MEMS 기반 저장장치의 기술 동향 및 소프트웨어적 관리 기법

이화여자대학교 ▌이소윤* · 반효경**

1. 서 론

최근 동영상, 음악 파일 등 대용량 멀티미디어 데이터를 모바일 기기에서 사용하는 수요가 급증함에 따라 많은 양의 정보를 담을 수 있는 저가의 모바일 저장 시스템에 대한 필요성이 증가하고 있다. MEMS 기반 저장장치는 이러한 수요에 부응할 수 있는 새로운 형태의 차세대 저장 매체로 부피가 작고 집적도가 높으며 낮은 전력 소모의 특성을 가지고 있어 플래시메모리와 함께 모바일 시스템의 저장장치로 사용될 수 있는 좋은 조건을 두루 갖추고 있다. 이와 함께 MEMS 기반 저장장치는 단위 공간당 가격이 플래시메모리에 비해 매우 저렴하면서 하드 디스크에 비해 접근 속도가 10배 이상 빨라 서버 및 데스크탑 환경에서의 저장 시스템으로도 사용될 가능성이 높은 저장 매체이다. 30년 이상 대용량 저장장치로 사용되었던 하드 디스크

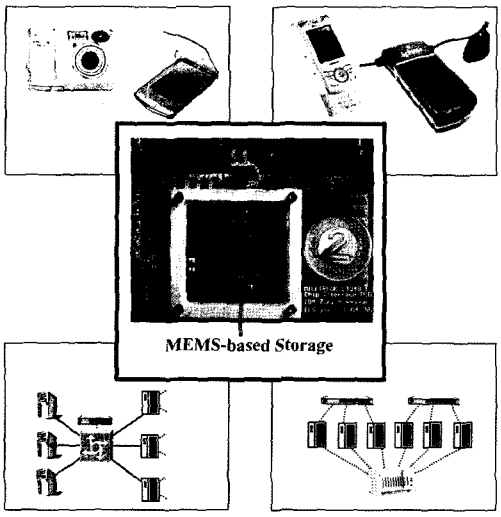


그림 1 모바일 스토리지에서 대용량 서버용 스토리지까지 다양한 MEMS 저장장치의 활용 분야

* 학생회원
** 종신회원

가 RAM과의 성능 격차가 매년 50% 이상씩 벌어짐에 따라 MEMS 기반 저장장치가 이러한 성능 차이를 완충시키는 역할을 수행할 수 있을 것으로 기대되고 있다[1].

본 고에서는 MEMS 기반 저장장치에 대한 기술 동향 및 소프트웨어적 관리 기법들에 대해 살펴본다. 2장에서 MEMS 기반 저장장치의 기본 구조와 장점, 하드디스크와 상이한 물리적 특성에 대해 언급하며 프로토타입으로 제시되고 있는 MEMS 기반 저장장치에 대해 소개한다. 3장에서는 MEMS 기반 저장장치의 사용에 필요한 I/O 스케줄링, 데이터 배치, 결함 허용성 관리, 에너지 관리 등 소프트웨어적 관리 기법에 대해 소개하고, 4장에서 결론을 맺는다.

2. MEMS 기반 저장 장치의 구조

2.1 기본 구조

MEMS 기반 저장장치는 그림 2와 같이 크게 데이터를 저장하는 매체 부분(media sled)과 데이터를 읽고 쓰는 헤드가 나열되어 있는 배열 부분으로 구성된다. 매체가 회전하고 하나의 헤드가 데이터를 읽고 쓰는 하드 디스크와 달리 MEMS 기반 저장장치는 수천 개의 헤드가 고정되어 있고 헤드 아래에 있는 매체가 직접 움직이면서 읽고 쓰려는 데이터를 헤드 위치로 이동시킨다. 사각형의 매체는 각 모서리에 스프링이 연결되어 있고 매체의 각 면에 달린 액츄에이터에 의해 x축 및 y축 방향으로 동시에 움직여 요청이 들어온 위치로 이동한다. 이동 후 데이터를 읽고 쓰는 작업은 매체가 y축 방향으로 움직이면서 이루어지게 된다. 저장 매체는 하드 디스크와 유사한 마그네틱 매체이고 수천 개의 영역(region)으로 나누어져 있다. 각 영역은 이를 전담하는 헤드가 하나씩 존재하여 해당 영역의 요청들을 처리한다. 각각의 헤드는 서로 다른 영역을 처리하지만 고정된 배열 형태로 존재하기 때문에 모든 헤드는 영역 내에서 상대적으로 동일한 위치를 접근해야 하며, 동시에 데이터를 읽고 쓸 수 있다. 그러

나 헤드가 동시에 활성화됨에 따라 발열 및 전력 소모 증가 등의 문제가 발생하여 동시에 활성화되는 헤드의 수를 제한하는 경우도 있다. 예를 들어, 카네기 멜론 대학의 MEMS 프로토타입 중 G2 모델은 6400개의 헤드 중 1280개의 헤드만이 동시에 데이터를 읽고 쓸 수 있다.

2.2 MEMS 기반 저장장치의 장점

MEMS 기반 저장장치는 모바일 스토리지에서 대용량 서버 스토리지에 이르기까지 다양한 환경에서 사용 가능한 차세대 저장장치이다. 본 절에서는 MEMS 기반 저장장치의 장점을 하드디스크 및 플래시메모리와 비교하여 설명하겠다.

· 작은 크기와 높은 집적도

그림 2에서 보는 바와 같이 MEMS 기반 저장장치는 가로와 세로가 각각 1cm, 두께가 2mm로 크기가 매우 작은 동시에 단위 공간당 저장할 수 있는 저장 용량이 매우 높다. 표 1에서 보는 바와 같이 $1cm^2$에 1-10GB의 정보를 저장할 수 있는 높은 집적도를 가지고 있다.

· 낮은 단위 공간당 비용

MEMS(MicroElctroMechanical System) 기반 저장장치는 단위 공간당 가격이 Gbyte당 $1-$3 정도로 RAM의 1/10, 플래시메모리의 1/100에 불과하다. 또한, byte당 가격을 고려하면 MEMS 기반 저장장치는 1-10GB의 용량을 가진 저장장치 중에서 가장 저렴한 저장장치이다. 하드디스크는 이보다 더 큰 용량인 100GB이상에서 가장 저렴하며 100MB 미만 용량에서는 RAM이 가장 저렴하다[2].

