VOD 서버에서 마감시간 초과율 감소를 위한 다중 디스크배열 I/O 스케줄링

(I/O Scheduling of Multiple Disk Arrays for Reducing Deadline Miss Rate on VOD Servers)

정 경 진 † 김 성 조 ††
(Kyong Jin Jeong) (Sung Jo Kim)

요 약 디스크배열 또는 RAID 시스템은 저렴한 비용으로 대용량 저장공간을 제공할 수 있으나, VOD 서비스와 같은 대규모 멀티미디어 서비스에 대해서는 아직 충분한 입출력 속도를 지원하지 못하고 있다. 보다 높은 대역폭을 지원하기 위해 다수의 디스크 컨트롤러를 채용한 다중 디스크배열에서 스트라이프를 전체 디스크에 효과적으로 분산시키기 위한 수직 스트라이핑 모델이 이용되고 있다. VOD 서비스 이용자가 고품질의 동영상을 감상하기 위해서는, 응용프로그램이 요청한 마감시간까지 저장장치에 기록되어 있는 데이타를 읽어와야 한다. 본 논문에서는 효과적인 VOD 서비스 지원을 위해, 다중 디스크배열에서 각 디스크 컨트롤러의 지역 요청큐로 분산된 입출력 요청들을 스케줄링하여 마감시간 초과율(deadline miss rate)을 줄이기 위한 알고리즘을 제안한다. 이 알고리즘은 VOD 서버와 같이 데이타 읽기 작업이 많은 멀티미디어 서비스에 적합하도록 설계되었다. 시뮬레이션 결과, 제안된 알고리즘이 마감시간 초과율을 평균 41.5% 감소시킴을 알 수 있었다.

Abstract Although conventional disk arrays or RAID systems can provide large data storage systems with low cost, their I/O throughput is still insufficient for supporting large-scaled multimedia services like VOD service. To address this problem, vertical striping scheme is widely used, which distributes each stripe to entire disks on multiple disk arrays. For better VOD service quality, data on the storage must be read on deadline which the VOD application requested. We propose an algorithm that schedules I/O requests which are dispersed into the local I/O queue of each disk controller on multiple disk arrays to reduce the deadline miss rate. This algorithm is designed to support efficiently a VOD server on which read operations dominate. According to our simulation, the proposed algorithm reduces the deadline misses by 41.5%.

1. 서 론

동영상, 음성 등과 같은 데이타의 처리를 요구하는 멀티미디어 서비스는 데이타의 용량과 처리속도 면에서 높은 시스템 성능을 요구한다. 이러한 서비스의 지원을 위해 저렴한 비용으로 기본 데이타 입출력 속도를 보장할 수 있는 디스크배열 또는 RAID 시스템이 많이 활용되고 있다[1,2,3].

• 본 논문은 중앙대학교의 연구기자재 구입 지원 프로그램의 도움을 받아 수행한 결과임.
† 비 회 원 : 중앙대학교 컴퓨터공학과
neutrino@konan.cse.cau.ac.kr
†† 종신회원 : 중앙대학교 컴퓨터공학과 교수
sjkim@cau.ac.kr
논문접수 : 1999년 8월 9일
심사완료 : 2001년 3월 19일

최근 들어, 멀티미디어 서비스는 컴퓨터 시스템의 성능 향상과 사용자 요구의 확대 등으로 인하여 VOD 서비스와 같이 더욱 고속·대용량화되고 있는 추세이다. 효율적인 VOD 서비스를 위해서는 많은 양의 비디오 타이틀들을 대용량 저장장치에 구축하고, 일정한 서비스율로 사용자에게 전송해야 한다. VOD 서비스는 대용량 데이타 처리, 고속 데이타 입출력, 실시간 스케줄링, QoS 관리 및 고속 네트워킹 등 다양한 기술들이 종합적으로 요구된다. 특히, 편당 수GB에 달하는 영화 타이틀을 수천 개 이상 보유하고 이를 실시간에 사용자에게 전송하여야 하므로, 일반적인 멀티미디어 서비스보다 훨씬 큰 데이타 저장공간과 빠른 데이타 입출력 속도를 요구한다. 이러한 요구사항을 충족시키기 위해 기존 디스크배열 시스템의 성능 개선이 필요하다. 디스크배열

시스템의 용량은 컨트롤러에 보다 많은 수의 디스크를 연결함으로써 쉽게 확장될 수 있다. 반면, 데이타 전송 대역폭을 확장하기 위해 보다 빠른 데이타 처리 속도를 갖는 디스크 컨트롤러의 도입을 고려할 수 있으나, 여러 가지 문제[1]로 인하여 다수의 디스크 컨트롤러를 시스템 버스에 병렬로 연결시킨 다중 디스크배열 모델이 더 적합하다[4,5,6].

파일 시스템에서 요구하는 입출력 요청이 빠른 시간 내에 처리될 수 있도록 디스크배열의 장치 드라이버는 스트라이핑(striping)을 이용하여 이들 요청을 가능한 한 전체 디스크로 균등하게 분산시켜야 한다[5,7,8]. 다중 디스크배열에서는 연속 미디어의 특성을 살리고 지연시간을 최소화함으로써 보다 빠른 시간 내에 처리될 수 있도록 전체 디스크에 대해 스트라이프를 분산시킴으로써, 전체 시스템에 균등한 부하 분배와 향상된 대역폭을 기대할 수 있다. [6]과 [8]에서는 그림 1과 같이, 스트라이프를 디스크 컨트롤러마다 수직으로 분산시키는 방안을 제시하였다.

