저전력과 입출력 성능이 향상된 n-블록 선반입 기반의 하이브리드 하드디스크 입출력 시스템 설계 및 구현

(Design and Implementation of Hybrid Hard Disk I/O System based on n-Block Prefetching for Low Power Consumption and High I/O Performance)

양 준 식 † 고 영 욱 † 이 찬 근 †† 김 덕 환 †††
(Junsik Yang) (Youngwook Go) (Chan-gun Lee) (Deok-Hwan Kim)

요 약 최근에 하드 디스크의 낮은 입출력 처리 성능을 개선하는 연구가 활발하게 진행 중이다. 하드웨어 연구는 좋은 성과를 보이고 있지만 시스템의 입출력 성능향상을 지원해야 할 시스템 소프트웨어 기술 발전이 미진하여 하드웨어 성능을 최대로 발휘하지 못하고 있는 상황이다. 본 논문에서는 n-블록을 플래시 메모리로 선반입하는 새로운 방법을 제안한다. 제안한 방법은 세 단계로 구성된다: (1) 블록 단위 읽기 요청의 패턴을 분석하여 n-블록단위로 플래시 메모리에 선반입한다; (2) 입출력 요청 시에 그 블록의 위치를 판단하여 입출력 서비스를 제공한다; (3) 블록 교체 정책에 따라 n-블록을 교체한다. 이 방법을 통해 하드디스크의 대기시간을 줄이고 전력 사용을 최적화 할 수 있다. 실험을 통해 제안한 동적 n-블록 방법이 기존의 AMP(Adaptive multistream prefetching) 방법과 비교하여 9.05%의 평균응답시간을 개선하고 평균전력소모를 11.11% 감소시킴을 확인하였다.

키워드 : 선반입, 하이브리드 저장장치, 비휘발성 메모리, 플래시 메모리

Abstract Recently, there are many active studies to enhance low I/O performance of hard disk device. The studies on the hardware make good progress whereas those of the system software to enhance I/O performance may not support the hardware performance due to its poor progress. In this paper, we propose a new method of prefetching n-blocks into the flash memory. The proposed method consists of three steps: (1)analyzing the pattern of read requests in block units; (2)determining the number of blocks prefetched to flash memory; (3)replacing blocks according to block replacement policy. The proposed method can reduce the latency time of hard disk and optimize the power consumption of the computer system. Experimental results show that the proposed dynamic n-block method provides better average response time than that of the existing AMP(Adaptive multistream prefetching) method by 9.05% and reduces the average power consumption than that of the existing AMP method by 11.11%.

Key words : Prefetching, Hybrid Storage, Non-Volatile memory, Flash memory

· 이 논문은 2008년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2008-313-D00822).
· 본 연구는 지식경제부와 한국산업기술진흥원의 전략기술인력양성사업으로 수행된 결과임
· 본 논문은 정보통신부 출연금으로 ETRI, SOC산업 진흥센터에서 수행한 IT SOC 핵심설계인력양성사업의 연구 결과임.
· 본 연구를 위한 실험 기자재인 하이브리드 하드디스크(HM16HJI)는 삼성전자로부터 지원을 받아 연구를 진행하였습니다.

† 학생회원 : 인하대학교 전자공학과
juneseek@iesl.inha.ac.kr
kyw@iesl.inha.ac.kr
†† 정 회 원 : 중앙대학교 컴퓨터공학부 교수
cglee@cau.ac.kr
††† 정 회 원 : 인하대학교 전자공학과 교수
deokhwan@inha.ac.kr
(Corresponding author임)

논문접수 : 2009년 1월 12일
심사완료 : 2009년 8월 19일

정보과학회논문지: 시스템 및 이론 제36권 제6호(2009.12)

1. 서 론

오늘날에는 개인용 컴퓨터뿐만 아니라 모바일 저장 장치에도 대용량 하드디스크가 장착되어 있다. 하드디스크의 기술은 크게 발전 하고 있지만 여전히 기계적 장치를 사용하여 입출력을 수행 하므로 몇 가지 문제점이 존재한다. 첫째, 하드디스크는 전력 소모가 높다. 하드디스크는 컴퓨터 시스템에서 전력 소모를 10~20% 차지하며 그 중 약 80~90% 정도의 전력 소모가 스핀들 모터에서 발생한다[1]. 둘째, 하드디스크는 필수적인 대기시간이 발생한다. 하드디스크의 데이터 전송구조를 살펴보면 플래터가 충분한 속도로 회전을 하면 암이 트랙을 찾고 섹터를 찾아서 그 데이터를 호스트에 전송한다. 이러한 과정 때문에 필수적으로 대기시간이 발생 한다.

유비쿼터스 컴퓨팅 환경에 필수적인 모바일 기기를 사용하기 위해서는 저전력 시스템이 필수적이다. PDA (Personal Digital Assistants), PMP(Portable Multimedia Player), MP3 Player 등에 하드디스크가 많이 사용되고 있는데 이러한 모바일 기기들은 전력 공급에 배터리를 주로 사용하기 때문에 전력 소비를 줄이는 연구가 중요하다[2]. 또한 앞서 설명한 하드디스크의 특성 때문에 하드디스크 보다 처리속도가 빠른 중앙처리장치 및 메인 메모리와의 병목현상이 일어나고 있다[3]. 이를 극복하기 위한 연구는 컴퓨터 시스템의 성능 향상 측면에서 아주 중요한 연구라 여겨진다. 한편, 최근에 하드디스크와 메인 메모리 사이에 플래시 메모리가 삽입 되어 있어 하드디스크의 캐시 기능을 가지는 하이브리드 하드디스크[4]와 인텔의 룹슨[5] 기술이 발표되었다. 이 기술들은 비휘발성 메모리인 플래시 메모리를 통하여 입출력을 함으로써 하드디스크의 스핀들 모터의 동작을 최소화 하는 하드웨어 기술이다. 하지만 플래시 메모리를 사용하여 입출력을 수행 한다고 해도 대용량의 하드디스크의 모든 주소를 플래시 메모리로 사상하여 입출력 서비스를 제공 할 수 없기 때문에 필연적으로 하드디스크가 스핀업 되어 입출력 서비스를 제공 하여야 한다[6,7]. 하드디스크가 스핀다운 상태에서 스핀업 상태로 변환 하는 동작은 성능뿐만 아니라 전력 소비 면에서도 아주 큰 비용이 발생 된다. 하이브리드 저장 시스템을 사용할 때 하드디스크 시동 빈도가 많아질수록 하이브리드 저장 시스템의 성능은 낮아지고 전력 소모량은 늘어나게 된다. 본 논문에서는 하이브리드 저장 장치에서 하드디스크의 스핀업 횟수를 줄이기 위해 n-블록 선반입 기법을 사용하여 플래시 메모리에 블록들을 선반입 시킨다. 여기에서의 선반입 이란 일반적인 디스크에서 메모리에서의 선반입이 아니라 하드디스크에서 플래시 메모리로의 선반입을 말한다. n-블록이란 하드디스크의 시동시간과 플래시 메모리의 남은 용량 등의 정보를 이용하여 계산 된 블록의 개수를 말한다. 결정된 n개의 블록을 플래시 메모리에 선반입 하면 호스트가 읽기 요청을 하였을 때 성능 향상뿐만 아니라 전력 소모를 줄일 수 있다. 플래시 메모리에 선반입 된 n개의 블록은 공간적으로 연속된 블록들과 시간적으로 연속된 블록들을 기준으로 모델링 된다. 본 논문에서는 제안한 방법의 성능 평가를 위해 전력소모와 응답시간 관점에서 성능 평가를 실시한다.

본 논문의 순서는 먼저 2장에서 관련 연구에 대해 소개하고 3장에서 제안하는 n-블록 선반입에 관하여 설명한다. 4장에서 n-블록 선반입 하이브리드 하드디스크 입출력 시스템의 설계 및 구현에 대해 설명하고 5장에서 각 n-블록 선반입 하이브리드 하드디스크 입출력 시스템을 적용한 시스템과 전형적인 하드디스크의 성능을 비교한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

본 장에서는 하이브리드 하드디스크의 구조와 컴퓨터 시스템에서 일반적으로 쓰이는 선반입에 관하여 설명한다.

