MEMS 기반 저장장치를 위한 병렬성 기반 스케줄링 기법

(Parallelism-aware Request Scheduling for MEMS-based Storages)

이 소 윤 † 반 효 경 †† 노 삼 혁 †††
(Soyoon Lee) (Hyokyung Bahn) (Sam H Noh)

요 약 MEMS 기반 저장장치는 높은 대역폭과 저전력성, 고집적도, 저가 등의 특성으로 인해 모바일 기기에서 대용량 서버 시스템에 이르는 다양한 환경에서 사용가능한 차세대 저장장치이다. MEMS 기반 저장장치는 원판이 회전하는 하드디스크와 달리 매체가 사각형 구조로 되어 있으며, 하나의 매체에 동시에 읽고 쓸 수 있는 수천 개의 헤드가 존재한다. 본 논문에서는 이와 같은 MEMS 기반 저장장치의 물리적 특성에 적합한 새로운 입출력 요청 스케줄링 기법을 제안한다. 새롭게 제안한 알고리즘은 사각형 평면 상에서의 헤드의 탐색 시간뿐 아니라 수천 개의 헤드에 의한 병렬적인 입출력을 고려한다. 또한, 기아 현상(starvation)을 극복하기 위해 알고리즘에 노화 요소(aging factor)를 반영한다. 트레이스 기반 모의 실험을 통해 본 논문이 제안한 스케줄링 알고리즘이 기존의 대표적인 알고리즘인 SPTF(Shortest Positioning Time First) 알고리즘에 비해 평균 응답 시간과 응답 시간의 편차 측면에서 각각 39.2%와 62.4%가 향상됨을 보였다.

키워드 : MEMS 기반 저장장치, 요청 스케줄링, 모바일 저장장치, 병렬성

Abstract MEMS-based storage is being developed as a new storage media. Due to its attractive features such as high-bandwidth, low-power consumption, high-density, and low cost, MEMS storage is anticipated to be used for a wide range of applications from storage for small handheld devices to high capacity mass storage servers. However, MEMS storage has vastly different physical characteristics compared to a traditional disk. First, MEMS storage has thousands of heads that can be activated simultaneously. Second, the media of MEMS storage is a square structure which is different from the platter structure of disks. This paper presents a new request scheduling algorithm for MEMS storage that makes use of the aforementioned characteristics. This new algorithm considers the parallelism of MEMS storage as well as the seek time of requests on the two dimensional square structure. We then extend this algorithm to consider the aging factor so that starvation resistance is improved. Simulation studies show that the proposed algorithms improve the performance of MEMS storage by up to 39.2% in terms of the average response time and 62.4% in terms of starvation resistance compared to the widely acknowledged SPTF (Shortest Positioning Time First) algorithm.

Key words : MEMS-based storage, request scheduling, mobile storage, parallelism

1. 서 론

최근 멀티미디어 등 대용량 데이타의 폭발적 증가에 따라 스토리지의 용량 수요가 급격히 증가하고 있다. 이러한 요구에 부응하기 위해 새로운 저장 매체 기술과 제품들이 하루가 다르게 등장하고 있다. 우선 대용량 서버 환경에서의 스토리지인 하드디스크의 사용량은 매년 60% 이상씩 증가하고 있으며 디스크 매체의 집적도 또한 지속적으로 높아지고 있다[1]. 다른 한편으로는 휴대기기의 사용이 보편화됨에 따라 모바일 스토리지에 대한

· 본 연구는 한국과학재단의 특정기초연구(R01-2004-000-10188-0)와 서울시 산학연협력사업(10661M0207332)의 지원으로 수행됨
† 학생회원 : 이화여자대학교 컴퓨터학과
sounie@ewhain.net
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
bahn@ewha.ac.kr
††† 종신회원 : 홍익대학교 정보컴퓨터공학부 교수
samhnoh@hongik.ac.kr
논문접수 : 2006년 8월 21일
심사완료 : 2006년 11월 13일

용량 수요가 급증하고 있다. 마이크로 디스크와 플래시 메모리가 모바일 스토리지 시장에 현존하는 대표적인 매체이다. MEMS(MicroElectro Mechanical Systems) 기반 저장장치는 모바일 스토리지와 대용량 서버 스토리지로 모두 사용될 수 있는 차세대 저장 매체의 한 종류이다[2,3]. MEMS 기반 저장장치가 이와 같이 서로 다른 환경에서 사용 가능한 것은 각 환경의 요구 사항을 모두 만족하기 때문이다. 표 1은 하드디스크와 플래시메모리, MEMS 기반 저장장치를 여러 가지 측면에서 비교한 것이다[1,4]. 표에서 보는 바와 같이 MEMS 기반 저장장치는 쓰기 속도가 상대적으로 느린 플래시메모리와 달리 읽기/쓰기 속도가 균일하게 빠르며 입출력 대역폭(bandwidth)이 매우 높다. 하드디스크에 비해서는 읽기/쓰기 속도가 10배 정도 빠르며, 비용 측면에서는 Gbyte당 최저 3달러 정도로 플래시메모리에 비해 매우 저렴하다. 이와 같은 특징은 MEMS 기반 저장장치가 대용량 서버 환경에서 하드디스크를 대체하거나 하드디스크와 함께 사용하기에 적합한 저장장치임을 보여준다. 또한, MEMS 기반 저장장치는 플래시메모리만큼 부피가 작으면서 물리적 충격에 강하고 전력 소모가 적어 모바일 스토리지로 사용하기에 적합한 특성을 가지고 있다.

