韓國컴퓨터情報學會 論文誌
第16卷 第10號, 2011. 10.

2011-16-10-1-3

저전력과 응답시간 향상을 위한 하이브리드 하드디스크의 입출력 기법

김 정 원*

I/O Scheme of Hybrid Hard Disk Drive for Low Power Consumption and Effective Response Time

Jeong-Won Kim*

요 약

최근 전력소모와 읽기 성능이 우수한 Solid state disk(SSD)가 많이 사용되고 있으나 가격이 고가이고 삭제 및 쓰기 연산의 효율이 낮은 것이 단점이다. 이것을 보완하기 위한 저장장치의 일종이 하이브리드 하드디스크(H-HDD: Hybrid Hard disk drive)인데 하드디스크 내부에 플래시 메모리(NVCache: Non-volatile Cache)를 장착하여 디스크블록의 캐시로 사용한다. 본 논문에서는 H-HDD의 저전력과 응답시간을 향상시키기 위해 NVCache의 선반입 및 관리 기법을 제안한다. 제안하는 기법은 NVCache를 읽기 캐시를 위주로 사용하고 쓰기캐시는 디스크 헤드와 스핀들의 상황에 따라 쓰기 연산을 지원한다. 읽기 캐시의 경우 시간적, 지역적 지역성을 동시에 고려하여 선반입을 통해 응답시간과 전력 소모를 감소시키고 쓰기 캐시의 경우 디스크 스핀들의 동작 상태에 따라 NVCache에 쓰기를 실시하여 저전력과 응답성을 향상시키고자한다.

▸ Keyword : 하이브리드 하드디스크, 선반입, 캐싱, 운영체제

Abstract

Recently, Solid state disk is mainly used because this device has lower power consumption as well as higher response time. But it features higher price and lower performance at delete and write operations compared with HDD. To compensate this defect, Hybrid hard disk with internal non-volatile flash memory was issued. This NVCache is used as a kind of cache for disk blocks. In this paper, an I/O scheme for H-HDD is proposed for improving low power consumption as well as response time. Our method is to use this NVCache as read cache mainly and write cache when

• 제1저자 : 김정원
• 투고일 : 2011. 04. 13, 심사일 : 2011. 05. 18, 게재확정일 : 2011. 08. 08
* 신라대학교 컴퓨터공학과(Dept. of Computer Science, Silla University)

write requests are concentrated. In read cache operation, disk blocks with higher priority determined on basis of time as well as spatial localities are prefetched, which can improve response time. The write operation is conducted only at write peak time as disk spindle up costs higher battery power as well as response time. Experiments results show that the suggested method can improve response time of H-HDD and lower the power consumption.

▸ Keyword : Hybrid hard disk, prefetching, caching, operating system

I. 서 론

컴퓨터시스템에서 하드디스크의 기술은 발전하고 있으나 CPU, 메모리와의 속도 차이로 인하여 주요한 병목현상을 초래하고 있다. 이를 해결하기 위하여 플래시 메모리를 병렬로 연결하여 하드디스크를 대체하고 있는 저장장치가 SSD(Solid state drive)이다. 이 플래시 메모리는 전력소모가 적고, 랜덤 접근 성능이 우수하며 충격에 강한 장점을 가지고 있는 반면 가격이 하드디스크에 비해 월등히 고가이고, 삭제하기 전에는 기록할 수 없으며 읽기, 쓰기 및 삭제 연산 값이 다르다는 특징이 있다. SSD에 사용되는 낸드 플래시 메모리에는 SLC(Single level cell), MLC(Multi level cell) 타입이 있는데 SLC는 속도가 빠르고 수명이 길지만 가격이 비싸며, MLC는 속도는 느리고 수명은 짧지만 용량이 크고 가격이 저렴한 특징이 있다.

한편, SSD와 HDD의 장점을 상호 보완하는 기술들이 등장하고 있는데 윈도우의 레디부스트(Ready boost), 인텔의 터보 메모리(Turbo memory), 그리고 하이브리드 하드디스크 등이다. 윈도우 비스타와 윈도우 7은 레디부스트 기능을 기본적으로 탑재하고 있는데, 이것은 일반적인 하드디스크를 사용하고 있는 PC에 플래시 메모리를 설치하여 이를 시스템 데이터용으로 사용하여 성능을 높이는 기능이다[1]. 또한 인텔의 터보 메모리는 PC의 메인보드에 플래시 메모리를 탑재해서 이를 시스템 데이터 저장용으로 사용하는 방식이다. 그리고 H-HDD는 하드디스크 내에 플래시 메모리를 장착하여 메인메모리와 물리디스크 간의 캐시역할을 수행하여 입출력 기능을 향상시키는 기술이다.

