멀티미디어 저장 서버에서의 동적 가중 제어를 이용한 디스크 스케줄링*

박은정° 박상수 이수형 장래혁 신현식
서울대학교 컴퓨터공학부
{ejpark,sspark,onlyjazz,naehyuck,shinhs}@cselab.snu.ac.kr

Disk Scheduling with Dynamic Weight Control in Multimedia Storage Servers

Eun-Jeong Park° Sang-Soo Park Soo-Hyung Lee
Nae-Hyuck Chang Heon-Shik Shin
School of Computer Science and Engineering, Seoul National University

요 약

멀티미디어 저장 서버에서는 실시간 클래스와 비실시간 클래스의 두가지 상이한 종류의 요청이 존재한다. 비디오와 오디오 등의 실시간 클래스는 정해진 시간 안에 추출되어야 하며 종료시한에 대한 만족이 보장되어야 한다. 반면 텍스트 데이터 등의 비실시간 클래스는 공정한 서비스와 빠른 응답 시간을 요구한다. 본 논문에서는 두 클래스의 요구사항을 동시에 만족시키기 위하여 각 클래스에 할당된 가중을 동적으로 제어하는 디스크 스케줄링을 제안하였다. 클래스의 가중은 각 클래스에 할당된 디스크 대역폭의 비율을 말한다. 또한 수용제어 알고리즘의 결과를 이용하여 각 클래스의 가중을 동적으로 변화시켜 유휴 대역폭이 효율적으로 이용되도록 하였다. 성능 측정 결과 동적 가중을 이용한 디스크 스케줄링은 비실시간 요청의 평균 응답시간 면에서 정적 가중을 이용한 디스크 스케줄링보다 좋은 성능을 나타낸다.

1. 서론

멀티미디어 기술이 발전함에 따라 저장서버와 네트워크를 통한 대용량 멀티미디어 데이터의 효율적 추출과 원거리 실시간 접근 방법에 대한 연구가 활발히 수행되고 있다. 비디오 또는 오디오 플레이어 등의 멀티미디어 응용들은 주기적으로 멀티미디어 데이터에 대한 서비스를 요청하고, 이들 데이터는 종료시한내에 서비스 되어야만 의미를 갖는 실시간 성격(real-time characteristics)을 나타낸다. 이러한 응용들은 종료시한의 만족과 동시에 보장된 서비스 성능을 제시하여 가능한 많은 사용자를 수용할 것을 요구한다. 반면, 워드 프로세서와 같은 상응형 최선 노력(interactive best-effort) 응용들은 텍스트 데이터에 대한 요청을 비주기적으로 발생시키며, 서비스의 공정성과 빠른 응답시간을 요구한다. 멀티미디어 저장 서버는 응용들의 다양한 성능 요구 사항을 동시에 만족시켜야 한다. 즉, 실시간 요청의 과부하가 비실시간 응용의 성능을 급격히 저하시키거나, 과도한 비실시간 요청들이 실시간 응용의 종료시한 만족에 대한 보장성을 떨어뜨려서는 안된다. 이와 같이 여러 응용을 동시에 지원하기 위해서는 멀티미디어 저장 서버의 디스크 요청을 효율적으로 스케줄링 하는 것이 중요하다. 이를 위해 본 논문에서는 실시간 요청과 비실시간 요청을 실시간 클래스와 비실시간 클래스로 분리하여 각 클래스에 전체 디스크 대역폭 중 사용 가능한 비율인 클래스 가중(class weight)을 할당하였다. 각 클래스에 할당된 디스크 대역폭을 이용하여 다른 클래스의 과부하에 따른 간섭으로부터 보호할 디스크 대역폭 중 사용 가능한 비율인 클래스 가중(class weight)을 할당하였다. 각 클래스에 할당된 디스크 대역폭을 이용하여 다른 클래스의 과부하에 따른 간섭으로부터 보호할 수 있다. 본 논문에서는 각 클래스의 요청들을 스케줄링하는 클래스 내부 스케줄러와 정렬된 각 클래스 큐의 요청을 통합하여 정렬하는 클래스 통합 스케줄러를 통해 2 단계의 스케줄링을 수행한다. 이렇게 2 단계로 나뉘어 동작하는 스케줄링 방식은 클래스 통합 스케줄러에서 각 클래스의 가중을 검사하여 과도한 대역폭의 사용을 금지시킨 후 각 클래스 큐를 통합하여 스케줄링하므로 클래스 간의 간섭을 최소화 할 수 있다. 그러나, 정적 클래스 가중을 이용한 디스크 스케줄링에서는 특정 클래스가 할당된 대역폭을 모두 사용하지 않을 경우 남는 대역폭을 타 클래스에서 충분히 활용할 수 없다는 점에서 비효율적인 측면이 있다. 이를 극복하기 위해 본 논문에서는 실시간 사용자 수의 변화에 따른 필요 대역폭의 변화를 클래스 가중을 통해 제어하는 방법을 사용하였다. 이로써 클래스 가중은 동적으로 조정되고, 그 결과 한 클래스에서 충분히 사용하지 못한 디스크 대역폭은 상대 클래스에서 효과적으로 사용 될 수 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 시스템 모델과 큐잉 메커니즘을 소개하고 3 장에서는 성능평가 결과를 분석한다. 그리고 4장에서는 결론을 제시하여 본 논문을 마친다.