한편, MEMS 기반 저장장치는 기존의 저장 매체들과는 상이한 물리적 특성을 가지고 있다. MEMS 기반 저장장치는 하드디스크와 유사한 마그네틱 장치이지만 원판이 회전하고 헤드가 이동하는 하드디스크와 달리 사각형 구조의 마그네틱 매체(magnetic media)가 스프링에 의해 x축 및 y축 방향으로 직접 움직이고 수천 개의 고정된 헤드가 이를 읽고 쓰는 방식으로 동작한다(그림 1). 이러한 물리적 구조 때문에 읽기 쓰기 동작에 있어 MEMS 기반 저장장치는 하드디스크와 크게 다른 2가지 특징을 가지고 있다[2,5,7]. 그 첫째는 데이타의 접근을 위해 x축과 y축 방향으로 동시에 매체를 이동시킨다는 점이고, 둘째는 수천 개의 헤드에서 데이타를 병렬적으로 읽고 쓸 수 있다는 점이다. 이러한 점은 하드디스크에서는 볼 수 없었던 MEMS 기반 저장장치만의 고유한 특성이며 이로 인해 디스크에서 사용되던 다양한 소프트웨어적 관리 기법들이 MEMS 기반 저장장치에 맞게 재구성되어야 한다[6].

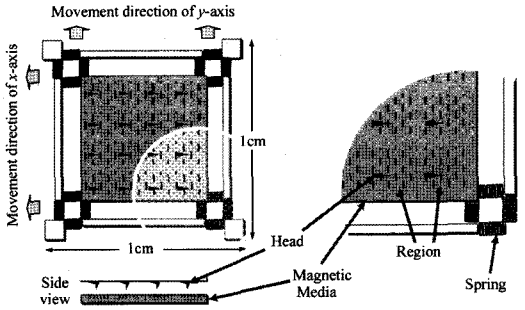


그림 1 MEMS 기반 저장장치의 물리적 구조

입출력 요청에 대한 스케줄링 기법은 MEMS 기반 저장장치의 이와 같은 특징의 영향을 크게 받는 메커니즘 중 하나이다. 하드디스크의 경우 헤드의 1차원적 이동을 제어하는 거리 기반 스케줄링을 수행하지만, MEMS 기반 저장장치에서는 2차원 평면 상에서 x축, y축 양방향의 이동을 제어해야 하므로 더욱 복잡한 스케줄링 문제가 발생한다.

본 논문에서는 MEMS 기반 저장장치의 물리적 특성을 고려하는 새로운 요청 스케줄링 알고리즘을 제안한다. 본 논문이 제안하는 스케줄링 알고리즘은 앞에서 언급한 MEMS 기반 저장 장치의 2가지 특성을 반영한다. 즉, x축과 y축 양방향의 이동을 고려하여 탐색 시간이 적으면서 여러 헤드가 동시에 데이타를 읽고 써야 하는 위치일수록 스케줄링의 우선 순위를 높여 빠른 서비스를 가능하게 한다. 하지만, 이와 같은 방법을 사용할 경우 탐색 시간이 길거나 요청이 하나밖에 안 들어와 있는 위치의 경우 대기 시간이 무한정 길어지는 기아 현상(starvation)이 발생할 수 있다. 이를 방지하기 위해 본 논문에서는 노화(aging) 기법을 활용한 새로운 버전

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	3-10
Bandwidth (MB/s)	17.3-25.2	100	75.9-320
Power consumption	High	Low	Low

의 스케줄링 알고리즘을 제안하여 기아 현상을 극복하고 요청들의 대기 시간에 대한 형평성을 확보한다. 트레이스 기반 모의 실험을 통해 본 논문에서 제안한 스케줄링 알고리즘이 기존 스케줄링 알고리즘인 FCFS(First-Come First-Served), SSTF(Shortest Seek Time First), SPTF(Shortest Positioning Time First) 등에 비해서 월등한 평균 응답 시간을 보임을 보였다.

2. 관련 연구

하드디스크에서의 입출력 요청 스케줄링 연구는 오래 전부터 이루어져 왔다. 먼저 도착한 요청을 먼저 처리해 주는 FCFS, 헤드의 이동이 가장 적은 지점을 먼저 서비스하는 SSTF[8], 헤드의 이동 시간(seek time)과 회전 지연 시간(rotational latency)을 함께 고려하는 SPTF[9] 등의 스케줄링 기법들이 제안된 바 있다. Griffin 등은 이러한 스케줄링 기법들을 MEMS 기반 저장장치에 그대로 매핑하여 사용할 수 있음을 보였다[6]. 이는 하드디스크에서의 헤드 이동 시간과 회전 지연 시간을 각각 MEMS 기반 저장장치의 x축 및 y축 방향 이동 시간으로 매핑할 경우 매우 유사한 결과를 얻을 수 있기 때문이다.