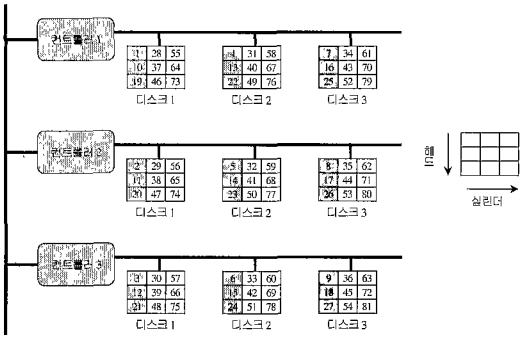

그림 1 수직 스트라이핑을 적용한 다중 디스크배열

다중 디스크배열과 수직 스트라이핑을 이용함으로써 저장장치의 대역폭을 향상시킬 수 있으나, VOD 서비스의 품질을 향상시키기 위해 디스크배열의 응답시간을 개선할 필요가 있다. 보다 빠른 입출력 응답시간을 제공함으로써 서비스 품질을 높일 수 있으나, 디스크배열의 각 디스크로 분산된 입출력 요청들을 적절히 스케줄하여 최대 응답시간을 줄이고 마감시간 초과율을 줄임으로써 VOD 서비스 이용자가 기다리는 횟수를 줄이도록 하는 방안도 VOD 서비스의 품질을 높일 수 있다.

본 논문은 다중 디스크배열 시스템을 제어하기 위해 장치 드라이버 수준에서 이용될 수 있는 I/O 스케줄링 기법을 제안한다. 다중 디스크배열 장치 드라이버는 파일 시스템에서 생성되는 디스크 입출력 요청을 분석하여 데이타가 저장된 목표 디스크의 컨트롤러들로 분산시키며, 제안된 알고리즘은 각 디스크 컨트롤러의 요청큐에 적재되어 있는 입출력 요청들을 스케줄링한다.

본 논문의 구성은 다음과 같다. 2장에서는 다중 디스크배열 시스템의 스케줄링 모델을 소개하고, 3장에서는 이러한 모델에서 마감시간 초과율을 줄일 수 있는 알고리즘을 기술한다. 4장에서는 시뮬레이션을 통해 제안된 알고리즘의 성능을 평가하며, 5장에서는 결론을 맺고 추후 연구과제를 제시한다.

2. 다중 디스크배열 스케줄링 모델

2.1 단일 디스크배열에서 입출력 요청의 처리과정

그림 2는 단일 디스크배열에서 입출력 요청의 처리과정을 보인 것이다. 파일 시스템으로부터 발생된 입출력 요청은 장치드라이버의 요청큐에 적재된다. 요청큐에서 대기하는 시간은 요청큐의 길이와 스케줄링 알고리즘의 효율성에 의존한다. 탐색시간은 물리적 디스크 입출력 처리 과정에서 가장 긴 시간을 차지하며 디스크 스케줄링 방법에 따라 큰 차이를 보이기 때문에, SCAN, C-SCAN, GSS 등의 기존 알고리즘에서 성능 개선의 주 대상이 되었다. 반면, 회전지연시간과 데이타 전송시간은 스케줄링 알고리즘에 거의 영향을 받지 않는다. 이러한 과정을 거쳐 읽혀진 데이타는 디스크 컨트롤러의 데이타 버스를 점유하기 위한 버스 경쟁과정을 거친 후, 파일 시스템으로 전송되기 위해 컨트롤러로 전달된다.

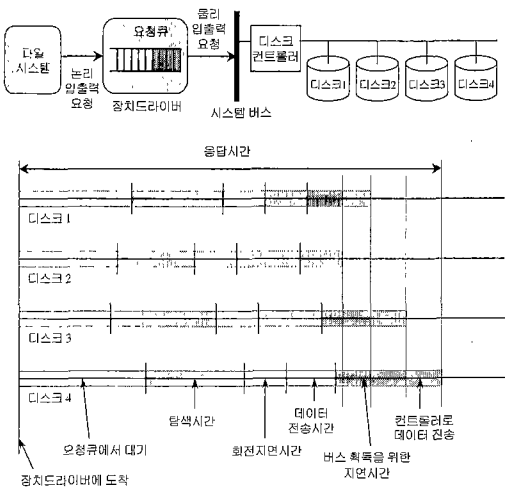

그림 2 단일 디스크배열에서 입출력 요청의 처리과정

2.2 응답시간과 마감시간

VOD 서버에서 동작 중인 서비스 프로그램은 파일 시스템에 요청한 데이타 읽기 요청이 일정 시간 내에 처리될 때 좋은 품질의 서비스를 제공할 수 있다. 저장 장치로부터 데이타 입출력이 늦어질 경우, 특별한 스케줄링 기법이 적용되지 않는 한 서비스 프로그램은 입출력이 완료될 때까지 대기할 수 밖에 없다. 이러한 문제를 해결할 수 있는 방안 중 하나는 데이타 입출력의 평균 응답시간을 줄이는 것이다. 예를 들어, 시스템이 단위 시간에 처리가능한 데이타 대역폭을 μ라 하고, 단위 시간에 응용프로그램들이 요청하는 데이타의 양을 λ라 가정하자. 만일 λ가 μ보다 크다면 시스템의 입출력 요청큐는 거의 포화상태일 것이며, 응용프로그램들의 데이타 입출력 완료 대기 시간이 길어지게 된다. 만일 평균 응답시간 x를 $x'(<x)$으로 줄일 수 있다면, μ의 변화 없이 응용프로그램의 처리율을 $(x/x'-1)$만큼 향상시킬 수 있다.

일반적으로 VOD와 같은 멀티미디어 서비스는 입출력 작업이 특정 시간 내에 종료되어야 한다. 따라서, 응용프로그램은 입출력을 요청할 때, 입출력 요청과 더불어 요구되는 입출력 마감시간(deadline)을 함께 시스템에 전달한다. 응용프로그램의 상태에 따라 입출력 데이타의 양이 적더라도 마감시간이 빠를 수 있으며, 반대로 데이타의 양이 많더라도 마감시간이 충분히 늦어 여유있게 처리될 수 있는 경우가 있다. 마감시간이 지켜지지 않을 경우 멀티미디어 데이타의 재생이 일시 정지되므로, 마감시간을 지키는 것은 서비스 품질 보장을 위해 매우 중요하다. 따라서, 일정한 수준의 서비스 품질을 유지하기 위해 운영체제는 입출력 요청이 마감시간 내에 처리될 수 있도록 해야하며, 이는 응답시간 향상보다 더 중요하다.

