효율적인 콘텐츠 저장을 위한 모바일 스토리지 기술 동향

고영욱, 양준식, 조원희, 이동환, 김덕환* (인하대학교)

차 례

1. 서론

지난 수십 년 간 진행되어 온 정보혁명은 전문가들이 예측한 것보다 훨씬 빠른 속도로 진행되어 왔고 급속한 IT기술발전에 힘입어 우리사회 전반을 변화시켜왔다. 최근 10여 년간 IT기술과 관련하여 콘텐츠와 모바일 이라는 단어가 우리사회에 가장 큰 영향을 미치는 키워드로 대두되게 되었고, 콘텐츠와 모바일 기술이 우리의 사회, 산업, 비즈니스, 교육, 오락, 의료, 교통 등 거의 모든 영역에서 지대한 영향을 발휘하며, 우리 삶의 터전과 의식구조 자체를 변화 시킬 하나의 거대한 흐름으로 우리 앞에 다가오고 있다.

과거의 모바일기기는 주로 문자나 멜로디의 콘텐츠를 주고받았으나 최근 데이터 전송속도가 빨라짐에 따라 모바일 환경에서 멀티미디어 콘텐츠의 활용이 가능하게 되었고, 앞으로는 콘텐츠를 언제 어디서나 접속할 수 있는 유비쿼터스 사회로 발전해 나갈 것으로 전문가들은 예측하고 있다[1].

세계적 흐름이 인터넷과 IT기술을 기반으로 한 모바일 환경으로 빠르게 개편되고 있는 현실에서, 특히 우리나라는 이 분야에서 지금까지 매우 경쟁력 있는 기술 개발과 산업 체제를 위해 노력을 해 왔다. 예를들어 휴대폰, 모바일 디지털TV, 네비게이션, PMP, PDA, MP3 등의 분야에서 세계적 우위를 차지하고 있다. 최근 기술개발 및 기술표준화를 위해서 노력하고 있는 휴대인터넷 서비스인 WiBro, 무선인터넷 플랫폼인 WIPI, 디지털 멀티미디어 방송(DMB), 텔레매틱스 기술, 유비쿼터스 환경의 RFID 등 모바일환경의 킬러 어플리케이션(Killer application)이 등장하고 있다.

모바일기기에서 콘텐츠의 검색, 처리 및 저장해야 할 콘텐츠의 양은 기하급수적으로 늘어나게 되고 [1]이러한 콘텐츠의 홍수 속에서 사용자에게 유용한 콘텐츠를 검색하고, 기록 및 저장하는 일은 정보보안 문제와 함께 매우 중요한 기능으로 떠오르고 있다. 또한 비디오, 메일, 영상녹화, 대형 게임의 수행, 대형 소프트웨어의 실행 등을 위해서는 모바일 기기에서의 콘텐츠 저장 용량을 데스크톱 PC나 노트북 수준으로 증가해야 하며, 이에 적합한 기술개발 노력에 박차를 가하고 있다. 모바일 기기 저장장치의 시장규모 또한 매년 증가하여 전체 저장장치 시장에서 차지하는 비율이 2000년 5%에서 2003년 12%, 2010년에는 25% 이상으로 확대될 것으로 전망된다[2].

본 논문에서는 효율적인 콘텐츠 저장을 위해 모바일 기기에서 가장 많이 사용되는 하드디스크, 플래시메모리, SSD의 특성, 제약 조건 등에 대하여 설명하고 국내외 기술동향에 대해 살펴본다.

2. 본론

2.1 하드디스크 기술 동향

모바일 기기를 이용하여 다양한 콘텐츠를 효율적으로 활용할 수 있게 되었으며, 이는 모마일 기기에 달린 저장장치인 하드디스크로 인한 것이다. 하드디스크가 있기 전에는 종이테이프 등의 저장매체를 사용하였는데, 용량

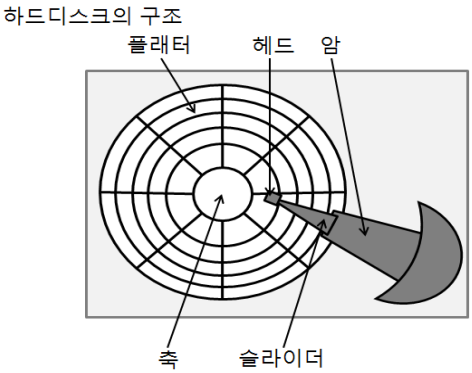
1) 본 논문은 2008년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구이며 (KRF-2008-313-D00822), 지식경제부와 한국산업기술 재단의 전략기술인력양성사업으로 수행된 연구결과임.

도 매우 작았으며 또한 사용하기 불편했다. 하드디스크가 발명되면서 모바일 기기에서 콘텐츠를 저장할 수 있는 용량은 급속도로 발전하기 시작했다.

2.1.1 하드디스크의 기계적 구성

하드디스크는 그림 1과 같이 플래터, 헤드, 암, 액추에이터, 축, 스핀들 모터, 논리보드 등으로 구성되어 있다[3]. 하드디스크는 플래터라고 불리는 원판 디스크를 사용하고 있다. 플래터는 양쪽 면에 자성물질로 코팅되어 있는데, 여기에 정보를 저장한다. 플래터의 한가운데에는 구멍이 있어서 축에 장착된다. 이 축은 스핀들 모터에 연결되어 있다. 그러므로 플래터는 이 모터에 의해 빠른 속도로 회전하게 된다.

헤드는 전자기적으로 데이터를 읽고, 쓰는 장치이다. 헤드는 슬라이더에, 슬라이더는 암에 장착되어 있다. 각각의 장치는 단일집합체로서 기계적으로 연결되어 있으며, 디스크 표면 위에 위치하고 있다. 액추에이터는 암에 동력을 연결해 주는 기능을 한다. 드라이브 바깥에 있는 논리보드는 하드디스크의 여러 장치의 동작을 조절하고, PC의 여타 부분들과 정보를 교환한다. 하드디스크의 스핀들 모터의 회전속도는 이 논리보드 회로에 의해 큰 편차 없이 안정적으로 조절된다.

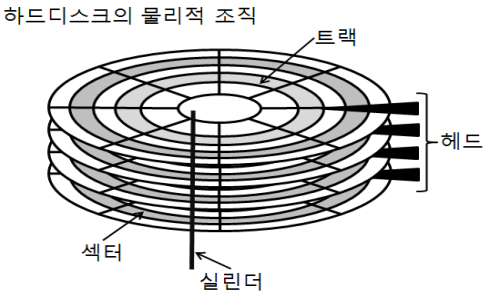


▶▶ 그림 1. 하드디스크의 기계적 구성

2.1.2 하드디스크의 플래터 구조 및 섹터 구조

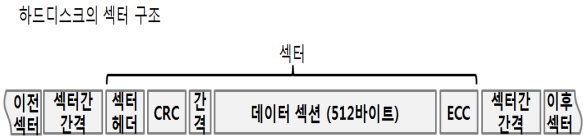
하드디스크의 플래터는 트랙, 섹터, 실린더로 구성되어 있다. 그림 2는 하드디스크의 플래터의 구조를 보여준다[3]. 헤드는 플래터의 양쪽 표면에 한 개씩 위치하고 있다. 이 그림에서는 4개의 플래터가 있고, 각각의 플래터 위에는 5개의 트랙이 있다. 실린더는 그림에서 실선에 의해 가로질러지는 8개의 트랙(표면당 2개)들로 구성된다. 모든 데이터는 플래터의 트랙에 기록된다. 트랙은 각각의 플래터 표면 위에 놓인 동심원들이다. 트랙은 바깥쪽에서 시작해서 안쪽으로 가면서 0부터 숫자로 세어진다. 최근 출시되고 있는 하드디스크는 각각의 플래터에 수십만 개의 트랙을 가지고 있다. 헤드가 디스크 안쪽에서 바깥쪽으로 움직임에 따라 데이터에 접근된다. 헤드는 암을 구동하는 액추에이터에 의해 동작된다. 플래터 동심원상에 데이터가 기록되어 있기 때문에 디스크의 어떤 부분으로도 쉽게 접근할 수 있도록 해준다. 그래서 하드디스크를 랜덤 액세스 저장장치라고 부른다.

