멀티미디어 화일을 위한 BSD 디스크 스케쥴링의 개선

김형수, Felix M. Villarreal
서강대학교 전자계산학과

An Enhanced Method of BSD Disk Scheduling For the Multimedia File

Hyoung Soo Kim, Felix M. Villarreal
Dept. of Computer Science, Sogang University

요 약

본 논문에서는 디스크상에 저장되는 멀티미디어 파일들의 반응시간의 특성을 반영하여 멀티미디어 데이터가 요구하는 실시간적(real-time) 특성을 고려하는 디스크 스케쥴링을 구현하여 보았다. 기존 운영체제의 디스크 스케쥴링 방식은 멀티미디어 파일들에 대한 처리에 있어서, 멀티미디어 파일들이 가지는 시간적인 특성을 고려하지 않는 문제를 가지고 있다. 본 논문에서는 디스크 읽기 요청 발생시에 각 요청의 처리가 이루어져야할 시간의 기한인 마감시간(deadline)을 설정하고, 가장 급한 마감시간을 가지는 요청들을 먼저 처리하되 같은 마감시간을 가지는 요청들에 대해서는 디스크 헤드의 이동방향상에 있는 요청들을 먼저 처리하게 하는 알고리즘을 적용한 디스크 스케쥴링을 구현했다. 요청들이 마감시간을 넘겨서 처리되는 비율과 초과시간을 평가 지수로 하여서 기존의 BSD 디스크 스케쥴링에 비해 개선된 점을 살펴보았다.

1. 서 론

기존 운영체제에서는 디스크상의 멀티미디어 데이터에 대한 I/O에 대해서 일반적인 디스크 I/O와 구분없이 동일한 작업으로 취급하여 디스크 스케쥴링이 일어나기 때문에 멀티미디어 데이타의 특성을 살릴 수 없다[2,4]. 즉, 기존의 운영체제에서 디스크 스케쥴링의 목적은 디스크상에서 찾기 시간을 줄이고 모든 요청들이 처리될 수 있도록 하는 것일 뿐이며, 멀티미디어 데이터의 실시간적 특성을 위해 필요한 시간적인 고려는 전혀 없다[3,6]. 이러한 문제점을 해결하기 위해서 디스크 스케쥴링시 디스크 입출력 요청에 대한 시간적인 특성이 반영되도록 하여야 한다. 이 보고서에서는 4.4BSD-lite에 기초하여 intel80386 이상의 기종에서 동작하는 UNIX 운영체제인 FreeBSD에서의 디스크 스케쥴링 알고리즘을 분석하고, 여기서 멀티미디어 데이타에 대한 디스크 읽기 요청시 생기는 문제점을 살펴보았다. 이때 디스크 요청들에 대해 그 요청이 처리되어야 할 시간의 기한인 마감시간(deadline)을 설정하여서 이 마감시간을 최대한 지킬 수 있도록 하는 디스크 스케쥴링의 개선을 이루고자 한다. 가장 급한 마감시간을 가지는 디스크 요청들을 먼저 처리하되, 같은 마감시간을 가지는 요청들에 대해서는 디스크 헤드가 움직이는 방향의 순서대로 처리하여 찾기 최적화를 이루는 SCAN-EDF 알고리즘을 적용하여서 FreeBSD 디스크 스케쥴링을 개선하고자 한다. 마감시간을 초과하여 끝나는 요청들의 비율이 전체 요청중에 어느 만큼을 차지하는가의 비율과 평균적인 마감시간 초과시간을 평가지수로 삼아 기존의 스케쥴링기법과 비교해 보았다.

2. FreeBSD 디스크 스케쥴링

FreeBSD 3.0 커널은 디스크 읽기 요청에 대해서 디스크 장치 구동기에서 호출되는 bufdisksort() 루틴을 사용하여 디스크 읽기 요청들의 처리 순서를 결정한다. 디스크 장치 구동기는 이른바 엘레베이터 정렬 알고리즘(elevator sorting algorithm)을 사용하여, 디스크 헤드가 이동하는 방향의 순서로 버퍼 큐(buffer queue)에 정렬시킨다. 이 알고리즘에 의해

서 요청들은 실린더 번호 오름차순이며 회전적인 순서(cyclic order)로 정렬되어, 요청들은 한 방향 스캔(one-way scan)으로 처리된다 (그림 1).