이 H-HDD는 하드디스크의 스핀들 모터의 동작을 최소화하여 전력소비를 감소시킬 수 있고 높은 접근 속도를 제공하여 모바일 환경에 적합한 기능을 제공할 수 있다. 따라서 하드디스크를 장착한 모바일용 컴퓨터용 제품, 즉 노트북, PDA, PMP 플레이어 등의 제품에 사용 가능성이 높은데 이러한 모바일 기기들은 배터리 의존도가 데스크톱 컴퓨터들 보다 높으므로 저전력 기술을 반드시 필요로 하기 때문이다. 또한, H-HDD는 플래시 메모리를 장착하고 있으므로 호스트의 요구 페이지가 플래시 메모리에서 히트되면 하드디스크의 스핀들 모터를 동작시킬 필요가 없으므로 배터리를 획기적으로 줄일 수 있다. 이것은 컴퓨터시스템에서 하드디스크가 전력의 20%를 차지하고 이중 스핀들 모터에서 90% 정도 발생한다는 점을 감안한다면[2] SSD가 HDD를 완전히 대체하기 전 H-HDD의 저전력 기술이 반드시 요구된다.

이러한 H-HDD를 위한 연구들이 다수 제안되고 있는데 [3]에서는 하드디스크의 시동시간과 플래시 메모리의 남은 용량 등의 정보를 이용하여 계산된 수만큼의 블록을 미리 선반입시켜서 전력소모를 줄이고자 했으며, [4]에서는 낸드 플래시 메모리를 SLC 뿐만 아니라 MLC도 통합하고 SLC를 MLC의 상위캐시로 사용하여 데이터가 플래시 메모리에 더 오래 머물 수 있도록 하여 전력소모를 줄이고자 하였다.

기존의 연구들이 주로 하드디스크의 스핀업 시간을 줄이기 위해 선반입을 하거나 다양한 캐싱 전략을 도입하였는데 현실적으로 H-HDD 내의 플래시메모리는 용량이 제한적이므로 읽기 요구 중에 스핀들 모터를 동작시키는 경우가 많이 발생하고 이것을 결국 사용자로 하여금 번거로움을 초래할 수 있다. 또한 일부 연구들이 NVCache를 쓰기용도로도 사용하고 있는데 플래시메모리의 쓰기 연산이 하드디스크에 비해 비효율적이고 쓰기 요구가 집중될 경우 NVCache가 병목현상을 초래할 수 도 있다. 따라서 본 연구에서는 플래시메모리에서 페이지를 읽는 중에 스핀들 모터를 동작시키는 횟수를 감소시키면서 H-HDD의 읽기 및 저전력 성능을 향상 시키고 쓰기 연산은 스핀들 모터가 유휴상태나 휴면상태일 경우만 동작하도록 하여 응답시간과 전력소모를 향상시키는 기법을 제안하고자 한다.

HDD의 스핀들 모터가 동작 중에는 추가의 전력 소모가 필요 없으나 스핀다운에서 스핀업될 때는 모터가 다시 동작하므로 상당한 전력소모를 필요로 하므로 스핀다운/업의 횟수를 줄이는 것이 모바일 응용을 위한 H-HDD의 핵심 기술 중의 하나이다. 이 스핀다운/업 회수를 최소화하기 위해서는 호스

트에서 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 NVCache에 선반입시켜서 스핀들 모터의 스핀다운/업 횟수를 감소시키고자 한다. 또한 스핀들 모터가 동작모드(Active mode)일 경우는 하드디스크에 쓰기 연산을 수행하고 유휴상태이거나 휴면 상태일 경우는 NVCache에 쓰기 연산을 수행하면 응용프로그램에게 높은 응답성뿐만 아니라 저전력성을 제공할 수 있을 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 H-HDD의 구조 및 다양한 캐싱기법을 소개하고 본 연구와의 관련성을 설명한다. 3장에서는 제안하는 선반입 기법을 설명하며 4장에서는 제안 기법의 성능 평가 결과를 소개하며 5장에서는 본 논문의 결론을 맺는다.

II. 관련 연구

그림 1은 H-HDD의 구조를 나타내고 있는데 일반적인 하드디스크의 구조와 비슷하나 NVCache가 있는 것이 주요한 차이점이다. H-HDD는 디스크의 신속한 접근을 위하여 자주 사용되는 데이터는 NVCache에 저장하고 접근 빈도가 낮은 데이터는 자기 디스크에 저장하는 장치로 NVCache에 히트되면 디스크의 스핀들 모터를 스핀다운시킬 수 있어 전력 소모를 감소시킬 수 있는 장점이 있다. 그러나 플래시 메모리의 추가로 일반 하드디스크보다 가격 경쟁력은 낮아질 수밖에 없다. 삼성전자의 HM16HJI 등 몇 년 전 H-HDD가 새로운 저장장치로 등장하였다가 현재는 HDD와 SSD가 대세를 이루고 있지만 최근 시게이트의 4GB의 SLC 낸드 메모리를 장착한 Momentus® XT의 등장으로 다시 주목 받고 있다.

