□ 기술해설 □

멀티미디어 서버에서의 실시간 처리 디스크 스케쥴링 기법

한국과학기술원 이종민* · 이귀영* · 이홍규**

● 목 차 ●

1. 서 론

오디오, 비디오와 같은 아날로그 데이타를 디지털 데이타로 만드는 디지털화 기법(digitalization)과 초당 수 기가 비트의 전송률을 가지는 통신망이 발달함에 따라서 기존의 텍스트 외에 이들 멀티미디어 데이타를 주고받는 그림 1과 같은 상호 분산 멀티미디어 시스템(interactive distributed multimedia system)이 가능하게 되었다[1]. 이러한 멀티미디어 시스템은 학계, 상업 분야, 정부, 소비자 등 여러 분야에서 주목을 받고 있다.

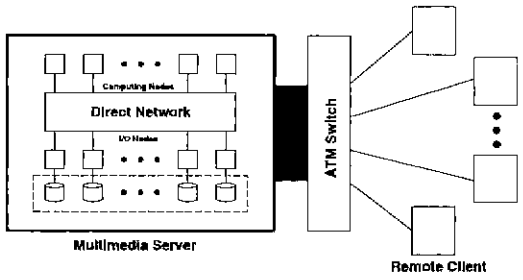


그림 1 상호 분산 멀티미디어 시스템의 예

멀티미디어 시스템은 주로 상호 서비스(interactive service)에 사용되는 데, 서비스 제공자의 관점에서 크게 단방향의 통신을 주로 하는 VOD(video-on-demand), AOD(audio-on-demand)와 같은 MOD(multimedia-on-demand)와, 쌍방향 통신을 주로 하는 원격화상회의(teleconferencing)로 나눌 수 있다[2]. MOD는 실시간 입출력을 보장하는 것이 주된 문제이며, 원격화상회의는 그 특성상 화상의 전송시간을 줄이는 것이 중요하다. 특히 MOD를 지원하기 위해서는 정보를 제공할 수 있는 멀티미디어 서버가 필요하며, 사용자의 비디오, 오디오 등에 대한 데이타 요청을 정해진 시간 안에 연속적으로 처리해 줄 수 있어야 한다. 따라서 이를 위한 실시간 처리가 필요하다.

멀티미디어 서버에서 실시간 처리가 필요한 부분은 크게 저장장치, 서비스 승인(admission) 제어, 통신망으로 분류할 수 있으며, 본 논문에서는 저장 장치에서의 실시간 처리 디스크 스케쥴링 알고리즘에 대하여 고찰한다. 저장 장치의 검색과 관련하여 많은 시간을 차지하는 부분은 디스크의 탐색 시간(seek time)으로서, 시간적인 제약 조건을 만족시키면서 이를 줄이

*학생회원
**종신회원

기 위한 연구가 많이 이루어져 왔다.

본 논문의 구성은 다음과 같다. 2절에서는 멀티미디어 서버의 일반적인 구조와 디스크 접근에 대해서 살펴보고, 3절에서는 실시간 처리를 고려하지 않은 기존의 디스크 스케쥴링 기법에 대해서 고찰한다. 4절에서는 멀티미디어 서버를 위하여 실시간 처리를 고려한 디스크 스케쥴링 기법에 대하여 고찰한다. 5절에서는 비실시간 디스크 스케쥴링 기법과 실시간 디스크 스케쥴링 기법에 대한 성능 평가를 요약하고, 사용된 방법에 따라서 분류한다. 마지막으로, 6절에서는 멀티미디어 서버에서의 실시간 디스크 스케쥴링 기법에 대해서 요약한다.

2. 관련 연구

2.1 멀티미디어 서버

멀티미디어 서버의 일반적인 구조는 그림 2와 같다[3]. 객체 관리 노드(object manager node)는 사용자로부터 오는 모든 요청을 받아서 적당한 인터페이스 노드(interface node)에게 서비스를 하도록 요청한다. 인터페이스 노드는 객체 관리 노드로부터 오는 요청들을 처리해 주며 자원을 효율적으로 관리한다. 서버 노드(server node)는 멀티미디어 데이타를 가지고 있으며 인터페이스 노드로부터 데이타를 요청 받으면 검색하여 전송해 준다. 수많은 사용자들로부터 오는 요청을 처리해 주려면, 멀티미디어 서버는 기존의 PC나 워크스테이션보다 훨씬 뛰어난 능력을 가지고 있어야 한다. 최소한 수 테라바이트의 이차 저장장치와 수 기가 바이트의 주기억장치를 필요로 하며, 고속의 원거리 통신망(wide-area network)에

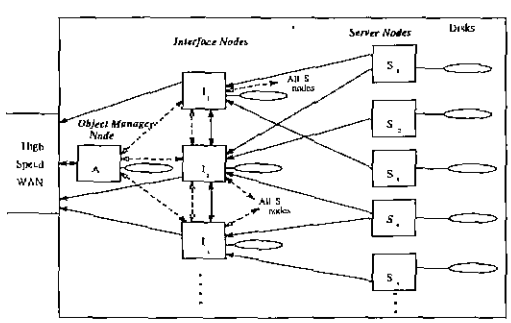


그림 2 멀티미디어 서버의 일반적인 구조

연결되어 있어야 한다. 이러한 멀티미디어 서버는 병렬컴퓨터나, PC 또는 워크스테이션의 클러스터(cluster)로 구현 가능하다.