2. 시스템 모델과 큐잉 메커니즘

본 논문은 실시간 요청과 비실시간 요청을 동시에 지원하는 멀티미디어 저장 서버를 대상으로 한다. 논문의 스케줄러 모델은 두가지 과정을 전제로 한다.

* 본 연구는 한국과학재단(과제번호: 97-0100-09-01-3)에 의해 수행되었습니다.

첫째, 저장 서버는 라운드 단위 방식으로 처리된다. 정해진 시간의 라운드 단위로 멀티미디어 사용자의 디스크 요청을 생성하고, 각 라운드의 시작시간에 해당 라운드의 모든 실시간 요청을 발생시킨다.

둘째, 저장 서버는 멀티미디어 사용자의 서비스 요구에 대해 수용제어 과정을 수행한다. 수용제어 알고리즘은 새로운 서비스 요청을 받으면 이미 서비스 받고 있는 사용자의 QoS(Quality of Service) 를 낮추지 않는 범위하에 새로운 요청을 받아들인다. 새로운 사용자 요청의 서비스 시간과 이미 수용된 사용자 요청들의 서비스 시간의 합인 총 서비스 시간을 기준으로 수용 여부를 판단한다.

이러한 전제를 바탕으로 설계한 멀티미디어 저장 서버의 디스크 스케줄러 모델은 그림1과 같다. 그림과 수식에서 사용할 기호는 표 1에 정리하였다.

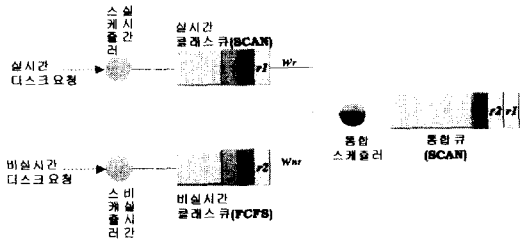

그림 1 :디스크 스케줄러 모델

실시간 요청의 요구 조건인 종료 시한의 만족과 높은 디스크 이용율을 위해서는 SCAN-EDF 스케줄링 방식이 적절한 것으로 연구된 바 있다[1]. 앞서 설명한 바와 같이 본 논문의 스케줄러 모델은 라운드 단위 방식과 수용제어를 전제하므로, 각 라운드마다 모든 실시간 요청의 종료시한은 라운드 종료시간이 되고, SCAN-EDF는 SCAN 방식으로 귀결된다. 비실시간 요청들은 비주기적으로 발생하고, 빠른 응답 시간과 기아의 한계 (starvation bound) 를 요구한다. 그러므로, 디스크 스케줄러는 공정한 서비스를 통하여 비실시간 요청의 기아를 방지하고, 디스크의 이용률을 높여야 한다. 이를 위해 비실시간 스케줄러는 비실시간 클래스 큐를 FCFS로 정렬하여 공정한 처리를 담당하고 통합 스케줄러는 통합큐를 SCAN으로 정렬하여 디스크 이용률을 높인다. 1단계에서 정렬된 두 클래스 큐의 요청들을 대상으로 통합 스케줄러는 2단계 스케줄링을 통해 통합큐를 구성한다. 통합 스케줄러가 클래스 i 의 요청 r_i 를 클래스 큐에서 통합 큐로 이동시키려면 먼저 해당 클래스가 과도한 서비스 시간을 이용하는지 검사하는 과정을 거쳐야 한다.