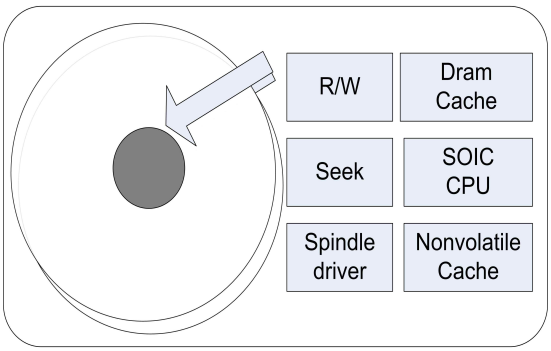

그림 1. H-HDD의 내부 구조
Fig. 1. Structure of H-HDD

다음은 기존의 선반입 기법에 대하여 간략히 소개한다. 리눅스는 연속된 위치의 블록을 미리 메모리로 읽어 선반입을 하는 RA(Read ahead)알고리즘을 사용하는데 이것은 지역성에 대한 힌트가 없는 경우이고 또한 공간적 지역성만을 고려하는 기법이다[5]. [6]에서는 AMP(Adaptive multistream prefetching)라는 선반입 기법을 제안하였는데 여러 어플리케이션이 병렬적으로 블록을 요청할 경우 순차성이 변하는 점에 착안하여 등급과 트리거 디스턴스 인자를 각 블록마다 기록하여 동적으로 선반입 블록을 결정하는 방법이다. [7]에서는 EXCES(External caching in energy saving storage system)라는 선반입 기법을 제안하였다. 여기에서는 하드디스크의 스핀다운/업하는 기계적 특성에 의한 전력소모를 감소하기 위해 하드디스크의 버퍼캐시로서 플래시메모리를 시스템에 장착하고 하드디스크의 동작은 최소화하고 플래시 메모리의 서비스 횟수를 증가시키는 방법이다. H-HDD가 디스크 내에 플래시 메모리를 가지고 있는 반면 이 기법은 디스크 외부에 플래시메모리를 가지고 있는 점이 다르다.

다음은 H-HDD를 위해 제안된 캐싱기법을 소개한다. [8]은 SLC/MLC를 동시에 장착한 하이브리드 하드디스크 구조를 제안하였는데 SLC를 MLC의 상위 캐시로 하는 구조로 SLC와 MLC의 용량 변화로 전력과 수명을 향상시키고자 하였다. 이 기법은 SLC를 MLC의 상위 캐시로 둠으로서 페이지 복사의 오버헤드가 높아 응답시간의 감소를 초래할 수 있다. [9]에서는 H-HDD내에서 자기디스크로부터 플래시메모리로 n-블록 선반입하여 하드디스크의 스핀업 횟수를 줄이고 호스트로부터 읽기 요청에 효율적으로 서비스하는 기법을 제안하였다. n-블록이란 하드디스크의 스핀업을 요구하지 않을 정도의 충분한 블록을 선반입한다는 것이다. 이 기법은 읽기 실패가 읽어나면 n-블록을 선반입하는데 서비스의 끊김현상이 발생할 것으로 보인다. [10]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 블록 교체 기법을 제시하였고 교체될 데이터 블록을 재배치하여 자기디스크로 플러싱하는 기법을 제안하였다. 위의 기법들은 블록 단위의 캐싱 기법으로서 H-HDD의 NVCache에서 읽기 요구 중 페이지가 존재하지 않고 스핀들 모터가 다운 상태이면 스핀들 모터를 다시 동작시키기 위해서는 상당한 전력소모와 서비스의 끊김 현상이 발생하고 이러한 경우가 빈번하게 발생된다면 상당한 성능 저하가 예상된다.

III. NVCache를 위한 선반입 기법

하드디스크는 전원이 인가된 후 스핀업(Spinup), 활성상태(Active), 유휴상태(Idle), 대기상태(Standby), 휴면상태(Sleep) 상태를 반복한다. 이중 대기상태와 휴면상태는 스핀들(spindle)이 회전하지 않는 상태로 다시 스핀업하기 위해서는 상당량의 전력을 필요로 한다. Seagate의 Momentus® XT의 경우 활성상태에는 평균 2.3W의 전력이 요구되나 스핀업하기 위해서는 최대 5W의 전력과 3초의 대기시간이 요구된다[11, 12]. 따라서 이 스핀업의 횟수를 감소시키는 것이 배터리 운용시간을 증가시킬 수 있다. H-HDD는 자주 사용되는 블록을 이 플래시 메모리에 복사해 두기 때문에 이 NVCache에서 읽기가 히트되면 디스크헤드의 탐색시간이 필요하지 않으므로 시스템의 응답성을 향상시킬 수 있다. 만약 디스크가 대기 또는 휴면 상태라면 전력과 시스템의 응답성은 더욱 향상될 것이다.

한편 플래시 메모리의 삭제 및 쓰기 연산이 하드 디스크에 비해 비효율적이므로 H-HDD에서는 NVCache를 읽기 캐시로 사용하는 경우가 대부분이다. 만약 NVCache에서 쓰기 캐시를 지원한다면 시스템의 전력소모는 감소시킬 수 있으나 빈번한 삭제 및 쓰기 연산으로 인하여 NVCache의 수명이 단축될 수도 있다. 따라서 삭제 및 쓰기 연산을 제한시켜서 전력소모, 수명, 그리고 시스템의 응답성을 동시에 만족시킬 수 있는 방법이 요구된다. 본 연구에서는 이러한 점을 감안하여 H-HDD에 대한 부하가 낮을 때는 HDD에 쓰기 연산을 수행하고 부하가 높거나 H-HDD의 상태가 유휴 또는 대기, 휴면 상태일 경우에만 쓰기 연산을 NVCache에 허용하는 기법을 제안한다. 이 기법은 NVCache에는 빈번한 쓰기 연산을 실행하지 않아 수명을 연장시키고 디스크 헤드가 움직이지 않을 경우만 NVCache에 쓰기 연산을 실시하여 응답성을 향상시키고 전력소모를 감소시킬 수 있다.