멀티미디어 서버에서 저장장치와 관련된 요구조건은 다음과 같다.

대용량의 저장장치 : MPEG1으로 인코딩된 데이타는 초당 1.5Mbits의 playback rate를 필요로 한다. 따라서 2시간 짜리 영화의 경우 약 1기가 바이트 정도의 저장장치를 필요로 한다. 또한 MPEG2로 인코딩된 경우에는 약 3.5 기가 바이트 정도의 저장장치를 필요로 한다. 따라서 여러 개의 영화를 가지고 여러 사용자의 요청을 만족시켜 주기 위해서는 멀티미디어 서버에 수 테라바이트 이상의 저장장치가 필요하다.

최소한의 반응시간(response time) : On-demand 서비스가 성공하기 위해서는 사용자 입장에서 보여지는 반응시간이 중요한 요소가 된다. 이를 위해서는 최악의 경우에도 반응시간 특성이 좋아야 하며, 실시간 처리가 필요한 요인이라고 할 수 있다.

실시간 트래픽과 비실시간 트래픽의 처리 능력 : 비디오, 오디오 데이타와 같은 실시간 트래픽뿐만 아니라, 새로운 프로그램을 리모트 서버로부터 다운로딩하는 과정, 요금 청구, 계정 확인 등과 같은 비실시간 트래픽도 일정 수준 이상의 서비스를 해 주어야 한다.

신뢰성 및 가용성 : 멀티미디어 서비는 많은 데이타의 처리로 인하여 신뢰성을 보장하는 것이 중요한 문제이다. 또한 수많은 데이타들을 잘 보관하기 위하여 고장 허용(fault tolerance)을 위한 메커니즘이 있어야 한다. 그리고 사용자의 요청이 언제 들어올 지 모르므로 다운 시간(down time)이 아주 적어서 가용 시간이 많아야 한다.

2.2 디스크 접근(Disk Access)

디스크에서 데이타를 읽거나 쓸 때 소요되는 시간은 그림 3과 같이 크게 탐색 시간(seek time), 회전 지연 시간(rotational latency), 전송 시간(transmission time)으로 나누어 질 수 있다[4].

디스크 상의 원하는 블럭에 접근하기 위하여

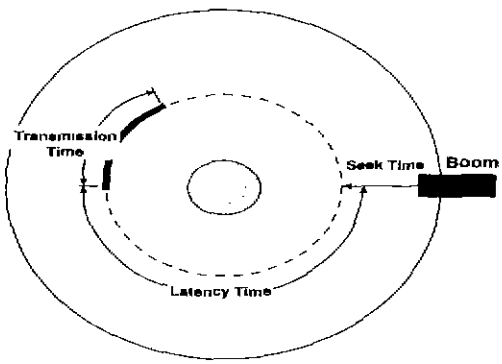


그림 3 디스크 접근의 요소

시스템은 먼저 특정 트랙 또는 실린더에 헤드를 위치시키게 되는데 이러한 동작을 탐색이라 하고, 그러한 동작을 완성하기까지 걸리는 시간을 탐색 시간이라 한다. 일단 헤드가 지정된 트랙 상에 위치하였더라도 원하는 블럭이 입출력 헤드 아래를 회전할 때까지 기다려야만 한다. 바로 이때 기다리는 시간이 회전 지연 시간이다. 끝으로 디스크와 주기억장치 사이에 정보의 실제적인 이동을 위하여 소요되는 시간을 전송 시간이라 한다. 이와 같이 한번의 디스크 요청을 처리하는 데 걸리는 총 시간은 탐색 시간, 회전 지연 시간, 그리고 전송 시간의 합이다.

그러나, 멀티미디어 서버에서 한번의 디스크 요청을 처리하는 데 걸리는 총 시간 δ_{to}는 탐색 시간, 회전 지연 시간, 전송 시간외에도 통신망을 지나는 데 걸리는 시간 등도 포함된다. 이는 (1)과 같이 표현될 수 있다[3]. δ_{rq}는 인터페이스 노드에서 서버 노드로 요청이 오는 시간이며, $\delta_{avg_{sk}}$, $\delta_{avg_{rl}}$, $\delta_{tr_{x}}$은 각각 서버 노드의 디스크에서 소요되는 탐색 시간, 회전 지연 시간, 전송 시간이다. $\delta_{nw_{x}}$는 서버 노드에서 인터페이스 노드로 요청된 데이타를 전송하는 데 걸리는 시간이다.

$$\delta_{to}=\delta_{rq}+\delta_{avg_{sk}}+\delta_{avg_{rl}}+\delta_{tr_{x}}+\delta_{nw_{x}} \quad (1)$$

3. 기존의 디스크 스케줄링 알고리즘

실시간 처리를 고려하지 않은 기존의 디스크 스케줄링 알고리즘 중 대표적인 것은 FCFS, SSTF, SCAN, C-SCAN이다. 다음은 이들 스케줄링 알고리즘에 대하여 간단히 요약한 것이다.