각각의 트랙에는 수천 바이트의 데이터가 저장된다. 디스크상의 최소 저장단위를 트랙으로 한다면 상당히 낭비가 심할 것이다. 왜냐하면 작은 파일을 커다란 트랙에 기록하면 큰 공간을 허비하는 것을 의미하기 때문이다. 따라서 트랙은 섹터라는 작은 단위들로 나누어져야 한다. 각각의 섹터는 512바이트의 정보를 보유한다. 또한 각 섹터에는 오류 탐색과 수정, 그리고 내부 드라이브 컨트롤을 위해 사용되는 수십 개의 바이트를 추가로 보유하고 있다.



▶▶ 그림 2. 하드디스크의 플래터 구성

그림 3은 섹터의 구조이다. 섹터와 섹터 사이에는 섹터를 구분 짓는 섹터간 간격이 있다. 1개의 섹터는 섹터 헤더, 섹터 헤드를 위한 CRC, 간격, 데이터 기록부, ECC로 구성되어 있다. 섹터 헤더는 섹터의 주소를 나타내며, CRC는 순환 중복 검사를 한다. 이것은 섹터 헤더의 끝부분에 위치하여 섹터 주소를 확실히 해준다. 데이터 섹션은 실제적인 데이터가 있으며, ECC는 데이터 섹션 뒤에 위치하는데 읽고 있는 데이터가 타당한 것인지 확인하기 위해 사용된다. 간격은 섹터의 영역들을 분리하는데 필요한 여분의 공간, 또는 컨트롤러가 더 많은 비트를 읽기 전에 이미 읽은 것을 처리해야 할 시간적 여유를 갖는데 필요한 공간이다[3].

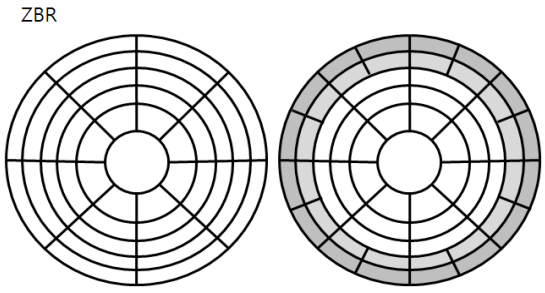


▶▶ 그림 3. 하드디스크의 섹터 구성

2.1.3 구역밀도 기록 방식(ZBR: Zone Bit Recording)

하드디스크의 용량과 속도를 증진시켜 왔던 한 가지 방법은 디스크의 바깥쪽에 있는 커다란 트랙을 효율적으로 사용하는 것이다. 초기의 하드디스크는 트랙 사이에서 변화되는 복잡한 배열을 다룰 수 없었기 때문에 모든 트랙의 섹터 수는 같다. 바깥쪽에 있는 트랙이 안쪽에 있는 트랙 보다 원주의 길이가 2배 정도 길다. 안쪽에 있는 동심원 트랙 내에 가능한 빽빽하게 데이터를 담으려 했다. 바깥쪽의 동심원에는 비트밀도를 줄여서 안쪽에 저장되는 데이터와 같은 양이 저장될 수 있도록 하였다. 그러므로 바깥쪽 트랙은 비효율적이다. 왜냐하면 바깥쪽 동심원에도 안쪽 동심원과 같은 밀도로 저장된다면 그만큼 더 많은 섹터를 가질 수 없기 때문이다. 이런 낭비 공간을 줄이기 위해서 최신 하드디스크는 ZBR 기술을 사용한다[4].

그림 4는 ZBR을 보여준다. ZBR은 트랙들을 디스크 중심으로부터 그들의 거리에 따라 구역으로 묶여진다. 각각의 구역에는 수많은 섹터 수가 할당된다. ZBR의 한 가지 특징으로는 안쪽 실린더를 읽을 때보다 바깥 쪽 실린더를 읽을 때에 디스크 데이터 전송이 빠르다. 이유는 바깥쪽 실린더는 더 많은 데이터를 포함하고 있지만, 플래터의 각 속도(angular velocity)는 어떤 트랙이 읽혀지더라도 같기 때문이다.

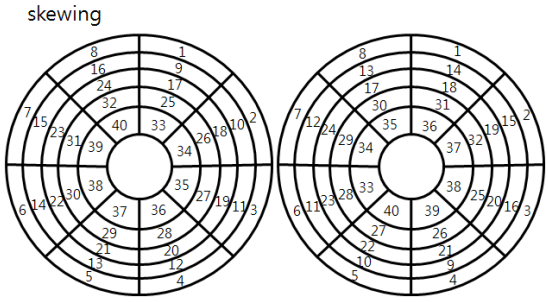


▶▶ 그림 4. ZBR (Zone Bit Recording)

2.1.4 스큐잉(Skewing)

하드디스크는 여러 개의 트랙으로 구성되어 있다. 헤드가 트랙에서 트랙으로 전환 할 때 헤드가 움직이는 지연시간이 발생한다. 이 문제를 해결하기 위해서 트랙을 바꿀 때 인접한 트랙의 시작 섹터를 조정함으로써 해결 할 수 있다. 이것을 스큐잉 이라 하면, 그림 5은 스큐잉을 적용한 플래터를 보여준다.

그림 5의 왼쪽 그림은 일반적인 플래터이고, 오른쪽 그림은 스큐잉이 적용된 플래터이다. 왼쪽 플래터는 섹터의 순서대로 번호가 정해져 있지만, 오른쪽 플래터는 3개의 섹터 오프셋(offset)이 적용되어 있다. 왼쪽 플래터는 1번 트랙을 처리하고 2번 트랙으로 전환 할 때 헤드가 움직이는 시간 때문에 헤드는 2번 트랙의 20번 섹터에 위치하게 된다. 그러면 2번 트랙의 9번 섹터를 처리하기 위해서는 플래터가 한 번 더 회전을 해야 하기 때문에 지연시간이 발생한다. 오른쪽 플래터에서는 1번 트랙을 처리하고 2번 트랙으로 전환 할 때 헤드가 움직이는 시간을 계산하여서 2번 트랙에 3개의 섹터 오프셋을 적용한다. 3개의 섹터 오프셋을 적용하면, 헤드의 이동 시간과 플래터의 회전속도의 타이밍이 맞기 때문에 불필요한 플래터 회전 없이 바로 2번 트랙의 9번 섹터를 처리 할 수 있다.



▶▶ 그림 5. 스큐잉(Skewing)

2.1.5 하드디스크의 스케줄링 기법

하드디스크 스케줄링 기법은 하드디스크 입출력 대기중인 요청들의 처리순서를 결정하는 기법으로서 시스템의 성능향상을 위해 사용된다. 하드디스크의 스케줄링 기법의 성능은 크게 세 가지로 평가 할 수 있다. 첫째, 단위 시간당 얼마나 많은 디스크 입출력 요구를 서비스하는가를 평가하는 단위 시간당 처리량과 둘째, 각 디스크 입출력 요구에 대해 얼마나 빠른 시간 내에 서비스 하는가를 측정하는 평균응답시간, 셋째, 예측성 판단을 위한 요소로서 응답시간의 분산에 사용되는 응답시간의 예측성 등을 사용하여 하드디스크의 스케줄링을 평가 할 수 있다. 이를 위한 스케줄링 기법으로는 FCFS(First

Come First Served), SSTF(Shortest Seek Time First), SCAN, C-SCAN, LOOK 등이 있다[5,6,7,8].