3.1 NVCache의 입출력 구조

그림 2는 H-HDD를 사용하는 시스템의 선반입 및 입출력 구조를 나타내고 있다. 전체적인 흐름은 현대 운영체제의 캐싱 구조와 비슷하다. 즉 가상메모리 구조에서 메인 메모리에 요구 페이지가 존재하지 않을 경우 HDD에서 페이지를 읽어온다. 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. 만약 NVCache에 존재하지 않으면 디스크헤드를 탐색하여 필요한 페이지를 가상메모리의 페이지 리스트에 복사하게 된다. 본 연구에서는 일반 운영체제의 메모리 캐시와 같이 하드디스크 블록이 메인 메모리에 캐싱되는 것처럼 NVCache에 즉각 캐싱되는 것이 아니라 사용자의 접근 빈도나 우선순위를 고려하여 NVCache에 선반입하는 구조이다. 이것은 NVCache의 용량이 제한적이고 블록 복사 횟수를 감소시키기 위한 것이다.

제안하는 기법에서는 NVCache에 두 가지 형태의 캐시 데이터를 유지하는데 파일 단위의 캐시와 블록 단위의 캐시이다. 물론 캐싱된 파일도 블록의 집합이므로 파일 단위의 캐시는 파일에 속한 모든 블록이 캐싱되어 있는 형태이다. 파일 단위의 캐시는 시스템 부팅에 필요한 파일이나 최근 사용빈도가 높은 파일들이 그 대상이 된다. 블록 단위의 캐시는 시간적, 공간적 지역성을 고려하여 우선순위가 높은 블록 들이 선반입된다. 또한 블록 단위의 캐시는 사용 빈도가 높은 데이터뿐만 아니라 시스템의 응답성과 전력소모를 감소시키기 위한 목적으로 인접한 블록까지 선반입한다. 본 논문에서는 사용자의 접근패턴을 분석하여 자주 사용되는 파일들과 페이지들을 NVCache에 선반입하여 시스템의 응답성을 향상시키고 전력소모를 감소시키고자 한다.

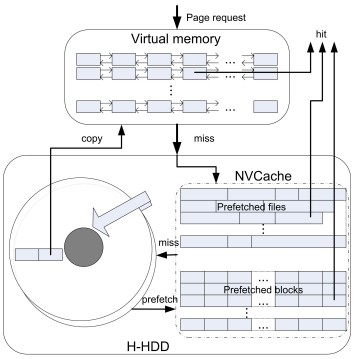

그림 2. H-HDD의 선반입 구조
Fig. 2. Prefetching structure of H-HDD

3.2 NVCache의 읽기 및 선반입 알고리즘

NVCache로의 선반입을 위하여 본 논문에서는 파일단위와 블록단위로 구분하여 알고리즘을 적용한다. 파일단위는 시스템의 부팅과 종료가 반복되는 동안 접근횟수를 파일시스템이 유지하여 접근 빈도가 높은 파일을 NVCache에 선반입을 시키는데 NVCache의 용량이 상당히 제한적이므로 주로 시스템 파일이나 사용자가 최근 접근 빈도가 높은 파일이 선반

입의 대상이 된다. 알고리즘의 동작 시기는 디스크 큐의 서비스 시간이 여유가 있을 때 실시한다. 또한 복사 위치는 플래시 메모리의 수명을 연장하기 위하여 정적 웨어레벨링(static wear leveling)을 적용한다.

```
File_Prefetching()
{
    sort Files_Priorities by Files_Reference_Count;
    if (NVCache_Available > 0) {
        while (MAX_File_Prefetch > 0) {
            if (MAX_File_Prefetch - File_Size > 0) {
                find location by static wear leveling;
                  copy target file to NVCache;
                Max_File_Prefetch_Size -= File_Size;
            }
        }
        decrease all files' priorities by 1;
    }
    else {
        find a file which has lowest priority;
        flush the file;
        call File_Prefetching();
    }
}
```

그림 3. 파일단위 선반입 알고리즘
Fig. 3. File based prefetching algorithm

그림 3은 파일단위 선반입 알고리즘이다. 이 File_Prefetching() 함수는 파일을 접근 빈도에 따라 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 우선순위만 유지하는 파일을 대체하기 위해서 복사가 일어나면 기존 파일들의 우선순위를 1씩 감소시켜 최근 접근 빈도가 높은 파일이 NVCache에 더 오래 남도록 한다.