FCFS(First-Come-First-Served)는 가장 먼저 온 데이타 요청이 가장 먼저 처리되는 방법을 말한다. 이는 프로그램하기 쉽고 본질적으로 공평성(fairness)이 유지된다는 점이 특징이다. 그러나, 부하(load)가 가벼울 때는 괜찮으나, 부하가 증가함에 따라서 포화되는 현상이 발생하여 반응 시간이 증가한다는 단점이 있다. 또한 데이타 요청이 디스크 표면상에 균일하게 분포되어 있으면 임의의 탐색 패턴을 보이게 되어 탐색 패턴의 최적화에 아무런 도움이 되지 않는다.

SSTF(Shortest-Seek-Time-First)는 주어진 디스크 요청을 처리하기 위해서는 헤드가 먼 곳까지 이동하기 전에 현재 헤드 위치에 가까운 모든 데이타 요청을 먼저 처리하는 것이 합리적일 것이라는 생각에 기초하고 있다. SSTF는 전반적인 탐색 시간을 감소시킬 수 있다. 그러나, SSTF는 근본적으로 SJF (Shortest-Job-First) 알고리즘의 형태이므로, 실린더의 제일 안쪽과 바깥쪽에서 디스크 요청의 기아(starvation) 현상이 발생할 수 있다. SSTF는 FCFS보다는 처리율(throughput) 면에서 보다 좋다고 할 수 있으며, 적정 부하에서 평균 반응 시간이 높으며, 반응 시간의 분산(variance)이 크다는 단점이 있다. 따라서 처리율이 관건인 일괄 처리(batch processing) 시스템에서 보다 효율적이라고 할 수 있으나, 반응 시간의 분산이 크기 때문에 멀티미디어 서버와 같이 상호 서비스를 제공하는 시스템에서는 좋지 않다.

SCAN은 요청 큐(request queue)의 동적 특성을 반영한 것으로서, 헤드가 디스크의 한쪽 끝에서 반대 방향으로 이동하면서 요청된 트랙에 대한 처리를 해 주다가 한쪽 끝에 이르면 반대 방향으로 헤드를 이동하면서 요청된 트랙에 대한 처리를 해 주는 방법이다. 이렇게 함으로써 SSTF에서와 같은 기아 현상을 제거하였다. 그러나 각 데이타 요청에 대한 분포가 균일하다고 가정할 때, 헤드가 한쪽 끝에 이르러 방향을 바꾸어야 할 시점에서 요청 밀도가 높은 쪽은 최초의 시작 부분이며, 나중에 처리

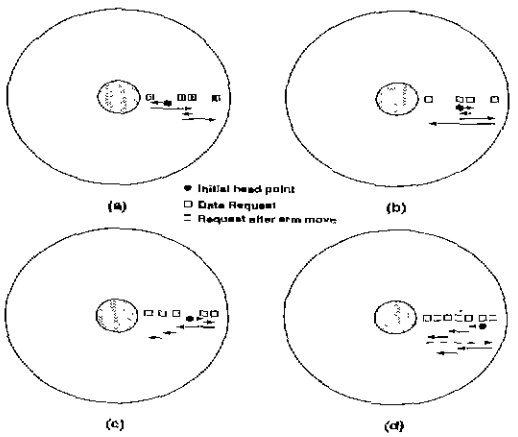


그림 4 기존의 디스크 스케줄링 알고리즘 : (a) FCFS (b) SSTF (c) SCAN (d) C-SCAN

된 헤드 바로 뒷부분은 비교적 밀도가 낮다. 따라서 밀도가 높은 쪽의 요청은 상당히 오랜 시간을 대기하게 되는 단점이 있다.

C-SCAN(Circular SCAN)은 SCAN에서의 불공평한 대기 시간을 좀더 균등하게 하려고 변형을 가한 것이다. SCAN에서와 같이 한쪽 방향으로 헤드를 이동해 가면서 요청을 처리하지만 한쪽 끝에 이르면 반대 방향으로 헤드를 이동하면서 요청을 처리해 주는 것이 아니라 다시 처음부터 처리를 진행하게 된다. 마치 처음과 마지막 트랙을 서로 인접시킨 것과 같은 원형처럼 디스크 요청을 처리해 주는 것이다.

이상의 방법들을 정리하면 그림 4와 같다.

4. 실시간 처리 디스크 스케줄링 알고리즘

멀티미디어 서버에서의 실시간 조건을 만족시키기 위한 여러 가지 스케줄링 알고리즘에 대한 연구의 필요성이 90년대 들어서 더욱 증가하고 있으나, 상대적으로 디스크 스케줄링에 대한 연구는 그다지 많지 않으며 기존의 기법들을 조합하여 상대적인 장점을 취하는 방법을 많이 채택하고 있다.

다음은 실시간 조건을 고려한 디스크 스케줄링 기법에 대하여 요약한 것이다. 실시간 조건을 고려하는 디스크 스케줄링에서는 디스크 요청이 필요한 트랙 번호뿐만 아니라, 시간적인 제약 조건을 나타내는 데드라인(deadline)을 가진다.