최근에는 MEMS 기반 저장장치의 물리적 특징을 고려한 스케줄링 연구들이 소개되고 있다[5,7]. Yu 등은 최소 신장 트리(minimum spanning tree)를 이용한 스케줄링 기법을 제안하였다[7]. 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[5]. 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[10]. 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.

3. 병렬성 기반 스케줄링 알고리즘

본 장에서는 먼저 스케줄링과 관련된 MEMS 기반 저장 장치의 물리적 특성들을 설명하고 이러한 특성을 반영하는 새로운 스케줄링 알고리즘에 대해 소개한다.

3.1 MEMS 기반 저장장치의 물리적 특성

카네기 멜론 대학[11], IBM 쥐리히 연구소[3], HP (Hewlett-Packard) 연구소[12] 등에서 MEMS 기반 저장장치에 관한 주요 프로젝트가 진행되고 있다. 이들 프로젝트에서 연구 개발 중인 MEMS 기반 저장장치는 그 물리적인 특성이 조금씩 다르지만 그림 1과 같이 기본적인 특성은 동일하다. 저장장치의 마그네틱 매체가 수천 개의 작은 영역(region)으로 구성되고 각 영역에는 헤드가 하나씩 존재하여 그 영역의 데이타를 읽고 쓸 수 있다. 헤드가 영역 내의 특정 (x, y) 위치에 있는 데이타를 읽고 쓰기 위해서는 해당 위치로 이동하는 위치 탐색 시간(positioning time)이 필요하며 하드디스크에서처럼 이 시간이 데이타 입출력 시간의 주요 부분을 차지하게 된다. 그러나, 하드디스크와 달리 MEMS 기반 저장장치에서는 헤드가 고정되어 있으며 마그네틱 매체 자체가 스프링의 제어에 의해 읽고 쓰려는 (x, y) 위치를 헤드 쪽으로 이동시킨다. 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 $time_{seek_y}$는 각각 x축과 y축 방향으로의 이동 시간을 의미한다. 현재 대부분의 MEMS 기반 저장장치 구조에서 $time_{seek_x}$에는 추가적인 정착 시간(settling time)이 필요하기 때문에 $time_{seek_y}$에 비

해 $time_{seek_x}$가 전체 탐색 시간에 지배적인 영향을 미친다. 정착 시간이란 마그네틱 매체가 이동 후 흔들림에 의해 이웃 섹터와의 간섭으로부터 벗어나는데 걸리는 시간을 뜻한다[2]. 정착 시간이 x축 방향으로만 소요되는 이유는 매체가 원하는 위치에 도달한 후 y축 방향으로 움직이면서 데이타를 읽고 쓰기 때문이다. 즉, 탐색 후 x축 방향으로의 진동은 트랙을 벗어나는 간섭을 일으키므로 정착이 필요한데 비해 y축 방향으로의 진동은 데이타를 읽고 쓰는 속도에만 약간 영향을 주게 된다[10].

3.2 병렬성 기반 스케줄링 알고리즘

MEMS 기반 저장장치에서는 수천 개의 고정된 헤드가 자신이 맡은 영역을 서비스하므로 각 영역에서 동일한 상대적 위치에 있는 요청들은 서로 다른 헤드에 의해 동시에 처리될 수 있다. 본 논문이 제안하는 스케줄링 알고리즘은 이러한 특징을 이용하여 모든 영역을 통틀어서 가장 많은 요청이 대기 중인 (x, y) 지점을 우선적으로 서비스하자는 것이 기본적인 원리이다. 이렇게 하면 많은 요청들을 큐에서 한꺼번에 제거할 수 있어 요청들의 평균 응답 시간이 빨라지게 된다. 이와 같은 병렬성 기반의 스케줄링 알고리즘은 SSTF나 SPTF와 같은 탐색 시간 기반의 알고리즘과 결합시킬 경우 더 좋은 성능을 추구할 수 있다. 본 논문에서는 SPTF를 기본적인 탐색 시간 기반의 알고리즘으로 사용하고 이를 병렬성과 함께 고려하는 P-SPTF(Parallelism-aware SPTF) 알고리즘을 제안한다. P-SPTF에서는 요청이 들어온 (x, y) 지점들에 대해 우선 순위를 아래와 같이 계산하고 우선 순위가 가장 높은 지점을 제일 먼저 서비스한다.

$$Priority(x, y) = N(x, y) \ / \ time_{position}(x, y) \quad (2)$$

이 때, $N(x, y)$는 모든 영역을 통틀어 (x, y) 위치에 들어온 요청의 수를 나타내며 $time_{position}(x, y)$는 (x, y) 위치로 이동하는데 걸리는 탐색 시간을 나타낸다. P-SPTF가 서비스 순서를 어떻게 결정하는지에 대해 그림 2의 간단한 예제를 통해 설명하도록 하겠다. 그림 2에서 점선은 현재 각 위치에 들어온 요청들을 나타낸다. 설명을 간단히 하기 위해 이 예제에서는 x축과 y축 방향으로 단위 거리를 이동하는데 걸리는 시간은 1로 동일하며, 정착 시간은 없는 것으로 가정하기로 한다. (실제 MEMS 기반 저장 장치에서는 영역의 중앙으로부터의 위치에 따라 동일 거리에 대한 탐색 시간이 달라지며, 4장의 실험에서는 이와 같은 특징들을 모두 반영하고 있다.) 그림 2에서 (11,1)과 (5, 4)의 $time_{position}(x, y)$ 값을 계산해 보면 각각 2와 3이 된다. 따라서, 탐색 시간이 최소인 지점을 먼저 서비스하는 SPTF는 (11,1)을 서비스하고 다음으로 (5, 4)를 서비스한다. 하지만,