2.3 다중 디스크배열의 스케줄링 모델

그림 3은 다중 디스크배열 시스템에서 입출력 요청의 처리과정을 보인 것이다. 운영체제의 파일 시스템은 다중 디스크배열 시스템을 단일 가상 디스크로 간주하고, 데이타의 논리 입출력 요청을 장치 드라이버에 전달한다. 논리 입출력 요청은 장치 드라이버의 전역 요청큐(global request queue)에 삽입된다. 또한, 다중 디스크배열의 각 컨트롤러는 자신의 지역 요청큐(local request queue)를 유지하고 있다. 장치 드라이버의 스케줄러는 먼저 전역 요청큐에서 입출력 요청을 하나씩 꺼내와 순번(sequence number)을 붙이고, 각 컨트롤러의 지역 요청큐로 분산시킨다. 이러한 입출력 요청의 분산 과정에서 수직 스트라이핑이 적용된다. 지역 요청큐에 들어 있는 입출력 요청들은 FIFO, Scan-EDF와 같은 다양한 스케줄링 과정을 거쳐 처리된다.

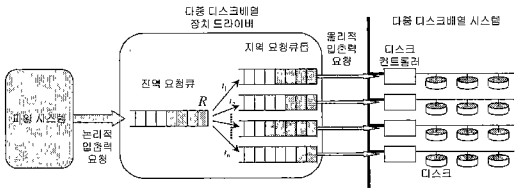

그림 3 다중 디스크배열에서 입출력 요청의 처리 과정

디스크 컨트롤러의 대역폭을 ρ, 디스크의 물리적 대역폭을 σ, 그리고 컨트롤러 당 디스크 수를 d라 하자. $\rho > d\sigma$일 때 컨트롤러는 모든 디스크의 대역폭을 수용하므로, 스케줄링 알고리즘은 각 디스크의 접근 속도 향상만을 고려하면 된다. 그러나, $\rho < d\sigma$인 경우에는 컨트롤러가 모든 디스크의 대역폭을 수용하지 못하므로, Scan-EDF와 같은 고속 디스크 헤드 스케줄링 알고리즘을 적용하여 단일 디스크 내에서의 입출력 요청이 완료되었더라도 컨트롤러는 아직 다른 디스크의 데이타를 전송 중일 확률이 높다. 이러한 경우에는 컨트롤러가 입출력 요청을 전달할 디스크를 어떤 순서로 선택하는가에 따라 처리시간이 크게 영향을 받게 된다. 따라서, 각 디스크로 흩어져 있는 물리 입출력 요청들이 보다 빠른 시간 안에 종료될 수 있도록 지역 요청큐를 스케줄링함으로써, 입출력 요청의 마감시간 초과율을 감소시킬 수 있다.

본 논문에서는 이 점에 착안하여, $\rho < d\sigma$의 조건을 갖는 다중 디스크배열 시스템에서 각 지역 요청큐를 스케줄하여 입출력 요청의 마감시간 초과율을 감소시켜 VOD 서비스의 서비스 품질을 향상시키기 위한 알고리즘을 제안한다.

3. 마감시간 초과율 개선을 위한 알고리즘

파일 시스템으로부터의 논리 입출력 요구는 입출력 방향, 논리적 시작 디스크 블록 번호, 데이타 블록의 수, 그리고 마감시간으로 구성되며(그림 4 참조), 디스크 컨트롤러의 지역 요청큐에 쌓이는 입출력 요청의 구조는 그림 5와 같다.

읽기/쓰기	시작 블록 번호	입출력할 블록의 수	마감시간

그림 4 논리 입출력 요청의 구조

순번	디스크 번호	읽기/쓰기	논리 블록 번호	처리할 섹터 수	지연 시간	실행 시간	마감 시간

그림 5 지역 요청큐 내 입출력 요청의 구조

본 논문에서 제안하는 알고리즘의 목적은 각 논리 입출력 요청들의 마감시간 초과율(deadline miss rate)을 줄일 수 있도록, 각 디스크 컨트롤러의 지역 요청큐를 스케줄하는데 있다. 단일 논리 요청 R이 $r_1, \ldots, r_n$의 물리 입출력 요청들로 분산되었을 때, R의 응답시간 τ_R은 식 (1)과 같다.

단, R. ArrivalTime과 r. EndTime은 각각 R이 전역 요청큐에 도착한 시간과 r의 실행 완료 시간을 나타낸다.

$$\tau_R = \max(r_1.\text{EndTime}, \cdots, r_n.\text{EndTime}) - R.\text{ArrivalTime} \quad (1)$$

마감시간 초과율을 줄이기 위해서는, 도착 시간과 완료 시간 간의 간격을 줄일 수 있도록 각 요청큐에 쌓인 요청들의 순서가 재배치되어야 한다. 즉, 도착 시간부터 마감시간까지의 간격 R. TimeToDeadline 값이 작고 τ_R 값이 큰 요청이 먼저 처리될 수 있도록, 각 물리 입출력 요청 $r_1, \ldots, r_n$을 요청큐 내에서 큐의 앞쪽(front)으로 전진시킨다.

스케줄링 알고리즘은 전역 요청큐의 입출력 요청이 각 디스크 컨트롤러의 지역 요청큐로 분산된 직후 동작한다. 예를 들어, 지역 요청큐 Q에 저장된 두 요청 v와 w가 상호 교환된 후, 이들을 각각 v', w'라 하자. Q 내의 입출력 요청들의 총 처리시간 QueueTotal Time을 식 (2)와 같이 정의할 수 있으며, 이 값은 Q에서 대기중인 요청의 수와 형태에 따라 변화된다. 입출력 요청이 실행될 때까지의 지연시간 v. Delay는 식 (3)과 같이 계산되며, Q의 길이가 길고 R. TimeToDeadline이 작을수록 큰 값을 갖는다. 이 때, v'와 w'의 지연시간은 각각 식 (4), (5)와 같이 계산된다. Execution Time은 입출력 요청을 처리하는데 요구되는 물리적 동작 시간으로, 실린더 탐색 시점부터 데이타 전송 완료 시간까지의 간격이다. 여기서 AST와 RL은 실제 디스크 입출력 요청의 형태에 따라 변화될 수 있는 값이다. 그러나, 2.3절에서 언급한 바와 같이 단일 컨트롤러에 연결된 디스크의 수가 많아질수록 디스크의 대역폭 총합이 컨트롤러의 대역폭보다 훨씬 커지며 컨트롤러의 데이타 버스 점유를 위한 경쟁이 심화되어, 각 디스크들은 다음 입출력을 위해 기다리는 시간이 커지게 된다. 또한, 디스크에 접근하기 위해 실린더, 헤드, 섹터 단위로 접근할 수 있다면 스케줄러가 AST 값을 어느 정도 예상할 수 있으나, 디스크배열 시스템의 95%를 차지하는 SCSI 시스템[9]에서는 단지 논리 블록 번호로만 접근이 가능할 뿐만 아니라 디스크 내부적으로 섹터 재사상(sector remapping) 기법 등이 이용되기 때문에, 현실적으로 스케줄러는 정확한 디스크 탐색시간을 예측하기 어렵다. 따라서, 본 논문에서는 알고리즘의 단순화를 위해 평균값을 이용하기로 한다.