2.1 하이브리드 저장장치(Hybrid Storage System)

그림 1은 WinHEC 2006에서 삼성전자가 플래시 메모리를 캐시로 사용하고 하드디스크를 주 저장장치로 사용하는 하이브리드 하드디스크의 개념도이다. 현재는 세계적으로 하이브리드 하드디스크의 유용성을 인정받아 차세대 운영체제인 Windows Vista에서 정식적으로 지원하기 시작했다.

Vista에서 지원하는 기술 중 하이브리드 하드디스크와 관련 있는 기술은 ReadyDrive와 ReadyBoost 기술이 있다[4]. 이 기술은 시스템과 하드디스크 간에 전송이 일어날 때마다 하드디스크를 구동시켜야 하는데, 이 구동 횟수를 줄이기 위해 중간에 플래시 메모리를 두는 기능이다. 따라서 하드디스크의 수명도 늘어나고, 전력도 적게 소모되며, 때에 따라서는 부팅 시 필요한 데이터를 플래시 메모리로 방향을 변경하여 저장하여 빠른 접근 속도로 인한 빠른 부팅 시간을 보여 준다[8]. 한편

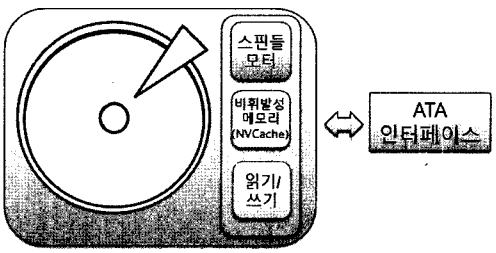

그림 1 하이브리드 하드 디스크 구성도

인텔에서는 롭슨 기술을 통해 하이브리드 저장 시스템을 제공한다. 하이브리드 저장장치와 관련하여 진행된 연구는 서울대학교의 SimHybrid[9]가 있다. SimHybrid는 트레이스 기반의 SimHybrid, 플래시 메모리 시뮬레이터인 SimFlash 및 호스트 시뮬레이터로 구성되어 있다. 마이크로소프트사의 Windows XP 환경에서 실험을 진행 하였고 부팅시간, 응용프로그램 실행 시간, 스핀-다운 시간과 에너지 소모량 등을 실험 하였다. UCSC의 Timothy[10,11]는 비휘발성 메모리장치인 플래시 메모리를 두어서 하드디스크의 스핀다운 시간을 늘렸다. 읽기 캐시는 많이 쓰이는 데이터를 선반입 함으로서 스핀-다운 시간을 늘렸다. 하지만 위의 연구에서는 하이브리드 저장장치의 기본적인 특성을 연구하고 있다. 블록의 개수를 정하고 선반입 시켜서 하드디스크의 스핀들 모터를 최대한 활동 하지 않게 하고 하드디스크의 주기를 동적으로 변경 하여 컴퓨팅 시스템의 성능 향상과 동시에 전력 감소 시키는 연구는 미진한 편이다.

2.2 선반입(Prefetching)

컴퓨터 시스템에서는 일반적으로 캐시, 메모리 및 디스크 등의 3단계의 계층적인 구조를 가진다. 그 중에서 메모리와 디스크간의 선반입은 하드웨어의 특별한 도움을 받을 수 없고 운영체제의 개입이 필수적이다. 메모리의 데이터 접근과 하드 디스크의 데이터 접근 시간은 최소 10^3배 이상 차이가 날 정도로 하드 디스크의 속도가 느리므로 선반입을 통해 하드 디스크와 메모리간의 병목 현상을 줄여야 한다[3]. 선반입의 기존 연구는 프로그램의 지역성과 순차성을 기준으로 나눌 수 있다. 지역성은 다시 시간적 지역성과 공간적 지역성으로 나눌 수 있다.

시간적 지역성은 최근에 사용된 정보는 다시 사용될 가능성이 높다는 것을 의미하며, 공간적 지역성은 하드디스크의 공간상에 현재 사용되는 데이터와 인접한 데이터는 사용될 가능성이 높다라는 것을 의미한다. 또한 응용프로그램을 기준으로 보았을 때 힌트를 이용한 선반입과 힌트를 이용하지 않는 선반입으로 나눌 수 있다. 대부분의 힌트를 이용한 선반입은 특수한 환경에서 사용된다. 응용프로그램이 힌트를 제공하는 연구에는 컴파일러 차원에서 힌트를 삽입하는 연구가 행해졌다[12]. 이 연구는 특정한 시스템에서만 가능하고 일반적인 환경에서는 검증된 바 없다고 알려져 있다. 응용프로그램 상에서 힌트를 주는 선반입은 응용프로그래머에게 부담이 될 뿐만 아니라 잘못된 정보가 유입될 가능성도 있고 호환성의 문제가 발생할 가능성이 크므로 특수한 상황에서만 사용 된다. 힌트가 없는 범용적인 선반입은 일반적인 운영체제에서도 사용할 수 있다. 리눅스에서는 RA(Read Ahead) 알고리즘을 사용하여 연속된 위치의 블록을 미리 읽어 들여 선반입을 실시한다[13]. 하드디스크는 1초에 최소 40메가바이트를 읽고 전송할 수 있으므로 RA 알고리즘은 부하가 발생하지 않는다.

UC 버클리의 Smith[14]는 계층적 구조의 메모리 시스템에서 순차적인 선반입이 작은 크기의 데이터 블록을 갖는 시스템에서 성능 개선 효과를 거둘 수 있으며, 특히 매 메모리 참조마다 순차적 선반입을 시도함으로써 유효 CPU 속도를 10%~25%까지 향상 시킬 수 있다고 하였다.

IBM 연구 센터의 B. Gill은 선반입을 등급과 트리거 디스턴스 인자를 받아서 히스토리를 생성 하고 이를 기반으로 하여 동적으로 선반입 등급과 트리거 디스턴스를 변화 시켜 여러 개의 어플리케이션이 서로 다른 요청을 동시에 요청 할 때 효과적으로 선반입을 수행하기 위한 AMP(Adaptive multistream prefetching in a shared cache) 선반입 방법을 제안하였다[15]. AMP 선반입 방법은 병렬적으로 여러 블록들이 요청될 경우에는 순차성이 달라질 수 있기 때문에 선반입을 수행할 때 등급과 트리거 디스턴스를 저장되어 있는 각각의 블록들에 대해 기록하고 이를 사용하여 선반입을 시행하였다.

또한 EXCES(EXternal Caching in Energy Saving Storage Systems) 시스템[16]은 하드디스크의 기계적인 특성에 의한 전력 소모를 줄이기 위하여 하드디스크의 캐시 기능을 가진 플래시 메모리를 탑재하고, 하드디스크의 동작을 최대한 줄이고 데이터 요청을 플래시 메모리에서 서비스 할 수 있도록 플래시 드라이브를 하드디스크의 버퍼로서 사용하고, 데이터의 조작 단위를 페이지 기반으로 한 캐싱과 선반입을 수행 하여 전력 소모를 줄였다.

서울시립대에서는 순차패턴 마이닝 기법을 이용하여 미래에 사용 될 파일을 검색 하여 플래시 메모리로 선반입 하는 시스템을 제안하였다[17]. 이 논문에서는 하이브리드 저장장치에서 캐시 역할로서 플래시 메모리를 사용하고 데이터 접근 속도를 높이기 위해서 미래에 접근 가능성이 높은 데이터를 파일을 기반으로 선반입을 수행 하였다.

하지만 지금까지 기술한 선반입 기법은 전통적인 선반입 기법인 하드 디스크와 메모리간의 선반입 기법이거나 플래시 메모리와 하드디스크를 혼합하여 사용하는 하이브리드 저장장치를 이용한 파일 기반의 선반입 방법이다. 제안하는 n-블록 선반입 하이브리드 하드디스크 입출력 시스템은 n-블록을 기반으로 하이브리드 하드디스크에 장착된 플래시 메모리로 선반입 함으로서 하드디스크의 전력 소모를 줄이고 시스템 입출력 성능을 향상시키는 방법을 제안한다.