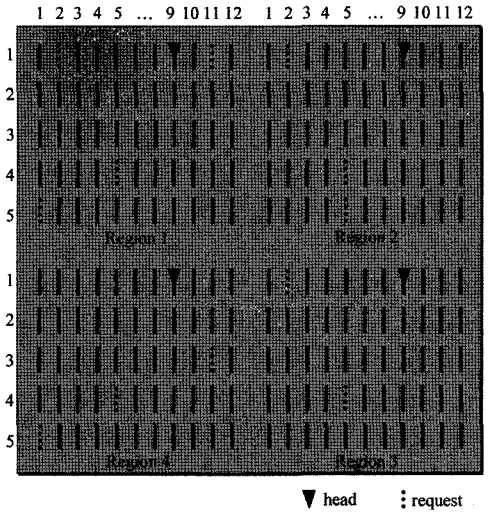


그림 2 P-SPTF 스케줄링의 예

(5, 4)에는 4개의 요청이 서로 다른 영역에 들어와 있고, (11, 1)에는 영역 1에만 요청이 들어와 있으므로 식 (2)에 의해 $Priority(5, 4)$는 4/3, $Priority(11, 1)$은 1/2의 값을 가지게 된다. 이를 근거로 P-SPTF는 (5, 4)를 다음 서비스 지점으로 선택하게 된다. 그 결과 (5, 4)에 들어와 있는 4개의 요청은 병렬적으로 처리되어 한꺼번에 큐에서 빠져나가게 된다. 본 예제의 계산에서는 자명한 두 지점에 대한 우선 순위만을 계산했지만 실제로는 모든 요청이 있는 지점에 대한 우선 순위의 계산이 필요하다. 한편, MEMS 기반 저장장치에서 동시 요청에 대한 처리에는 약간의 제약 사항이 있다. 전력 소비와 발열 등의 문제로 인해 일반적으로 모든 헤드를 동시에 활성화시키지는 않는다. 예를 들어 카네기 멜론 대학의 MEMS 기반 저장장치 모델에서는 6,400개의 전체 헤드 중 1,280개의 헤드만이 동시에 활성화 될 수 있다[6]. P-SPTF는 이와 같은 제한점을 다음과 같은 방법으로 쉽게 해결할 수 있다. 만약 $N(x, y)$ 값이 동시에 활성화 될 수 있는 헤드의 수인 Max를 초과하게 되면 Max개의 요청만을 우선적으로 처리하고 남은 요청들은 큐에 대기 중인 다른 요청들과 동일하게 취급한다. 그러나, 남은 요청들의 탐색 시간인 $time_{position}(x, y)$이 거의 0에 가까운 값을 가지므로 이 위치의 우선순위인 $Priority(x, y)$가 여전히 매우 높은 값을 가지게 되어 대기 중인 요청들 중에 가장 먼저 처리될 가능성이 높게 된다. 즉, 동일 지점의 요청들은 동시 처리 요청 수의 제한에 걸리더라도 연이어 처리될 가능성이 매우 높아 이와 같은 제약 사항이 알고리즘의 운영에 큰 문제가 되지는 않는다.

3.3 형평성을 고려한 병렬성 기반 알고리즘

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 발생시키는 것은 적절하지 않다. P-SPTF 알고리즘에서는 현재 헤드의 위치에서 멀리 떨어진 곳이나 요청의 수가 적은 위치인 경우는 스케줄링에서 계속적으로 소외될 수 있다. 디스크 스케줄링 분야에서는 이와 같은 문제를 해결하기 위해 Jacobson과 Wilkes가 SPTF 알고리즘에 노화(aging) 요소를 적용시킨 ASPTF 알고리즘을 제안한 바 있다[13]. 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.

본 논문에서는 P-SPTF 알고리즘을 확장하여 ASPTF처럼 요청간 대기 시간의 형평성을 고려하는 PA-SPTF (Parallelism-aware with Aging extension SPTF) 알고리즘을 제안한다. PA-SPTF는 P-SPTF와 유사하게 각 지점에 대한 우선 순위 $Priority(x, y)$를 계산하고 우선 순위가 가장 높은 지점을 제일 먼저 서비스한다. $Priority(x, y)$는 식 (3)과 같이 계산된다.

$$Priority(x, y) = \sum W(x, y) \ / \ time_{position}(x, y) \quad (3)$$

$time_{position}(x, y)$는 (x, y) 위치로 이동하는데 걸리는 탐색 시간을 뜻하며 $\sum W(x, y)$는 (x, y) 위치에 있는 요청들의 대기 시간의 합을 뜻한다. (x, y) 지점에 대기 중인 요청이 많을수록 $\sum W(x, y)$ 값이 커지므로 이 알고리즘은 기본적으로 병렬성을 고려하고 있으며, 요청이 기다린 시간이 길어질수록 $W(x, y)$ 값이 커지므로 노화 요소를 반영하여 기아 현상을 해소할 수 있다.