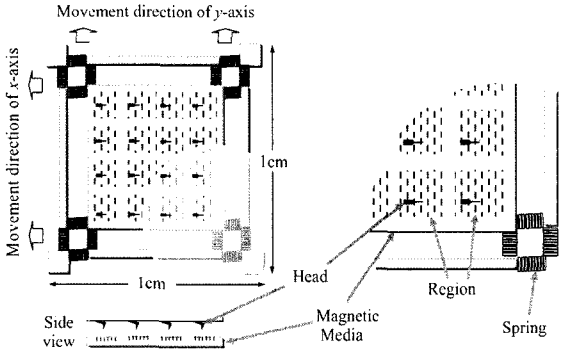


그림 2 MEMS 기반 저장장치의 물리적 구조

· 저전력성

원판이 회전하면서 데이터를 읽고 쓰는 하드디스크와 달리 매체의 회전이 없고 매체의 크기가 작아 발열과 소음이 적으며 전력소모가 하드 디스크에 1/100 이하로 매우 효율적이다.

· 높은 처리율

MEMS 기반 저장장치에서 하나의 헤드가 처리할 수 있는 처리량은 100Kbits/s에서 1Mbits/s이지만 수천 개의 헤드가 동시에 데이터를 읽고 쓸 수 있기 때문에 동시 활성화된 헤드들의 전체 처리량은 최대 1GB/s에 이른다. 이것은 하드디스크의 100배를 넘는 처리율이다.

· 빠른 데이터 접근시간

MEMS 기반 저장장치는 하드 디스크와 같이 헤드가 이동하면서 데이터를 읽고 쓰는 기계식 저장장치이지만 하드 디스크에 비해 10배 이상 빠른 데이터 접근 속도를 나타낸다. 그러나, 기계장치의 한계에 의해 반도체 장치인 플래시메모리에 비해서는 데이터 접근 속도가 상대적으로 느리다.

2.3 MEMS 기반 저장장치의 연구

MEMS 기반 저장장치에 대한 대표적인 연구 기관으로는 미국의 카네기 멜론 대학(Carnegie Mellon University, CMU)과 IBM Zurich 연구소가 있다.

카네기 멜론 대학의 MEMS 저장장치 연구팀은 G1(1st generation), G2(2nd generation), G3(3rd generation)에 이르는 MEMS 저장장치 프로토타입을 제시하였다[3-5].

표 1 MEMS 기반 저장장치와 플래시메모리, 하드디스크의 특성 비교(하드디스크와 플래시메모리는 현존하는 제품 사양에 근거한 값이지만 MEMS 기반 저장장치는 아직 제품이 출시되지 않은 관계로 카네기 멜론 대학의 모델에 기반한 값임. 따라서, 매체 간의 정확한 상대적 비교는 적절하지 않을 수 있음).

	하드디스크	NAND형 플래시메모리	MEMS 기반 저장장치
Size(mm)	101.6×147.0×26.1	12.0×17.0×1.0	10.0×10.0×2.0
Density(GB/cm^2)	0.14-0.24	0.49	1-10
Read access time(ms)	5-10	0.000025-0.025	0.56-0.80
Write access time(ms)	5-10	0.2-1.5	0.56-0.80
Shock resistance	Low	High	High
Cost($/GB)	0.21	45	1-3
Bandwidth(MB/s)	17.3-25.2	100	75.9-320
Power consumption	High	Low	Low

표 2 Gerneration MEMS 저장장치 프로토타입 [3]

	G1	G2	G3
bit width(nm)	50	40	30
sled acceleration(g)	70	82	105
access speed(kbit/s)	400	700	1000
X settling time(ms)	0.431	0.215	0.144
total tips	6400	6400	6400
active tips	640	1280	3200
max throughput(MB/s)	25.6	89.6	320
per-sled capacity(GB)	2.56	4.00	7.11
bidirectional access	no	yes	yes

표 2에서 보는 바와 같이 G1, G2, G3는 동시 활성화 가능한 헤드의 개수, 집적도, 정착시간, 매체의 접근 속도 측면에서 서로 다른 파라미터 값을 갖는다. 매체가 움직이고 매체 위의 고정된 헤드가 데이터를 읽고 쓰는 MEMS 기반 저장장치의 기본 구조는 동일하지만 G1에서 G3 모델로 갈수록 성능 척도에서 더 나은 파라미터 값을 보여주고 있다. 특히 G1 모델인 경우는 y축 방향으로 움직이며 데이터를 읽고 쓸 때 위에서 아래 방향으로 이동하는 동안에만 입출력이 가능하지만 G2, G3 모델은 양방향 이동시 모두 입출력이 가능하다. 대부분의 MEMS 기반 저장장치 연구에서는 하드웨어 측면에서 안정적으로 평가된 G2 프로토타입을 가장 많이 사용한다.

IBM은 Milipede 프로젝트를 통하여 MEMS 기반 저장장치 개발을 진행하고 있다[6-8]. IBM의 MEMS 기반 저장장치 프로토타입인 Milipede는 사각형 평면의 매체와 수천 개의 헤드를 포함하는 등 기본 구조가 카네기 멜론 대학과 동일하나 매체가 아닌 헤드가 움직이면서 데이터를 읽고 쓴다는 점과 헤드의 하드웨어적인 메커니즘에서 차이점을 나타내고 있다.

이외에도 Hewlett-Packard[9], Kionix[10], Nanochip[11] 등에서 MEMS 기반 저장장치에 관한 연구가 진행되고 있다.