$$\text{QueueTotalTime} = \sum_{x=1}^{t} (AST + RL + DT_{r_x}) \quad (2)$$

t: 요청큐의 요청 수
AST: 평균 탐색시간
RL: 회전지연시간
DT_{r_x}: r_x의 데이타 전송시간

$$v.\text{Delay} = \text{QueueTotalTime} - R.\text{TimeToDeadline} \quad (3)$$

$$v'.\text{Delay} = v.\text{Delay} - w.\text{ExecutionTime} \quad (4)$$

$$w'.\text{Delay} = w.\text{Delay} + v.\text{ExecutionTime} \quad (5)$$

v'. Delay와 w'. Delay는 각각 교환 결과에 의해 줄어든 v의 지연시간 및 늘어난 w의 지연시간이다. 이 때, 식 (6)과 같이 늘어난 w의 지연시간 w'. Delay이 v. Delay보다 작다면, v와 w의 교환으로 인하여 입출력 응답시간이 향상될 수 있으므로, 지역 요청큐에서 v와 w의 순서를 교환하여 v를 앞으로 전진시킨다.

$$w'.\text{Delay} < v.\text{Delay} \quad (6)$$

Delay 값은 입출력 요청이 실행되기까지 대기하여야 하는 시간이므로, 이는 입출력 요청의 시작 시간으로 고려될 수 있다. v_s와 v_x를 각각 v의 시작시간과 실행시간, w_s와 w_x를 w의 시작시간과 실행시간이라 가정하자. 이 때, v는 w의 바로 다음 요청이기 때문에 $v_s = w_s + w_x$이다. 여기서 교환전의 평균 시작시간 $v_s + w_s$와 교환 후의 평균 시작시간 $v'_s + w'_s$을 고려하자. 식 (4), (5) 및 $v_s = w_s + w_x$에 의해 $v'_s + w'_s = (w_s + v_x) + w_s$이다. 여기서 식 (6)이 참이라면 $w_s + v_x < v_s$이므로 $v'_s + w'_s < v_s + w_s$이 된다. 즉, 교환된 후의 평균 시작시간이 교환전의 평균 시작시간보다 짧아졌음을 의미한다.

Delay와 ExecutionTime을 각각 D와 T로 표현할 때 그림 6에서 보인 바와 같이, 파일시스템에서 전달된 입출력 요청 V, W는 여러 개의 v와 w로 분리되어 컨트롤러의 지역 요청큐로 삽입된다. 이 때, V와 W의 응답시간은 각각 11, 6이다. 여기서 컨트롤러 2의 지역 요청큐에 들어있는 입출력 요청 v와 w를 서로 교환하여 v', w'가 되었을 때 V와 W의 응답시간은 각각 6, 9가 되었으며, 따라서 V와 W의 평균 응답시간은 8.5에서

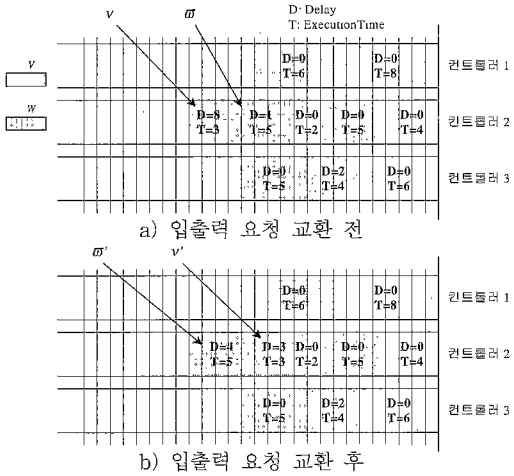

그림 6 물리 입출력 요청의 전진

7.5로 줄었음을 알 수 있다.

VOD 서비스에서는 저장장치에 새로운 타이틀을 저장하는 작업보다 저장된 타이틀로부터 데이타를 읽어 네트워크로 전송하는 작업이 월등히 많으며, 데이타 쓰기 작업은 서비스 품질에 직접적인 영향을 미치지 않으므로 알고리즘을 적용하지 않는다. 즉, v가 쓰기 작업인 경우에는 입출력 요청의 전진작업을 수행하지 않는다. 또한, v, w가 각각 읽기 및 쓰기 작업인 경우, 읽기-쓰기 데이타 충돌 현상이 발생할 수 있다. 예를 들어, v가 w의 쓰기 작업에 따라 변경된 데이타를 읽어와야 하나, v와 w가 서로 교환된 경우 v는 변경되지 않은 데이타를 읽게 될 수 있다. 따라서, 이 경우에는 v와 w가 접근하는 디스크 영역이 겹치는가를 검사하여, 데이타 충돌 현상을 초래할 경우 v의 전진작업을 피하여야 한다. 물론, 쓰기 요청에 대해 동기화 처리를 할 경우 이러한 쓰기 충돌 검사를 할 필요가 없을 수도 있으나, 본 논문에서는 읽기 요청 처리 성능을 보다 향상시키기 위해 쓰기 요청의 동기화를 배제한다.