3.4 병렬성 기반 알고리즘의 일반화

본 절에서는 P-SPTF와 PA-SPTF 알고리즘을 일반화시켜 두 알고리즘을 포함하는 일련의 알고리즘의 스펙트럼이 존재함을 보인다. 이와 같은 일반화는 앞에 설명된 식 (3)을 식 (4)와 같이 간단히 변형함으로 얻을 수 있다.

$$Priority(x, y) = \sum W^{\alpha}(x, y) \ / \ time_{position}(x, y) \quad (4)$$

대기 시간을 단순히 합하는 방식 대신 식 (4)에서는 대기 시간을 α 승한 값을 더하게 된다. 이 때, α는 0과 1사이의 값을 가진다. α가 0이면 $W^{\alpha}(x, y)$는 1이 되어 $\sum W^{\alpha}(x, y)$는 (x, y) 지점에 들어와 있는 요청의 수를 의미하게 되며 이 알고리즘은 P-SPTF 알고리즘과 같은 의미를 가진다. α 값이 증가함에 따라 스케줄링 알고리즘은 노화 요소에 높은 가중치를 부여하게 되며 α가 1이 되었을 때 이 알고리즘은 PA-SPTF 알고리즘이 된다. 이와 같은 일반화된 알고리즘은 스케줄링이 필요할 때마다 대기 중인 모든 요청에 대해 지수 함수를 계산해야 하는 추가적인 오버헤드가 필요하므로 본 논문에서는 알고리즘의 존재성만 보이고, 실험에서는 P-SPTF와 PA-SPTF 두 극단의 알고리즘만을 고려하기로 한다.

4. 실험 결과

MEMS 기반 저장장치는 아직 연구 개발 단계에 있으며 상이한 물리적 파라미터들을 가지는 모델이 선도 연구 기관들에 의해 각각 제시되고 있다. 본 논문의 실험에서는 기본적으로 카네기 멜론 대학에서 제안한 MEMS 기반 저장장치의 모델을 그대로 따랐다[2,6]. 이 모델에서는 하나의 MEMS 기반 저장장치에 6,400개의 영역이 있으며 각 영역은 $x \times y$ 평면상에서 2500×2440 비트를 가진다. 각 영역은 2500×27개의 섹터로 구성되며 그림 3에서 보는 바와 같이 섹터 하나는 8 바이트의 데이타가 인코딩된 80 비트로 구성된다. 그리고, 영역과 영역 사이에는 섹터 구분을 위한 10 비트의 부가 정보(servo information)가 저장된다. 표 2는 본 논문의 실험에서 사용한 MEMS 기반 저장장치의 구체적인 파라미터들을 보여주고 있다.