bufqdisksort()알고리즘은 요청들을 두 개의 큐로 유지되는 버퍼큐(buffer_queue)에 정렬시킨다 각각의 큐는 실린더 번호의 오름차순으로 정렬된다 지금 들어온 요청이 현재 디스크

```
bufqdisksort(bufq,bp)

  buf_queue_head  * bufq
  buffer          * bp

{
  if (queue = empty) {
    place the request at the tail of 1st queue;
    return;
    }

  /* request lies before the first active request) */

  if(request's cylinder is behind that of 1st request in 1st queue)
    locate the beggining of the 2nd request queue;
    sort request into the 2nd request queue;
    }

  else    /* request is at/after the current request */
    sort in the 1st request queue;
}
```

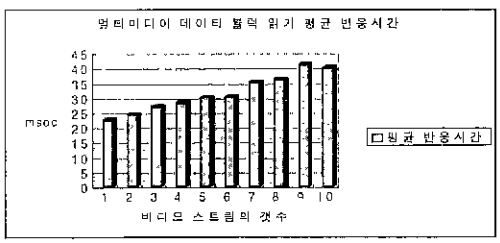
<그림 1> FreeBSD 디스크 스케줄링 알고리즘

헤드가 움직이는 방향상에 있으면, 이 요청은 현재 서비스되는 큐에 정렬되어 삽입된다. 이 큐는 buffer queue head의 insert point에 의해 가리켜지는 큐이다. 만일 이미 디스크 헤드가 지나간 실린더에 대한 요청이라면 다음번에 처리될 큐에서 정렬되어 삽입된다. 이 큐는 buffer queue head의 switch point에 의해 가리켜진다 현재 서비스 되는 큐의 끝에 도달하면 두 개의 큐에 대해서 insert point와 switch point가 가리키는 큐를 바꿈으로서 다시 새로운 큐에 대한 서비스를 시작한다. 이렇게 FreeBSD 3.0에서는 반복적으로 두 개의 큐에 블럭에 대한 오름차순의 요청들을 유지함으로서 한 방향 SCAN 디스크 스케줄링(one way disk scheduling), 즉 C-SCAN을 구현한다.

3. FreeBSD 디스크 스케줄링의 문제점

3.1. 멀티미디어 프로세스의 디스크 읽기 요청에 대한 시간적인 고려를 하지 않음

그림 2는 동시에 실행되는 동영상 재생 프로세스 및 텍스트 파일을 읽어들이게 하는 프로세스의 갯수를 늘려가면서 수행할 때 디스크상의 멀티미디어 데이터 블럭 읽기 요청에 대한 반응시간(response time)을 나타낸 것이다.



<그림 2> 멀티미디어 데이터 블록 읽기 평균 반응시간

여기서 반응시간은 읽어들여야 할 블럭에 대한 읽기 요청이 발생해서, 그 요청에 대한 디스크 블럭을 찾아서 데이터를 읽어 들이기 까지의 시간을 말한다

실행되는 멀티미디어 프로세스의 수가 증가할수록 멀티미디어 데이터 블록을 읽어들이는 평균 반응시간이 증가하고 있음을 알 수 있다 FreeBSD에서는 디스크 읽기 요청들을 그 디스크 블록 번호에 따라 C-SCAN알고리즘으로 정렬하여 처리하기 때문에 멀티미디어 데이터 읽기 요청이라 하더라도 먼저 처리하지를 않는다. 프로세스의 수가 증가할수록 평균 반응시간은 더 길어지므로 멀티미디어 프로세스의 실시간적 특성은 점점 지키기 힘들어진다.