이 알고리즘의 주요 목표는 하나의 논리 입출력 요청이 각 디스크 컨트롤러의 요청큐로 분산되었을 때, 분산된 각 입출력 요청들의 시작 및 종료 시간의 편차를 줄임으로써 마감시간 초과율을 줄이는데 있다. 입출력 요청 V가 $v_1, \ldots, v_n$으로 분리될 때, QueueTotalTime이 크고 V. TimeToDeadline값이 작을수록 v_x. Delay $(1 \le x \le n)$의 값이 커지며, 식 (6)의 조건에 부합될 가능성이 커진다. 따라서, 각 지역 요청큐 길이의 편차가 크고 마감시간 값이 작을수록 입출력 요청의 전진 가능성이 높아진다.

다중 디스크배열 시스템 운용을 위한 작업은 초기화(MDA_Initialization), 물리 입출력 요청의 각 지역 요청큐로의 분산(MDA_LQueueInsertion), 지역 요청큐에서 처리 완료된 입출력 요청 삭제(MDA_LQueue Removal) 등으로 이루어진다. 알고리즘이 필요로 하는 전역 데이타에는 논리 입출력 요청이 저장될 전역 요청큐 GQueue와 각 컨트롤러마다 처리해야 할 물리 입출력 요청이 저장될 지역 요청큐 LQueue 배열이 있다. 각 지역 요청큐 LQueue는 큐 내에 있는 입출력 요청들의 총 실행시간인 QueueTotalTime을 유지한다.

초기화 알고리즘 MDA_Initialization은 모든 LQueue의 QueueTotalTime값을 0으로 설정한다. 분리된 입출력 요청 $r_1, \ldots, r_n$을 각 지역 요청큐로 삽입하고, 지역 요청큐를 스케줄하는 알고리즘 MDA_LQueue Insertion은 다음과 같다.

```
algorithm MDA_LQueueInsertion
{
    /*  1 단계  ..... */
    get a logical request R from GQueue;
    split R into physical requests r_1, ..., r_n;
    for each r_x where 1≤x≤n {
        find index l of LQueue into which r_x is to be inserted;
        r_x. Delay = LQueue[l].QueueTotalTime -
                     R. TimeToDeadline;
        LQueue[l].QueueTotalTime += r_x. ExecutionTime;
        insert r_x into LQueue[l];
    }
    /*  2 단계  */
    if( R is a read request ) {
        for each r_x {
            set r_x to v;
            loop until r_x is the first request in the local queue {
                set the next request of r_x to w;
                if ( w is write) and
                ( v and w have same target address), ..... ①
                break;
                v'. Delay = v. Delay - w. Execution Time;
                w'. Delay = w. Delay + v. Execution
                            Time;
                if ( v'. Delay < 0), break; ..... ②
                if v. Delay ≤ w'. Delay, break; ..... ③
                swap v and w;
            }
        }
    }
}
```

MDA_LQueueInsertion은 전역 요청큐로부터 하나의 논리 입출력 요청 R을 가져와 물리 입출력 요청 $r_1, \ldots, r_n$

으로 분리하여 각 지역 요청큐로 삽입하는 1단계와, 제안된 알고리즘을 적용하여 각 지역 요청큐에 삽입된 입출력 요청들을 앞으로 전진시키는 2단계로 구성되어 있다. 1단계에서는 수직 스트라이핑이 이루어지며, 각 지역 요청큐의 QueueTotalTime값을 조정한다. 2단계에서는 R이 읽기 요청인 경우에만 입출력 요청의 전진작업을 수행한다. 각 지역 요청큐에 새롭게 삽입된 물리 입출력 요청 r_x에 대해 전진작업을 수행할 때, 쓰기 요청과 겹치는 경우(①), v'.Delay가 음수인 경우(②), 그리고 전진 효과가 없는 경우(③)에는 전진작업을 실시하지 않는다.

MDA_LQueueRemoval은 처리가 완료된 입출력 요청 r 을 지역 요청큐에서 제거하고, QueueTotalTime에서 r.ExecutionTime 을 빼준다.

이 알고리즘은 다음과 같은 조건 하에서 동작하는 것으로 가정한다.

- 읽기 요청이 쓰기 요청에 비해 압도적으로 많다.
- 시스템에 연결된 모든 하드디스크는 동일한 모델이다.
- 입출력이 끝난 각각의 입출력 데이타의 재조립(reassembly)은 고려하지 않는다.
- 시스템 버스의 대역폭은 모든 디스크 컨트롤러의 대역폭을 수용할 수 있다.

알고리즘 MDA_LQueueInsertion의 복잡도를 계산해 보면 다음과 같다. 1단계는 파일시스템에서 전달된 입출력 요청을 실제 물리적 요청으로 분리하여 각 LQueue에 삽입하는 작업이므로, 컨트롤러의 수와 컨트롤러 당 디스크 수를 각각 c, d라 할 때 $O(cd)$의 수행시간을 갖는다. 2단계는 각 LQueue에 삽입된 입출력 요청을 알고리즘에 의해 전진시키는 작업이므로, LQueue의 크기를 q이라 할 때 $O(cdq)$의 수행시간이 요구된다. 따라서, 알고리즘의 전체 수행시간은 $O(cd)+O(cdq)=O(cdq)$이다. 각 분리된 지역 요청큐의 매 입출력 요청마다 알고리즘이 적용된다. 알고리즘은 전진 작업을 위해 지역 요청큐를 한 번씩 탐색하게 됨을 의미하며, 이는 매 입출력 완료 시 다음 입출력 요청을 선택하기 위해 요청큐를 한 번씩 탐색하는 Scan/EDF와 같은 일반적인 디스크 스케줄링 알고리즘의 복잡도와 갖게 되나, 실제 계산 과정은 Scan/EDF에 비해 추가의 계산 작업이 요구된다.

4. 시뮬레이션

4.1 시뮬레이션 환경 및 유사 모델 제시

본 연구에서 제안된 알고리즘의 성능을 평가하기 위해, 다음과 같은 4가지 모델들에 대하여 시뮬레이션을 수행하였다. 각 모델들은 모두 수직 스트라이핑이 적용된 것으로 가정하였다.