통합 스케줄러는 r_i 의 서비스 시간 s_i 인하여 클래스 i 의 총 서비스 시간 τ_i 가 한라운드 길이 R 중 점유하는 비율이 클래스 가중 w_i 를 넘는지 여부를 검사한다. 즉 라운드 시간 R 동안 클래스 i의 요청들은 $R \times w_i$ 의 시간을 사용할 수 있다. 점유 비율이 w_i 를 넘지 않으면 해당 요청은 통합 큐로 이동된다. 만약 τ_i 가 $R \times w_i$ 보다 크다면, 해당 요청은 클래스 큐에서 대기하게 되고 이 요청이 최종 요청 큐로 이동 가능한 시점은 현재 라운드에서 디스크 휴지(disk idle) 상태가 되었을 때 또는 다음 라운드 시작시 다시 통합 디스크 스케줄러에 의해 검사된 후가 된다.

디스크 스케줄러 모델에서의 작업을 세분해 보면, 먼저 실시간 클래스의 총 서비스 시간에 대한 검사가 수용제어 과정에서 이루어진다.

기호	정의
R	저장 서버의 라운드 길이
w_i	클래스 i의 가중 ($0 \le w_i \le 1$, $\sum w_i = 1$)
W_i	클래스 i의 최대가중 ($0 \le w_i \le 1$, $\sum w_i = 1$)
r_j	디스크 요청 j
l_j	라운드 길이 R동안 디스크 요청 r_j를 위해 추출된 양 (byte)
t_{seek}	디스크 최대 탐색 시간
t_{rot}	디스크 최대 지연 시간
R_{disk}	디스크 전송율 (byte/sec)
B	블록 크기 (byte)
s_j	디스크 요청 r_j의 서비스 시간
τ_i	한 라운드 동안 클래스 i의 요청에 소요된 서비스 시간

표 1 : 사용 기호의 정리

수용제어 과정에서는 새로운 실시간 사용자의 요청을 포함한 실시간 클래스의 요청들에 대해 총 서비스 시간을 계산하여 $R \times W_r$ 의 시간 내에서 서비스 가능한지 검사한다. 총 서비스 시간의 계산은 다음의 식과 같다[2].

$$\tau_r = 2 \times t_{seek} + \sum_{j=1}^{n} \left\lceil \frac{l_j}{B} \right\rceil \times t_{rot} + \sum_{j=1}^{n} \frac{\left\lceil \frac{l_j}{B} \right\rceil \times B}{R_{disk}}$$

계산된 실시간 클래스의 총 서비스 시간을 바탕으로 새로운 동적 가중 w_r 과 w_{nr} 을 다음과 같이 계산할 수 있다.

$$w_r = \frac{\tau_r}{R} \quad (w_r \le W_r) \quad , \quad w_{nr} = 1 - w_r$$

실시간 클래스에서 초기에 할당한 디스크 대역폭을 완전히 사용하지 않을 경우, w_r 을 동적으로 조절하여 유휴 대역폭을 비실시간 클래스에서 효율적으로 이용하도록 하였다. 계산된 w_{nr} 은 비주기적으로 도착하는 비실시간 클래스의 요청에 대한 검사에 사용되고, 비실시간 클래스의 총 서비스 시간에 대한 검사 후 통합 큐로의 삽입이 결정된다. 비실시간 클래스의 계산 과정은 다음과 같다.