3. n-블록 선반입 기법

3.1 n-블록 선반입 기법

본 논문에서 제안하는 n-블록 선반입 기법은 n개의 블록을 플래시 메모리로 한번에 선반입 하여 시스템의 성능을 높이는 정책이다. 선반입을 수행 하는 시기는 시스템의 성능 저하를 막기 위해 수시로 수행 하지 않고 플래시 메모리에 선반입 된 데이터로부터 읽기 실패가 일어났을 때 선반입을 수행 한다. 프로그램이 원활하게 동작하기 위해서는 가능한 많은 블록들을 선반입 하는 것이 좋다. 하지만 플래시 메모리의 선반입 캐시의 크기가 제한되므로 선반입할 블록의 개수를 정하는 것이 중요하다.

그림 2는 하드디스크의 상태변화와 하드디스크가 시동하는데 걸리는 대기시간과 전력 소모량을 보여 준다. 즉, 명령어 실행이 시작되면 디스크가 대기상태에서 활성 상태로 전환 되며 이것을 스핀업 이라고 한다. 예를 들면 시스템이 부팅과정에 있거나 프로그램을 로딩 하고 있을 때 하드디스크 시동시간 및 탐색 시간 동안 대기시간이 발생 되고 추가적으로 전력 소모가 발생 된다. 명령어 실행이 끝나면 하드디스크는 활성 상태에서 유휴 상태 또는 대기상태로 상태가 변화 된다. 그림 2에서는 하드디스크가 명령어 요청을 받아 명령어 실행을 시작할 때에 4.2초의 대기시간과 4.5w의 전력이 소모됨을 알 수 있다.

그림 3은 n-블록 선반입 기법을 적용하는 경우 하드디스크의 응답속도와 전력 소모를 보여준다. 입출력 요청이 발생 되었을 때 하드디스크가 대기 상태이거나 휴면 상태 또는 유휴 상태라면 일반적인 시스템에서는 하드디스크가 활성 상태로 전환하기 위한 대기시간이 발생된다. 하지만 그림 3처럼 하드디스크가 아닌 플래시 메모리에서 요청된 블록을 읽어 오게 되면 하드디스크의 지연시간을 줄일 수 있다. 하드디스크가 시동 하는데 걸리는 대기시간 동안에 플래시 메모리에서 이미 선반입 된 블록을 메인 메모리로 이동 시켜 하드디스크의 대기시간 없이 입출력 요청을 수행할 수 있다. 또한 하드디스크가 대기 상태로 지속되기 때문에 전력 소모를

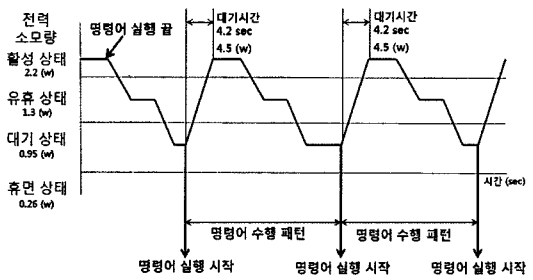

그림 2 일반적인 하드디스크의 전력 소모와 응답속도

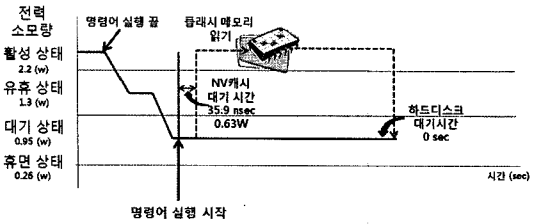

그림 3 제안한 하드디스크의 전력 소모와 응답속도

줄일 수 있다.

본 논문에서는 하드디스크의 스핀업 동작을 최소화 하면서 입출력 요청을 수행하는데 필요한 최소의 n-블록을 구하는 방법을 제안한다. "최소 n-블록"은 다음과 같이 구할 수 있다.

$$nMinBlocks = \frac{HSpinUpLT}{(FlashAccessT + BlockTransT)} \quad (1)$$

호스트가 요청한 n개의 블록을 하드디스크 대신에 플래시 메모리에서 읽어드리면 시스템은 하드디스크의 시동으로 인해 발생되는 지연시간을 상쇄시킬 수 있으며 하드디스크의 기계적인 동작으로 인해 발생되는 전력 소모를 줄일 수 있다. $nMinBlocks$는 시스템 성능 입장에서 보았을 때 한번에 선반입 시킬 최소한의 블록의 숫자를 결정하는 것이며, $HSpinupLT$는 하드디스크의 스핀업 지연시간을 의미한다. $FlashAcessT$는 플래시 메모리에 접근 하는 시간을 의미하고, $BlockTransT$는 한 블록을 옮기는데 걸리는 시간을 의미한다. 블록의 개수를 정하는 것은 선반입을 수행 할 때에 공간적 지역성을 유지 하기 위하여 요청된 블록들뿐 만 아니라 요청된 블록과 공간상으로 가까운 연속된 n개의 블록을 한번에 선반입 하는 정책이다. $FlashAcessT$와$BlockTransT$의 합은 한 개의 블록을 플래시 메모리에서 메인 메모리로 옮기는데 걸리는 시간이다. 한 개의 블록을 메인 메모리로 옮기는데 걸리는 시간으로 $HSpinUpLT$을 나누면 블록의 개수가 나온다.

$$nDynamicBlocks_k = \frac{FlashExtraSize * BlockPrio_k}{(BlockSize * HPrefetchBlockCount * nMinBlocks} + nMinBlocks \quad (2)$$

2번 식은 "동적 n-블록" 즉, 동적으로 선반입을 할 수 있는 블록의 개수 n을 구하는 과정을 보여 준다. $FlashExtraSize$는 플래시 메모리가 선반입을 할 수 있는 가용한 자유 영역의 크기이며, $BlockSize$는 플래시 메모리에 할당된 있는 한 블록의 크기이며 디스크의 섹터 크기와 같다. 즉, 블록은 플래시 메모리의 페이지의 단위인 512바이트를 사용한다. $HPrefetchBlockCount$는 입출력 요청을 분석하여 플래시 메모리로 선반입될 블록들의 개수이다. $BlockPrio_k$는 k번째 블록의 우선순위

이다. 우선순위가 높으면 많은 수의 블록을 선반입 하고 우선순위가 낮으면 적은 수의 블록을 선반입 하는 동적인 선반입 방법이다. 단, k번째 블록의 동적인 선반입 개수가 스핀업 동작을 최소의 횟수로 하는데 필요한 최소 n-블록 보다는 커야 하므로 $nMinBlocks$를 더해준다.

3.2 블록 우선순위 결정과 블록 교체

그림 4는 블록 우선순위 결정과 블록 교체를 위해 사용되는 입출력 시스템의 구조를 보여준다. (1) DRAM에 존재하는 입출력 요청 큐에는 입출력 요청된 블록들이 저장된다. (2) DRAM에 존재하는 선반입 요청 큐에는 최근성과 최빈성에 의해 우선순위가 정해진 블록들이 재정렬 된다. 이렇게 저장된 선반입 요청 큐의 블록들은 식 (1)과 식 (2)에 따라 플래시 메모리에 선반입 된다. (3)플래시 메모리에 선반입된 블록들은 LRU 정책에 따라 접근 빈도가 낮은 블록 교체를 수행한다. (4)DRAM에 존재하는 교체 우선순위 큐는 접근 빈도가 낮은 블록을 교체 하기 위하여 플래시 메모리에 선반입 된 목록들의 접근빈도 별 교체 우선순위를 관리한다.

3.2.1 블록의 선반입 우선순위

본 논문에서는 선반입을 하기 위하여 실시간으로 입출력 요청을 분석 한다. 입출력 요청을 분석하고 블록의 최근성과 최빈성에 따라 플래시 메모리로 선반입을 수행 하게 된다. 무작위로 선택된 블록들을 선반입 하는 것은 전력을 많이 소모할 뿐만 아니라 탐색을 위한 지연시간이 길어진다. 또한 중요도가 낮은 블록을 선반입 하면 시스템의 성능이 떨어지고 추후에 추가적인 블록 교체에 따른 전력 소모와 지연시간을 감수 해야 한다.