- VS: 각 컨트롤러의 지역 요청큐에 대해 FIFO 방식으로 처리하는 모델
- VSA: 제안된 알고리즘을 적용한 모델
- Scan-EDF: 각 디스크 별로 Scan/EDF 알고리즘을 적용한 모델
- SE-WFQ: 각 디스크 별로 Scan/EDF 알고리즘을 적용한 후, WFQ(Weighted Fair Queuing)을 적용하여 각 디스크의 요청수가 많을수록 스케줄되는 디스크의 우선순위를 달리하는 모델

또한, 디스크 컨트롤러는 Ultra2 SCSI 타입으로 대역폭이 80MB/s이며, 모든 디스크 드라이브는 Ultra2 SCSI 타입의 Quantum Atlas 10K II 모델[10]을 참조하여 다음과 같은 제원을 가진 것으로 가정하였다.

- 디스크 용량: 18 GB
- 평균 회전 지연시간: 3 ms
- 평균 탐색 시간: 4.7 ms
- 최대 탐색 시간: 12 ms
- 트랙간 탐색 시간: 600 μs
- 최대 연속 데이타 전송율: 26MB/s

시뮬레이션 프로그램은 데이타 파일로부터 논리 입출력 요청을 읽어와 전역 요청큐에 삽입하는 GQIO_Generator, 전역 요청큐에서 논리 입출력 요청을 꺼내와 물리 입출력 요청들로 분리하여 각 디스크 컨트롤러의 지역 요청큐에 삽입한 후 스케줄링 알고리즘을 적용하는 GQIO_Scheduler, 그리고 각 물리 입출력 요청을 처리하는 LQIO_Consumer 등의 3 부분으로 구성된다. GQIO_Generator는 그림 4와 같은 논리 입출력 요청을 생성하며, 각 입출력 요청의 마감시간은 입출력 요청의 크기에 비례하여 평균 2~4초의 범위 내에서 정규분포를 이루도록 설정한다. GQIO_Scheduler는 프로그램 옵션에 따라 VS, VSA, Scan-EDF, SE-WFQ 중 하나의 알고리즘을 적용하여 지역 요청큐를 스케줄하며, LQIO_Consumer는 각 지역 요청큐에서 입출력 요청을 하나씩 꺼내와 요청의 정보에 따라 필요한 만큼의 시간을 지연한다. 이 때, 각 디스크의 현재 및 다음 헤드 위치를 고려하여 예상되는 탐색 시간을 계산하고, 여기에 평균 회전 지연시간과 데이타 전송시간을 더하여 해당 시간만큼을 지연한다. 또한, 시뮬레이션 프로그램은 일반적인 SCSI 버스를 그대로 모델링하여 디스크 입출력 요청이 끝나더라도 SCSI 버스 획득을 위한 경쟁과정을 고려하였으며, 이 과정에서 데이타 전송중인 디스크의 대역폭

총합이 컨트롤러의 대역폭을 넘지 않도록 조정된다.

본 절에서는 주어진 각 파라미터를 이용하여 데이타 파일을 생성하고, 이를 이용하여 시뮬레이션을 수행한다. MPEG-1으로 압축된 타이틀들이 저장되어 있고, 다수의 사용자가 동시에 접속하여 실시간에 비디오를 감상할 수 있는 가상의 VOD 서버가 존재하는 것으로 간주하여, 이 환경에서 발생할 수 있는 데이타를 파일로 기록하였다. 각 타이틀에 대해 인기도를 각기 다르게 설정하였으며, 서비스 이용자들도 인기도에 따라 최신 타이틀들을 보다 많이 감상하는 것으로 가정하여 데이타를 생성하였다. 생성된 데이타 파일은 각 알고리즘에 공통적으로 적용되어 시뮬레이션을 수행한다.

이러한 환경에서 동시 사용자 수, 저장된 타이틀 수, VOD 서비스 응용프로그램의 평균 입출력 요청 크기, 다중 디스크 배열 시스템의 컨트롤러 수, 컨트롤러 당 디스크 수 등 다양한 파라미터들이 변화될 수 있으며, 알고리즘은 이러한 파라미터들에 의해 영향을 받을 수 있다. 본 절에서는 평균 입출력 요청의 크기와 컨트롤러 당 디스크 수의 변화에 대해 시뮬레이션을 수행하며, 각 지역 요청큐로 분산된 물리 입출력 요청들의 전진 횟수를 분석한다.

4.2 입출력 요청의 크기

논리 입출력 요청의 크기가 클수록 분리되는 물리 입출력 요청의 수가 많아진다. 제안된 알고리즘은 각 지역 요청큐로 분리되어 삽입된 물리 입출력 요청간의 처리 시간 완료 편차를 고려하도록 설계되었기 때문에, 물리 입출력 요청의 수에 영향을 받게 된다.

그림 7은 논리 입출력 요청의 평균 크기가 변화함에 따라, 4.1절에서 언급한 4 모델들의 마감시간 초과율을 백분율(%)로 표시한 것이다. 컨트롤러 수, 컨트롤러 당 디스크 수, 타이틀 수, 동시 사용자 수가 각각 4개, 16개, 100개, 5000명일 때, VOD 서비스 응용프로그램이 시스템에 요청하는 입출력 요청의 평균 크기를 10KB/s ~ 200KB/s로 변화하면서 정규분포를 이루는 입출력 요청들을 생성하여 시뮬레이션을 수행하였다. 알고리즘을 적용한 VSA 모델이 대체적으로 마감시간 초과율이 낮아지는 효과를 보였다. 알고리즘을 적용하지 않은 VS 모델에 비해, VSA 모델은 마감시간 초과 횟수를 평균 43.92% 정도 감소시켰다. 60KB/s 부분까지 마감시간 초과율이 증가한 것은, 입출력 데이타의 크기에 비해 동작되어야 할 디스크의 수가 많기 때문인 것으로 판단된다. 즉, 단일 스트라이프 크기보다 근소하게 큰 입출력 요청의 경우 최대 3개의 디스크가 참여할 수 있게 되며, 이는 마감시간 초과율을 높이게 된다.

