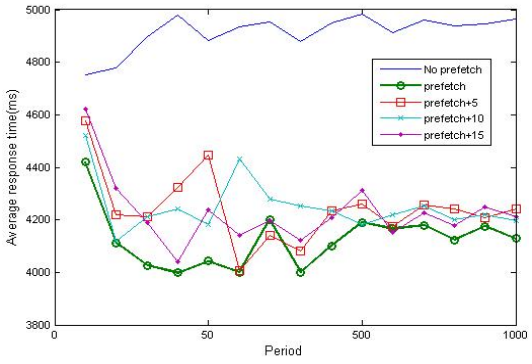

그림 4. 선반입 크기에 따른 응답시간
Fig. 4 Response time based on Prefetching Size

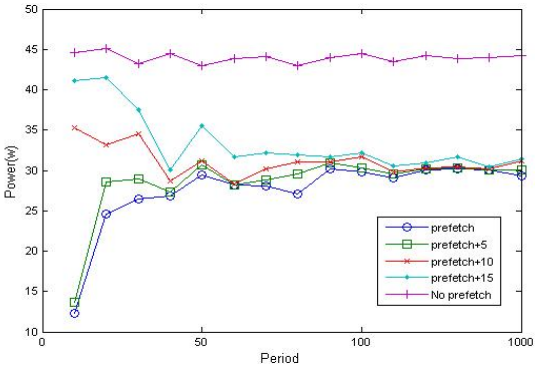

그림 5. 선반입 크기에 따른 전력 소모
Fig. 5 Power Consumptrion based on Prefetching Size

3.2 선반입 크기에 따른 전력 소모

그림 5는 선반입의 크기에 따른 전력소모를 나타내고 있는데 본 실험에서는 전력 소모를 각 주기별로 측정하여 성능을 평가하였다. 선반입을 하지 않는 경

우의 전력 소모는 요구블록을 서비스하는 데 소모되는 양과 유휴상태시 소모되는 양의 합으로 나타낼 수 있다(식 5). 각 주기의 요구블록 수는 $ReqBlks$이고 탐색시 전력소모는 $Seek_w$, 읽기 모드시 전력소모는 $read_w$이다. 디스크 큐의 크기는 $SizeOfQueue$이고 유휴상태의 전력 소모는 $Idle_w$이다.

$$ReqBlks*(seek_w + read_w) + (SizeOfQueue - ReqBlks)*Idle_w \quad (5)$$

선반입을 할 경우 전력소모는 디스크에서 소모되는 양과 NVCache에서 소모되는 양, 그리고 유휴상태시 소모되는 양의 합으로 나타낼 수 있다(식 6). 식 (6)에서 $Hits$는 NVCache에서 히트된 블록의 수를 나타내고 $NVCache_w$는 NVCache에서 읽을 때 전력 소모량을 나타낸다. 수식에서 보듯이 NVCache에서 히트가 많이 될수록 전체 전력소모량은 감소될 수 있다.

$$(ReqBlks - Hits)*(seek_w + read_w) + Hits*NVCache_w + (SizeOfQueue - ReqBlks)*Idle_w \quad (6)$$

그림 5에서 보듯이 선반입 하지 않는 경우 평균 30%이상의 성능 저하를 보이고 있다. 또한 선반입을 많이 할수록 성능이 월등히 개선되고 있지 않은 데 이것은 디스크 스핀들 모터의 동작과 디스크 헤드의 움직임이 상대적으로 증가했기 때문이다.

Ⅳ. 결 론

본 논문에서는 최근 다시 주목받고 있는 H-HDD에 적합한 선반입 기법을 제시하였다. 제안하는 기법에서는 디스크 서비스를 주기로 나누고 운영체제가 요구한 블록을 서비스하고 남은 여유 시간에 공간적, 지역적 우선순위를 고려하여 디스크내에 있는 NVCache에 파일 및 블록을 선반입하는 방법이다. 여유 시간에 선반입을 하게 되므로 서비스의 끊김현상이 감소하고 HDD의 스핀업시 소요되는 전력소모를 감소시킬 수 있다. 향후연구로는 NVCache에 제한적이나마 쓰기 연산을 지원하여 전력 소비를 더욱 효율적으로 하고 NVCache에 적합한 블록 재배치기법을 통하여 NVCache의 수명을 최대화하는 것이다.

참고 문헌

[1] R.Panabaker, “Hybrid hard disk & ReadyDrive technology: improving performance and power for Windows Vista mobile PCs”, Proc. of Microsoft WinHEC, 2006.

[2] Hong-jae Lee, “Toward Understanding Hard Disk”, Electronic Times, Apr., 2003.

[3] Park S.H, “A Mixed Flash Translation Layer Structure for SLC-MLC Combined Flash Memory System”, International Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability, pp. 56-63, 2008.

[4] S. Im and D. Shin, "Storage Architecture and Software Support for SLC/MLC Combined Flash Memory", Proc. of 24th ACM Symposium on Applied Computing, pp. 23-30, 2009.

[5] Y.J. Kim et al, "I/O Performace Optimization Techniques for Hybrid Disk-Based Mobile Consumer Devices", IEEE Transactions on Consumer Electronics, Vol. 53, No. 4, pp. 35-42, 2007.

[6] 홍성철, 신동군, “SLC/MLC 혼합 플래시 메모리를 이용한 하이브리드 하드디스크 설계”, 정보과학회논문지, 제16권, 제7호, pp. 789-793, 2010.

[7] 양준식, 고영욱, 이찬근, 김덕환, “저전력과 입출력 성능이 향상된 n-블록 선반입 기반의 하이브리드 하드디스크 입출력시스템 설계 및 구현”, 정보과학회논문지: 시스템 및 이론, 제36권, 제6호, pp. 451-462, 2009.

[8] 박광희, 이근형, 김덕환, “하이브리드 하드디스크를 위한 효율적인 데이터 블록 교체 및 재배치 기법”, 정보과학회논문지: 컴퓨팅의 실제 및 레터, 제16권, 제1호, pp. 1-10, 2010.

[9] http://www.seagate.com.

저자 소개

김정원(Jeong-Won Kim)

1995년 부산대학교 전자계산학과 (학사)
1997년 부산대학교 대학원 전자계산학과 (석사)
2000년 부산대학교 대학원 전자계산학과 (박사)
2000년~2001년 기술신용보증기금 기술평가역(차장)
2002년~현재 신라대학교 컴퓨터공학과 교수
※ 관심분야 : USN, 임베디드시스템, 모바일 시스템