본 논문에서는 블록 별로 시간상의 빈도수를 가중치에 의해 계산 하여 우선 순위를 결정 한다. k번째 블록의 우선순위를 결정 하는 식은 다음과 같다.

$$BlockPrio_k = \frac{1}{r}\sum_{t=1}^{r}(recent_t * ref_t) \quad (3)$$

식 (3)에서 ref_t는 k번째 블록의 시간대 별 입출력 요청 빈도를 의미한다. 선반입의 특성상 최근의 블록 접근 빈도에 관한 정보를 신뢰해야 해야 하므로 시간에 관한 상수 r을 사용한다. r은 제안한 선반입 시스템에서 정한 상수 로서 타임 스탬프를 찍는 횟수를 말한다. 즉, 타임 스탬프를 1분으로 설정 하고 1분 동안의 입출력 요청 빈도를 r 만큼만 저장 하는 것이다. 입출력 요청 빈도에 가중치 함수인 $recent_t$를 곱하면 최근성을 구할 수 있다. 가중치 함수는 다음과 같다.

$$recent_t = 1 - \frac{t}{r} \quad (4)$$

식 (4)는 우선순위를 구하기 위한 가중치 값을 구하는 과정을 보여 준다. $recent_t$의 값은 0에서 1사이의 값으로 정규화 시킨다. 정규화 된 값을 이용하여 식 (3)의 블록 우선순위를 구할 수 있다.

여기에서 r은 히스토리를 얼마나 저장 하는지에 관한 변수이다. r이 커지면 많은 히스토리를 저장할 수 있어서 학습효과를 많이 받을 수 있지만 선반입이라는 것은

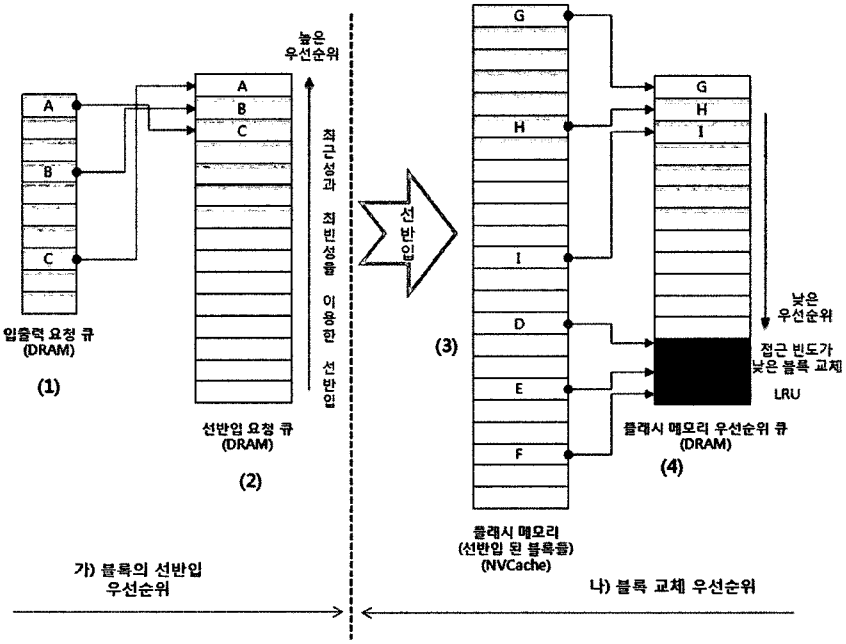

그림 4 선반입 할 블록 선택과 플래시 메모리 블록 교체 정책

과거의 정보를 너무 신뢰해서는 안되며, 많은 히스토리를 처리하는데 드는 비용을 무시할 수 없기 때문에 본 논문에서는 실험을 통하여 r은 100이라는 상수로 정하여 수행한다.

그림 5는 선반입 요청 큐의 블록 우선순위를 관리하는 알고리즘이다. 선반입 요청 큐에는 페이지캐시의 래딕스 트리내에 존재하는 블록이나 플래시 메모리에 선반입이 되어 있는 블록들은 제외된다. 중복된 블록을 관리 할 필요가 없기 때문이다.

3.2.2 블록 교체 우선순위

그림 6은 플래시 메모리에 선반입 되어 있는 플래시 메모리 우선순위 큐의 교체우선순위를 설정하고 플래시 메모리의 블록들을 교체 하는 알고리즘이다. 블록교체의 경우에는 플래시 메모리의 크기의 80%이상이 넘어가게 되면 블록교체를 수행하고 하드디스크가 스핀업 되었을 때도 블록 교체를 수행 한다. 왜냐하면 스핀업이 되었을 때는 선반입 요청 큐의 블록들이 선반입이 되기 때문에 신규로 선반입 되는 블록을 플래시 메모리 우선순위 큐에 추가 해주어야 하기 때문이다.

4. n-블록 선반입 기반 하이브리드 하드디스크 입출력 시스템의 설계 및 구현

본 논문에서 제안하는 n-블록 선반입 기반의 하이브리드 하드디스크 입출력 시스템은 호스트로부터 읽기 요청이 발생 하였을 경우 커널의 페이지 캐시를 검사하고 페이지 캐시에 해당 페이지가 존재 하지 않을 경우에 하드디스크를 스핀업 시키고 블록을 요청 하는 일반적인 하드디스크를 탑재한 리눅스 시스템 대신에 하이브리드 하드디스크의 플래시 메모리에 선반입 시켜 놓은 블록 데이터를 전송 하게 함으로써 하드디스크의 스핀업 횟수를 줄여 준다. 하이브리드 하드디스크는 일반적인 하드디스크에 플래시 메모리가 탑재된 제품으로써 일반적인 리눅스 시스템에서의 입출력을 플래시 메모리로 리다이렉션 시켜야 한다. 본 논문에서 제안 하는 서브시스템은 리눅스 커널 2.6.24 버전에서 구현하였고 전체 구조는 다음과 같다.

4.1 입출력 개요

그림 7과 같이 호스트로부터 블록 요청이 들어오면 입출력 분석기는 최근성과 최빈성을 고려하여 실시간으로 데이터 블록의 우선순위를 설정한다. 요청한 블록을 페이지 캐시에서 검색하고 이를 실패하였을 경우 플래시 메모리에 선반입된 블록을 관리하는 플래시 메모리 우선순위 큐에서 검색한다. 검색이 실패하면 읽기 실패가 발생하며 하드디스크에서 데이터를 읽어 들인다. 이

```
최근성과 최빈성을 고려한 선반입 요청 큐의 블록 우선순위 관리 알고리즘
입력 : 블록번호
{
1       입출력 요청 큐에 있는 블록들에 대해 식3과 같이 최근성과 최빈성을 이용 하여 우선순위결정;
2       우선순위 순서에 따라 선반입 요청 큐 갱신;
3       if( 해당 블록이 플래시 메모리에 선반입 되어 있는지 여부 검색 )
4       {
5               플래시 메모리에 선반입된 해당 블록 데이터를 호스트에 전송;
6               플래시 메모리 우선순위 큐의 해당 블록의 우선순위 증가;
7       }
8       else
9       {
10              하드디스크 스핀업;
11              하드디스크에 저장되어 있는 해당 블록 데이터를 호스트로 전송;
12              식1 또는 식2를 이용하여 선반입 요청 큐에 있는 n-블록을 플래시 메모리로 선반입;
13      }
}
```

그림 5 블록 우선순위 관리 알고리즘

```
블록 교체 알고리즘
{
1       블록 접근 횟수를 기반으로 플래시 메모리 우선순위 큐의 교체 우선순위 설정;
2       if( 플래시 메모리에 선반입된 블록들이 80% 넘으면)
3       {
4               접근빈도가 낮은 블록 제거;
5               신규로 선반입된 블록을 플래시 메모리 우선순위 큐에 추가;
6               블록 접근 빈도에 따라 교체 우선순위 재설정;
7       }
}
```