$$S_j = t^{j}_{pre(seek)} \pm \left\lceil \frac{l_j}{B} \right\rceil \times t_{rot} + \frac{\left\lceil \frac{l_j}{B} \right\rceil \times B}{R_{disk}}$$

$$\tau_{nr} = \tau_{nr} + S_j$$

s_j 는 새로 도착한 비실시간 요청의 서비스 시간을 나타내고 이 값을 더하여 새로운 τ_{nr} 이 계산된다. 새로운 비실시간 클래스의 총 서비스 시간에 대하여 $\tau_{nr} \le R \times w_{nr}$ 의 식을 검사한 후 통합 큐로의 삽입이 결정된다. 위에서 소개한 동적 가중 제어를 통한 디스크 스케줄링 동작 과정의 예는 그림 2 에서 볼 수 있다. (a)의 정적 가중을 이용한 디스크 스케줄링의 경우, 서버에서 지정한 초기의 클래스 가중 w_r 은 고정되어 있고 실제 실시간 클래스의 부하는 그보다 적다. 실시간 클래스의 남은 대역폭을 비실시간 클래스의 요청이 이용하기 위해서는 현재 라운드의 모든 실시간 요청이 서비스 된 후 디스크 유휴시간을 이용해 비실시간 요청 $r4$와 $r5$가 차례로 한 요청씩 서비스된다. 한 요청씩 서비스 되므로 요청 $r4$부터는 FCFS의 순서로 스케줄링 된다. 이 예에서 $r6$ 는 현재 라운드의 종료시간 내에

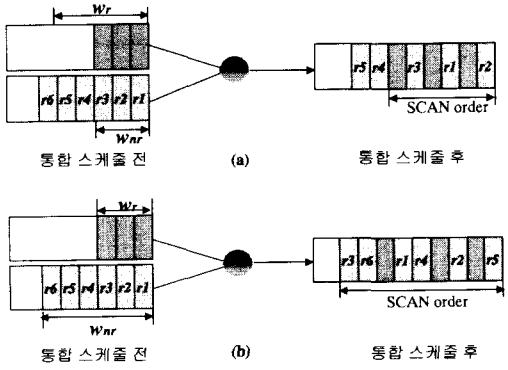

그림 2 : 스케줄링 결과 (a) 정적 가중 (b) 동적 가중

서비스 되지 못하는 경우이다. 이러한 순서는 디스크 헤드의 이동을 증가시켜 디스크 탐색 시간이 증가한다. 이에 비해 (b)의 동적 가중을 이용한 디스크 스케줄링에서는 수용 제어에서 미리 조정된 W_r을 적용하여 상대적으로 W_{nr}은 늘어나게 된다. 이에 따라 많은 수의 비실시간 요청이 SCAN 순서로 스케줄링 되고, 탐색시간이 줄어듦에 따라 현재 라운드에서 서비스 할 수 있는 요청의 수도 늘어나게 된다.

3. 성능평가

동적 가중 제어를 이용한 디스크 스케줄링의 성능을 평가하기 위하여 본 논문에서는 두 가지 실험을 하였다. 먼저, 범용 운영 체제인 리눅스의 디스크 스케줄링과 가중 제어를 이용한 디스크 스케줄링의 성능을 비교하였다. 실험에서 멀티미디어 저장 서버의 라운드 길이는 1초이고, 실제 비디오 파일을 대상으로 하였다. 비실시간 사용자 요청은 평균 800ms의 도착간 시간을 가지는 지수분포를 따르고 요청의 크기는 32KB를 평균값으로 갖는 정규분포를 따른다. 본 실험에서는 실시간 사용자 8명을 대상으로 하였으며 가중을 이용한 디스크 스케줄링의 비실시간 초기 가중 W_{nr} 은 0.3 이다

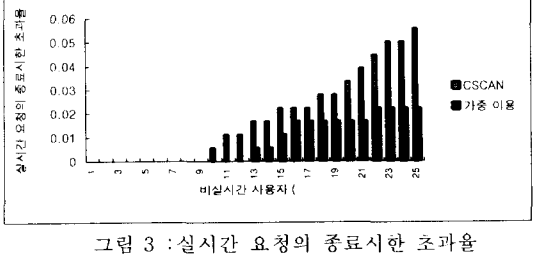

그림 3 :실시간 요청의 종료시한 초과율

또한 본 논문에서는 예약된 대역폭에 비해 적은 실시간 요청이 요구될 경우, 비실시간 요청의 평균 응답시간을 감소시키기 위해 동적 가중을 이용하는 디스크 스케줄링을 제안하였고, 이를 이를 실험한 결과가 그림 4에 나타내었다. 이 실험에서 실시간 요청은 1초마다 도착하고, 비실시간 요청은 평균 800ms의 도착간 시간을 가지는 지수 분포를 따른다. 실시간 요청은 실제 비디오 파일을 대상으로 하고, 비실시간 요청의 크기는 평균 32KB의 정규분포를