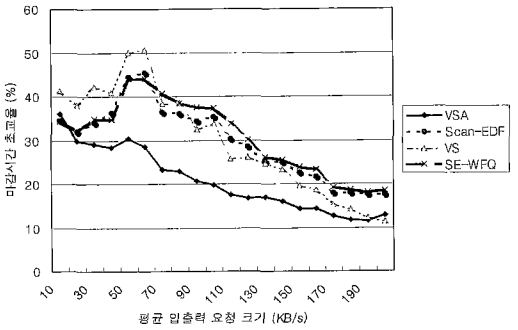

그림 7 입출력 요청의 평균 크기에 따른 마감시간 초과율의 변화

4.3 컨트롤러 당 디스크 수

그림 8은 컨트롤러에 연결된 디스크의 수를 변화시켰을 때, 각 모델들의 마감시간 초과율을 백분율(%)로 표시한 것이다. 컨트롤러 수, 타이틀 수, 동시 사용자 수, 평균 입출력 요청의 크기가 각각 4개, 100개, 5000명, 150KB/s일 때, VOD 서비스 응용프로그램의 입출력 요청을 생성하여 시뮬레이션을 수행하였다. 컨트롤러 당 디스크 수가 적은 경우에는 알고리즘을 적용했을 경우 오히려 마감시간 초과율이 높아졌으나, 디스크 수가 8보다 큰 경우 알고리즘을 적용했을 때 마감시간 초과율을 줄일 수 있었다. 디스크가 8개 이상일 때 디스크 대역폭의 총합이 컨트롤러의 대역폭보다 커지게 되므로, 알고리즘의 효과가 더 커지는 것으로 판단된다. 이 경우, VSA 모델은 VS 모델에 비해 마감시간 초과비율이 약 39.07% 감소하였다. 디스크 수가 17, 27, 34개일 때 마감시간이 급격히 증가한 것은, 논리 입출력 요청이 분리되는 평균 물리 입출력 요청의 수에 따라 입출력에 참

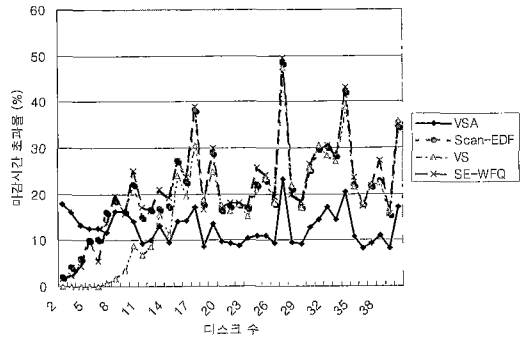

그림 8 컨트롤러 당 디스크 수에 따른 마감시간 초과율의 변화

여하는 디스크 수가 달라지기 때문인 것으로 판단된다.

4.4 물리 입출력 요청의 전진 횟수

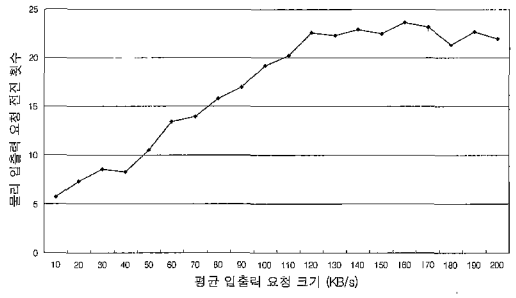

그림 9 물리 입출력 요청의 전진 횟수

그림 9는 본 연구에서 제안한 알고리즘을 이용할 때 각 물리 입출력 요청들이 지역 요청큐에서 전진하는 평균 횟수를 보인다. 컨트롤러 수, 컨트롤러 당 디스크 수, 타이틀 수, 동시 사용자 수가 각각 4개, 16개, 100개, 5000명일 때, VOD 서비스 응용프로그램이 시스템에 요청하는 입출력 요청의 평균 크기를 10 ~ 200 KB/s의 범위에서 정규분포를 이루도록 입출력 요청들을 생성하여 시뮬레이션을 수행하였다. 그림에서 나타난 바와 같이, 입출력 요청의 크기가 커질수록 물리 입출력 요청이 지역 요청큐 내에서 전진하는 횟수가 점차 증가함을 알 수 있다. 이는 입출력 요청의 크기가 커질수록 분리되는 물리 입출력 요청의 수가 많아지며, 따라서 입출력 요청의 처리 시간에 영향을 주는 지역 요청큐의 수가 증가한다. 알고리즘은 각 지역 요청큐의 지연 시간 편차가 클수록 더욱 많이 증가하도록 설계되었으므로, 물리 입출력 요청의 전진 횟수는 점차 증가한다. 그러나, 그림 9에서 입출력 요청의 크기가 120 KB보다 커지면서 전진 횟수가 둔화되는데, 이는 분리된 물리 입출력 요청들이 추가되는 지역 요청큐의 수가 더 이상 증가하지 않기 때문인 것으로 판단된다.

5. 결 론

디스크배열 시스템의 저장 용량과 함께 입출력 대역폭을 높이기 위해서는 다중 디스크배열 모델이 적합하다. 또한, 고속 데이타 처리가 요구되는 대용량 VOD 서버에서 다중 디스크배열 시스템의 성능을 극대화하기 위해서는 수직 스트라이핑의 이용이 필수적이다. 본 논문은 이러한 수직 스트라이핑을 채택한 다중 디스크배열 시스템의 장치 드라이버에서 이용될 수 있는 I/O 스케줄링 알고리즘을 연구하였다.

입출력 요청의 마감시간 초과율을 줄이기 위해, 이 알고리즘은 마감시간을 고려하여 각 입출력 요청의 처리 지연시간을 구하고 지역 요청큐 내의 입출력 요청 순서의 교환을 통해 평균 마감시간이 감소하도록 설계되었다. 이 때, 실제의 환경에 부합되도록 하기 위해, 탐색시간, 회전 지연시간, 그리고 데이타 전송시간을 고려하여 각 입출력 요청에 대한 지연시간을 계산하였다. 알고리즘은 또한 쓰기 요청에 따른 읽기-쓰기 충돌 문제(read-write conflict)를 해결하기 위해 쓰기 입출력 요청에 대해서는 지역큐 내의 지역 요청 전진 작업을 적용하지 않았으며, 읽기 입출력 요청에 대해서 앞의 요청이 같은 영역에 대한 쓰기 요청인 경우에 한하여 물리적 입출력 요구의 전진 작업을 하지 않도록 하였다. 시뮬레이션 결과, 본 논문에서 제안한 알고리즘을 사용한 경우 마감시간 초과율을 41.5% 정도 줄일 수 있었다.