기존의 비 실시간 디스크 스케줄링 알고리즘은 서비스 순서를 결정할 때 각각 입출력 요구가 요청하는 데이터의 물리적 위치만을 고려하기 때문에 실시간적인 특성을 갖는 입출력 요구, 즉 종료시한 이내에 서비스를 받아야 하는 디스크 입출력 요구들을 서비스 하는 데에는 적합하지 않다. 이러한 문제점을 해결하기 위하여 여러 가지 실시간 디스크 스케줄링 알고리즘이 제안되었고 그것에 관한 많은 연구 결과가 발표 되었다. 그중 대표적인 실시간 디스크 스케줄링 알고리즘은 EDF(Earliest Deadline First)[9,10], EDF에 SCAN기법을 추가한 P-SCAN(Priority SCAN)[11] SCAN-EDF[12], 큐를 이용한 SSEDO/ SSEDV(Shortest Seektime and Earliest Deadline by Ordering/Value)[13], 디스크 입출력 요구들의 긴급도에 기반 한 UG-SSTF(Urgent Group and Shortest Seek Time First)[14] 등이 있다.

EDF(Earliest Deadline First)는 가장 잘 알려진 실시간 디스크 스케줄링 알고리즘으로 종료시간에 가장 가까운 요구를 먼저 처리해 주는 알고리즘이다. 즉, 종료시간이 가장 빠른 요구가 가장 높은 우선순위를 갖고 서비스를 받는다.

P-SCAN(Priority SCAN)은 디스크 큐 안에 있는 모든 입출력 요구들을 몇 개의 우선순위 단계로 분류하는 전략에 기반을 두고 있다. 그리고 각각의 우선순위 단계에서는 SCAN을 이용하여 입출력 요구를 서비스 해 준다. 가장 우선순위가 높은 단계에 속해 있는 입출력 요구를 모두 서비스하면 다음으로 우선순위가 높은 단계에 속해 있는 입출력 요구들을 서비스 해 준다. 각각의 입출력 요구를 서비스 한 후에 디스크 스케줄러는 더 우선순위가 높은 입출력 요구가 서비스를 기다리고 있는지 확인한 후 있으면 우선순위 단계를 높여서 서비스를 해 준다.

SCAN-EDF는 EDF에서 비슷한 종료기한을 가진 요구들을 SCAN방식으로 처리하여 탐색시간을 줄인 알고리즘이다. 디스크에 대한 입출력 요구들이 도착하면 종료시한에 따라 큐 안에 정렬해 넣고, EDF와 마찬가지로 종료기한이 가장 빠른 것부터 서비스 한다. 이때 종료시한이 같은 입출력 요구가 있다면 SCAN의 서비스 순서대로, 즉 요청하는 데이터가 있는 트랙을 순차적으로 탐색하면서 서비스해 준다.

SSEDO/SSEDV(Shortest Seektime and Earliest Deadline by Ordering/Value)는 Chen에 의해 제안된 알고리즘으로 디스크 큐 속에 있는 입출력 요구들을 종료시한에 따라 정렬하고, 알고리즘의 수행시간을 줄이기 위해 m개의 선두 입출력 요구들로 정의된 윈도우에서 스케줄링 결정을 내린다. 이 두 알고리즘은 각기 다른 방법을 이용하여 윈도우 내의 입출력 요구들에게 우선순위 값을 부여하고 우선순위 값이 가장 작은 것부터 서비스 한다. 큐 안에 정렬되어 있는 입출력 요구들 중에 i 번째 요구의 종료시한을 d_i라 하고 현재 헤드의 위치로부터의 탐색거리를 l_i 라 하면, 우선순위 P_i 는 식 1과 2로 정의 할 수 있다.

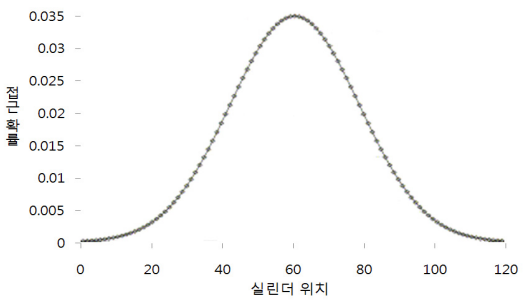
$$P_i = w_i * d_i, i = 1,2,3,.....,m \quad (1)$$
$$where\ w_i = B^{i-1}(B \geq 1), i = 1,2,3...,m$$

$$P_i = a * d_i + (1-a) * l_i (0 \leq a \leq 1), i = 1,2,3,..,m \quad (2)$$

식 1과 2의 방법에 따라 윈도우 내의 입출력 요구들의 우선순위 값을 정한 후 그 값이 가장 작은 것부터 서비스 한다. 즉 탐색거리가 짧고 종료시한이 빠를수록 높은 우선순위를 갖는다.

UG-SSTF(Urgent Group and Shortest Seek Time First)는 디스크 입출력 요구들을 긴급도의 정도에 따라 세 개의 그룹으로 구분하고 같은 그룹에 속한 입출력 요구들을 탐색시간이 작은 것부터 서비스를 해 주는 알고리즘이다. 이 알고리즘은 입출력 요구들을 세 개의 그룹으로 나누어서 서비스함으로써 낮은 종료시한 실기 율을 달성하고 연성 실시간 시스템의 전형적인 성능 요구 사항인 종료시한에 관한 관리를 수행한다.

2.1.6 하드디스크의 파일 재배치 기법



▶▶ 그림 6. Organ pipe arrangement 방법

하드디스크 파일 재배치 방법은 하드디스크의 탐색 지연시간을 줄이기 위한 방법이다. 이는 하드디스크의 참

조 패턴이 지역성을 띄는 것을 이용하는 것으로서 실제로 디스크의 참조 형태를 분석하고, 이를 기반으로 하여 디스크에서 블록의 위치를 동적으로 바꾸어 디스크의 배치형태를 탐색지연 시간이 극소화 될 수 있는 형태로 재배치하는 방법을 말한다. 하드디스크 파일 재배치 방법의 경우 사용자의 디스크 접근에 대한 정보를 계속적으로 기록하고 이를 토대로 하여 최적의 재배치 형태를 계산한 후, 이를 실제로 적용하는 방법을 사용하고 있다. 그래서 디스크의 접근에 대한 정보를 얻기 위해 디바이스 드라이버 계층에서 디스크의 접근을 감시하고 이를 다시 디스크에 저장 하는 방법을 사용 하고 있다. 디스크 재배치 형태는 그림 6과 같이 접근이 빈번한 실린더를 디스크의 중앙 실린더에 배치하고, 그 다음으로 접근이 빈번한 실린더를 그 옆의 실린더에 차례로 배치시켜서, 디스크의 중앙 실린더 부분에서 멀어질수록 접근빈도가 낮게 만드는 오르간 파이프 편곡 Organ pipe arrangement[15] 방법을 사용한다.

2.1.7 하드디스크의 선반입 기법

메모리의 데이터 접근과 하드디스크의 데이터 접근 시간은 최소 배 이상 차이가 날 정도로 하드 디스크의 속도가 느리므로 선반입을 통해 하드 디스크와 메모리간의 병목 현상을 줄여야 한다[16]. 선반입의 기존 연구는 프로그램의 지역성과 순차성을 기준으로 나눌 수 있다. 지역성은 다시 시간적 지역성과 공간적 지역성으로 나눌 수 있다.