EDF(Earliest-Deadline-First) 알고리즘은 데드라인에 가장 가까운 디스크 요청부터 먼저 처리해 주는 것을 말한다. EDF는 요청의 서비스 시간이 미리 알려져 있다면 최적이라고 알려져 있다[5]. 원래 EDF는 선점 가능한 태스크(preemptable task)를 가정하고 있는 것이어서, 현재의 디스크와 같이 선점 불가능한 동작을 하는 장치에는 적합하지 않다고 알려져 있었다. 그러나 최근의 연구에서는 태스크가 선점 불가능할 때도 최적이라는 것으로 밝혀졌다[6]. 그러나, 탐색 패턴을 고려하지 않기 때문에 탐색 시간이 많아지게 되어서 EDF를 그대로 적용하게 되면 탐색 시간이 과다하게 되고 디스크의 사용률이 낮아지게 된다는 단점이 있다.

P-SCAN(Priority SCAN) 알고리즘은 I/O 큐에 들어 있는 디스크 요청을 여러 개의 우선순위(priority)를 가지는 클래스(class)로 분류를 한 후, 우선 순위가 가장 높은 클래스의 디스크 요청을 모두 서비스한 후, 남아 있는 클래스 중에서 가장 높은 우선 순위를 가지는 클래스의 디스크 요청을 서비스해 주는 것이다[7]. 각 클래스의 디스크 요청들은 SCAN 알고리즘에 따라 서비스된다. [7]에서는 우선 순위를 가지는 클래스의 수는 적게 하는 편이 I/O 성능 면에서 좋다고 연구되었으나, 각 요청들에 대해서 어떤 방식으로 우선 순위를 부여하는 가에 대해서는 언급하지 않고 있다.

FD-SCAN(Feasible Deadline SCAN) 알고리즘[8]은 SCAN 알고리즘의 변형으로서, 가장 빠른 실행 가능한(feasible) 데드라인을 가지는 디스크 요청이 SCAN 방향을 결정하는데 사용된다. 데드라인이 실행 가능하다는 것은 그 요청이 데드라인 전에 서비스될 수 있다는 것을 말한다. 각 요청의 실행 가능성(feasibility) 결정은 현재 디스크 암(arm)의 위치와 요청된 트랙의 위치를 알 수 있으면 그 요청에 대한 서비스 시간을 결정할 수 있으므로 쉽게 결정할 수 있다. 즉, 현재의 시간에 서비스 시간을 더한 것보다 디스크 요청의 데드라인이 크면 실행 가능한 디스크 요청이라고 할 수 있

다. 각 스케쥴링 시점에서 모든 디스크 요청을 검사하여 그 중에서 가장 빠른, 실행 가능한 데드라인을 가지는 디스크 요청을 결정하여 그 디스크 요청을 서비스해 주는 방향으로 SCAN 방향을 결정한 후, 디스크 암을 그 방향으로 움직이면서 그 방향에 있는 모든 디스크 요청에 대하여 서비스를 해준다.

SSEDO(Shortest Seek and Earliest Deadline by Ordering) 알고리즘[9]은 기본적으로 데드라인에 가까운 디스크 요청에 우선순위를 주어서 처리하는 방식이다. 윈도우 개념을 사용하여 각 디스크 요청 r_i중 데드라인 L_i에 따라서 정렬(sorting)된 디스크 요청 큐에서 처음 m개의 요청에 대해서만 적용된다. 즉, 윈도우 크기가 m인 알고리즘이라고 할 수 있다. 스케쥴링은 각 디스크 요청 r_i에 가중치(weight) w_i를 부가하여 $w_i\,d_i$값 중 가장 작은 값을 가지는 디스크 요청을 서비스해 주는 것이다. 여기서, d_i는 현재의 디스크 암 위치와 디스크 요청 r_i의 트랙 위치의 차이이다. $w_i\,d_i$는 디스크 요청 r_i에 대한 우선 순위 값(priority value)이라고 정의한다. 스케쥴링시 같은 우선 순위 값을 가지는 디스크 요청이 여러 개 있을 경우, 가장 빠른 데드라인을 가지는 디스크 요청이 선택되어 서비스된다. 이때, w_i를 (2)와 같이 두면, $\beta=i$일때는 SSTF 알고리즘과 비슷한 결과를 보이며, 윈도우의 크기가 커질수록 더욱 SSTF 알고리즘과 비슷하게 된다. 또한 윈도우의 크기가 1이면, EDF 알고리즘과 같아지게 된다. 따라서, 윈도우의 크기와 β의 값을 적절하게 선택하는 것이 중요하게 된다.