그림 6 블록 교체 알고리즘

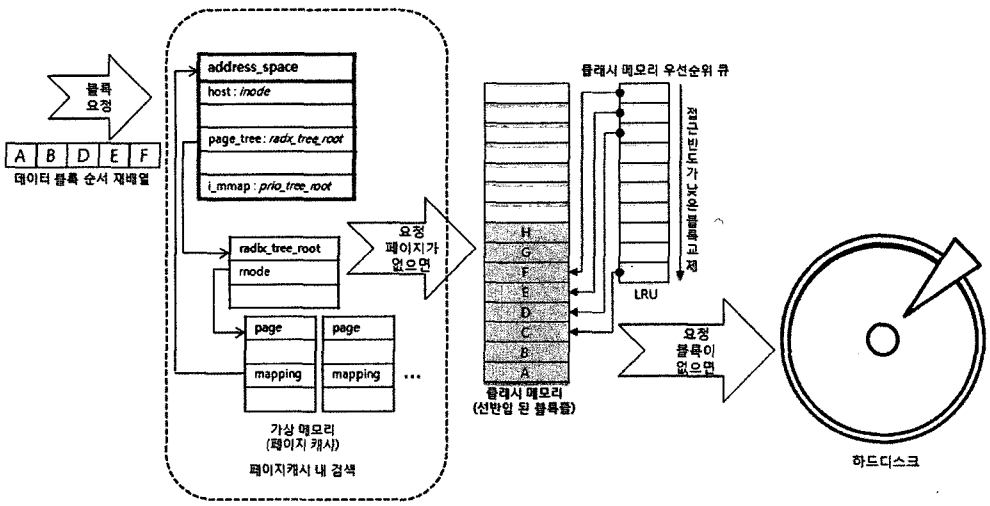

그림 7 n-블록 선반입 하이브리드 하드디스크 입출력 시스템 흐름도

때 하드디스크는 스핀업 동작을 수행 하고 입출력 요청 분석기에서 설정한 우선순위를 고려 하여 블록들을 선반입 하고 플래시 메모리에서 접근빈도가 낮은 블록들을 교체 한다. 호스트가 요청한 블록을 페이지 캐시에서 검색할 때는 래딕스 트리에서 검색을 하는데 그 이유는 인덱스를 수 비트씩 분할하여 계층화 하여 빠른 검색을 보여주기 때문이다.

4.2 n-블록 선반입 하이브리드 하드디스크 입출력 시스템 구조

본 장에서는 n-블록 선반입 하이브리드 하드디스크 입출력 시스템의 구조에 대해 설명한다. 그림 8의 회색 모듈은 본 논문에서 제안하는 시스템의 구성 요소이며, 흰색 모듈은 커널의 구성요소이다. 커널의 입출력 스케줄러의 요청을 입출력 요청 분석기를 통하여 블록의 분석을 실시 하며 분석된 블록은 최근성과 최빈성을 고려하여 우선순위를 할당받는다. 요청한 블록이 페이지 캐시에 있는지 검색을 하여 페이지 캐시에 존재하면 바로 전송하고 그렇지 않을 시에는 하이브리드 하드디스크 관리자를 통하여 플래시 메모리에 선반입 되어 있는지 검색한다. 플래시 메모리에도 존재 하지 않으면 기존의 디바이스 드라이버를 통하여 하드디스크를 스핀업 시키고 블록을 하드디스크로부터 서비스 받는다. 이때 하이브리드 하드디스크 관리자는 선반입을 수행 하며, 전력 측정 관리자 또한 전력 측정을 실시간으로 체크한다. 이와 같이 n-블록 선반입 하이브리드 하드디스크 입출력 시스템은 구성되며 각각의 모듈들은 아래에서 설명한다.

4.2.1 입출력 요청 분석기

입출력 요청 분석기는 읽기 요청을 커널로부터 받아서 분석하고 이를 블록 별로 정제하는 역할을 하는 모

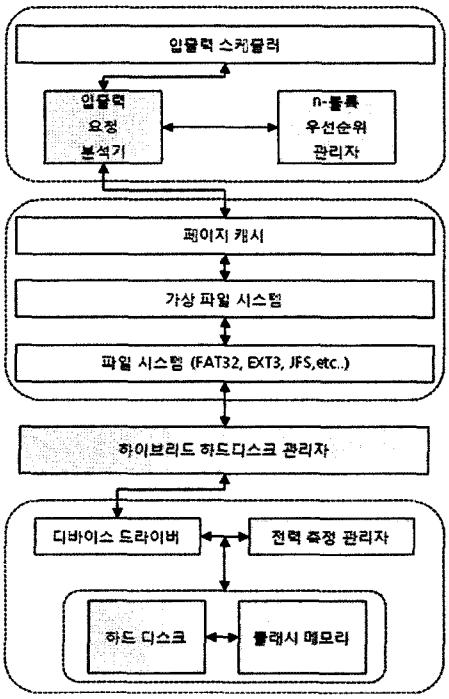

그림 8 n-블록 선반입 하이브리드 하드디스크 입출력시스템 구조도

듈이다. 읽기 요청에 대한 분석은 페이지 캐시에 존재하는 블록과 플래시 메모리에 존재하는 블록을 구분하지 않고 모든 요청을 저장한다. 읽기 요청된 블록들은 연결 리스트를 통해서 관리 하고 실시간으로 빈도, 시간을 기준으로 정렬을 한다. n-블록 우선순위 관리자는 블록 교체 정책 절에서 설명하였다.

4.2.2 하이브리드 하드디스크 관리자

하이브리드 하드디스크 입출력 관리자는 제안하는 서브시스템의 중간에 위치하여 모든 정보를 수집하고 결정을 내리는 모듈이다. 그림 8에서 보는 것과 같이 모듈의 중간에서 모든 데이터의 흐름을 통제하는 모듈이다.

4.2.3 전력 측정 관리자

전력 측정 관리자는 하이브리드 하드디스크의 전력 소모량을 측정하며 그림 9와 같이 크게 두 부분으로 구성되어 있다.

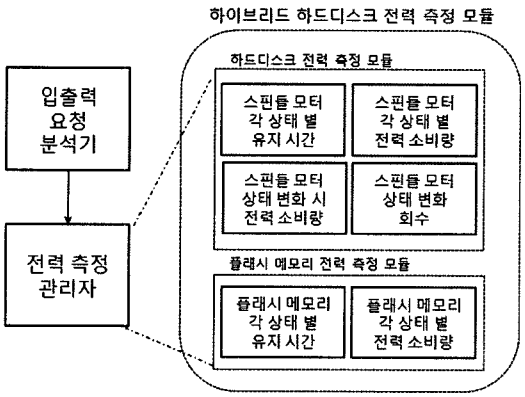

그림 9 하이브리드 하드디스크 전력 측정 모듈 구조

1. 스핀들 모터 상태 변화 기반으로 한 하드디스크의 전력 소모 측정.
2. 플래시 메모리의 전력 소모 측정

4.2.3.1 하이브리드 하드디스크 전력 측정

하이브리드 하드디스크의 전력 소모 측정 모듈은 하드디스크 관리자를 통한 데이터를 근거로 하드디스크의 전력 소모를 측정하며[18], 플래시 메모리 관리자를 통한 데이터를 근거로 플래시 메모리의 전력 소모를 측정한다. 하이브리드 하드디스크가 사용한 총 전력 소모량은 하드디스크의 상태 유지 시간과 하드디스크 스핀들 모터 상태가 변화한 횟수와 플래시 메모리의 상태 별 유지시간과 각각의 상태 별 전력 소비량을 통해 측정할 수 있다.

$$E_{Hybrid} = \sum_{j} P_j T_j + \sum_{k} \sum_{l} N_{kl} E_{kl} + \sum_{m} G_m T_m \quad (5)$$

식 (5)에서 j는 활성, 유휴, 휴면 상태를 나타내고 P_j는 각각의 디스크 상태에 대한 서로 다른 전력 소모량을 나타낸다. T_j는 각각의 디스크 상태에 대한 유지 시간을 나타낸다. E_{kl}은 디스크 상태가 k에서 l 상태로 변할 때 전력 소모를 나타내며 N_{kl}는 k에서 l 상태로 변한 횟수를 나타낸다. 또한 m은 플래시 메모리의 상태(읽기, 쓰기, 지우기 모드)를 나타내고 G_m는 각각의 플래시 메모리의 상태에 대한 서로 다른 전력 소모량을 나타낸다. T_m은 각각의 플래시 메모리의 상태에 대한 유지 시간을 나타낸다.