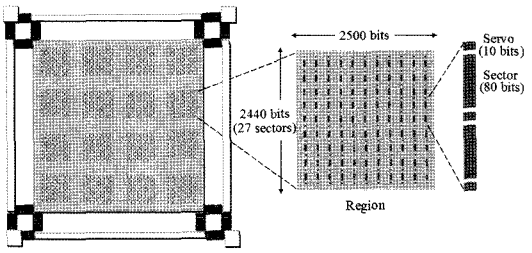


그림 3 실험에 사용된 MEMS 기반 저장장치의 구조

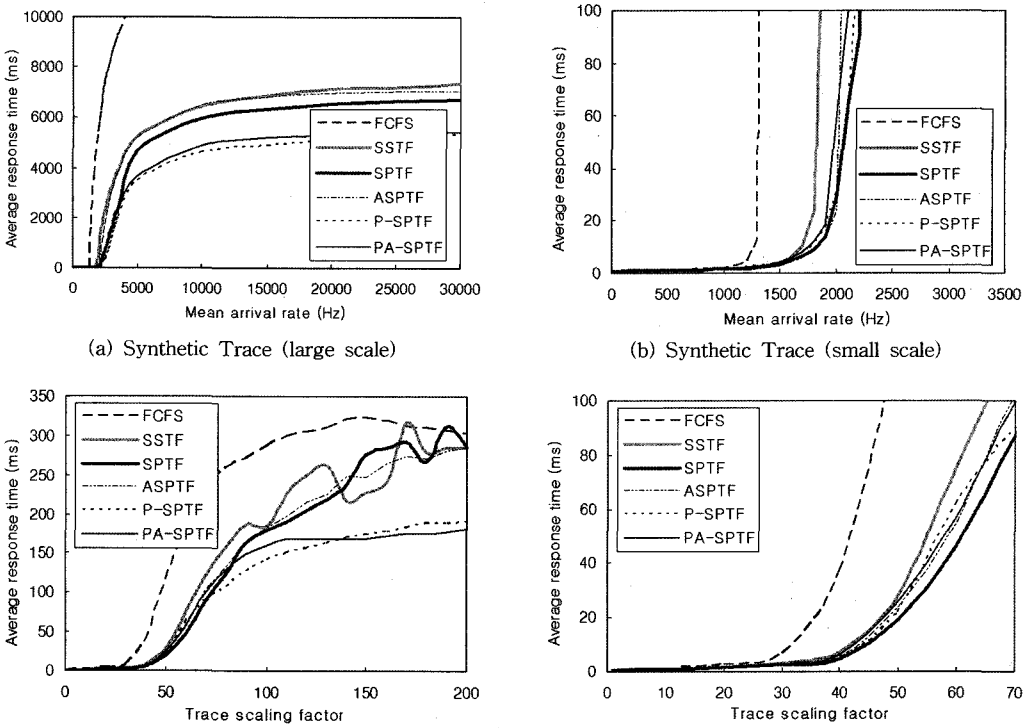
표 2 실험에 사용된 MEMS 기반 저장장치 파라미터

Number of regions	6400
Number of heads	6400
Maximum concurrent heads	1280
Device capacity	3.2 GB
Physical sector size	8 bytes
Servo overhead	10 bits per sector
Bits per region	2500 × 2440
Settling time	0.22 ms
Average turnaround time	0.07 ms
Spring factor	75%
Media bit cell size	40 × 40 nm
Sled acceleration	803.6 m/s^2
Sled access speed	28mm/s
Per head data rate	0.7 Mbit/s
Command processing overhead	0.2 ms/request
On-board cache memory	0MB

512 바이트의 논리적 블록은 64개의 서로 다른 영역에서 상대적으로 같은 (x, y) 위치의 8 바이트 섹터로 매핑하여 병렬적인 접근이 가능하도록 하였다. 또한, 인접한 논리적 블록들은 y축 방향을 따라 순차적으로 할당하여 부가적인 헤드 탐색 시간을 없애도록 하였다. 본 논문에서 제안한 스케줄링 알고리즘의 효율성을 평가하기 위해 트레이스 기반 모의 실험을 수행하였으며, 실험에 사용한 트레이스는 합성 트레이스(synthetic trace)와 실제 시스템의 추출 트레이스를 모두 사용하였다. 합성 트레이스 경우, 다양한 평균 도착률을 지수 분포에 따라 생성하였으며 읽기와 쓰기 연산의 비율은 다른 연구에서 널리 사용한 67%와 33%으로 하였다[6]. 요청된 데이타의 크기는 평균이 4Kbyte인 지수 분포에 따랐으며 요청들의 배치는 전체 MEMS 기반 저장장치 전역에 걸쳐 균일하게 분포시켰다.

추출 트레이스의 경우에는 MEMS 기반 저장장치를 이용한 직접적인 트레이스를 구할 수 없기 때문에 널리 사용되는 Cello99 디스크 트레이스를 사용하였다. Cello99 트레이스는 HP 연구소의 HP-UX 운영체제를 사용하는 시분할 시스템에서의 디스크 입출력 데이타를 수집한 트레이스이다. Cello99 트레이스는 [14]에서 얻을 수 있다. 부하의 범위를 다양하게 실험하기 위해 요청들의 도착 간격을 다양하게 조절하여 실험을 하였다. 예를 들어 스케일링 요소(scaling factor)가 2인 경우 부하는 원래 트레이스에 비해 2배로 빈번한 요청이 이루어지는 것을 뜻한다. 트레이스에 나타난 데이타의 크기가 MEMS 기반 저장장치 하나가 수용할 수 있는 용량인 3.2Gbyte를 넘어서는 경우에는 여러 개의 매체를 사용하는 것으로 가정하였다.

실험에서는 본 논문에서 제안한 P-SPTF와 PA-SPTF 알고리즘을 FCFS, SSTF, SPTF, ASPTF와 비교하였으며 성능 척도로는 평균 응답 시간과 응답 시간의 분포를 사용하였다. 그림 4(a), (b)는 합성 트레이스에서 평균 도착율의 변화에 따른 여섯 가지 알고리즘에 대한 요청들의 평균 응답 시간을 보여주며 그림 4(c), (d)는 Cello99 트레이스에서 스케일링 요소가 증가함에 따른 요청의 평균 응답 시간을 보여주고 있다. 그림 4(a)와 4(c)에서 볼 수 있듯이 P-SPTF와 PA-SPTF는 요청의 평균 도착률이 증가함에 따라 FCFS, SSTF, SPTF, ASPTF보다 큰 성능 향상을 보임을 알 수 있었다. Cello99 트레이스에서는 스케일링 요소가 150 이상인 경우 본 논문이 제안한 두 알고리즘이 SPTF보다 39.2%에 이르는 성능 향상을 나타내었다. 이는 영역 내의 상대적 위치가 동일한 요청들에 우선 순위를 높여



(a) Synthetic Trace (large scale)

(b) Synthetic Trace (small scale)

(c) Cello99 Trace (large scale)

(d) Cello99 Trace (small scale)