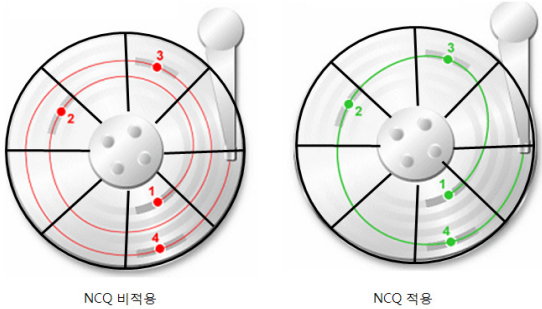
시간적 지역성은 최근에 사용된 정보는 다시 사용될 가능성이 높다라는 것을 의미하며, 공간적 지역성은 하드디스크의 공간상에 현재 사용되는 데이터와 인접한 데이터는 사용될 가능성이 높다라는 것을 의미한다. 또한 응용프로그램을 기준으로 보았을 때 힌트를 이용한 선반입과 힌트를 이용하지 않는 선반입으로 나눌 수 있다. 대부분의 힌트를 이용한 선반입은 특수한 환경에서 사용된다. 응용프로그램이 힌트를 제공하는 연구에는 컴파일러 차원에서 힌트를 삽입하는 연구가 행해졌다[17]. 이 연구는 응용프로그래머에게 부담이 될 뿐만 아니라 잘못된 정보가 유입될 가능성도 있고 호환성의 문제가 발생할 가능성이 크므로 특수한 상황에서만 사용 된다. 힌트가 없는 범용적인 선반입은 일반적인 운영체제에서도 사용할 수 있다. 리눅스에서는 미리 읽기 알고리즘을 사용하여 연속된 위치의 블록을 미리 읽어 들여 선반입을 실시한다. 하드디스크는 1초에 최소 40메가바이트를 읽고 전송 할 수 있으므로 미리 읽기 알고리즘은 부하가 발생 하지 않는다[18].

하드디스크에 플래시 메모리가 탑제되어 있는 하이브리드 하드디스크에서 n-블록을 플래시 메모리에 선반입하는 방법은 하드디스크의 입출력 성능을 향상시키고 전력 소비를 감소 시켰다[19]. 하이브리드 하드디스크를 스핀다운 상태로 두고 플래시 메모리에 선반입된 n-블록을 사용 하여 입출력 요청을 서비스 함으로서 시스템 입출력 성능과 전력 소비를 감소 시킬 수 있었다.

2.1.8 하드디스크의 NCQ 기법

Native Command Queueing(NCQ)는 기존의 기존 데이터 탐색 및 저장 방식의 비효율적인 운영 방식을 개선한 것으로 기존까지는 데이터를 처리하기 위해 HDD에 데이터를 저장하거나 검색할 때는 해당 명령의 순서대로 데이터를 검색하거나 저장하기 때문에 원형으로 회전하면서 데이터를 저장 검색하는 하드디스크의 경우 여러개의 데이터를 처리하기 위해 불필요한 회전을 가지게 되어 지연값이 증가하는 문제가 있다.

하지만 NCQ는 이 문제를 해결 하기 위해 명령을 그림 7과 같이 데이터의 요청을 재배열하여 처리한다. 따라서 불필요한 회전 지연값을 최소화 함으로써 하드디스크의 물리적인 수명의 한계도 증가시키고 데이터 처리를 더욱 빠르게 해준다[20].



▶▶ 그림 7. NCQ 방법

2.1.9 하드디스크의 스핀다운을 통한 저전력 기법

하드디스크는 컴퓨터 시스템에서 전력 소모를 20~30% 차지하며 그 중 약 80~90% 정도의 전력 소모가 스핀들 모터에서 발생한다. 스핀들 모터에서 발생하는 전력을 감소시키기 위해서는 스핀들 모터의 동작을 정지시킬 수 있다. 스핀들 모터의 동작을 효율적으로 관리하기 위한 방법으로는 다음과 같은 알고리즘이 있다. 고정

된 스핀다운(Fixed Spindown) 알고리즘[21]은 정해놓은 시간만큼 하드디스크에 요청이 없으면 하드디스크를 스핀다운 상태로 변경함으로서 하드디스크의 스핀들 모터가 소비하는 전력을 줄이는 가장 일반적인 하드디스크 스핀다운 알고리즘이다.

동적 스핀다운(Dynamic Spindown) 알고리즘[22]은 고정된 스핀다운 알고리즘의 정해진 시간을 동적으로 계산 하여 스핀다운을 시행하는 알고리즘이다. 새로운 고정된 시간은 하드디스크의 유휴주기(Idle period)에 계산 되어 지며 스핀다운 횟수를 줄이면서 전력 소비를 줄였다.

적응적인 스핀다운(Adaptive Spin-down) 알고리즘[23]은 스핀업 대기시간을 줄이기 위하여 스핀다운을 하기위한 정해진 시간을 변화 시키는 알고리즘이다. 사용자의 패턴을 분석하여 스핀업 횟수를 최대한 줄이고 스핀업 대기 시간을 줄여서 전력소비를 최대한 줄이는데 목적이 있다.

2.2 플래시 메모리 기술동향

플래시 메모리는 비휘발성 특징을 가지고 있으며, 하드디스크에 비해 견고하다. 저 전력으로 동작이 가능하며 접근 시간이 하드디스크 보다 빠르다. 또한 크기가 작아 휴대 기기에 적합하다. 그러나 단점으로 하드 디스크에 비해 가격이 5~10배정도 비싸며, 이미 데이터가 있는 공간에 새로운 데이터를 쓰고자 할 때는 지움 과정을 수행한 다음에서야 데이터를 저장 할 수 있다. 또한 읽는 속도는 매우 빠르지만, 쓰기 속도와 지우는 속도가 상대적으로 느리고, 한 번에 지울 수 있는 크기가 일정하며, 상온에서 지울 수 있는 회수가 정해져 있다[24]. 또한 이런 플래시 메모리는 NOR 형태와 NAND 형태의 플래시 메모리 형태로 구분 할 수 있다. 표 1에서는 플래시 메모리의 기본 연산 수행시간을 보여준다[25].

표 1. 플래시 메모리 기본 연산 수행시간

Media	Read	Write	Erase
DRAM	60ns (2B) 2.56us (512B)	60ns (2B) 2.56us (512B)	N/A
NOR Flash	150ns (1B) 14.4us (512B)	211ns (2B) 3.53us (512B)	1.2s (128KB)
NAND Flash	10.2us (1B) 35.9us (512B)	201us (2B) 226us (512B)	2ms(16KB)
Disk	12.4ms (512B) (average)	12.4ms (512B) (average)	N/A

2.2.1 NOR 플래시 메모리

초기에 개발된 NOR 플래시 메모리는 전원공급이 끊겨도 데이터를 잃지 않는 비휘발성 메모리로서, 바이트 단위로 읽기 가능한 RAM 형태의 인터페이스를 가진다. 하지만 쓰기의 경우에는 해당 바이트의 1인 비트를 0으로 바꾸는 동작만 가능하다. 0인 비트를 쓰기가 가능한 1인 상태로 바꾸는 동작을 삭제(erase)라고하며 이는 읽기, 쓰기작업 보다 훨씬 큰 영역인 소거 블록(erase block)단위로만 가능하다[26]. NOR 플래시 메모리는 읽기 성능은 비교적 좋지만 쓰기와 소거 성능은 매우 낮다. 상기한 동작 특성 외에 중요한 특성으로는 소거 횟수에 제한이 있다는 것이다. 일반적으로 100,000번 정도의 삭제 사이클을 지원하며 이 후에는 정확한 동작을 보장할 수 없게 된다. 이런 현상을 마모(wear)된다고 표현한다.

2.2.2 NAND 플래시 메모리

NOR 플래시 메모리보다 나중에 개발된 NAND 플래시 메모리는 집적도를 높일 수 있는 구조적인 특성으로 인해 보다 적은 비용으로 대용량의 저장장치를 구성할 수 있게 되었다. 하지만 NOR 플래시 메모리와 같은 바이트 단위의 접근은 가능하지 않으며 일정 영역(페이지) 단위로만 읽기 및 쓰기가 가능하다. 이런 특성으로 인해 일반적인 RAM 인터페이스를 이용할 수 없고 별도의 인터페이스 장치가 필요하며 읽기 성능 또한 NOR 플래시 메모리보다 낮다[26]. 하지만 쓰기 및 소거 성능은 NOR 플래시 메모리보다 현저하게 높다. 일반적인 페이지의 크기는 512Byte, 2Kbyte 등이다.