2.4 논리적 블록과 물리적 섹터의 맵핑

MEMS 기반 저장장치는 그림 3에서 보는 바와 같이 사각형 매체가 수천 개의 영역(region)으로 구분되며 각 영역에서는 섹터 단위로 데이터의 저장 및 접근이 이루어진다. 카네기 멜론 대학의 MEMS G2 프로토타입인 경우 하나의 영역은 가로 2500 bits, 세로 2440 bits로 구성되고 하나의 섹터는 인코딩된 8 bytes의 데이터를 저장하기 위한 80 bits로 구성되며 섹터와 섹터 사이에는 10 bits의 서보 정보(servo information)를 두어 하드웨어 자체가 사용하는 부가정보를 담는다. 따라서 하나의 영역에는 2500x27개의 섹터가 존재한다. 보통 입출력의 기본 단위인 논리적 블록의 크기가 512 bytes인데 비해 MEMS 기반 저장장치의 섹터 크기는 8 bytes에 불과하다. 논리적 블록과 섹터의 크기가 512 bytes로 동일한 하드 디스크와 달리 그 크기가 상이한 MEMS 기반 저장장치에서는 하나의 논리적 블록을 64개의 서로 다른 영역에 존재하는 물리적 섹터로 맵핑시켜 동시에 읽고 쓸 수 있도록 하고 있다.

2.5 데이터 접근 방법

헤드가 영역 내의 특정(x, y) 위치에 있는 데이터를 읽고 쓰기 위해서는 해당 위치로 이동하는 위치 탐색 시간(positioning time)이 필요하며 하드디스크에서처럼 이 시간이 데이터 입출력 시간의 주요 부분을 차지하게 된다. 헤드가 1차원 상을 움직이며 데이터를 읽고 쓰는 하드 디스크와 달리 MEMS 기반 저장장치는 2차원 평면 상을 움직이며 데이터에 접근한다. 이때 x축, y축 방향으로의 이동은 각각 독립적으로 동시에 진행된다. 따라서, 특정(x, y) 위치로의 위치 탐색 시간 $time_{position}(x, y)$는 식 (1)과 같이 계산될 수 있다.

$$time_{position}(x, y) = max(time_{seek\_x}, time_{seek\_y}) \quad (1)$$

이 때, $time_{seek_x}$와 $timeseek_y$는 각각 x축과 y축 방향으로

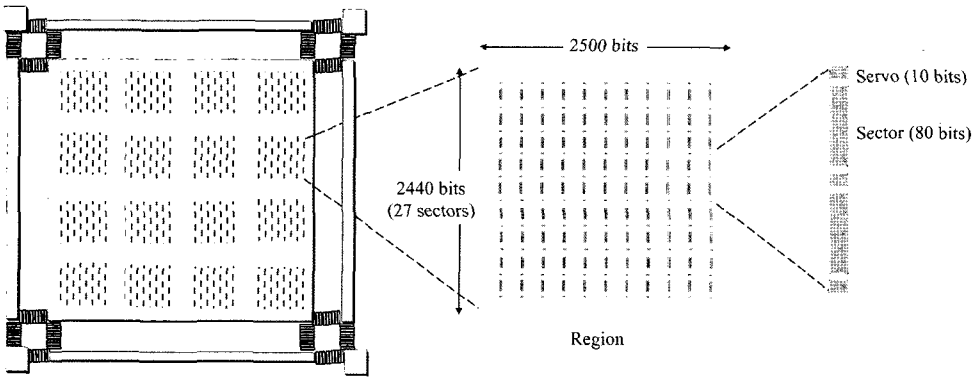


그림 3 MEMS 기반 저장장치의 물리적 구조와 기본 배치

의 이동시간을 의미한다. x축 방향으로의 이동시간인 $time_{seek_x}$는 실제 이동 시간과 추가적인 정착 시간(settling time)의 합으로 결정된다. 정착 시간이란 스프링에 매달린 매체가 이동 후 흔들림에 의해 이웃 섹터와의 간섭으로부터 벗어나는데 걸리는 시간을 뜻한다[4]. 매체의 이동으로 인한 흔들림은 x축과 y축 방향으로 모두 발생하지만 y축 방향의 진동으로 인한 정착 시간은 x축 방향에 비해 큰 의미를 가지지 않는다. 이는 매체가 원하는 위치로 이동한 후 y축 방향으로 움직이면서 데이터를 읽고 쓰기 때문에 이웃 섹터와 간섭을 일으키는 x축 방향과 달리 y축 방향의 흔들림은 단지 데이터를 읽고 쓰는 시간에 약간의 영향을 미칠 뿐이기 때문이다. 한편, y축 방향의 이동 시간 $time_{seek_y}$에는 정착 시간이 포함되지 않는 대신 전향시간(turn-around time)이 포함된다. 전향시간이란 매체의 이동 방향을 반대 방향으로 전환할 때 추가되는 지연시간을 의미한다.

2.6 하드 디스크와 상이한 물리적 특징

MEMS 기반 저장장치는 마그네틱 매체이며 기계적으로 헤드의 이동이 수반된다는 점에서 하드 디스크와 유사하다. 그러나 하드 디스크의 소프트웨어적 관리 기법을 그대로 적용하기에는 MEMS 기반 저장장치 고유의 물리적 특징들이 아래와 같이 상이하고 이에 따라 소프트웨어적 관리 기법을 매체의 특성에 맞게 설계하는 것이 필요하다.

· 비회전 매체

원판이 회전하는 하드 디스크와 달리 MEMS 기반 저장장치는 매체의 회전 없이 스프링의 제어에 의해 x축 및 y축 방향으로 이동하여 데이터에 접근한다. 하드 디스크의 위치탐색시간(positioning time)은 데이터가 있는 트랙의 위치로 헤드를 이동시키는 접근 시간(seek time)과 헤드가 읽고 쓰려는 섹터 위치에 도달하기까지 걸리는 회전 지연 시간(rotational latency)의 합으로 구성된다. 이때 회전 지연 시간은 매체의 회전 속도와 헤드가 원하는 트랙에 도착해서 회전 지연이 시작되는 시점의 불명확성 때문에 정확한 탐색 시간을 예측하기 힘들게 하는 요소였다. 그러나 MEMS 기반 저장장치는 2차원 평면상의 x축 및 y축 방향 이동을 고려한 계산이 가능하기 때문에 탐색 시간이 예측 가능하다. 또한 거리가 3mm 이하의 가까운 위치의 데이터를 읽을 경우, 항상 회전으로 인한 지연 시간이 추가되는 하드 디스크에 비해 더욱 효율적인 탐색 시간을 보인다[8]. 하드 디스크는 매체 회전으로 인해 전체 시스템 소모 에너지의 20-54%를 소비하여 에너지 효율성이 낮고 소음, 발열 등의 문제점을 가진다. 그러나 MEMS 기반 저장장치는 매체의 회전이 없고 매체의 크기가 디스크보다 작아 하드 디스크 에너지 소모량의 1/100 수준을 보인다.