다음은 블록단위 선반입 알고리즘에 대하여 설명한다. 선반입의 목적은 디스크 큐의 서비스 시간과 NVCache에 여유가 있을 때 블록을 미리 복사하여 시스템의 응답성을 향상시키고 스핀다운 상황에서 페이지 요구가 발생했을 때 가능한 스핀업의 횟수를 감소시키고 불가피하게 스핀업을 하더라도 사용자의 대기시간을 감소시키는 것이다. 선반입을 위한 고려요소는 선반입시기, 대상 블록, 그리고 선반입을 위한 최소 개수이다. 선반입의 시기는 디스크큐와 NVCache에 여유가 있을 때이다. 대상 블록은 시간적, 지역적 지역성에 의하며 선반입 최소 개수는 디스크 스핀다운/업 또는 유휴/활성 지연시간동안 페이지 블록이 NVCache에서 가상메모리로 전송할 수 있는 수에 기반을 둔다.

선반입의 시기를 결정하기 위해서 디스크의 성능을 참고하여 디스크큐의 크기를 결정한다. 디스크 스케줄링 알고리즘이 SCAN일 경우 디스크 헤드의 full-strock 시간동안 읽을 수 있는 블록의 개수를 디스크 큐의 크기로 정한다. 예를 들어 Segate Momentus® XT H-HDD 디스크의 경우 full-strock은 22 ms가 소요된다[12,13]. 이 디스크는 SATA 인터페이스의 경우 평균 전송률이 58.4MB/s(4KB 블록 기준) 이므로 한 블록을 전송하기위해서 0.067ms가 소요된다. 따라서 디스크큐의 크기는 최소한 330블록(22ms/0.067ms ≒ 328.36) 으로 지정할 수 있다.

```
Block_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) {
              find locations by static wear-leveling;
                 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 4. 블록단위 선반입 알고리즘
Fig. 4. Block based prefetching algorithm

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

$$\min Prefetch Blks = \frac{SpinUpTime + AverageSeekTime}{TransferRatesPerOneBlk} \quad \cdots\cdots\cdots\cdots (4)$$

수식 (1)에서 SpinUpTime 은 스핀업 지연시간이고 AverageSeekTime은 디스크 헤드의 평균탐색시간이다. 그리고 TransferRatesPerOneBlk는 한 블록을 메모리로 전송하는 데 걸리는 시간이다. Seagate Momentus® XT의 경우 스핀업 지연시간은 3sec이고, 평균 탐색시간은 11ms이

며, TransferRatesPerOneBlk 은 평균 0.067ms이므로 minPrefetchBlks 는 44,940 개다.

만약 디스크가 대기상태나 휴면상태가 아니고 유휴상태인 경우는 스핀들은 회전중이지만 디스크헤드는 tracking, floating, parked의 세 가지 상태로 구분된다. 이 상태에서 동작상태로의 전환 시 평균탐색시간이 각각 11ms, 15ms, 22ms 일 경우 수식 (2)를 적용하면 minPrefetchBlks는 각각 164, 223, 299개의 블록이다.

$$\min PrefetchBlks = \frac{Idle\,To\,Active\,Seek\,Time}{Transfer\,Rates\,Per\,One\,Blk} \quad \cdots\cdots (2)$$

그림 4는 단기 선반입 알고리즘이다. 디스크큐가 minPrefetchBlks보다 크다면 선반입이 실행된다. 선반입 대상 블록을 결정하기 위하여 블록의 우선순위를 결정하는데 공간적 지역성과 시간적 지역성을 고려한다. 지역성은 최대 관찰 시간의 윈도우, 즉 W를 정하는데 1에서 W까지의 시간 동안 참조회수(Acct)를 누적하고 각 슬롯별 가중치(Weightt)를 정하는데 가중치는 다음과 같이 최근 참조가 발생한 경우 가중치 값이 높은 방식으로 다음의 수식 (3)과 같다.

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

이 식에서 W값은 블록 참조 패턴의 시간별 최대 이력을 정하는 것으로 클수록 보다 정확한 이력을 반영할 수 있으나 복잡도를 고려하여 본 실험에서는 10초로 정한다. 참조회수에 이 가중치를 모든 윈도우에 대하여 곱하게 되면 참조 빈도가 높은 블록일수록 높은 값이 산출된다. 이 값을 MaxPriority 값으로 나누어 참조 빈도가 높은 블록이 최대 우선순위 값으로 수식 (4)와 같이 부여된다. 블록단위 선반입에서도 플래시메모리의 수명을 고려하여 정적 웨어 레벨링을 적용하여 복사 위치를 결정한다.

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