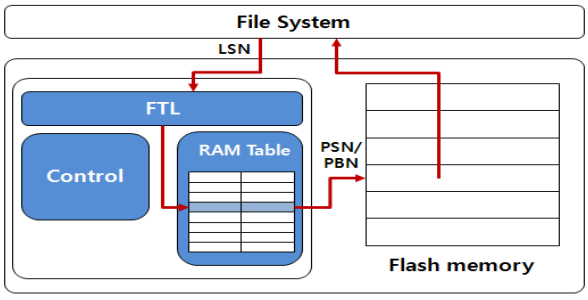
2.2.3 플래시 메모리의 시스템 소프트웨어 기술

플래시 메모리의 특성으로 인하여 기존 파일시스템을 플래시 메모리에 바로 적용할 수 없으므로 플래시메모리 전용 파일 시스템에 대한 연구가 활발히 진행되어 왔다. 대표적인 예로 FTL(Flash Trans lation Layer), NFTL(NAND Flash Translation Layer), CFTL(Clustered Flash Translation Layer), JFFS2(Journaling Flash File System), YAFFS(Yet Another Flash File System) 등이 있다[27]. FTL은 순차적인 플래시 공간이 디스크의 섹터처럼 보이도록 하기 위해 매핑(mapping) 관리를 수행하는 드라이버 형식으로 구현되어 있다.

2.2.3.1 플래시 메모리의 FTL, NFTL, CFTL 기술

플래시 메모리는 데이터를 기록하기 전에 데이터 영역이 지워져 있어야 하는(erase-before-write) 제약이 있으며, 비대칭적인 읽기, 쓰기, 삭제 연산의 처리속도, 블록 당 소거 횟수의 제한과 같은 특징을 지닌다. 위와 같은 특징으로 인하여 플래시 메모리는 데이터 겹쳐 쓰기(overwrite)가 발생할 경우 물리적인 고정된 위치에 데이터를 기록하지 않고 미사용중인 빈 영역을 찾아 쓰기 연산을 수행한다. 그러므로 파일 시스템에서 요청한 섹터 번호와 실제 데이터가 기록된 물리적인 위치는 서로 다르게 된다. 이와 같은 문제점을 극복하고 플래시 메모리를 효율적으로 사용하기 위해 플래시 변환 계층(Flash translation layer)이라는 미들웨어를 사용한다 [28].

그림 8과 같이 플래시 변환 계층은 주소 변환 테이블을 통한 논리적인 주소를 물리적인 주소로 사상하는 방법과 무휴 데이터(invalid data)들을 처리하기 위한 가비지 컬렉션(garbage collection) 기능 그리고 균등한 쓰기를 보장하기 위한 마모도 평준화(wear-leveling) 기능을 제공한다. 이러한 플래시 전환 계층을 통하여 FAT와 같은 전통적인 파일 시스템을 사용이 가능하게 된다. 플래시 전환 계층은 주소사상 방법에 따라 섹터 사상(sector mapping), 블록 사상(block mapping), 혼합 사상(hybrid mapping)으로 나뉘며 대표적으로 FTL, NFTL, CFTL 등이 있다.



LSN: Logical Sector Number
PSN: Physical Sector Number
PBN: Physical Block Number

▶▶ 그림 8. 플래시 변환 계층

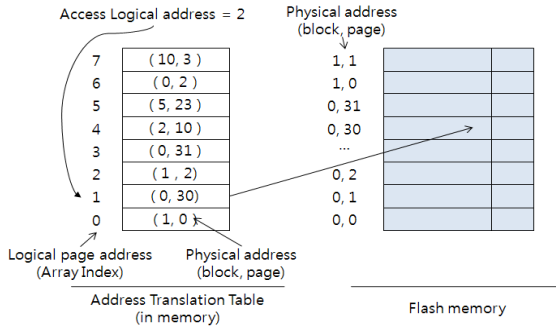
가. FTL(Flash Tranlation Layer)

FTL은 초기에 논리 주소와 물리 주소를 섹터 단위로 1:1 주소 변환을 하는 섹터 사상을 사용하였다 [29]. 예를 들어 그림 9와 같이 파일 시스템으로부터 논리 주소 2에 대한 물리 주소의 값을 찾는 요청이 들어오면 주소 변환 테이블의 2번째 엔트리의 (0, 30)을 찾게 된다. (0, 30)은 NAND 플래시 메모리의 물리주소로 0번 블록의 30번째 페이지 주소를 일컫는다. 이와 같이 섹터 사상을 하면 한 번의 주소 변환 테이블의 검색을 통해 실제 플래시 메모리의 물리적인 섹터 번호를 알 수 있으므로 매우 빠르게 동작할 수 있는 장점을 갖는다.

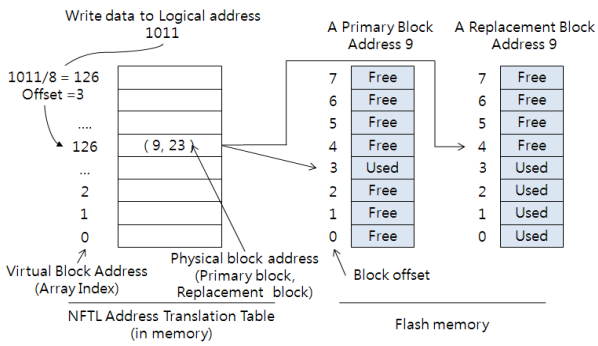
하지만 FTL은 주소 변환 테이블의 크기가 용량에 따라 선형적으로 커지므로 NAND 플래시 메모리와 같이 대용량 플래시 메모리에 사용할 경우 주소 변환 테이블을 위한 메모리 비용이 매우 커지게 되는 문제가 발생한다.

나. NFTL(NAND Flash Translation Layer)

NFTL은 M-System사에서 설계한 NAND 플래시 메모리 전용 FTL로 위에서 설명한 FTL의 단점인 주소 변환 테이블의 크기를 줄이기 위해 그림 10과 같은 블록 단위의 주소 변환 방식을 사용한다 [30,31,32]. 블록 사상 방법은 논리 주소 요청에 대하여 논리 주소를 블록 당 섹터 개수로 나누어 몫을 논리 블록 번호(Logical block number)로 사용하고 나머지 값은 섹터 단위의 오프셋으로 사용한다. 그리고 주소 변환 테이블로 접근 시 논리 블록 번호에 해당하는 엔트리에 접근하여 해당하는 물리 블록 주소를 찾는다. 이와 같이 블록 단위로 주소 변환 테이블을 구성하므로 메모리 비용을 절감할 수 있다.



▶▶ 그림 9. 섹터 사상

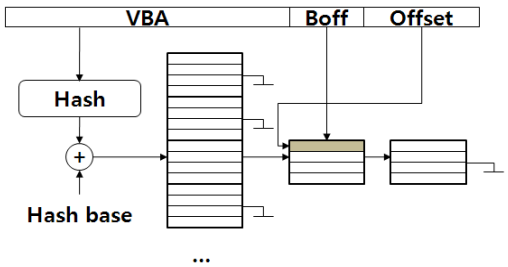


▶▶ 그림 10. NFTL의 주소 변환 방법

하지만 블록 사상을 사용할 경우 겹쳐 쓰기가 발생할 때마다 주소 변환 테이블의 물리 블록 주소가 바뀌므로 이와 관련된 블록 내의 모든 데이터를 바뀐 물리 블록으로 복사해야하는 오버헤드가 발생한다. 이러한 블록 사상의 오버헤드를 줄이기 위해 NFTL에서는 Primary 블록과 Replacement 블록을 사용하여 해결한다. Primary 블록은 기존의 블록 사상에서 사용하는 겹쳐 쓰기 전의 데이터가 저장되는 블록이다. Replacement 블록은 겹쳐 쓰기에 대한 효율성을 높이기 위해 사용되는 블록으로 모든 블록이 Replacement 블록을 갖는다. 만약 겹쳐 쓰기가 발생할 경우 그림 10과 같이 Replacement 블록의 섹터에 오프셋에 상관없이 순차적으로 저장하여 블록 사상으로 인해 발생하는 불필요한 복사를 감소시킨다. 하지만 Replacement 블록에 존재하는 데이터를 찾기 위해서는 Replacement를 순차적으로 검색해야 하므로 섹터 사상보다 주소 변환 성능이 느리다.