· 탐색 시간의 25% 이상을 차지하는 정착시간

하드 디스크와 MEMS 기반 저장장치는 헤드의 이동 후 원하는 위치에 헤드가 정착하기 위해 추가적인 시간을 필요로 한다. 일반적으로 디스크에서의 정착 시간은 1-15ms의 전체 탐색시간 중에서 0.5ms 정도를 차지한다. 이에 비해 MEMS 기반 저장장치에서는 0.2-0.8ms의 전체 탐색 시간 중에서 0.2ms 정도를 정착시간으로 사용한다[5]. MEMS 기반 저장장치에서 전체 탐색 시간 중 정착시간이 차지하는 비율이 크고 정착시간은 x축으로의 이동 시에 발생하기 때문에 x축으로의 이동이 y축으로의 이동에 비해 탐색 시간에 주도적인 영향을 미친다.

· x축 및 y축 방향으로의 독립적 이동

MEMS 기반 저장장치는 2차원 평면상을 이동하면서 요청들을 처리한다. 이때 x축과 y축 방향으로의 이동은 서로 영향을 받지 않고 독립적으로 이루어지는 특징을 가지고 있다. 따라서, x축과 y축 방향으로의 이동 시간 중 큰 값으로 위치 탐색 시간이 결정된다.

· 병렬적 입출력 지원

하나의 헤드로 데이터를 읽고 쓰는 하드 디스크와 달리 MEMS 기반 저장장치는 수천 개의 헤드가 존재하며 이들은 동시에 활성화되어 데이터를 병렬적으로 읽고 쓸 수 있다. 카네기 멜론 대학의 G3 MEMS 모델인 경우 전체 6400개의 헤드 중 최대 3200개의 헤드가 동시에 활성화 될 수 있다.

· 빠른 read-modify-write 연산

MEMS 기반 저장장치는 메타 데이터의 업데이트처럼 동일한 위치에 대해 반복적인 접근 및 변경을 수행하는 read-modify-write 연산을 빠르게 처리할 수 있다. 하드 디스크인 경우 동일한 블록에 대한 읽기 연산과 갱신 및 쓰기 연산이 연속적으로 일어나기 위해서는 적어도 디스크가 한번 회전하는 시간만큼 대기하여야 한다. 그러나 MEMS 기반 저장장치에서는 y축 방향으로 움직이면서 데이터를 읽고 스프링의 복원력에 의해 평형 상태로 되돌아가면서 반대 반향으로 헤드를 전향하여 동일한 위치에 곧바로 데이터를 쓰는 것이 가능하다. 그림 4에서 보는 바와 같이 read-modify-write 연산을 위한 재탐색 시간은 MEMS 기반 저장장치가 하드 디스크의 1/85 수준임을 알 수 있다.

	Atlas 10K Disk	MEMS
Read	0.14	0.13
Reposition	5.98	0.07
Write	0.14	0.13
Total	6.26	0.33

그림 4 4Kbytes 크기의 데이터에 대한 read-modify-write 연산 비교(단위: ms) [3]

· 탐색 거리와 탐색 시간

하드디스크에서의 탐색 시간은 탐색 거리의 대략적인 상수 배가 된다. 그러나 MEMS 기반 저장장치에서는 이와 같은 사항이 적용되지 않는다. 그림 5의 왼쪽 그래프와 같이MEMS 기반 저장장치는 각 모서리가 스프링에 연결되어 있기 때문에 가장자리의 섹터를 접근할 때는 매체의 정 중앙의 섹터를 접근할 때보다 더 많은 탐색시간을 갖는다. 특히 정 중앙에서 수십 bits 사이까지의 탐색시간은 탐색거리에 비해 급속히 증가하며 그 구간을 지나면 탐색 거리에 비해 탐색 시간의 증가분이 완만해짐을 알 수 있다.

3. MEMS 기반 저장장치의 소프트웨어 관리 기술

MEMS 기반 저장장치에 대한 소프트웨어적 관리 기법 연구는 기존에 활발히 연구되었던 하드 디스크 관리 기법들을 MEMS 기반 저장장치에 적용시키는 연구와 MEMS 기반 저장장치의 특징을 반영하여 새로운 기법을 제안하는 연구로 분류할 수 있다. 각각의 관리 기법들은 디바이스 레벨에서부터 시스템 레벨에까지 다양한 계층에서 적용할 수 있도록 활발하게 연구되어 왔다. 본 장에서는 I/O 스케줄링, 데이터 배치, 그리고 저전력을 위한 에너지 관리 기법 등에 대한 연구를 소개한다.

3.1 I/O 스케줄링

하드디스크에서의 입출력 요청 스케줄링 연구는 오래 전부터 이루어져 왔다. 먼저 도착한 요청을 먼저 처리해주는 FCFS, 헤드의 이동이 가장 적은 지점을 먼저 서비스하는 SSTF[12], 헤드의 이동 시간(seek time)과 회전 지연 시간(rotational latency)을 함께 고려하는 SPTF[12] 등의 스케줄링 기법들이 제안된 바 있다. Griffin 등은 이러한 스케줄링 기법들을 MEMS 기반 저장장치에 그대로 맵핑하여 사용할 수 있음을 보였다[3]. 이는 하드디스크에서의 헤드 이동 시간과 회전 지연 시간을 각각 MEMS 기반 저장장치의 x축 및 y축 방향 이동 시간으로 맵핑할 경우 매우 유사한 결과를 얻을 수 있기 때문이다.