5. 성능 평가

성능평가를 위해 시행한 실험은 본 논문에서 제안한 n-블록 선반입 하이브리드 하드디스크 입출력 시스템과 기존의 선반입 기법과 비교 및 평가하였다. 실험환경은 인텔 코어 2 듀오 2.4Ghz CPU가 탑재되어 있고 메인 메모리 2 기가바이트가 장착되어 있고, 160 기가바이트의 하이브리드 하드디스크로 구성된 PC를 사용하며, 리눅스 커널 2.6.24 환경에서 C 언어를 사용하여 구현하였다.

5.1 실험 환경

실험은 n-블록 선반입 하이브리드 하드디스크 입출력 시스템과 기존의 선반입 정책을 적용한 시스템과 비교 분석 하였다. 실험에 사용한 하이브리드 하드디스크의 특징은 표 1과 같다. 하드디스크와 하드디스크에 내장된 플래시 메모리의 전송속도, 전력 소비에 관한 특징들이다[19,20].

시간에 관한 상수 r은 100으로 정하고 사용하였다. r의 값이 너무 커지면 시스템 성능이 떨어지고 너무 작으면 블록의 우선순위를 결정하는 것이 정확해 지지가 않기 때문에 이 부분에 대해서는 실험을 통하여 100이라는 수치를 산출하였다.

표 1 실험에 쓰인 하이브리드 하드디스크 환경

변수 \ 모델		하이브리드 하드디스크 (SAMSUNG HM16HJI)
용량(GB)		160
분당 디스크 회전 수 (RPM)		5400
평균 탐색 시간(ms)		12
평균 대기 시간(ms)		5.6
NV Cache 읽기(max)		100MB/s
NV Cache 쓰기 (max)		9MB/s
전력 소비 (W)	활성상태	2.00
	유휴상태	0.60
	대기상태	0.25
	휴면상태	0.20
	탐색상태	2.10
	스핀업	4.50
	NV Cache 읽기	0.85
	NV Cache 쓰기	0.75

5.2 실험 트레이스

본 실험에서는 BLTK(Linux Battery Life Tool Kit) [21]를 사용하여 전력 소모량과 응답속도에 관한 실험을 진행 하였다. 응답속도는 BLTK에 내장되어 있는 프로그램이 화면에 표시되는 시간을 의미한다 이것을 통하여 입출력 성능을 측정한다. 본 실험에서는 문서편집기, 컴파일러, 그림판, 동영상 플레이어를 사용하여 응답속도를 측정하여 입출력 성능을 비교하고, 플래시 메모리의 크기변화에 따른 전력소비량 비교, 스핀업 횟수, 적중률을 비교한다.

리눅스 커널 2.6.24 버전 에서 선반입 기법을 사용하지 않는 "일반 리눅스" 시스템, 기존의 선반입 정책인 "RA" 방법, "AMP" 방법, "EXCES" 방법, 본 논문에서 제안한 "최소 n-블록" 방법과 "동적 n-블록" 방법에 대한 성능을 비교 분석한다.

5.3 부팅 프로세스

리눅스 시스템을 부팅하여 최종적으로 명령을 내릴 수 있는 쉘이 화면에 표시 될 때까지의 시간을 부팅 시간이라 정하고 실험을 총 100회 반복 진행 하였다. "일반 리눅스" 보다 "동적 n-블록"을 사용한 선반입을 하였을 때 16초 가량 부팅 시간이 단축 하였다. "AMP"와 "RA"는 선반입 정책을 적용 하지 않은 "일반 리눅스" 보다는 빠른 부팅 시간을 보여 주었지만 n-블록 선반입 시스템 보다는 느린 부팅 시간을 보여 주었다. 부팅할 때 하드디스크를 접근하지 않고 플래시 메모리의 선반입된 블록들을 사용하기 때문에 "동적 n-블록" 선반입 시스템이 "일반 리눅스" 보다 179.80% 감소하였다. 모바일 컴퓨팅을 하기에 좋은 결과라 볼 수 있다. 스핀업 횟수는 "일반 시스템" 방법, "AMP" 방법, "RA" 방법의 경우에는 스핀업 상태로 유지되어 있기 때문에 시스템이 시동될 때 한번만 스핀 다운에서 스핀업 상태로 변화 하였고 "최소 n-블록" 방법의 경우에는 5번 변화하였다. "동적 n-블록" 방법의 경우에는 3번의 변화가 일어났다. "일반 리눅스" 방법의 경우에는 끊임 없이 하드디스크로 입출력 요청을 하기 때문에 대기 상태까지 상태 변화를 하지 않았다. n-블록 선반입 시스템의 경우에는 하드디스크의 접근을 최소화하기 때문에 스핀다운 알고리즘에 따라 유휴 상태와 대기 상태로 하드디스크의 상태가 변화하였다.

표 2 컴퓨터 부팅 시간 및 전력 소모

분류 \ 장치	n-블록 선반입 시스템		일반 리눅스	AMP	RA
	최소 n-블록	동적 n-블록			
활성 상태 유지 시간 (s)	20	10	62	36	58
유휴 상태 유지 시간 (s)	5	5	27	45	27
대기 상태 유지 시간 (s)	50	58	0	0	0
휴면 상태 유지 시간 (s)	0	0	0	0	0
회전 대기 시간(s)	4	4	4	4	4
스핀들 모터 변화(회)	5	3	1	1	1
시스템 부팅 시간(s)	79	77	93	85	89
부팅 시 필요한전력 소비량(J)	78	51	142.7	103.5	136.7

전력 소모와 응답 속도 면에서 보았을 때 "일반 리눅스" 방법과 비교하여 "최소 n-블록" 방법의 경우에는 82.95% 전력 감소를 보였고 12.05%의 시스템 응답 속도의 향상을 보였다. 또한 "동적 n-블록" 방법과 "일반 리눅스" 방법과 비교하였을 경우에는 179.80%의 전력 소모 감소를 보였으며 시스템 응답 속도는 20.78%의 향상을 보였다.

n-블록 선반입의 경우에는 플래시 메모리에 선반입 되어 있는 블록들을 사용 하여 부팅을 수행하기 때문에 하드디스크의 상태와 상관없이 빠른 부팅과 저전력을 유지 할 수 있었다.

5.4 응답 속도 및 전력 소비량

자주 쓰이는 응용프로그램을 실행 할 때 걸리는 시간을 응답시간이라 하고 실험을 100회 반복 하여 측정하였다. 그림 10은 6가지 방법에 대한 응답속도를 측정한 평균 결과를 보여 준다. 컴파일러와 문서 편집기 같이 문서 편집 응용 프로그램의 경우에는 작은 크기의 파일들을 병렬적으로 요청 하므로 병렬적인 블록 요청을 효율적으로 서비스 하기 위한 선반입 방법인 "AMP"와 n-블록 선반입 방법이 유사한 성능을 보여 주며, "일반 리눅스"의 경우에는 하드디스크에 할당된 블록들을 탐색하는데 걸리는 시간이 추가 적으로 소모 되므로 응답 속도가 높게 나왔다. "RA"의 경우에는 연속적인 데이터를 선반입 하기 위한 선반입 방법이므로 문서 편집의 응용에서는 낮은 성능을 보여 주었다. 또한 "EXCES" 방법은 저전력을 위하여 하드디스크의 블록 데이터를 페이지 단위로 선반입을 수행하고 있으므로 "최소 n-블록" 방법과 유사한 성능을 보여 주었다. 동영상 플레이어와 그림판 같은 브라우징 응용 프로그램의 경우에는 연속적인 블록들을 입출력 하기 때문에 모든 방법이 유사한 성능을 보여주었다. 하지만 그림판의 경우에는 동영상 플레이어와 다르게 입출력 요청의 크기가 작으므로 "일반 리눅스"에 비교하여 "최소 n-블록" 방법과