다. CFTL(Clustered Flash Translation Layer)

CFTL은 주소 변환 테이블의 크기를 줄이면서 주소 변환 성능을 향상시키기 위해 혼합 사상을 사용한다. 혼합 사상 기법은 메모리 요구량이 적은 블록 사상을 기본을 사용하고 빠른 주소 변환을 위해 소수의 블록들에 대해 섹터 사상을 위한 주소 변환 테이블을 사용하는 기법이다. CFTL은 블록 단위의 주소 변환을 위한 Coarse-grained 군집형 해시 테이블과 섹터 단위의 주소 변환을 위한 Fine-grained 군집형 해시 테이블을 사용하여 주소 변환 성능을 향상시키고 메모리 소모를 감소시켰다 [33, 34]. CFTL은 그림 11의 군집형 해시 테이블을 사용하여 유사한 주소들을 동일한 버켓(Bucket)에 그룹화 하므로 지역성 확보가 가능하다. 또한 Fine-grained 해시 테이블을 2단계로 설계하여 미스 페널티를 최소화 시키고 상대적으로 주소변환 오버헤드가 큰 블록 단위 군집형 해시 테이블의 접근 빈도를 감소시킨다.

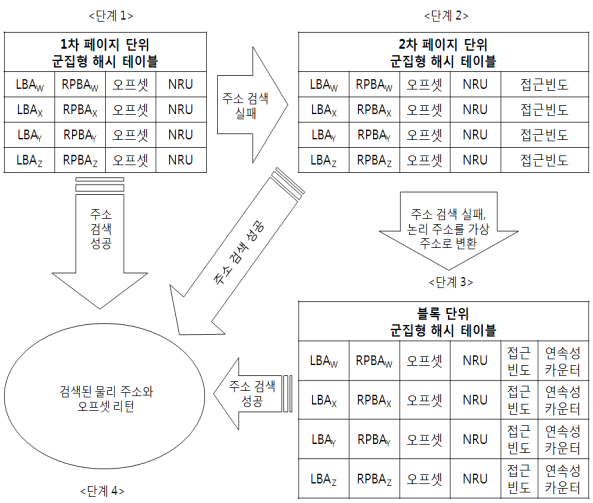


▶▶ 그림 11. 군집형 해시 테이블

CFTL은 블록 단위 군집형 해시 테이블의 주소 변환 성능을 향상시키기 위해 연속성 카운터를 사용하여 연속적인 논리 주소 변환 요청시 주소 변환 부담을 줄인다. 주소 변환 요청 시 섹터 단위 군집형 해시 테이블에 요청된 논리 주소가 존재할 경우 그 논리 주소가 소속된 버켓의 전체 페이지를 미리 선반입하여 주소 변환 성능을 향상시켰다.

그림 12는 CFTL의 주소 변환 방법의 전체 알고리즘을 자료구조와 흐름도로 나타낸 것이다. 우선 특정 논리 주소에 대한 주소 변환이 호스트로부터 들어오면 가장 먼저 1차 섹터 단위 군집형 해시 테이블을 검색하고 매칭되는 물리 주소를 찾으면 주소 변환은 종료된다.

만약 1차 섹터 단위 군집형 해시 테이블에서 검색에 실패 했을 경우 2차 섹터 단위 군집형 해시 테이블을 검색하여 매칭되는 물리 주소 변환은 종료 된다. 하지만 2차 섹터 단위 군집형 해시 테이블에서도 검색에 실패한 경우 최종적으로 블록 단위 군집형 해시 테이블에서 매칭되는 주소 변환 정보를 찾는다. 블록 단위 군집형 해시 테이블은 NFTL과 동일하게 우선 논리 주소를 물리 블록 주소와 오프셋으로 변환한다. 이렇게 구해진 물리 블록 주소를 해싱하여 Primary 물리 블록 주소와 Replacement 물리 블록 주소가 저장되어 있는 버켓을 찾는다. 그리고 매칭된 Primary 물리 블록 주소와 오프셋에 있는 페이지 데이터가 유효하지 않을 경우 Replacement 물리 블록 주소를 순차적으로 검색하여 해당 물리 주소를 찾는다.



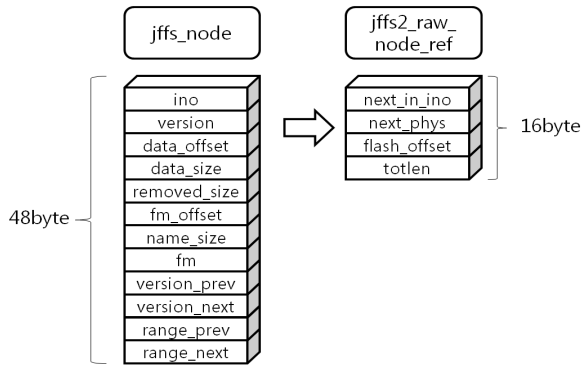
▶▶ 그림 12. CFTL의 주소 변환 방법

CFTL과 같이 섹터 사상과 블록 사상을 혼합한 혼합 사상을 사용한 플래시 변환 계층에 대한 연구가 활발하며 주소 변환 성능뿐만 아니라 효율적인 가비지 컬렉션 기법과 마모도 평준화 기법에 대한 연구가 진행되고 있다. 또한 FTL의 한계를 극복하기 위해 플래시 메모리 전용 파일 시스템에 관한 연구도 활발하다.

2.2.3.2 플래시 메모리 파일시스템 JFFS2, YAFFS 기술

가. JFFS2(Journaling Flash File System2)

JFFS2는 LFS(Log-structured File System) 방식의 플래시 메모리 파일 시스템으로서 우수한 안정성으로 인해 널리 사용되고 있다. NOR 플래시 메모리에 적합한 파일 시스템으로 개발되었다. JFFS는 데이터를 로그 형태로 플래시 메모리에 순차적으로 쓰고, 읽기 연산은 로그를 역순으로 검색하여 가장 최신의 데이터를 읽어 들인다.



▶▶ 그림 13. JFFS 플래시 메모리 구조

그림 13은 JFFS 플래시 메모리 구조를 나타낸다. JFFS에서는 저널링 노드로서 jffs_node라는 구조체를 사용하는데, 이것의 크기는 48Bytes로 상당히 크다. 오버헤드를 줄이기 위해 JFFS2는 next_in_ino, next_phys, flash_offset, totlen 값만으로 구성된 jffs2_raw_node_ref라는 구조체 (16Bytes)로 jffs_node를 대체하여 메모리에 저장한다. 또한, 플래시 메모리의 공간을 효율적으로 활용하기 위해 데이터 압축 기능을 사용한다. 메모리 사용량이 48Bytes에서 16Bytes로 줄었다고는 하지만, 플래시 메모리가 128MBytes일 때, 218페이지를 필요로 할 수 있기 때문에 jffs2_raw_node_ref를 위한 공간만 222Bytes, 즉 4MBytes를 할당해야 하는 문제점이 있고, 노드를 찾고 파일의 구조를 결정하기 위한 스캔 시간이 길다. JFFS2는 NOR 플래시 메모리를 기반으로 설계되었기 때문에 NAND 플래시 메모리용 파일 시스템으로 사용되기에는 메모리 사용량, 마운트 및 플래시 메모리 스캔 시간 그리고 가비지 컬렉션 시간 등에서 여러 가지 문제점이 있다 [35].