최근에는 MEMS 기반 저장장치의 물리적 특징을 고려한 스케줄링 연구들이 소개되고 있다[13-15]. Yu 등은 최소 신장 트리(minimum spanning tree)를 이용한 스케줄링 기법을 제안하였다[8]. 이 방법에서는 큐에 들어온 요청들을 x축, y축 방향의 이동 거리에 따라 최소 신장 트리를 구성한 후 요청된 작업들을 트리를 순회하는 순서로 서비스한다. 실험 결과에 의하면 이 방법이 SSTF보다는 우수한 성능을 나타내었으나 SPTF와는 대부분의 경우 거의 유사한 성능을 나타내었다. 이 방법은 최소 신장 트리를 생성하는 부가적인 시간이 필요하고, 온라인 스케줄링을 위해서는 새로운 요청이 들어올 때마다 트리를 재구성해 주는 시간이 필요하다는 약점이 있다. 또한, SPTF나 SSTF에서와 마찬 가지로 일부 요청이 지나치게 오래 기다려야 하는 기아 현상이 발생할 수 있다.

Hong 등은 기아 현상을 어느 정도 완화시키고 위치에 따른 고른 서비스를 보장하기 위해 MEMS 기반 저장장치의 각 영역(region)을 여러 개의 지역으로 나누어 지역 내의 요청은 SPTF 순서로 처리하고, 지역과 지역 간에는 C-SCAN 순서로 스케줄링하는 새로운 스케줄링 방법을 제시하였다[14]. 이 방법은 평균 응답 시간 측면에서는 SPTF보다 약간의 성능 저하가 발생했으나 요청간 응답 시간의 편차에 있어서는 개선된 결과를 보여주었다.

Schlosser 등은 MEMS 기반 저장장치의 탐색 시간 중 x축 방향의 움직임에는 부가적인 정착 시간이 소요되므로 x축 방향의 이동 거리만을 고려하는 것이 x축과 y축 양방향의 이동 거리를 모두 고려하는 방법에 비해 효율적이라는 의견을 제시하였다[5]. 그들은 SDF (Shortest Distance First)라는 알고리즘을 통해 이와 같은 사실을 보였다. SPTF 알고리즘이 x축과 y축 양방향의 이동 시간을 함께 고려한 총 탐색 시간이 가장 짧은 지점을 먼저 서비스하는 것과 달리 SDF는 x y 평면 상의 기하 거리(Euclidean distance)가 가장 가까운 위치를 먼저 서비스한다. 실험 결과에 의하면 SDF는 x축 방향의 이동거리만을 고려하는 SSTF보다도 좋지 않은 성능을 나타내었다. 이 결과를 토대로 그들은 MEMS 기반 저장 장치의 탐색 시간이 x축 방향의 이동 및 정착 시간에 크게 좌우되며, 이는 하드디스크에서 헤드 이동 시간이 회전 지연 시간에 비해 상대적으로 크다는 점과 유사하므로 디스크 스케줄링을 MEMS 기반 저장 장치에 그대로 적용하는 것이 적절하다고 언급하고 있다.

Lee 등은 탐색 시간 기반 알고리즘인 SPTF와 헤드의 병렬성을 함께 고려하는 새로운 알고리즘을 제안

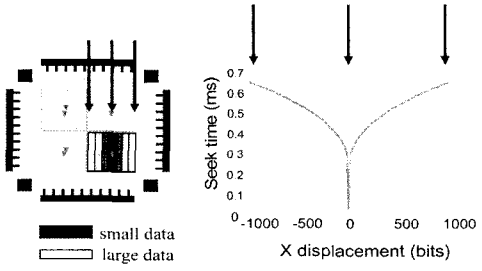
하였다[15]. 이 방법은 요청이 들어온(x, y) 지점 중 현재 헤드 위치와의 거리가 가까우면서 해당 위치에 들어온 요청의 개수가 많은 지역을 우선적으로 서비스해주는 것을 기본 아이디어로 한다. MEMS 기반 저장장치에서는 수천 개의 고정된 헤드가 자신이 맡은 영역을 서비스하므로 각 영역에서 상대적으로 동일한 (x, y) 위치에 있는 요청들은 서로 다른 헤드에 의해 동시에 처리될 수 있다. P-SPTF 스케줄링 알고리즘은 이러한 특징을 이용하여 모든 영역을 통틀어서 많은 요청이 대기 중인(x, y) 지점에 대해 서비스 우선순위를 높여주자는 것이 기본적인 원리이다. 이러한 위치가 먼저 서비스될 경우 많은 요청들을 큐에서 한꺼번에 제거할 수 있어 요청들의 평균 응답 시간이 빨라지게 된다.

스케줄링 알고리즘은 요청들의 평균 응답 시간을 줄이는 것이 가장 큰 목표이지만 요청들 간의 응답 시간의 편차가 너무 클 경우 형평성 차원에서 적절하지 않다. 즉, 평균 응답 시간을 향상시키기 위해 일부 요청을 무한정 기다리게 하는 기아 현상을 방지하는 것도 빠른 응답 시간과 함께 고려되어야 할 사항이다. P-SPTF 알고리즘에서는 현재 헤드의 위치에서 멀리 떨어진 곳이나 대기 중인 요청의 수가 적은 위치인 경우 스케줄링에서 지속적으로 소외되어 기아현상을 초래할 수 있다. 디스크 스케줄링 분야에서는 이와 같은 문제를 해결하기 위해 Jacobson과 Wilkes가 SPTF 알고리즘에 노화 요소(aging factor)를 적용시킨 ASPTF 알고리즘을 제안한 바 있다[12]. ASPTF는 $time_{position} - w\ time_{wait}$ 값이 가장 작은 지점을 우선적으로 서비스하며, $time_{position}$과 $time_{wait}$은 각각 헤드의 이동 시간과 큐에서 기다린 시간을 의미하며 w는 상수이다. PA-SPTF (Parallelism-aware with Aging extension SPTF) 알고리즘은 P-SPTF 알고리즘을 확장하여 요청간 대기 시간의 형평성을 고려한 알고리즘이다[15]. (x, y) 위치에 쌓여 있는 요청의 개수를 헤드의 이동 시간으로 나누어 서비스 우선순위를 계산하는 P-SPTF 알고리즘과 달리 PA-SPTF는 (x, y) 위치에 대기 중인 요청들이 기다린 시간의 합을 헤드의 이동 시간으로 나눠 그 값이 큰 지점을 우선적으로 서비스한다. (x, y) 지점에 대기 중인 요청이 많을수록 요청들이 기다린 시간의 합이 커지므로 이 알고리즘은 기본적으로 병렬성을 고려하고 있으며, 비록 대기 중인 요청이 하나뿐이고 헤드의 현재 위치에서 멀리 떨어진 위치라 하더라도 기다린 시간이 길어지면 우선순위가 높아지므로 노화 요소를 반영하여 기아 현상을 해소할 수 있다.