$$w_i=\beta_i(\beta\geq i),\ i=1,\ 2,\ \cdots,\ m \qquad (2)$$

SSEDV(Shortest Seek and Earliest Deadline by Value) 알고리즘[9]은 SSEDO 알고리즘이 각 디스크 요청의 데드라인의 순서 정보만을 이용하여 스케쥴링 하는 것을 보완한 것이다. SSEDO 알고리즘에서는 다음과 같은 예가 발생할 수 있다. γ_1, γ_2 두개의 디스크 요청이 있을 때 γ_1의 데드라인은 현시점에서 매우 가깝지만, γ_2의 데드라인은 매우 멀리 있다고 가정하자. 이때 γ_2의 위치가 현재 디스크 암의 위치와 매우 가깝다면 SSEDO 알고리즘은 γ_2를 선택하여 서비스해 줄 것이다. 그러면 γ_1은 데드라인을 지나서 서비스해 주지 못하게 될 것이다. 이는 각 디스크 요청의 데드라인 순서 정보만을 가지고 스케쥴링을 해 주었기 때문이다. SSEDV(Shortest Seek and Earliest Deadline by Value) 알고리즘[9]은 이러한 점에 착안하여 각 디스크 요청 γ_i에 우선 순위 값을 다음 식과 같이 주었다.

$$ad_i+(1-a)l_i,\ 0\leq a\leq 1 \qquad (3)$$

여기서, α는 스케쥴링 변수로 조정 가능하며, l_i(L_i−현재시간)는 디스크 요청 r_i의 남아있는 생존시간(life time)이다. $\alpha=0$이면 EDF 알고리즘과 같아지게 되고, $\alpha=1$이면 SSTF 알고리즘과 비슷하게 됨을 알 수 있다.

SCAN-EDF 알고리즘[10]은 탐색 패턴의 최적화 기법과 EDF 알고리즘을 적절히 조합하여 사용한다. 가장 빠른 데드라인을 가지는 디스크 요청이 먼저 서비스되나, 여러 개의 디스크 요청이 같은 데드라인을 가지고 있다면 이들 디스크 요청들은 트랙 위치에 따라 SCAN 방향으로 서비스되어 탐색 패턴을 최적화한다. 이를 좀더 자세하게 기술하면 그림 5와 같다.

SCAN-EDF 알고리즘은 같은 데드라인을 가지는 디스크 요청들에 대해서만 탐색 패턴의 최적화 기법을 적용한다. 따라서 얼마나 많은 디스크 요청들이 같은 데드라인을 가지느냐 하는 것이 중요한 문제가 된다. 이를 위하여 [10]에서는 각 디스크 요청들이 주기 p의 배수로 릴리스 시간(release time)을 가지도록 한다. 릴리스 시간은 디스크 요청이 시작되는 시간을 말한다. SCAN-EDF 알고리즘은 EDF 알고리즘을 조금 수정하여 구현 가능하다. 디스크 요청 r_i의 데드라인을 D_i, 트랙 위치를 N_i라고 하자. 그러면 EDF에서는 가장 작은 D_i값

Step 1 : Let T=set of tasks with the earliest deadline
Step 2 : If T=1, service that request
else let T_i be the first task in T in scan direction, service t_i
go to Step 1

그림 5 SCAN-EDF 알고리즘

을 가지는 디스크 요청이 서비스를 받으나, SCAN-EDF에서는 가장 작은 $D_i+f(N_i)$값을 가지는 디스크 요청이 서비스를 받으면 된다. 여기서 $f(N_i)=N_i/N_{max}-1$(N_{max}=디스크의 최대 트랙 번호)이다.

GSS(Grouped Sweeping Scheduling) 알고리즘[11,12]은 디스크 스케줄링 기법중 고정 순서 방식과 SCAN 방식을 조합하여 만든 것이다. 서비스를 기다리는 디스크 요청들을 g개의 그룹으로 나눈 후, 각 그룹들은 고정 순서 방식으로 처리하며, 그룹내의 디스크 요청들은 SCAN 방식에 따라 서비스된다. n을 디스크 요청의 수라고 하였을 때, $g=n$이면 FCFS 방식과 같아지고, $g=1$이면 SCAN 방식과 같아지게 된다. 그러나, GSS 알고리즘은 실시간 처리의 효율성보다는 버퍼를 얼마나 적게 사용하는가에 초점을 맞추었다. 즉, 디스크 요청이 들어왔을 때 그 요청에 대한 서비스를 위한 버퍼 영역을 얼마나 적게 사용하는 가가 최적화의 목적이다. 따라서 다른 디스크 스케줄링 알고리즘과의 비교 평가는 어렵다.

5. 성능 평가

본 절에서는 여러 가지 디스크 스케줄링 알고리즘들을 비교, 평가한 Chen과 Reddy의 연구[9,10]를 간단히 요약하고, 실시간 처리 디스크 스케줄링 알고리즘들을 사용된 방법에 따라서 분류한다.

5.1 Chen의 연구

각 디스크 스케줄링 알고리즘의 비교 평가에 사용된 모의실험 모델은 그림 6과 같다. 각 사용자는 평균 1초의 지수 분포(exponential distribution)를 가지고 트랜잭션(transaction)을 만들어 낸다. 각 트랜잭션은 1~20단계의 임의의 오퍼레이션(operation)으로 구성된다. 각 오퍼레이션 단계는 한번의 디스크 접근을 필요로 한다. 디스크 접근시 읽기 확률 p_r은 0.8로 고정하였다. 각 트랜잭션의 데드라인 $Trans\ Deadline$은 (4)와 같이 정의된다. 이때, $CPU\ Time$과 $IO\ Time$은 각각 FCFS 기법에서 각 단계에서의 예상 평균 CPU 시간과 디스크 서비스 시간이며, η는 $[0.25*Num\ Users, 4*0.25*Num\ Users]$ 구간에서 균일한 분포를 가지는 확률 변수이다. 여기서 0.25와 4는 실험적인 수치이다.