나. YAFFS(Yet Another Flash File System)

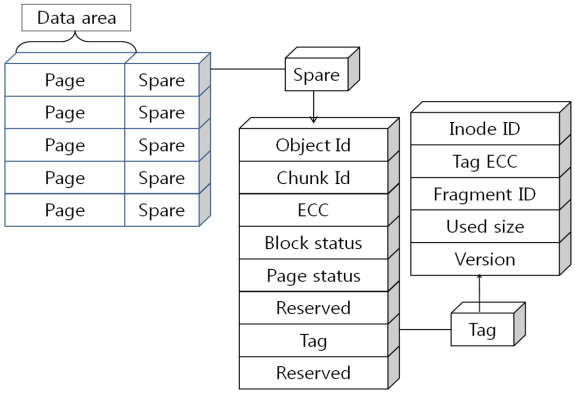
YAFFS는 JFFS2가 저널링에 사용되는 메모리 소모량이 큰 것과, 느린 마운팅 속도를 해결하기 위해 개발된 NAND 플래시 전용 파일 시스템이다. YAFFS는 페이지의 크기가 512Byte인 플래시 메모리만이 사용가능하며 최대 파일 크기 512MB, 최대 파일 수 260,000, 최대 파일 시스템 크기 1GB인 제약을 가진다. YAFFS2의 경우 2Kbyte 페이지 플래시 메모리가 사용가능하며 최대 8GB 파일 시스템 크기를 가질 수 있다. 플래시 메모리에서 읽기와 쓰기는 페이지 단위로 수행되며, 삭제는 블록 단위로 수행된다[36]. YAFFS는 그림 14와 같이 파일 데이터를 하나의 페이지와 동일한 크기인 Chunk로 나누어 플래시 메모리에 저장한다. Chunk는 파일의 데이터와 스페어 영역으로구분되며, 데이터 영역에 파일의 정보를 관리 하는 헤더 (Object Header) 가 저장된다 [37].

헤더가 저장되는 경우는 스페어 영역의 ChunkID가 0으로 되어 있다. 이 헤더는 파일의 이름과 파일의 크기, 수정 시간, 상위디렉터리에 포인터 등으로 구성된다. ChukID가 0이 아니면 파일의 데이터라고 간주한다. 또한 각 Chunk마다 스페어 영역이 존재한다. 스페어 영역에는 블록상태, 페이지상태, ECC영역 및 Tag가 존재한다. 페이지 상태 영역에는 유효(valid)/무효(invalid)페이지를 나타내는 정보가 있다. Tag영역은 페이지에 들어있는 데이터의 크기, 파일 업데이트 횟수 및 아이노드 식별자 등의 자료가 들어 있다.

파일 갱신과 파일 삭제 하는 경우에 그 파일이 들어있는 페이지를 무효화하게 된다. 추후 가비지 컬렉션을 할 때 무효화 된 페이지들을 포함한 블록을 삭제하게 된다.

YAFFS는 마운트 시에 플래시 메모리의 스페어 영역만을 스캔하여 헤더가 저장되어 있는 페이지가 발견되면 그 페이지 데이터의 내용과 스페어영역에 저장된 파일상태를 읽어서 해당 파일의 오픈 파일 테이블을 주 메모리에 동적으로 생성한다. 따라서 플래시 메모리의 전체 영

역을 읽어야 하는 JFFS2에 비해 마운트시간이 훨씬 짧다. 그러나 마운트 과정에서 최소한 모든 페이지의 스페어 영역을 읽어야 하기 때문에 플래시 메모리의 크기가 커질수록 마운트 시간이 증가하게 된다.



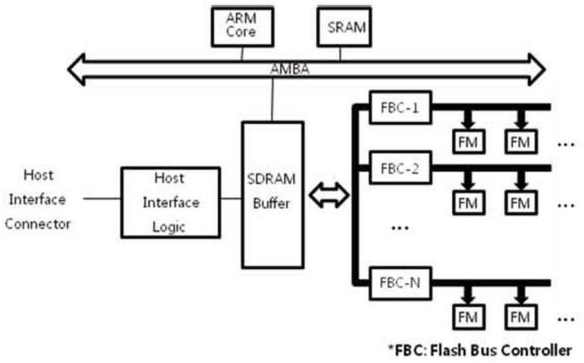
▶▶ 그림 14. YAFFS 플래시 메모리 구조

2.3 SSD의 기술동향

SSD는 다수의 NAND 플래시메모리를 부착하고 호스트에서 요구하는 명령을 받아 수행하는 대용량 저장장치이다. 플래시 기반 SSD는 NAND 플래시 기반으로 전기적인 방법으로 데이터를 일련의 명령순서에 의해 입출력할 수 있으며, 전력소모가 적고 고속프로그래밍이 가능하기 때문에 노트북 등에 시장을 형성하고 있는 기술이다. 기존의 하드디스크와는 달리 플래시메모리에서 사용하는 블록기반 구조에 마모도에 따른 수명제한이라는 특성을 가진다[38].

그림 15는 MTRON사에서 개발한 SSD의 내부구조이다. SSD의 기본 구성은 ARM 프로세서, SRAM 캐시, SDRAM 버퍼, 그리고 다수의 플래시 메모리 칩과버스로 이루어진다. ARM프로세서는 SSD를 제어하는 FTL 코드를 수행하고, SDRAM버퍼는 읽기/쓰기 요청된 데이터를 보관한다. 또한 요청들을 SDRAM에 보관하고 이들을 다수의 버스를 통해 병렬적으로 처리하여 전체적인 성능을 높인다. 그림 15에서 보듯이 하나의 버스에는 다수의 칩들이 연결된다. 하나의 버스에 연결되는 칩들의 수가 증가하면 SSD의 전체 용량은 증가하고 또한 병렬로 동작할 수 있는 칩이 증가하여 성능도 증가할 수 있다. 그러나 버스 하나의 데이터 전송양은 한정되어 있으므로 버스가 병목이 될 수 있다. 이를 해결하는 한 가지 방법으로 버스 수를 증가시키면 병렬로 전송하는 데이터가 증가하여 SSD의 처리속도가 증가하게 된다.[39].

성능 측면에서 SSD 저장장치는 기존의 디스크에 비해 탐색시간 및 회전 지연이 없는 매체의 특성에 의해 I/O 성능향상을 보장받을 수 있다. 하지만 기존의 연구는 SSD를 위한 마운트 속도나 마모도 평준화를 고려한 수명 향상의 연구는 미비한 상태이다. 따라서 앞서 기술한 플래시 메모리를 위한 파일시스템기술을 SSD에 적용하여 빠른 마운트 속도와 수명향상 기술 연구가 필요하다.



▶▶ 그림 15. SSD 구조 블록 다이어그램

3. 결론

본 논문에서는 언제 어디서나 콘텐츠를 접속 할 수 있는 모바일 기기의 저장장치인 하드디스크, 플래시메모리, SSD의 기술 동향을 살펴보았다. 인터넷과 IT기술 발전, 개인 문화의 성향변화 등으로 콘텐츠의 유형이나 취급용량이 빠른 속도로 증가되며 품질도 다양화되고 있다. 때문에 모바일 기기에서 저장해야 할 콘텐츠의 양은 기하급수적으로 늘어나게 되므로, 이를 대비하기 위한 저장장치의 특성과 기술적인 문제점 그리고 이를 극복하기 위한 기술 등을 분석하였다.

IT시장에서 모바일 기기 저장장치의 시장규모는 5년 이내에 250~300억 달러가 될 것으로 예상한다[40]. 현재 모바일 기기에서 사용하는 저장장치는 하드디스크, 플래시메모리가 대부분 이다. 하지만 대용량 콘텐츠의 수요증대 및 새로운 킬러 어플리케이션의 등장 등으로 앞으로는 기존의 저장장치의 단점인 처리속도, 신뢰성, 전력소모 등의 개선을 위해 낸드플래시메모리가 결합된 하이브리드 하드디스크, 플래시메모리가 고집적된 형태인 SSD가 많이 사용될 것 이며, 이에 따른 저장장치 기술개발이 이루어질 것으로 전망된다.

참고문헌

[1] 백문철, 강광용, “차세대 PC 모바일 광저장장치,” 한국광학회 광 정보처리기술 워크샵, 2004.

[2] 과학기술부, “고성능정보처리 및 저장장치 국가기술지도,” 2002.

[3] 이홍재, “하드디스크의 이해,” 전자신문사 2004.