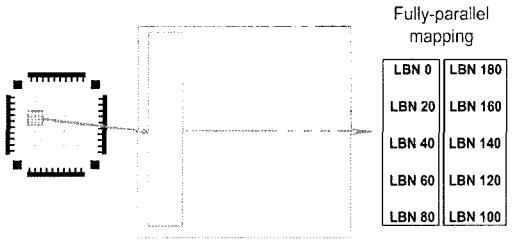
3.2 데이터 배치

I/O 스케줄링과 더불어 데이터 배치는 성능 향상을 좌우하는 중요한 소프트웨어적 기법 중 하나이다. Ganger 등은 MEMS 기반 저장장치를 위한 이분배치(bipartite layout) 방법을 제안하였다[5]. MEMS 기반 저장장치는 각 모서리가 스프링에 의해 연결되어 헤드 이동 시간이 영역 내의 위치에 따라 급격히 달라진다. 그림 5에서와 같이 영역의 중심 부분은 상대적으로 작은 거리 이동에도 많은 이동 시간의 차이를 보인다. 각 모서리에 위치한 스프링의 장력이 평형이 되어 영역의 외곽에 비해 상대적으로 이동이 자유롭다. 이 때문에 사이즈가 작고 빈번히 요청되는 데이터나 메타데이터는 영역의 중심 부분에 배치하고 한번 헤드가 움직이면 많은 양을 한꺼번에 읽어야 하는 멀티미디어 데이터와 같이 사이즈가 큰 데이터는 외곽에 배치하는 것이 효율적이라는 것이 이분 배치의 기본 아이디어이다.

이웃하는 논리적 블록들을 수천 개의 헤드로 동시에 접근할 수 있도록 배치하여 헤드의 병렬성을 최대한 활용하는 데이터 배치기법도 제안되었다[16]. MEMS 기반 저장장치에서는 하나의 논리적 블록이 64개의 서로 다른 영역에서 상대적으로 동일한 섹터에 맵핑되므로 동시 활성화가 가능한 1280개의 헤드를 고려하면 최대 20개의 이웃한 논리적 블록을 동시에 읽고 쓸 수 있도록 데이터를 배치할 수 있다. 그리고 2장에서



이분배치(bipartite layout) 기법[4]



병렬성을 고려한 지그재그형 데이터 배치[15]

그림 5 다양한 MEMS 데이터 배치 기법

언급되었듯 MEMS 기반 저장 장치는 x축 방향의 이동 및 정착 시간이 전체 탐색시간의 주요 부분을 차지하므로 x축 방향의 움직임을 최소화할 수 있는 데이터 배치가 요구된다. 헤드의 병렬성을 최대한 활용하는 데이터 배치 기법은 그림 5와 같이 논리적 블록 20개를 동시에 접근 가능한 서로 다른 영역의 동일한 위치에 배치한다. 그리고 x축 방향의 불필요한 이동을 없애기 위해 논리적 블록을 y축을 따라 연속적으로 배치한다. 이 때의 배치도 논리 블록 20개가 동시에 접근 가능하도록 배치한다. 또한, y축 방향의 끝까지 데이터를 배치한 경우, 헤드의 양방향 접근을 이용하기 위해 바로 인접한 섹터에 y축을 따라 반대방향으로 데이터를 배치한다. 이 기법은 데이터 참조에 있어 공간 지역성(space locality)을 가지고 있는 경우 연속된 논리적 블록을 불필요한 탐색없이 접근하게 되어, 순차 참조의 경우 데이터 접근 시간을 절반 이하까지 줄일 수 있게 된다.

3.3 결함 허용성(Fault Tolerance)

3.3.1 디바이스 수준

MEMS 기반 저장장치는 수천 개의 헤드를 가지고 있고 미세한 나노 수준의 공정이 이루어지므로 확률적으로 헤드의 하드웨어적 고장이 발생할 가능성이 높다. 이 경우 여러 개의 헤드 중 일부 헤드를 결함 복구를 위한 여유분 헤드로 설정하여 사용할 수 있다[4].

또한 하드 디스크와 같이 마그네틱 매체 자체의 결함이 발생할 수 있다. 그러나 MEMS 기반 저장장치에서 매체 결함은 보통 1-4개 정도의 영역에서만 발생한다[4]. 이러한 경우 MEMS 기반 저장장치의 섹터마다 저장되는 ECC를 이용하면 이러한 하드웨어적인 결함을 해결할 수 있다. 예를 들어, 하나의 논리적 블록이 64개의 섹터에 나누어 저장되는 것처럼 데이터를 병렬적으로 읽고 써야 하는 경우 헤드나 매체의 결함으로 일부 손실된 데이터가 발생할 수 있다. 이때 각 섹터에 저장된 ECC들을 이용하여 손상된 헤드나 데이터 손실을 발견하고 결함을 복구 할 수 있으며 여분의 헤드를 이용하여 RAID 처럼 안정성을 높일 수 있는 방법도 있다.

MEMS 기반 저장장치의 기계적 특징이 RAID 5와 같은 결함내구성 구조의 구현에 효율적일 수 있다. MEMS 기반 저장장치에서 제공되는 빠른 read-modify-write 연산은 RAID 5 구조에서 자주 쓰이는 연산이다. 예를 들어, RAID 5에서의 데이터 쓰기 연산은 이전 데이터와 패리티를 각각 읽고 새로운 패리티를 계산한 후 새로운 데이터와 패리티를 쓰는 적어도 두 번의 read-modify-write을 필요로 한다. 다섯 대의 하드 디스크로 구성된 RAID 5의 경우, 네 대의 하드 디스크에 데이터 분할 배치를 한 경우에 비해 응답시간이 77% 증가하나 MEMS 기반 저장장치에서는 27% 증가만을 보였다[4].