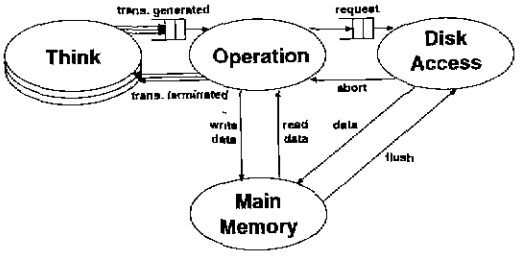


그림 6 모의실험 모델

$$Trans\_Deadline = T_{min} * \eta, \quad (4)$$
$$T_{min} = (CPU\_Time + IO\_Time) * Num\_Steps$$

성능 평가에 사용된 척도(measure)는 트랜잭션 분실 확률(transaction loss probability)과 평균 반응 시간(average response time)이다. 트랜잭션 분실 확률은 멀티미디어 서버에서 QOS와 관련이 있으므로 중요한 요소라고 할 수 있다. 평가 대상의 스케줄링 알고리즘은 실시간 처리를 고려하지 않은 기존의 알고리즘인 FCFS, SSTF, SCAN, C-SCAN과, 실시간 처리를 고려한 EDF, P-SCAN, FD-SCAN, SSEDO, SSEDV이다. SSEDV와 SSEDO의 스케줄링 변수 α, β는 각각 0.8과 2로 고정하였다.

그림 7과 그림 8은 각각 여러 디스크 스케줄링 알고리즘들의 트랜잭션 분실 확률과 평균 반응 시간을 보여준다. 평균 반응 시간 면에서는 모든 알고리즘들이 비슷하나, 트랜잭
분실 확률은 알고리즘에 따라서 차이가 있음을 알 수 있다. 보다 좋은 서비스를 하기 위해서는 트랜잭션에 대하여 반응 시간은 빠르면서도 트랜잭션 분실 확률은 적게 하는 것이 중요하며, 이러한 성질을 가지는 스케줄링 알고리즘의 개발이 필요함을 알 수 있다.

5.2 Reddy의 연구

표 1은 모의실험에 사용된 디스크 변수에 대한 가정이며, 그림 9는 [10]에서 사용된 실시

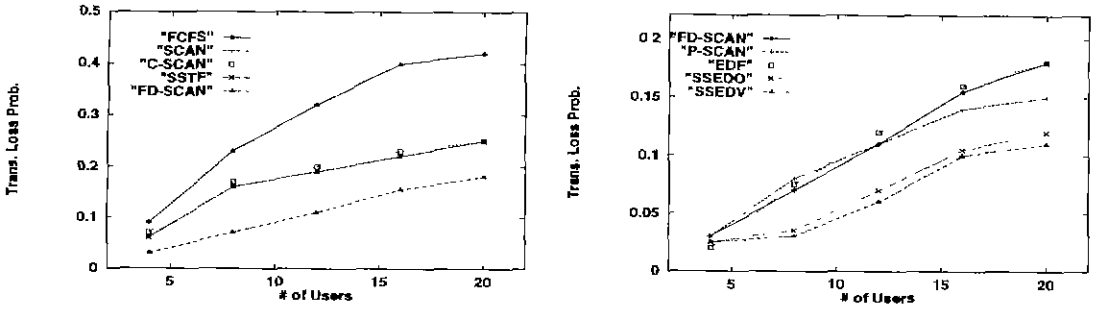


그림 7 트랜잭션 분실 확률

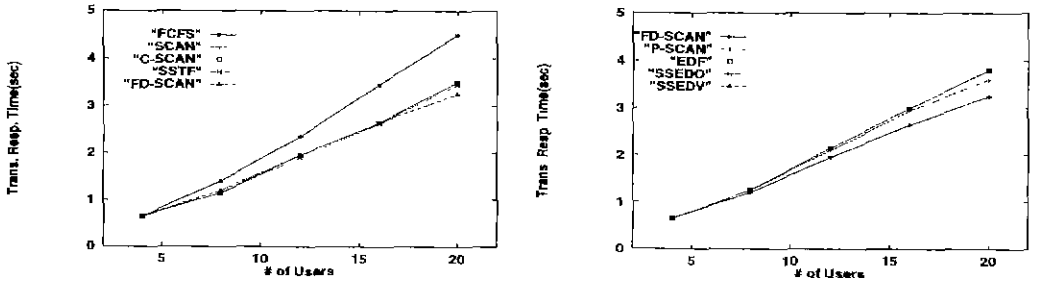


그림 8 평균 반응 시간

표 1 디스크 변수

Time for one rotation	11.1ms
Average Seek	9.4ms
sectors/track	84
tracks/cylinder	512 bytes
cylinders/dusk	2577
seek cost function	nonlinear
min. seek time	1.0ms