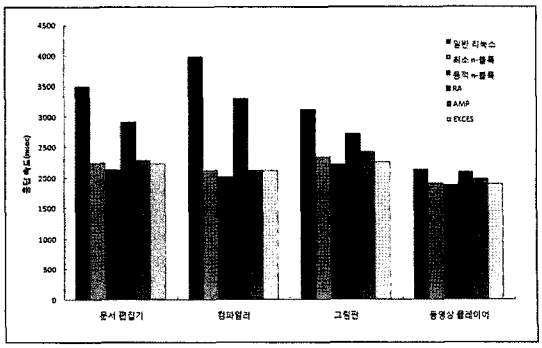

그림 10 프로그램 별 응답 속도

"동적 n-블록" 방법의 경우 응답시간이 각각 33.32%와 41.16%의 향상을 보였다. 왜냐하면 제안하는 n-블록 선반입 방법은 하드디스크 보다 접근 속도가 빠른 플래시 메모리에 선반입을 수행하는 방법이므로 선택된 n-블록이 가상메모리로 할당되므로 빠른 응답속도를 보여준다.

자주 쓰이는 응용프로그램을 실행할 때 소모되는 전력소모량을 100회 반복하여 측정하였다. 그림 11은 6가지 방법에 대한 전력소모를 측정한 평균 결과를 보여준다. 그림 10에서와 같이 그림 11의 실험에서도 브라우징 응용프로그램과 편집 응용프로그램으로 나눌 수 있다. 컴파일러와 문서 편집기의 경우에는 병렬적인 블록 요청을 하기 때문에 하드디스크 탐색 시간이 길어져서 접근이 많은 "일반 리눅스" 방법을 수행할 때 가장 많은 전력 소모를 보였다. 그림판과 동영상 플레이어 같은 연속적인 블록 요청을 하는 브라우징 응용 프로그램을 사용할 때 "동적 n-블록" 방법은 "일반 리눅스" 방법에 비하여 19.57% 전력 소모 감소를 보여주었다. "EXCES" 방법은 "최소 n-블록" 방법과 유사한 성능을 보여 주었다. 그 이유는 "EXCES" 방법은 플래시 메모리에 페이지 단위로 선반입을 수행하는 방법이므로 하드디스크를 스핀다운 상태로 유지하는 시간이 메모리에 선반입을 수행하는 다른 방법보다 길기 때문이다. 하지만 동적으로 블록의 숫자를 조정하는 "동적 n-블록" 방법보다 "EXCES" 방법이 평균 8.28%의 많은 전력 소비를 보여준다.

플래시 메모리의 크기를 변경하여 선반입 할 수 있는 캐시의 크기를 변경하고 그림 12와 같이 100회 반복 실험을 하였다. 그림 12는 플래시 메모리 선반입 캐시 크기의 변화에 따른 평균 전력 소비량을 나타낸다. 선반입 캐시 크기에 따른 전력 소비의 추이를 살펴보면 선반입 캐시의 크기가 증가함에 따라 "최소 n-블록" 방법과 "동적 n-블록" 방법의 전력 소모량은 줄어듦을 알 수 있다. 그러나 2기가 바이트 이상의 선반입 캐시에서는

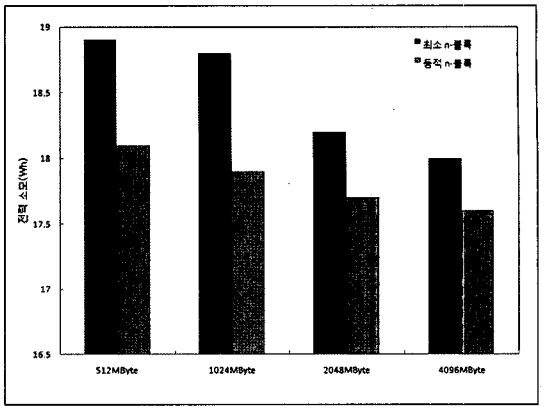

그림 12 플래시 메모리 선반입 캐시 크기 변화에 따른 전력 소비량

전력소모량의 차이가 크게 나지 않는다. 따라서 2기가 바이트가 적절한 캐시 크기로 사료 된다. "동적 n-블록" 방법이 "최소 n-블록" 방법 보다 평균 3.52% 적은 전력 소모량을 보여 주고 있다. 그 이유는 동적 선반입의 경우에는 우선순위가 높은 블록의 경우에는 동적으로 n개의 개수를 변경하여 선반입을 수행 하기 때문이다.

5.5 스핀업 횟수 및 선반입 적중률

그림 13은 "EXCES", "RA", "AMP", "일반 리눅스", "최소 n-블록", "동적 n-블록"의 6가지 방법에 대한 평균 스핀업 횟수를 나타낸다. 스핀업 횟수는 시스템의 성능과 전력 소비를 결정하는 요소이기 때문에 스핀업이 얼마나 자주 일어나는지에 대한 실험을 진행하였다. 하이브리드 하드디스크를 사용하여 실험을 진행한 "최소 n-블록" 방법, "동적 n-블록" 방법 및 "EXCES" 방법은 선반입 캐시의 크기를 2 기가 바이트로 고정 하였다. "동적 n-블록" 방법의 경우에 "일반 리눅스" 방법보다 36.75% 적은 스핀업 횟수를 보였으며, "최소 n-블록"과 비교 하였을 때에는 19.23%, "RA" 방법과 비교 하였을 때는 26.83%, "AMP" 방법과 비교하였을 때에는 21.45%

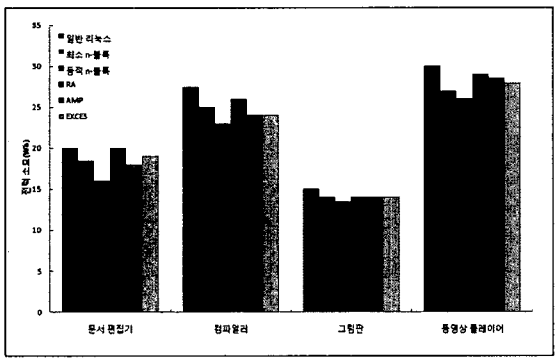

그림 11 프로그램 별 전력소모

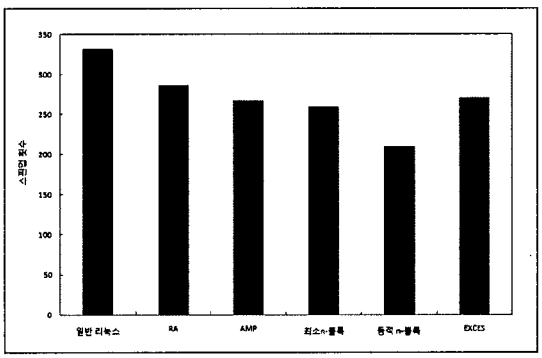

그림 13 스핀업 횟수의 비교

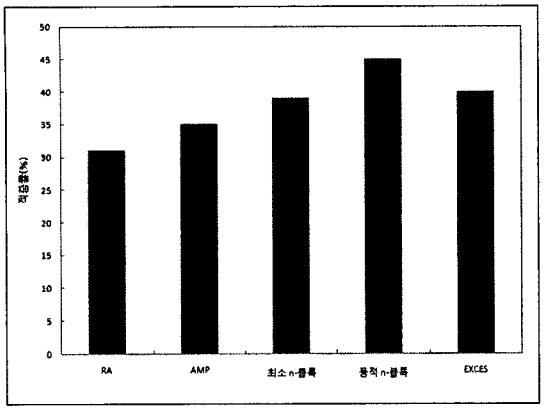

그림 14 적중률의 비교

스핀업 횟수가 감소 함을 보여주었다. 이는 "동적 n-블록" 선반입 기법은 선반입 캐시에 동적으로 n개의 블록을 선반입 함으로서 하드디스크를 거의 접근 하지 않았기 때문이다. 또한 같은 용량의 하이브리드 저장장치를 사용할 때 "동적 n-블록" 방법이 "EXCES" 방법과 비교하여 3.85%의 스핀업 횟수 감소를 보여 주었다.