본 연구에서 제안된 다중 디스크배열 시스템 상에서의 I/O 스케줄링 알고리즘은 VOD 서비스와 같은 환경에서 대용량 멀티미디어 데이타의 입출력 속도를 향상시키고, 서비스 품질을 높이는데 큰 기여를 할 것으로 기대된다. 이 알고리즘은 멀티미디어 데이타 처리에 이용될 고성능 RAID 시스템 구현에 응용될 수 있다.

본 논문에서 제안된 알고리즘을 개선하기 위해서는 다음과 같은 연구가 추가적으로 필요하다. 제안된 알고리즘은 디스크의 헤드 위치를 고려하지 않고 평균 탐색시간을 사용하고 있으나, 보다 정확한 디스크 스케줄링을 위해 입출력 요청의 블록 번호를 고려하여 보다 적절한 탐색시간을 계산하여야 한다. 제안된 알고리즘이 실제 환경에서 적용되기 위해서는, 특정 디스크 또는 컨트롤러 장애 시의 복구를 고려하여 높은 안정성 및 고가용성을 지원하여야 한다. 따라서, RAID-5 모델과 같은 패리티 블록 배치 방안과 효율적 패리티 블록 계산 기법이 연구되어야 한다. 또한, 처리 시간 향상을 위해 캐시 메모리가 추가된 다중 디스크배열 시스템의 지원 방안에 대한 연구도 필요하다.

참 고 문 헌

[1] C. L. Elford and D. A. Reed, "Technology Trends and Disk Array Performance," *Journal of Parallel and Distributed Computing*, Vol 46, pp. 136-147, 1997

[2] D. A. Patterson, G. Gibson, and R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks(RAID)," *Proceedings of the Conference on Management of Data*, pp. 109-116, 1988.

[3] F. A. Tobagi, J. Pang, R. Baird, and M. Gang,

"Streaming RAID™ - A Disk Array Management System for Video Files," *Proceedings of the Conference on Multimedia '93*, pp. 393-400, 1993.
[4] P. M. Chen, E. K. Lee, G. Gibson, R. Katz, and D. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, Vol. 26, No. 2, June 1994.
[5] M. Schulze, G. Gibson, R. Katz, and D. Patterson, "How Reliable is a RAID?," *Procedures of the IEEE Computer Society International Conference(COMPCON89)*, March 1989.
[6] Spencer W. Ng., "Crosshatch Disk Array for Improved Reliability and Performance," *Proceedings of the 1994 International Symposium on Computer Architecture*, pp. 255-264, April 1994.
[7] R. Jain, K. Somalwar, J. Werth, and J. C. Browne, "Heuristics for Scheduling I/O Opera- tions," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 3, pp.310-320, March 1997.
[8] C. S. Lee and T. M. Parng, "Performance Modeling and Evaluation of a Two-Dimens- ional Disk Array System," *Journal of Parallel and Distributed Computing*, Vol 38, pp.16-27, 1996.
[9] ANSI, *"Information Technology - SCSI-3 Architecture Model (SAM),"* 1996.
[10] http://www.quantum.com/downloads/pdfs/atlas_10kII.pdf
[11] V. Catania, A. Puliafito, S. Riccobene, and L. Vita, "Design and Performance Analysis of a Disk Array System," *IEEE Transactions on Computers*, Vol. 44, No. 10, pp.1236-1247, October 1995.
[12] A. Merchant and P. S. Yu, "Analytic Modeling and Comparisons of Striping Strategies for Replicated Disk Arrays," *IEEE Transactions on Computers*, Vol. 44, No. 3, pp.419-433, March 1995.
[13] Y. Rompogiannakis, G. Nerjes, P. Muth, M. Paterakis, P. Triantafillow, and G. Weikum, "Disk Scheduling for Mixed-Media Workloads in a Multimedia Server," *Proceedings of the 6th ACM International Conferences on Multimedia*, pp.297-302, 1998
[14] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID Hierarchical Storage System," *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, pp.96-108, 1995.
[15] 김종훈, 엄세웅, 노삼혁, 원유헌, "스트라이핑이 소프트웨어 RAID 파일시스템의 성능에 미치는 영향," *정보과학회논문지(A)*, 제 25권 제 5호, pp.458 -468, 1998. 5.
[16] 김정녀 외, "UNIX SVR4 MP 환경 하에서 다중처리 기능 검사 도구," *'93 가을 정보과학회 학술발표논문집*, 제 20권 2호, pp.539-542, 1993.
[17] http://samba.anu.edu.au/samba
[18] http://www.adaptec.com/products/overview/scsi3950u2.html
[19] http://www.emc.com/products/enterprise_storage_systems/symm5000/
[20] http://www.mhbizlink.com/ccreseller/Content/1997/05-07/f02_features.html

정 경 진

1995년 중앙대학교 전자계산학과 공학사. 1997년 중앙대학교 컴퓨터공학과 공학박사. 1997년 ~ 현재 중앙대학교 컴퓨터공학과 박사과정. 관심분야는 시스템 성능 분석, 멀티미디어 운영체제, 디스크 배열임.

김 성 조

1975년 서울대학교 응용수학과 공학사. 1977년 한국과학기술원 전산과 이학석사. 1977년 ~ 1980년 ADD(연구원). 1980년 ~ 현재 중앙대학교 컴퓨터공학과 교수. 1984년 ~ 1987년 Univ. of Texas at Austin 이학박사. 1987년 ~ 1988년 Univ. of Texas at Austin(Research Fellow). 관심분야는 병렬 및 다중처리, 디버깅, 시스템 망 관리, 멀티미디어, 이동 컴퓨팅임.