```
NVCache_Write(blocks)
{
  if (Disk Queue service state is peak &&
      Spindle is Idle, Stanby, Sleep mode) {
    find location by static wear leveling;
    write target block on NVCache;
  }
  else {
      write target block on HDD;
  }
}
```

그림 5. NVCache 쓰기 알고리즘
Fig. 5. Write algorithm of NVCache

3.3 NVCache의 쓰기 알고리즘

플래시 메모리의 쓰기 연산의 비효율성을 감안하여 본 논문에서는 디스크큐의 서비스가 복잡하고 스핀들이 유휴, 대기 및 휴면 상태일 경우만 NVCache에 쓰기 연산을 수행하고 여유가 있을 경우는 HDD에 쓰기 연산을 수행한다. 만약 NVCache에 존재하는 블록일 경우는 일관성 문제가 발생하므로 해당 블록을 무효화시키고 디스크 큐의 서비스 시간이 남는 타임 슬롯에 우선순위에 의하여 해당 블록을 다시 복사한다. 그림 5는 쓰기 연산의 알고리즘이다.

IV. 실험 결과

제안하는 기법의 성능을 확인하기 위하여 다양한 실험을 실시하였는데 실험의 목적은 선반입을 통하여 읽기 및 쓰기 연산에 있어서 응답시간의 향상, 전력의 사용의 감소를 확인하는 것이다.

제안하는 기법의 성능을 측정하기 위해 H-HDD 시뮬레이터를 작성하여 실험을 진행하였다. 실험에서는 윈도우7에서 매트랩 및 시뮬링크를 사용하여 시뮬레이터를 작성하였다. 보다 정확한 실험을 위해서는 실제 H-HDD의 펌웨어를 재구성해야 하는데 이것은 현실적으로 곤란하므로 시게이트사의 H-HDD인 Momentus® XT(ST93205620AS)의 사양을 사용하여 프로그래밍하였다.

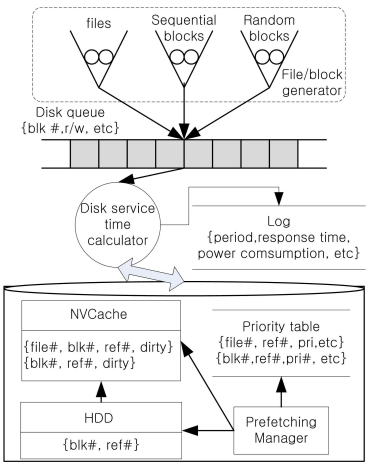

그림 6. H-HDD 시뮬레이터의 구조
Fig. 6. Structure of simulator

그림 6은 실험에서 작성한 시뮬레이터의 구조이다. 먼저 file/block generator는 요구 파일 또는 블록을 생성하는데

블록은 임의 블록 또는 순차 블록의 형태로 만들어지며 각 주기에 디스크 큐에 도착하는 블록은 포아송 분포를 따른다. Disk queue는 한 주기 동안 도착한 요구 블록을 저장하고 SCAN 알고리즘에 의해 블록을 정렬한다. Disk service time calculator는 한 주기 동안 도착한 블록들을 전부 서비스하는데 걸리는 시간, 이때의 전력 소모량 등을 계산한다. 편의상 블록들의 탐색시간은 디스크의 평균 탐색시간을 사용한다. Log 모듈은 실험 결과의 분석을 위하여 주기, 응답시간, 전력 소모량 등의 데이터를 저장한다. NVCache와 HDD는 블록에 대한 메타데이터를 저장하고 Priority table은 요구 블록의 참조 회수 및 주기에 기반하여 우선순위 값을 계산한 테이블이다. Prefetching Manager는 논문에서 제안하는 알고리즘에 의하여 우선순위에 기반하여 디스크 큐의 유휴시간에 블록을 선반입하는 모듈이다.

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

Specification	ST93205620AS
Capacity	320GB
DRAM Cache	32MB
Spindle speed	7200(RPM)
NVCache	4GB
Stanby to Ready	3 Sec
Average seek	11ms(read), 13ms(write)
track to track seek	1.5 ms
Transfer rates	1.23Gb/s(internal),3.0Gb/s(I/O)

표 1은 실험에서 사용된 Seagate Momentus® XT(ST93205620AS)의 사양이다.

4.1 쓰기 연산의 부하에 따른 응답시간

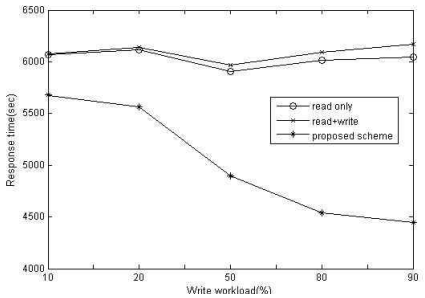

그림 7. 쓰기 연산의 부하에 따른 응답 시간
Fig. 7. Response time based on write workload

그림 7은 쓰기 연산의 부하(10~90%)에 따른 응답시간을 실험한 결과이다. 응답시간은 전체 시뮬레이션 시간 동안 각 주기의 응답시간을 합한 값이다. 그림에서 'proposed scheme'은 논문에서 제안하는 기법이고 비교를 위하여 'read only'는 NVCache를 읽기 캐시로 사용하는 경우이고, 'read+write'는 읽기 및 쓰기 캐시로 사용하는 경우이다. 실험에서의 부하는 디스크 큐에 쓰기 연산이 10%에서 90%까지 변할 때 세 가지 기법의 총 응답 시간을 측정하였다. 제안하는 기법이 나머지 두 기법에 비하여 응답 시간이 우수한 것을 볼 수 있는데 제안 기법이 선반입을 통하여 NVCache에 히트율이 높기 때문이다. 또한 부하가 증가할수록 응답성이 더 향상되고 있는데 이것은 평상시에는 HDD에 쓰기연산을 수행하고 디스크가 유휴 또는 대기 상태에서만 NVCache에 연산을 수행하기 때문이다. 그림에서 'read+write'기법이 'read'기법에 비해 응답성이 다소 낮은 것은 이 기법이 쓰기 연산의 부하가 증가하게 되면 NVCache에 쓰기를 빈번하게 실행하여 더 많은 시간을 요구하기 때문이다.