그림 14는 "EXCES", "RA", "AMP", "최소 n-블록", "동적 n-블록"의 5가지 방법에 대한 평균 적중률을 비교하였다. 하이브리드 하드디스크를 사용하였으며, "EXCES", "최소 n-블록"과 "동적 n-블록" 방법은 하이브리드 저장장치를 사용하였으며, 선반입 캐시의 크기를 2 기가 바이트로 고정 하였다. "동적 n-블록" 방법이 44.45%의 가장 좋은 적중률을 보여 주었으며, "RA"방법이 32.19%의 가장 낮은 적중률을 기록 하였다. "RA" 방식은 연속된 블록들에 대해서 선반입이 이루어 지므로 순차적이지 않은 블록 요청이나 병렬적인 요청에 대해 낮은 성능을 보여 준다. 그에 비해 "동적 n-블록" 방법은 선반입 캐시의 우선순위에 따라 블록들의 선반입이 되므로 적중률이 높게 나왔다.

6. 결론

본 논문에서는 저전력과 시스템 성능 향상을 위해 블록 기반의 입출력 패턴 분석과 선반입 기반의 서브시스템을 설계 및 구현하고 이를 평가하여 일반적인 하드디스크와 제안한 시스템과의 성능 평가를 수행하였다. 본 논문에서 제안한 n-블록 선반입 기법은 최소 n-블록 방법과 동적 n-블록 방법으로 나눌 수 있다. 이 방법들은 하이브리드 저장장치 기반으로 설계되었으며 이 방법들을 사용하면 기계적인 장치를 사용하는 하드디스크의 단점인 전력 소모 측면과 시스템 입출력 성능을 향상시킬 수 있었다. 또한 제안한 방법들은 하드디스크에 저장되어 있는 자주 쓰이는 블록을 선별하고 블록의 개수를 결정하여 비휘발성 메모리인 플래시 메모리로 선반입 하기 때문에 지속적으로 하드디스크를 스핀다운 상태로 둘 수 있어 전력 소모를 줄일 수 있으며, 상대적으로 빠른 플래시 메모리의 접근 속도로 인해 빠른 입출력 성능을 보장할 수 있었다. 성능평가를 통해 부팅 시에는 "동적 n-블록" 방법이 기존의 방법인 "AMP" 방법과 비교하여 9.05%의 시스템 응답 속도 향상과 11.11% 전력소모량 감소를 실험을 통해 확인하였다. 또한 하이브리드 저장장치를 사용하는 "EXCES" 방법과 비교하여 3.43%의 시스템 응답 속도 향상 및 8.28%의 전력 소모 감소를 확인하였다.

참 고 문 헌

[1] Hong-jae Lee, "Toward Understanding Hard disk," Electronic Times, April 2003.

[2] Jungwan Choi, Youjip Won, "Power Constraints: Another Dimension of Complexity in Continuous Media Playback," *Lecture Note in Computer Science, Springer-Verlag*, vol.2515, pp.288-299, November 2002.

[3] Windsor W. Hsu, Alan Jay Smith and Honesty C. Young, "The Automatic Improvement of Locality in Storage Systems," *ACM Transactions on Computer Systems(TOCS)*, vol.23, Issue 4, pp.424-473, November 2005.

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

[5] http://www.intel.com/design/flash/nand/turbo-memory/index.htm

[6] Kwanghee Park, Junsik Yang, Joon-Hyuk Chang, and Deok-Hwan Kim, "Anticipatory I/O Management for Clustered Flash Translation Layer in NAND Flash Memory," *ETRI Journal*, vol.30, no.6, pp.790-798, December 2008.

[7] Shea Yun Lee, "Hybrid HDD: An Application of Flash Memory for Enhancing Storage Performance Characteristics," *Journal of KIISE*, vol.25, no.6, pp.29-34, June 2007.

[8] Geunhyung Lee, Deok-Hwan Kim, "Design and Implementation of Hybrid Hard Disk Simulator based on Linux Environment," IEEK Summer Conference, pp.649-650, June 2008.

[9] Young-Jin Kim, Sung-Jin Lee, Kangwon Zhang, and Jihong Kim, "I/O performance optimization technique for hybrid hard disk-based mobile consumer devices," *IEEE Transactions on Consumer Electronics*, vol.53, Issue 4, November 2007.

[10] Timothy Bisson, Scott A. Brandt. "Adaptive Disk Spin-Down Algorithms in Practice," *3rd USENIX Conference on File and Storage Technologies*, Work in Progress Proceedings, FAST, 2004.

[11] Timothy Bisson, Scott Brandt, and Darrell D.E. Long, "NVCache: Increasing the effectiveness of disk spin-down algorithms with caching," *Proceedings of the 14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, IEEE, 2006.

[12] Todd C. Mowry, Angela K. Demke, and Orran Krieger, "Automatic Compiler-Inserted I/O Prefetching for Out-of-Core Applications," In *USENIX 2nd Symposium on Operating Systems Design and Implementation*, October 1996.

[13] Daniel Plerre Bovet, Marco Cesati, "Understanding the Linux Kernel (3/E)," *O'REILLY*, November 2005.

[14] A. J. Smith, "Sequential Program Prefetching in Memory Hierarchies," *Computer*, pp.7-21, December 1978.

[15] B. Gill and L. Bathen. AMP: Adaptive multistream prefetching in a shared cache. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies(FAST)*, 2007.

[16] Luis Useche, Jorge Guerra, Medha Bhadkamkar, Mauricio Alaron, and Raju Rangaswami, "EXCES: EXternal Caching in Energy Saving Storage Systems," In *HPCA-13: Proceedings of the 13th International Symposium on High-Performance Computer Architecture*, 2008.

[17] Un-Keun Yoon, Han-joon Kim, "Sequential Pattern-based Prefetching Technique for Hybrid Storage Device," *KIISE Fall Conference, Korea Computer Congress*, vol.35, no.2, pp.23-28, October 2008.

[18] Youngwook Go, Geunhyung Lee, Kwang-Hee Phark, Deok-Hwan Kim," Design and Implementation of Power Consumption Measurement Simulator for Hard Disk on Mobile computing System," *KIISE Summer Conference, Korea Computer Congress*, vol.35, no.2, pp.459-463, October 2008.

[19] http://www.samsung.com/us/consumer/detail/detail.do?group=computersperipherals&type=harddiskdrives&subtype=hybridhdd_flashon&model_cd=HM16HJI

[20] Samsung Elec., "NAND-type Flash Memory, "http://www.samsung.com/Products/Semicon ductor/Flash/index.htm.

[21] L. Brown, K. A. Karasyov, V. P. Lebedev, A. Y. Starikovskiy, and R. P. Stanley. Linux laptop battery life: Measurement tools, techniques, and results, February 2007.

양 준 식

2008년 인하대학교 컴퓨터정보공학부 학사. 2008년~현재 인하대학교 전자공학과 석사과정. 관심분야는 임베디드 시스템 소프트웨어, 하이브리드 저장장치, 모바일 플랫폼 등

고 영 욱

2008년 강원대학교 정보통신 전자공학부 전자공학과 학사. 2008년 2월~현재 인하대학교 전자공학과 석사과정. 관심분야는 임베디드 시스템 소프트웨어, 하이브리드 저장장치, 저전력 시스템 등

이 찬 근

2005년 Univ. of Texas at Austin 전산학과 박사. 2007년~현재 중앙대학교 컴퓨터공학부 조교수. 관심분야는 실시간 소프트웨어, 수행시간 모니터링, 소프트웨어 테스팅 등

김 덕 환

2003년 한국과학기술원 정보통신공학과(컴퓨터공학)박사. 2006년~현재 인하대학교 전자공학부 부교수. 관심분야는 임베디드 시스템, 시스템 소프트웨어, 스토리지 시스템, 멀티미디어, 시각정보처리 등