3.3.2 시스템 수준

갑작스런 컴퓨터 시스템 고장으로 인해 파일 시스템이나 데이터베이스에서의 데이터 일관성이 깨어지는 문제를 해결하기 위해 하드 디스크에서는 저널링 기법을 사용한다. 그러나 MEMS 기반 저장장치는 하드 디스크에 비해 빠른 서비스 시간을 제공하기 때문에 데이터 갱신 시 동기적으로 쓰기 연산을 수행하는 방법을 고려해 볼 수 있다[4].

3.4 에너지 관리 기법

모바일 스토리지로서의 효과적인 사용을 위해 MEMS 기반 저장장치의 에너지 소모를 줄이는 연구가 디바이스 레벨과 시스템 레벨에서 각각 연구되어 왔다.

3.4.1 디바이스 수준

가장 대표적인 디바이스 레벨에서의 에너지 관리 기법 연구는 공격적 스핀-다운(Aggressive spin-down) 방법이다[17]. 이 방법은 MEMS 기반 저장장치의 에너지 상태를 활성과 비활성 모드로 구분하고 I/O 큐가 비어 있을 때 곧바로 MEMS 기반 저장장치를 비활성 모드로 전환하여 불필요한 에너지를 절감하는 방법을 사용한다. 이러한 모드 전환 방법은 하드 디스크에서 많이 사용되어 온 저전력 관리 기법이다. 그러나 하드 디스크에서는 비활성모드에서 활성 모드로의 전환 시 정지된 매체를 다시 회전시켜야 하기 때문에 많은 에너지 소모와 함께 시간적 지연이 발생한다. 따라서 이러한 접근 방법이 하드 디스크에서는 정교한 미래 요청 시점에 대한 예측을 필요로 했다. 그러나 MEMS 기반 저장장치는 매체의 회전이 없고 매체 크기가 디스크에 비해 작기 때문에 비활성 모드에서 활성 모드로의 전환이 용이하다. 15-25초의 매체 활성화 시간이 필요한 하드 디스크와 달리 MEMS 기반 저장장치는 매체를 활성화시키는데 0.5 ms 이하의 시간이 소요되고 그에 비해 에너지 소모는 전체 에너지 소모의 50% 이상을 절감시킬 수 있는 것으로 평가되었다.

3.4.2 시스템 수준

시스템 레벨에서의 에너지 관리 기법으로는 저장장치로의 요청 빈도를 최대한 줄여 에너지를 절감하기 위한 캐싱 방법[18]과 NVRAM을 이용한 방법[19]이 연구되었다. 이들은 2차 저장장치의 종류에 상관없이 적용되는 방법이다.

MEMS 기반 저장장치는 하드디스크와 같이 헤드의 움직임으로 데이터를 접근하는 기계적 특성으로 인해 요청이 들어온 위치로의 이동에 많은 시간과 에너지를 소모하게 된다. Yin 등의 요청 병합(request merging) 방법은 한 번의 탐색 과정(seek process)으로 여러 요청을 동시에 처리하여 에너지를 줄이기 위한 접근 방법이다[17]. 기본적으로 MEMS 기반 저장장치가 여러 개의 헤드를 가지고 있고 병렬적으로 입출력이 가능하다는 물리적 특성을 반영하였다. 서로 다른 영역이지만 영역내 요청의 위치가 동일한 경우는 동시 처리가 가능하기 때문에 시스템 레벨에서 이들을 하나의 요청으로 병합하여 불필요한 요청 탐색 횟수의 감소로 에너지를 줄일 수 있다.

MEMS 기반 저장장치는 수천 개의 헤드로 동시에 읽고 쓸 수 있는 병렬성을 지원하지만 발열 및 에너지 손실 문제로 일반적으로 모든 헤드를 동시에 활성화하지는 않는다. 헤드의 활성화에 드는 에너지는 매체를 움직이는데 드는 에너지보다 훨씬 크기 때문에 이에 대한 제한을 두고 있다. 최대 활성화 헤드 개수를 제공하는 카네기 멜론의 G3 프로토타입인 경우에도 전체 6400개의 헤드 중 동시에 활성화 가능한 헤드를 3600개로 제한하고 있다. 부분 섹터 접근(subsector access) 방법은 필요한 데이터가 저장된 영역의 헤드만을 활성화시켜 에너지 소모를 줄이는 방법으로 요청되는 데이터가 하나의 512 byte 논리 블록보다 작은 크기의 데이터 일 때 적용된다[17]. 이는 2장에서 언급했듯이 MEMS 기반 저장장치에서 하나의 섹터가 8byte로 512 byte 크기의 디스크 섹터보다 작기 때문에 가능한 기법이다. 디스크에서는 ECC 정보와 데이터 탐색 시간의 부하로 인해 512 byte 논리적 블록보다 작은 단위로 데이터를 접근하는 것이 불가능하다. 그러나 MEMS 기반 저장장치는 각 섹터 별로 ECC가 저장되어 하나의 논리적 블록 중 필요한 섹터를 부분적으로 읽고 나머지 헤드는 비활성화하여 에너지 소모를 줄일 수 있다.

3.5 MEMS 기반 저장장치의 활용

본 절에서는 MEMS 기반 저장장치의 실제 응용 분야에 대한 연구를 몇 가지 소개하겠다.

3.5.1 메타데이터 저장장치

Hong 등은 실제 파일 데이터는 하드 디스크에 저장하고 파일 시스템의 메타데이터를 MEMS 기반 저장장치에 저장하는 새로운 구조를 제안하였다[20].

3.5.2 관계형 데이터베이스와 MEMS 기반 저장장치

관계형 데이터베이스 설계 시 하드디스크를 이용하게 되면 보통 인덱스를 이용하여 행중심 순서나 열중심 순서로 저장한다. 그러나 행중심으로 저장한 경우 열중심의 접근이 필요하게 되면 불필요한 데이터 접근 지연시간이 발생하게 된다. Yu 등은 MEMS 기반 저장장치의 2차원 특성을 활용하여 관계형 데이터베이스를 저장함에 있어서 행중심과 열중심 접근이 모두 가능하도록 설계하여 데이터 접근 효율성을 높였다.