표 2. 디스크 모드별 전력 소모
Table 2. Power dissipation

Power dissipation	+5V Input average
Spinup(max)	5 W
Seek	2.2 W
Read, Write	2.4W, 2.3 W
Idle	1.4 W
Stanby, Sleep	0.4 W
NVCache	0.2 W

4.2 쓰기 연산의 부하에 따른 전력 소모

그림 8은 표 2의 전력 소모 파라메터를 참고하여 쓰기연산의 부하에 따른 전력 소모를 실험한 결과이며 세 기법 모두 수식 (2)에 의해 선반입을 실시하였다. 전력의 소모는 각 주기에서 요구블록을 서비스하는 데 소모되는 양, 남은 주기 동안(유휴 또는 대기모드) 소모되는 양, 그리고 선반입을 위해 필요한 양의 합으로 나타낼 수 있다. 읽기와 쓰기의 전력소모가 다르므로 읽기 블록은 수식 (5), 쓰기 블록은 수식 (6)을 적용하였다.

$$ReadHits * NVCache_w + DiskReads * (Seek_w + Read_w) + (SizeOfQueue - ReqBlks) * IdleStanby_w + Prefetch_w \quad \cdots\cdots (5)$$

각 주기의 요구블록 수는 ReqBlks이고 ReadHits는 NVCache에 히트된 블록의 수이고 DiskReads는 디스크에서 읽어야할 블록의 수이고 디스크 큐의 크기는 SizeOfQueue이다. NVCachew는 캐시메모리의 전력소모이고, 디스크헤드의 탐색시 전력소모는 Seekw, 읽기 연산의 전력소모는 readw,이다. 그리고 유휴 또는 대기모드의 전력 소모는 IdleStanbyw이며 Prefetchw는 선반입시 소모되는 전력소모이다. 따라서 수식 (5)의 첫 번째 항은 캐시 히트시 전력소모의 양을 나타내고 두 번째 항은 디스크 읽기 연산의 전력소모이고 세 번째 항은 유휴 또는 대기상태의 전력소모이며

마지막 항은 선반입시 소모되는 전력소모이다.

$$WriteHits*NVCache_w + DiskWrites*(Seek_w + Write_w) + (SizeOfQueue - ReqBlks)*IdleStanby_w + Prefetch_w \quad \cdots\cdots (6)$$

수식 (6)은 쓰기 연산의 전력소모를 나타내는데 Write Hits는 쓰기 히트의 수를 나타내고 DiskWrites는 디스크 쓰기 연산의 수이고, Writew는 쓰기 연산의 전력소모를 나타낸다. 그림 8의 실험에서 'read only'기법은 수식 (6)에서 첫 번째 항이 0이고, 제안하는 기법은 디스크의 모드에 따라서 0이 될 수도 있고 아닐 수도 있다.

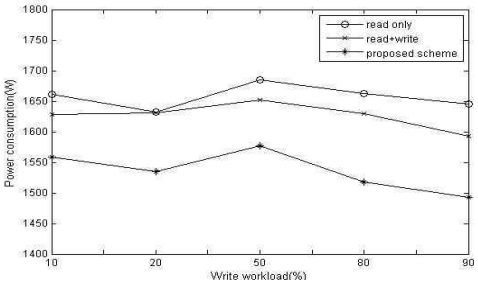

그림 8. 쓰기 연산의 부하에 따른 전력 소모
Fig. 8. Power consumption based on write workload

그림 8에서 보듯이 제안하는 기법이 두 기법에 비하여 낮은 전력 소모를 보이는데 이것은 스핀다운 상태에서 제안 기법은 NVCache에 쓰기 연산을 수행하므로 스핀업의 전력 소모를 최대한 감소시키기 때문이다. 두 기법은 스핀다운 상태에서도 HDD에 쓰기 연산을 수행하므로 유휴 또는 대기모드의 횟수가 증가할수록 더 많은 전력소모를 요구하는 것으로 분석된다. 또한 'read+write'기법은 'read only'기법에 비해 전력소모가 다소 적은데 이것은 NVCache에 쓰기 연산을 수행하기 때문이다.

4.3 선반입 크기에 따른 응답 시간

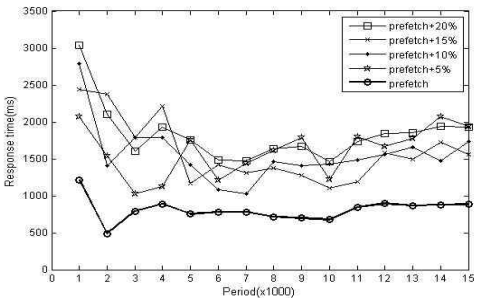

그림 9. 선반입의 크기에 따른 응답 시간
Fig. 9. Response time based on prefetching size

그림 9는 선반입의 크기에 따른 각 주기의 평균 응답시간을 나타낸다. 그림에서 'prefetch'는 수식 (2)에 의하여 선반입을 하는 경우 이고 'prefetch+n'은 선반입의 크기에 따른 성능 비교를 위해 n% 블록을 추가한 경우이다. 그림에서 보듯이 제안하는 기법이 선반입 양을 증가한 나머지 기법보다 평균 15% 이상의 성능 개선을 보이고 있다. 선반입 블록의 수를 증가하더라도 응답시간이 개선되지 않는데 이것은 선반입을 하기위해 추가의 디스크 헤드 이동이 소요되고 이것이 전체 응답시간을 증가시키기 때문이다. 응답 시간을 나타내는 그래프의 전체적인 패턴에서 볼 때 초기의 응답 시간이 다소 높은데 이것은 NVCache의 히트율이 낮기 때문이고 주기가 지날수록 선반입의 개수에 관계없이 응답 시간이 수렴하는 것을 볼 수 있다. 따라서 수식 (1), (2)와 같은 선반입 수의 최적화로 H-HDD의 응답시간을 최대로 향상시킬 수 있음을 알 수 있다.

V. 결 론