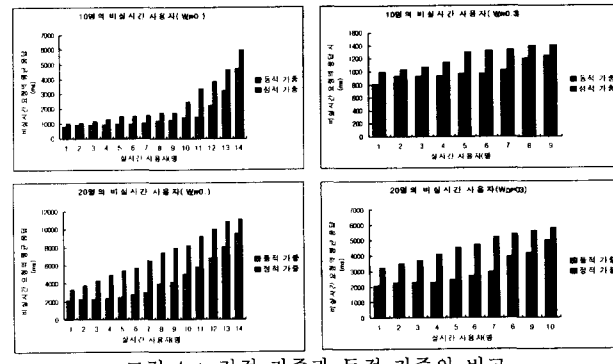

그림 4 : 정적 가중과 동적 가중의 비교

따른다. 실험은 1000개의 비실시간 요청을 대상으로 하였고, 결과는 그림 4 와 같다. 실험 결과에 따르면 동적 가중 제어를 이용한 디스크 스케줄링의 경우 평균 응답 시간은 정적 가중 제어의 디스크 스케줄링보다 서서히 증가함을 볼 수 있다.

4. 결론

멀티미디어 저장 서버의 실시간 요청은 종료시한의 만족을 요구하며 보장된 성능을 바탕으로 가능한 많은 사용자를 수용할 것을 목표로 한다. 이와 달리 비실시간 요청은 공정한 서비스와 빠른 응답시간을 요구하고, 기아의 방지를 위한 지원을 필요로 한다. 이에 본 논문에서는 디스크 요청을 실시간과 비실시간의 두 가지 클래스로 분류하고, 각 클래스가 사용할 수 있는 디스크 대역폭인 클래스 가중을 할당하였다. 요청들은 먼저 각 클래스 큐 내에서 클래스의 특정 요구조건을 만족하도록 정렬되고 통합 스케줄러에 의해 해당 클래스 가중의 한도내에서 통합큐로 삽입되어 서비스된다. 가중을 이용한 디스크 스케줄링은 범용 디스크 스케줄링에 비해 실시간 요청의 종료시한 초과율 면에서 좋은 성능을 나타내었다. 그러나, 정적 가중을 이용한 디스크 스케줄링은 멀티미디어 서버에 수용제어기법을 적용할 경우 실시간 클래스의 여분의 대역폭을 비실시간 클래스에서 효율적으로 이용하지 못한다. 본 논문에서는 실시간 클래스의 부하를 반영하는 동적 가중을 이용하여 비실시간 클래스에서 유휴 디스크 대역폭을 효율적으로 이용하게 하였다. 동적 가중을 이용한 디스크 스케줄링은 수용 제어의 결과를 클래스 가중으로 변환하여 비실시간 클래스의 평균 응답 시간을 줄일수 있다.

5. 참고 문헌

[1] C L. Liu and J W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 30:47-61, 1973

[2] N. Kim, J. Kim, and H. Shin. Statistical admission control algorithm with capability to control transient overload in multimedia storage server. T he 25th Korea Information Science Society Conference, pages175-177, 1999

[3] H. M. Vin, P.Goyal, Alok Goyal, and Ansh uman Goyal. A statistical admission control algorithm for multimedia server. In *Proceedings of ACM Multimedia*, 1994

[4] P. J. Shenoy and H. M. Vin. Cello: A disk scheduling framework for next generation operating systems. In *Proceedings of ACM SIGMETRICS*, June 1998

[5] E. Park, N. Kim, S.Park, J.Kim, and H.Shin. Dynamic dsk scheduling for multimedia storage servers. IEEE Region 10 Conference, 1999