간 데이타 요청과 비주기적(aperiodic)인 비실시간 데이타 요청의 서비스 모델이다. 실시간 요청 스트림(stream)은 150KB/sec의 데이타 전송률을 요구한다고 가정한다. 150KB/sec은

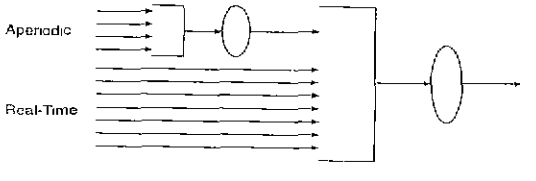
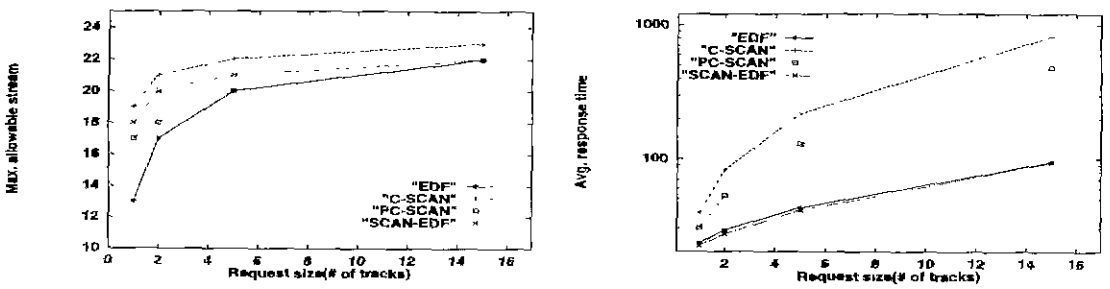


그림 9 데이터 요청의 서비스 모델

CDROM 데이타 스트림의 전송률과 같다. 비주기적인 요청이 많을 경우, 실시간 데이타 요청이 영향을 받을 수도 있으므로 이를 위하여 실시간 데이타 요청에 영향을 많이 주지 않도록 비주기적인 요청을 제어할 필요가 있다. 그러나, 이러한 비주기적인 요청은 우선 순위가 높아서 오는 즉시 바로 서비스를 받는다고 가정한다. 비주기적인 데이타 요청은 한 트랙만큼의 데이타를 요구한다고 가정하고, 실시간 데이타 요청은 1, 2, 5, 15 트랙만큼의 데이타를 요구한다고 가정한다.

성능 평가에 사용된 척도는 최대 허용 가능한 스트림의 수와 비주기적인 요청의 반응 시간이다. 최대 허용 가능한 스트림의 수는 데이타 요청을 하는 스트림의 수 n을 점진적으로 증가시켜 가면서 $n+1$개의 스트림에서 더이상 데드라인을 만족시켜 줄 수 없을 때 구한다. 평가 대상으로 사용된 알고리즘은 EDF, C-SCAN, PC-SCAN, SCAN-EDF이다. C-SCAN에서는 비주기적인 요청에 우선 순위를



(a) (b)

그림 10 (a) 최대 허용 가능한 스트림의 수 (b) 비주기적인 요청의 반응 시간

주지 않고 요청된 트랙 번호에 따라 SCAN 방향으로 서비스를 해준다. PC-SCAN은 C-SCAN의 변형으로서, 비주기적인 요청에 대해서 C-SCAN에서의 디스크 암이 움직이는 방향과 다르게 될 수 있다. 이 경우는 비주기적인 요청에서 요구하는 트랙 번호가 현재 디스크 암 뒤에 남아있는 트랙의 수의 절반보다 작을 때이다.

그림 10은 모의실험을 통하여 얻은 그래프로, 각각 최대 허용 가능한 스트림의 수와 비주기적인 요청의 반응 시간이다. 요청하는 트랙의 수가 많아질 수록 최대 허용 가능한 스트림의 수에 대한 성능 향상이 있으며, 각 알고리즘들의 성능이 비슷해짐을 알 수 있다. 이는 서비스하는 데이타의 양이 많아질 수록 전송 시간이 탐색 시간에 비하여 많아지기 때문이다. 그러나, 이에 따른 버퍼의 크기가 증가해야만 한다는 단점이 있다. 비주기적인 요청의 반응 시간도 같이 증가함을 알 수 있다. 그러나, 이 연구는 CDROM 정도의 데이타 전송률을 가정하고 있어서 멀티미디어 서버에서의 요구에 맞지 않으며, 현재의 멀티미디어 서버 요구에 맞는 가정을 사용하여 연구할 필요가 있음을 알 수 있다.

5.3 실시간 처리 디스크 스케줄링 알고리즘의 분류

다음은 실시간 처리 디스크 스케줄링 알고리즘을 사용한 방법에 따라 분류한 것이다. 표에서 알 수 있듯이 여러 알고리즘들이 평균 반응 시간 특성을 좋게 하기 위하여 SCAN 알고리즘의 기법을 혼용하여 사용하고 있음을 알 수 있다. 또한 사용 목적에 맞게 다른 기법들을 조합하여 사용하고 있음을 알 수 있다.