4. 결 론

본 고에서는 차세대 저장장치인 MEMS 기반 저장장치의 기본 구조, 하드 디스크와의 특징 비교와 함께 I/O 스케줄링, 데이터 배치, 결함 허용성, 저전력 관리 등의 소프트웨어적 관리 기법을 소개하였다. 상용화되지 않은 차세대 저장장치인 만큼 저장장치 운용을 위한 기본 소프트웨어적 관리 기법들은 중요한 기반 기술이 될 것이다. MEMS 기반 저장장치의 사용이 모바일 스토리지에서부터 대용량 서버 스토리지까지 그 범위가 확장 가능한 저장장치인 만큼 이에 대한 원천적인 연구가 지속적으로 이루어져야 할 것이다.

감사의 글

본 연구는 서울시 산학연 협력사업에 의해 지원되었음.

참고문헌

[1] E. Grochowski, "IBM leadership in disk storage technology", IBM Corporation, 2000.

[2] Y. Zhu, "An Overview on MEMS-based Storage, Its Research Issues and Open Problems," Proceedings of 2nd International Workshop on Storage Network Architecture and Parallel I/Os, Sept. 30, 2004.

[3] S. Schlosser, J. Griffin, D. Nagle, and G. Ganger, "Designing computer systems with MEMS-based storage," *9th Int'l Conf. Architectural Support for* Programming Languages and Operating Systems, 2000.

[4] J. Griffin, S. Schlosser, G. Ganger, and D. Nagle, "Modeling and performance of MEMS-based storage devices," ACM SIGMETRICS Conf., pp. 56-65, 2000.

[5] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle, "Operating system management of MEMS-based storage devices," in 4th Symp. on Operating Systems Design & Implementation(OSDI), pp. 227-242, San Diego, California, Oct 2000.

[6] M. Despont, J. Brugger, U. Drechsler, U. Durig,

W. Haberle, M. Lutwyche, H. Rothuizen, R. Stutz, R. Widmer, H. Rohrer, G. Binnig, and P. Vettiger. "VLSI-NEMS Chip for AFM Data Storage," In Proceedings 12th International Workshop on Micro Electro Mechanical Systems, pages 564{569, Orlando, FL, 17{21 Jan. 1999.

[7] B. Schechter and M. Ross. Leading The Way In Storage. IBM Research Magazine, 35(2), 1997. P. Vettiger, M. Despont, U. Drechsler, U. Durig, W. Haberle, M. I. Lutwyche, E. Rothuizen, R. Stutz, R. Widmer, and G. K. Binnig. "The 'Millipede' – More Than One Thousand Tips for Future AFM Data Storage," IBM Journal of Research and Development, 44(3):323–340, 2000.

[8] J. F. Alfaro and G. K. Fedder, "Actuation for probe-based mass data storage," in Proceedings of the 2002 International Conference on Modeling and Simulation of Microsystems, 2002, Vol. 1, pp. 202–205.

[9] J. W. Toigo. Avoiding a Data Crunch|A Decade Away: Atomic Resolution Storage. Scienti_c American, May 2000. http://www.sciam.com/2000/0500issue/0500toigbox6.html.

[10] T. Davis. Realizing a Completely Micromechanical Data Storage System(Kionix, Inc.). In Diskcon 99 International Technical Conference, Sept. 1999.

[11] Nanochip Inc. Nanochip, Inc. Product Overview. In Diskcon 99 International Technical Conference, Sept. 1999.

[12] B. Worthington, G. Ganger, and Y. Patt, "Scheduling Algorithms for Modern Disk Drives," ACM SIGMETRICS Conf., pp. 241–251, 1994.

[13] H. Yu, D. Agrawal, and A. Abbadi, "Towards optimal I/O scheduling for MEMS-based storage," 20th IEEE/11th NASA Goddard Conf. Mass Storage Systems and Technologies, 2003.

[14] B. Hong, S. Brandt, D. Long, E. Miller, K. Glocer, and Z. Peterson, "Using MEMS-based Storage in Computer Systems-Device Modeling and Management," ACM Transaction on Storage, Vol. 2, No.2, 2006, pp 139–160.

[15] Soyoon Lee, Hyokyung Bahn, and Noh, S.H., "Parallelism-Aware Request Scheduling for MEMS-based Storage Devices," 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems(MASCOTS 2006), Sept. 2006.

[16] Soyoon Lee and Hyokyung Bahn, "Data allocation in MEMS-based mobile storage devices," IEEE Transactions on Consumer Electronics, Vol. 52, No. 2, May. 2006.

[17] Y. Lin, S. Brandt, D. Long, and E. Miller, "Power conservation strategies for MEMS-based storage devices," International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, FortWorth, TX, October 2002.

[18] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber, "Storage alternatives for mobile computers," in Proceedings of the 1st Symposium on Operating Systems Design and Implementation(OSDI), pp. 25 - 37, Monterey, CA, Nov. 1994.

[19] M. Wu and W. Zwaenepoel, "eNVy: a non-volatile, main memory storage system," in Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS), pp. 86 - 97. ACM, Oct. 1994.

[20] Bo Hong, Feng Wang, Scott A. Brandt, Darrell D. E. Long, Thomas J. E. Schwarz, S. J., "Using MEMS-based storage in computer systems---MEMS storage architectures," ACM Transaction on Storage, Vol. 2, No.1, 2006, pp 1–21.

이 소 윤

2004 이화여대 컴퓨터학과 학사
2006 이화여대 컴퓨터학과 석사
2006~현재 이화여대 컴퓨터학 박사과정
관심분야 : 스토리지 시스템 관리, 시스템 최적화, 지능형 스토리지 시스템, 저전력시스템
E-mail : sounie@ewhain.net

반 효 경

1997 서울대학교 계산통계학과 학사
1999 서울대학교 전산과학과 석사
2002 서울대학교 컴퓨터공학부 박사
2002~현재 이화여자대학교 컴퓨터학과 교수
관심분야 : 운영체제, 저장 시스템 관리, 저전력 시스템, 웹 캐싱 등
E-mail : bahn@ewha.ac.kr