본 논문에서는 플래시 메모리를 사용하는 H-HDD의 응답성과 전력소모를 감소시키기 위한 NVCache 관리 기법을 제시하였다. 제안하는 기법은 응답성을 향상시키기 위하여 주기당 디스크 큐를 서비스하고 남은 여유 시간에 블록의 공간적, 지역적 우선순위를 고려하여 NVCache에 파일 및 블록을 선반입한다. 여유 시간에 선반입을 하게 되므로 서비스의 끊김 현상이 감소하고 HDD의 스핀업시 소모되는 전력소모를 감소시킬 수 있다. 또한 시간적, 공간적 접근 패턴에 따라 선반입 우선 순위를 결정하므로 캐시 히트율을 높일 수 있다. 쓰기 연산의 경우 디스크 쓰기 연산이 집중되거나 유휴 또는 대기 상태에서만 NVCache에 쓰기 연산을 수행하여 시스템의 응답성과 전력 소모를 제한적이나마 향상시킬 수 있다. 따라서 모바일 응용에 본 연구의 결과가 구현된다면 H-HDD의 입출력 성능을 극대화할 수 있을 것으로 기대된다.

참고문헌

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

[2] Hong-jae Lee, "Toward Understanding Hard Disk," Electronic Times, April 2003.
[3] Park S.H. Park et al, "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, Aug 2008.
[4] S. Im and D. Shin, "Storage Architecture and Softw are Support for SLC/MLC Combined Flash Memory," Proc. of 24th ACM Symposium on Applied Computin g, Sep 2009.
[5] Daniel Pierre Bovet, Marco cesati, "Understanding the Linux Kernel (3/E)," O'REILLY, Nov, 2005.
[6] B. Gill and L. Bathen. AMP: Adaptive multistream prefetching in a shared cache. In Proceeding in a shared cache. In Proceedings of the 5th USENIX Conference on File and Storage Technologies, 2007.6.
[7] Luis Useche, Jorge Guerra, Medha Bhadkamkar, "EXCESS: External Caching in Energy Saveing Storage Systems," Proceedings of the 13th International Symposium on High-Performace Computer Architecture, 2008.7.
[8] Y.J. Kim et al, "I/O Performace Optimization Techniques for Hybrid Disk-Based Mobile Consumer Devices," IEEE Transactions on Consumer Electro nics, vol.53, no.4, 2007.4.
[9] Seongcheol Hong, Dongkun Shin, "Designing Hybrid HDD using SLC/MLC combined Flash Memory," Journal of KIISE : Computing Practices and Letters, vol.6, no.7, 2010.7.
[10] J.S. Yang, Y.W. Go, C.G.Lee, D.H. Kim, "Design and Implementation of Hybrid Disk I/O System based on n-Block Prefetching for Low Power Consumption and High I/O Performance," Journal of KIISE : Computer Systems and Theory, vol.36, no.6, 2009.12.
[11] K.H. Park, G.H. Lee, D.H. Kim, "An Efficient Data Block Replacement and Rearrangement Technique for Hybrid Hard Disk Drive," Computing Practices and Letters, vol.16, no.1, 2010.1.
[12] http://www.seagate.com
[13] http://www.overclockersclub.com/reviews/seagate_momentus_xt_500gb/7.htm

저 자 소 개

김 정 원

1995년 : 부산대학교 전자계산학과(학사)
1997년 : 부산대학교 대학원 전자계산학과 (석사)
2000년 : 부산대학교 대학원 전자계산학과 (박사)
2000년~2001년 : 기술신용보증기금 기술평가역(차장)
2002년~현재 : 신라대학교 컴퓨터정보공학부 교수
관심분야 : 내장형시스템, 멀티미디어, 운영체제
Email : jwkim@silla.ac.kr