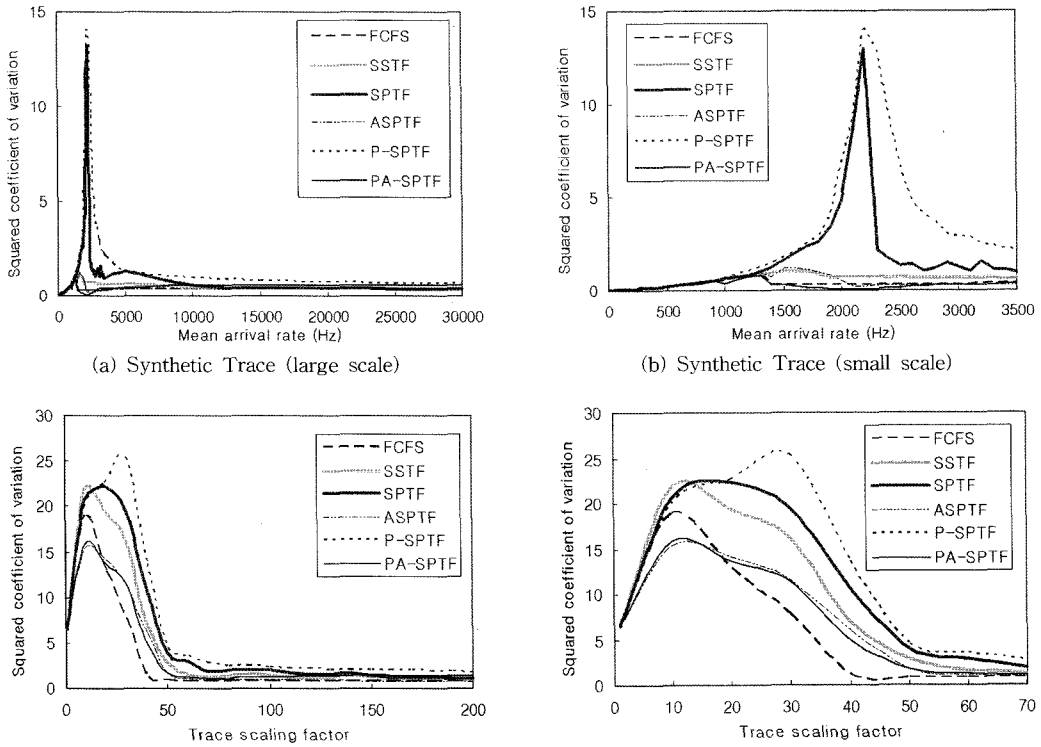
그림 4 스케줄링 알고리즘들의 평균 응답 시간

주는 동시에 헤드의 이동 시간을 줄였기 때문으로 평가할 수 있다. 그러나, 그림 4(b)와 4(d)에서와 같이 부하가 적을 때에는 본 논문이 제안한 알고리즘이 SPTF와 유사한 성능을 나타낸 것을 확인할 수 있다. 이는 부하가 적은 경우 동시 처리가 가능한 요청의 수가 상대적으로 적어지기 때문으로 분석할 수 있다. 기존 알고리즘들 간의 성능 차이는 다른 연구의 결과들과 마찬 가지로 SPTF가 SSTF보다 우수한 성능을 나타내었으며, FCFS는 SSTF보다 크게 저하된 성능을 나타내었다[5,6,10]. ASPTF는 트레이스와 부하의 상태에 따라 SPTF와 SSTF의 성능과 유사한 결과를 나타내었다. 이러한 실험 결과를 바탕으로 본 논문이 제안한 알고리즘은 기존 알고리즘에 비해 확장성이 매우 높아 멀티미디어 서버나 웹 서버처럼 대용량의 스트림 요청을 처리해야 하는 환경에서 사용할 경우 효율적일 것이라는 것을 알 수 있었다.

그림 5는 평균 도착률과 트레이스 스케일링 요소의 증가에 따른 요청들의 응답 시간 편차에 대한 표준 계수(σ^2/μ^2)를 보여주고 있다. 여기에서 σ는 응답 시간의 표준편차이며 μ는 평균 응답 시간이다. 이 성능 척도는 각각의 요청에 대한 응답 시간이 평균과 큰 차이가 없는 경우 작은 값을 가지게 된다. 따라서, 이러한 성능 척도는 기아 현상의 파악과 요청간 스케줄링의 형평성 평가에 널리 사용된다[9]. 대기 시간의 합을 통하여 노화 요소를 추가한 PA-SPTF는 형평성 측면에서 다른 알고리즘에 비하여 모든 경우 가장 월등한 성능을 보여주고 있다. Cello99 트레이스와 합성 트레이스에서 각각 50 이하의 스케일링 요소와 3,500 이하의 낮은 도착률을 가지는 경우, FCFS와 PA-SPTF는 다른 알고리즘에 비해 일관성 있게 좋은 성능을 나타내었으며, ASPTF는 생성된 트레이스의 경우 평균 도착률이 2,000 이하일 때 FCFS와 PA-SPTF에 비해 약간의 성능 저하가 있지만 대체로 우수한 성능을 나타내었다.