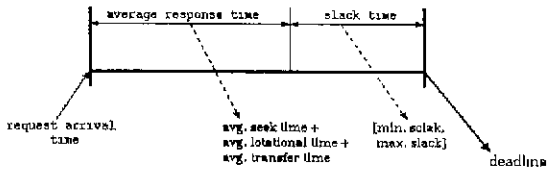
4. 제안하는 디스크 스케줄링

4.1. 마감시간의 설정

디스크 스케줄링에 있어서 마감시간이란 디스크 읽기 요청이 완료되어야하는 시간의 기한이다[5]. 본 실험에서는 디스크 읽기 요청에 대해 다음과 같이 마감시간을 설정하였다

마감시간 = 디스크 읽기 요청 발생시간 + 평균 반응시간 + slacktime

디스크 읽기 요청 발생시간은 읽어들일 데이터 블록에 대한 요청이 최초로 생성되는 시간을 말한다. 평균반응시간은 요청에 대해 디스크 제어기가 실질적으로 데이터 블록을 찾아서 읽어들이는 시간의 평균이다. slacktime은 디스크 읽기 요청이 평균적인 반응시간내에 끝날 때 여유 시간으로서 [*minslack,maxslack*]의 사이에서 정해진다(그림 3).



<그림 3> 마감시간

4.2. 제안하는 스케줄링 방식

FreeBSD 디스크 스케줄링 방식에 마감시간을 고려하는 SCAN-EDF 알고리즘을 적용하였다 SCAN-EDF는 찾기 시간의 최적화(SCAN)와 EDF(Earliest Deadline First)의 장점을 조합한 기법이다[1,2].

```
/* SCAN-EDF algorithm */
if(disk read request occurs)
    check deadline of current request.
DL  for(first item of buffer queue
    deadline of current request ≥ deadline of an item of buffer queue
    get next item of buffer queue)
      compare deadline of an item of buffer queue with that of current request
  if(deadline of current request == deadline of an item of buffer queue){
    deadline of current request +=(Bi/Bmax); /* Bi  block # of current request
    goto DL; }                               Bmax  block # of the last block of disk */
  insert current request after the last item compared (in the buffer queue)
  return.
```

<그림 4> SCAN-EDF알고리즘

가장 빠른 마감시간을 가지는 디스크 읽기 요청을 먼저 처리하되, 여러개의 디스크 요청이 같은 마감시간을 가지고 있

다면 이들 디스크 요청들은 블록 위치에 따라 새로운 마감시간을 설정 받는다. 새로운 마감 시간은 가장 안쪽 블록에 위치하는 요청일수록 더 조금 증가하게된다. 즉 탐색 패턴의 최적화(찾기 시간의 최소화)를 이룬다(그림 4)

4.3. 구현한 디스크 스케줄링의 성능평가

두 가지의 평가지수를 통해서 디스크 스케줄링의 개선을 살펴보았다[5].

- 평균 마감시간 초과시간(tardy time) : 마감시간을 넘긴 디스크 읽기 요청의 실제 종결시간과 마감시간의 차이

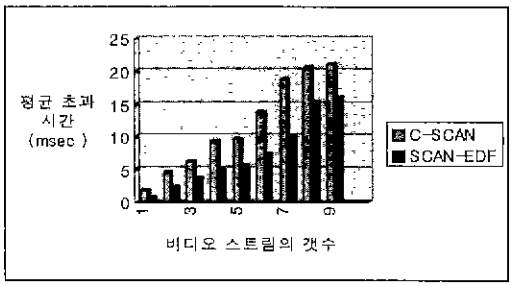
  초과시간 = 실제 요청이 끝난 시간 - 마감시간

- 분실율(miss ratio) : 전체 디스크 읽기 요청중 마감시간을 넘긴 디스크 읽기 요청의 비율

  분실율 = (마감시간을 초과한 요청들/전체 발생한 디스크 읽기 요청들) × 100 (%)

위의 평가지수는 모든 디스크 읽기 요청들이 마감시간을 지킬 수는 없다는 가정하에서 살펴보는 것이다. 실제로 마감시간이 설정된 디스크 요청들이 모두 마감시간내에 처리되지 않기 때문에, 현재의 디스크 스케줄링에서 생기는 평균 마감시간 초과시간과 분실율을 낮춤으로서 디스크 스케줄링을 개선을 살펴보는 것이 목적이다

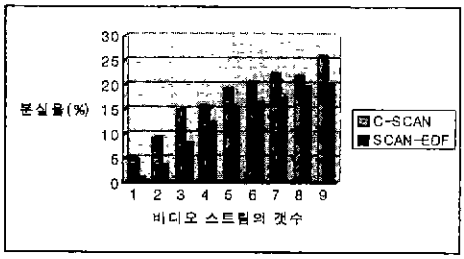
그림 5, 6은 minslack=10msec, maxslack=20msec인 경우이다 실험에 사용한 디스크의 평균 반응시간은 22 msec이므로 디스크 읽기 요청들은 32msec. ~ 42 msec.의 마감시간을 가지게 된다. 나중에 발생하는 디스크 읽기 요청일수록 반응시간이 길어지는 경향을 보이므로(그림 2), 더 여유있는 마감시간을 가지도록 하였다. 동시에 다수의 비디오 스트림을 재생시키면서 마감시간 초과시간의 평균과 전체 요청의 횟수에 대한 마감시간을 초과한 요청들의 비율을 살펴보았다.