표 2 디스크 스케줄링의 분류

알고리즘	Scan	Group	Window	Arm 위치
FCFS				
SSTF				*
SCAN	*			
C-SCAN	*			
EDF				
P-SCAN	*	*		
FD-SCAN	*			
SSEDO			*	*
SSEDV			*	*
SCAN-EDF	*			
GSS	*	*		

6. 요 약

본 논문에서는 실시간 처리 디스크 스케줄링 알고리즘에 대하여 살펴보았다. 이러한 실시간 처리 디스크 스케줄링 알고리즘은 멀티미디어 서버의 저장 장치로 사용되는 하드 디스크에 접근할 때 각 디스크 요청의 데드라인을 최대한 만족시키면서 처리율을 좋게 하기 위하여 사용될 수 있다. 멀티미디어 서버에서 한정된 자원을 효과적으로 사용하면서 많은 사용자에게 실시간으로 비디오, 오디오 등의 데이타를 제공하기 위해서는 이러한 스케줄링 기법의 도입이 필수적이며, 그 중 중요한 부분이 바로

직접 비디오, 오디오 등의 데이타를 읽고 쓰는 디스크이다. 요청된 데이타의 탐색 시간을 줄이면서 시간적인 제약조건을 만족시켜 주기 위한 연구가 많이 이루어져 왔다. 그러나 아직까지는 멀티미디어 서버의 성능에 대한 일관성이 없어서 각 스케줄링 알고리즘들에 대한 가정이 많이 달라서 이들을 객관적으로 비교 평가하기 어려운 상황이며, 이를 위한 연구가 필요한 것으로 평가된다.

참고문헌

[1] D. Jadav and A. Choudhary, "Designing and Implementing High-Performance Media-on-Demand Servers," *IEEE Parallel and Distributed Technology*, vol. 3, pp. 29-39, Summer 1995.

[2] T. D. C. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand." *IEEE Multimedia*, vol. 1, pp. 14-24, Fall 1994.

[3] D. Jadav, C. Srinilta, A. Choudhary, and P. B. Berra, "Design and Evaluation of Data Access Strategies in a High Performance Multimedia-On-Demand Server," in *Proc. 2nd Int'l Conf. on Multimedia Computing and Systems*, May 1995.

[4] H. M. Deitel, *Operating Systems*. Addison Wesley, 2nd ed., 1990.

[5] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of ACM*, pp. 46-61, 1973.

[6] K. Jeffay, D. F. Stanat, and C. U. Martel, "On Non-preemptive Scheduling of Periodic and Sporadic Tasks," in *Proc. of Real-Time Systems Symp.*, pp. 129-139, Dec. 1991.

[7] M. J. Carey, R. Jauhari, and M. Livny, "Priority in DBMS Resource Scheduling," in *Proc. of the 15th VLDB Conf.*, pp. 397-410, 1989.

[8] R. Abbott and Garcia-Molina, "Scheduling I/O Requests with Deadlines : A Performance Evaluation," in *Proc. of Real-Time Systems Symp.*, pp. 113-124, 1990.

[9] S. Chen, J. A. Stankovic, F. Kurose, and D. Towsley, "Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems," *Journal of Real-Time Systems*, vol. 3, pp. 307-336, Sept. 1991.

[10] A. Reddy and J. Wyllie, "Disk Scheduling in a Multimedia I/O System," in *Proc. ACM Int'l Conf. on Multimedia*, pp. 225-233, Aug. 1993.

[11] P. S. Yu, M.-S. Chen, and D. D. Kandlur, "Design and Analysis of a Grouped Sweeping Scheme for Multemedia Storage Management," in *Proc. of the 3rd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, Nov. 1992. San Diego, California.

[12] M.-S. Chen, D. Kandlur, and P. Yu, "Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams," in *Proc. ACM Int'l Conf. on Multimedia*, pp. 235-242, Aug. 1993.

이 종 민

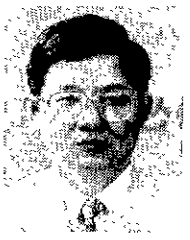
1992 경북대학교 공과대학 컴퓨터공학과 학사

1994 한국과학기술원 전산학과 석사

1994~현재 병렬처리, 웜홀라우팅, 멀티미디어 시스템

이 귀 영

1992 부산대학교 자연과학대학 전자계산학과 학사
1994 한국과학기술원 전산학과 석사
]1995～현재 한국과학기술원 전산학과 박사 과정
관심분야 : 실시간 운영체제, 결함 허용 시스템, 멀티미디어 컴퓨팅



이 흥 규

1978 서울대학교 공과대학 전자공학과 학사
1981 한국과학기술원 전산학과 석사
1984 한국과학기술원 전산학과 박사
1985～86 Univ. of Michigan, U.S.A. Research Scientist
1986～현재 한국과학기술원 전산학과 부교수
관심분야 : 고장 감내 시스템, 엄격한 실시간 시스템