5. 결 론

본 논문에서는 MEMS 기반 저장장치의 물리적 특성을 고려한 P-SPTF와 PA-SPTF 알고리즘을 제안하였다. P-SPTF 알고리즘은 2차원 평면 상의 헤드의 이동 시간과 동시 처리가 가능한 요청의 수를 고려하였으며, PA-SPTF 알고리즘은 여기에 기아 현상을 극복할 수 있는 노화 요소를 추가하여 요청간 응답 시간의 형평성



(a) Synthetic Trace (large scale)

(b) Synthetic Trace (small scale)

(c) Cello99 Trace (large scale)

(d) Cello99 Trace (small scale)

그림 5 스케줄링 알고리즘들의 응답 시간 편차

을 추구하였다. 트레이스 기반 모의 실험을 통해 본 논문이 제안한 알고리즘은 평균 응답 시간과 요청별 응답 시간의 편차 측면에서 기존의 알고리즘들보다 우수한 성능을 나타냄을 보였다. 특히 요청이 빈번한 환경에서는 기존에 가장 우수한 성능을 나타낸 SPTF보다 평균 응답 시간이 39.2%까지 향상됨을 보였다.

본 논문의 향후 연구에서는 MEMS 기반 저장장치에서의 데이타 배치와 사전 인출(prefetching) 기법을 스케줄링 기법과 함께 고려할 계획이다. 본 논문에서는 효과적인 데이타 배치 기법을 고려하지 않았지만 MEMS 기반 저장장치의 성능은 데이타 배치와 깊은 연관이 있는 것으로 알려져 있고[6] 스케줄링의 효율성도 데이타 배치 방법에 따라 다르게 나타나게 되므로 이들을 함께 고려한 연구의 수행이 반드시 필요하다. 이와 더불어 수천 개의 헤드가 동시에 데이타를 읽고 쓸 수 있으므로 실제 요청이 들어와 있지 않은 지점의 데이타를 헤드가 미리 읽어 오는 사전 인출(prefetching) 기법을 고려할 필요가 있다. 사전 인출 메커니즘이 추가되었을 때 스케줄링 알고리즘은 사전 인출 요청과 실제 들어온 요청에 대해 우선 순위를 다르게 취급해야 하며 이는 스케줄링 문제를 더욱 복잡하게 만들게 된다.

참 고 문 헌

[1] 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.

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

[3] P. Vettiger, M. Despont, U. Drechsler, U. Dürig, W. Häberle, M. Lutwyche, H. Rothuizen, R. Stutz, R. Widmer, and G. Binnig, "The Millipede - More than one thousand tips for future AFM data storage," *IBM Journal Research and Development*, Vol.44, No.3, pp.323-340, 2000.

[4] Samsung Flash Memory, http://www.samsung.com/products/semiconductor/NANDFlash/SLC_LargeBlock/8Gbit/K9K8G08U1A/K9K8G08U1A.htm.

[5] B. Hong, S. Brandt, D. Long, E. Miller, K. Glocer, and Z. Peterson, "Zone-based Shortest Positioning Time First Scheduling for MEMS-based Storage Devices," *11th IEEE/ACM Symp. Modeling, Analysis, and Simulation of Computer and Tel. Systems*, 2003.

[6] J. Griffin, S. Schlosser, G. Ganger, and D. Nagle, "Operating system management of MEMS-based storage devices," *4th USENIX Symp. Operating Systems Design and Implementation*, pp. 227-242, 2000.

[7] 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.

[8] P. Denning, "Effects of scheduling on file memory operations," *AFIPS Spring Computer Conf.*, 1967.

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

[10] S. Schlosser and G. Ganger, "MEMS-based storage devices and standard disk interfaces: A square peg in a round hole?" *3rd USENIX Conf. File and Storage Technologies*, 2004.

[11] Center for Highly Integrated Information Processing and Storage Systems, Carnegie Mellon University, http://www.ece.cmu.edu/research/chips/

[12] Hewlett-Packard Laboratories Atomic Resolution Storage, http://www.hpl.hp.com/research/storage.html.

[13] D. Jacobson and J. Wilkes, "Disk scheduling algorithms based on rotational position," *Technical Report HPL-CSP-91-7rev1*, HP Labs, 1992.

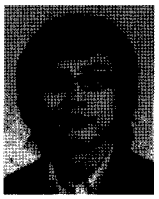
[14] Public Software, Storage Systems Department at HP Labs, http://tesla.hpl.hp.com/public_software/

이 소 윤

2004년 이화여대 컴퓨터학과 학사. 2006년 이화여대 컴퓨터학과 석사. 2006년~현재 이화여대 컴퓨터학 박사과정. 관심분야는 스토리지 시스템 관리, 시스템 최적화, 지능형 스토리지 시스템, 저전력시스템

반 효 경

1997년 서울대학교 계산통계학과 학사 1999년 서울대학교 전산과학과 석사 2002년 서울대학교 컴퓨터공학부 박사 2002년~현재 이화여자대학교 컴퓨터학과 교수. 관심분야는 운영체제, 스토리지 관리, 저전력 시스템, 웹 캐싱 등



노 삼 혁

1986년 서울대학교 컴퓨터공학과 학사 1993년 메릴랜드대학교 컴퓨터과학과 박사. 1993년~1994년 조지워싱턴대학교 객원 조교수. 1994년~현재 홍익대학교 정보컴퓨터공학부 교수. 관심분야는 운영체제, 플래시메모리소프트웨어, 내장형시스템, 차세대저장장치