<그림 5> 기존의 디스크 스케줄링과 제안된 디스크 스케줄링 적용시 디스크 읽기 요청들의 평균 마감시간 초과 시간의 비교

그림 5는 디스크 읽기 요청들이 마감시간을 넘어서서 처리되었을 때 평균 초과시간의 결과이다. SCAN-EDF의 경우 기존의 C-SCAN 디스크 스케줄링에 비해 더 나은(초과 시간이 작은)결과를 보였다. 디스크 읽기 요청을 하는 비디오 스트림의 개수를 늘려가면서 관찰하였다 처리해야 할 요청의 빈도가 높아질수록 마감시간을 초과하는 정도가 높아짐을 알 수 있다

그림 6은 전체 디스크 읽기 요청 발생 횟수에 대해서 마감시간을 넘겨서 처리된 요청들의 비율을 나타낸 것이다 처리해 주어야 할 디스크 읽기 요청의 발생 비율이 높을수록 마감시간을 넘겨서 처리되는 요청들의 비율도 높아졌다 이 때



<그림 6> 기존의 디스크 스케줄링과 제안된 디스크 스케줄링 적용시 디스크 읽기 요청들의 평균 분실율의 비교

기존의 스케줄링에 비해서 제안된 기법이 더 낮은 분실율의 결과를 보이고 있다. 즉, 멀티미디어 데이터 블록 읽기 요청이 마감시간을 넘기는 비율이 줄어듬으로써 멀티미디어 프로세스의 실시간적인 특성 손실을 줄일 수 있었다.

5. 결 론

멀티미디어 데이터를 고려할 때 , 기존의 운영체제의 디스크 스케줄링은 모든 요청을 시간적인 특성에 대한 고려없이 처리하는 문제점을 가진다. 이에 대해 멀티미디어 데이터의 실시간 특성을 고려한 새로운 기법을 FreeBSD 디스크 스케줄링에 적용하였다. 디스크 읽기 요청들이 처리되어야 할 마감시간을 설정해 보았으며, 급한 마감시간을 가지는 디스크 읽기 요청들을 먼저 처리하되, 같은 마감시간을 가지는 요청들은 디스크상의 블록번호에 따라 새로운 마감시간을 설정받게 하는 스케줄링을 구현하였다. 요청들이 처리될 때 얼마나 마감시간을 넘겨서 처리되는가에 대한 점을 살펴봄으로써, 수정된 디스크 스케줄링이 기존의 기법에 비해 멀티미디어를 위한 개선된 기법임을 보았다.

6. 참고문헌

[1] A.L.N.Reddy, J.Wyllie, *"Disk Scheduling in multimedia I/O system"*, Proceedings of the conf. on multimedia, p225-233, 1993

[2] P.J.Sheonoy, H.M.Vin, *" Cello : A disk Scheduling Framework for Next Generation Operating Systems"* , Proceedings of ACM SIGMETRICS, 1998

[3] D.J.Gemmel, *"Multimedia Sotrage Servers : A Tutorial " ,Computer Vol28 No5* , pp.40-49, May 1995

[4] B. Ozden, R,Rastogi, *A.Silberschatz, " Research Issues in Multimedia Storage Servers" , ACM computing Surveys,* Dec. 1995

[5] R.K.Abbott, H.G.Molina, *"Scheduling I/O Request with Deadlines:a Performance Evaluation"*, Proc. of Real-Time Systems symp., 1990

[6] H.Jen, Thomas D.C., *"Storage Allocation Policies for Time-Dependent Multimedia Data"* , IEEE Transactions on Knowledge and Data Engineering, Vol.8, No.5, October 1996, pp.855-864