[4] 조경선, 원유집, 신일훈, 고건, “구역분할 디스크를 사용하는 멀티미디어 서버에서 새로운 세션 시작에 따른 스케줄링 지연 현상의 최소화,” 정보과학회, 시스템 및 이론 제 31 권 제 8 호, 2004. 8.

[5] P. J. Denning, “Effect of Scheduling on File Memory Operations,” In Proceedings of AFIPS SJCC, volume 30, pages 9-21, 1967.

[6] N. C. Wilhelm, “An Anomaly in Disk Scheduling: a Comparison of FCFS and SSTF Seek Scheduling Using an Empirical Model for Disk Accesses,” CACM, 19(1),pages 13-17, 1972.

[7] M. Hofri, “Disk Scheduling: FCFS vs. SSTF Revisited,” CACM, 23(11) pages 9-21, 1967.

[8] T. J. Teorey, “Properties of Disk Scheduling Policies in Multi programmed Computer Systems,” In Proceedings of AFIPS SJCC, volume 41, pages 1-11, 1972.

[9] B. Kao and R. Cheng, “Disk Scheduling,” In REAL-TIME DATABASE SYSTEMS, pages 97-107, Kluwer, 2002.

[10] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” Journal of ACM, pages 46-61, 1973.

[11] M. J. Carey, R. Jauhari, and M. Livny, “Priority in DBMS Resource Scheduling,” In Proceedings of the 15th VLDB Conf, pages 397-410, 1989.

[12] A. L. N. Reddy and J. Wyllie, “Disk Scheduling in a Multimedia I/O System,” In Proc. of the first ACM International Conference on Multimedia, pages 225-233, Anheim, CA, 1993.

[13] S. Chen, J. A. Stankovic, J. F. Kurose and D. Towsley, “Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems,” The Journal of Real-Time Systems, 3(3), 1991.

[14] K. Hwang and H. Shin, “Real-time disk scheduling based on urgent group and shortest seek time first,” in Proceedings of 5th EUROMICRO Workshop on Real-Time Systems, page 124-130, 1993.

[15] Yannis Manolopoulos J. (Yannis) G. Kollias,“Optimal data placement in two-headed disk systems.

[16] 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.

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

[18] Daniel Plerre Bovet , Marco Cesati, "Under- standing the Linux Kernel (3/E)," O'REILLY.

[19] 양준식,고영욱,김덕환,“저전력과 입출력 성능이향상된 n-블록 선반입 기반의 하이브리드 하드디스크 입출력 시스템 설계 및 구현",정보과학회, 시스템및 이론(개제예정).

[20] http://commons.wikimedia.org/wiki/File:NCQ.svg.

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

[22] D. Helmbold, D. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In In Proceedings of the 2nd ACM International Conference on Mobile Computing, November 1996.

[23] Timothy Bisson, Scott Brandt, and Darrell D.E. Long. A Hybrid Disk-Aware Spin-Down Algorithm with I/O Subsystem Support. Proceedings of the International Performance Conference on Computers and Communication (IPCCC), IEEE, 2007.

[24] 이동환, 조원희, 김덕환 "다중 블록 지우기 기능을 적용한 퓨전 플래시 메모리의 FTL 성능 측정도구 설계 및 구현, "대한전자공학회, 제31권, 제1호, pp.647-648, 2008년 6월.

[25] 변시우 “F-Tree : 휴대용 정보기기를 위한 플래시 메모리 기반 색인 기법”, 한국데이타베이스학회, 학술저널, 제13권, 제4호, pp.257-271, 2006년 12월.

[26] 이수관, 민상렬, 조유근, “플래시 메모리 관련 최근 기술 동향, ”정보과학회지, 제24권, 제 12호, pp.99~106, 2006년 12월.

[27] 조원희, 이동환, 김덕환 “NAND플래시 파일 시스템을 위한 내용기반 블록관리기법을 이용한 마운트 시간감소와 지움 정책, ”전자공학회논문지, SD편, 제3호, pp.41~50, 2009년 3월.

[28] Eran Gal and Sivan Toledo, “Algorithms and Data Structures for Flash Memories,” ACM Computing Surveys, 37(2), Jun. 2005.

[29] Intel Coporation, "Understanding the Flash-Tra-nslation Layer (FTL) Sepcification," 1998.

[30] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, and Yookun Cho, "A space-efficient flash translation layer for CompacFlash systems," IEEE

Transaction on Consumer Electronics, Vol. 48, No. 2, pp.366-375.
[31] Ban, A. 1995. Flash file system. US patent 5,404,485. Filed March 8 1993; Issued April 4, 1995; Assigned to M-Systems.
[32] Ban, A. 1999. Flash file system optimized for page-mode flash technologies. US patent 5,937,425. Filed October 16, 1997; Issued August 10, 1999; Assigned to M-Systems.
[33] Kwanghee Park, Junsik Yang, Joon-Hyuk Chang, and Deok-Hwan Kim, "A Clustered Flash Translation Layer for CompactFlash Systems," IEEE International Conference on Consumer Electronics, Las Vegas, NV, U. S. A., Jan. 2008.
[34] 박광희, 김덕환, "플래시 메모리를 위한 효율적인 선반입과 비동기 쓰기 기법," 한국정보과학회,정보과학회논문지:컴퓨팅의 실제 및 레터 제 15권 제 2호, pp.77-88 2009. 2.
[35] 박상오, 김성조 "NAND플래시 메모리 기반 파일 시스템을 위한 빠른 마운트 및 안정성 기법, "정보과학회논문지, 제34권, 제11-12호, pp.683-695, 2007년 12월.
[36] 한대만, 김성범, 구용완 "플래시 메모리를 위한 파일 시스템의 개선 방안," 한국인터넷정보학회,제6권, 제2호, pp.733-736, 2005년.
[37] Samsung Electronics Co., .NAND Flash Memory & SmartMedia Data Book., 2002.
[38] 배영현 "고성능 플래시 메모리 SSD(Solid State Disk)설계 기술," 정보과학회지, 제25권, 제6호, pp.18-28, 2007년 6월.
[39] 윤진혁, 남이현, 성윤제, 김홍석, 민상렬, 조유근 "고성능 플래시 메모리 솔리드 스테이트 디스크,"정보과학회논문지, 제14권, 제4호, pp.378-388, 2008년 6월.
[40] 전자신문, "저장장치, 총성없는 전쟁," 기획자료 2005.10.

저자소개

● 고 영 욱 (YoungWook Go)

▪2008년 2월 : 강원대학교 정보통신 전자공학부 전자공학과 (공학사)
▪2008년 2월 ~ 현재: 인하대학교 전자공학과 (석사과정)
<관심분야> : 임베디드 시스템, 스토리지 시스템

● 양 준 식 (Junsik Yang)

▪2008년 2월 : 인하대학교 컴퓨터정보공학부 (공학사)
▪2008년 2월 ~ 현재: 인하대학교 전자공학과 (석사과정)
<관심분야> : 임베디드 시스템, 스토리지 시스템

● 조 원 희 (Won-Hee Cho)

▪2007년 2월 : 명지대학교 정보통신공학부 통신공학과 (공학사)
▪2008년 2월 ~ 현재 : 인하대학교 전자공학과 (석사과정)
<관심분야> : 임베디드 시스템, 스토리지 시스템

● 이 동 환 (Dong-Hwan Lee)

▪2008년 2월 : 인하대학교 전자공학과 (공학사)
▪2008년 2월 ~ 현재: 인하대학교 전자공학과 (석사과정)
<관심분야> : 임베디드 시스템, 스토리지 시스템

● 김 덕 환 (Deok-Hwan Kim) 정회원

▪2003년 2월 : 한국과학기술원 정보 및 통신공학과 컴퓨터전공 (공학박사)
▪2006년 2월 : ~ 현재: 인하대학교 전자공학과 (부교수)
<관심분야> : 임베디드 시스템, 스토리지 시스템, 시각정보처리, 추천시스템