연성 실시간 서비스를 위한 디스크 스케줄링 기법

이용수°, 이승원, 김정원, 정기동
부산대학교 전자계산학과

Disk Scheduling Scheme for Soft Real-Time Service

YoungSoo Lee°, SeungWon Lee, JeongWon Kim, KiDong Chung
Dept. of Computer Science, Pusan National Univ.

요 약

본 연구에서는 연성 실시간 태스크 즉 비디오, 오디오와 같은 태스크의 실시간 적인 특성을 만족시키기 위하여 현재 각광 받고 있는 리눅스 커널을 사용하여 연성 실시간 커널을 개발 하고자 한다. 리눅스 파일 시스템의 디스크 스케줄링은 오디오와 비디오와 같은 연성 실시간 태스크의 실시간 요건을 고려하지 않았다. 본 연구에서는 우선 주기 당 일정 대역폭을 요구하는 멀티미디어 서비스를 지원하기 위해 디스크 대역폭 예약이 가능한 디스크 스케줄링 기법을 제시하여 태스크간 공정한 디스크 서비스를 제공한다.

1. 서론

기존의 범용 운영체제에서는 디스크 스케줄링 알고리즘들이 탐색 시간 및 비용을 줄이고 처리율을 높이며 모든 프로세스에게 공정한 디스크 접근만을 제공하였다. 실 시간적인 특성을 요구하는 VOD (Video On Demand)나 NOD (News On Demand)와 같은 멀티미디어 서비스에는 기존의 디스크 스케줄링 알고리즘들이 적합하지 않다.

현재 멀티미디어 서비스를 위해 고려되고 있는 알고리즘으로는 EDF (Earliest Deadline First)[1], SCAN-EDF[2], GSS(Group Sweeping Scheduling) [3]등이 있다.

위의 디스크 스케줄링 알고리즘들은 모든 시간임계 작업들의 마감시간을 맞추고, 디스크 탐색 시간의 최소화, 그리고 버퍼 요구량을 낮추는 것을 목표로 하였다. 반면 멀티미디어 서비스는 주기 당 일정 수준의 대역폭을 요구하는 연성 실시간 서비스이다. 그러나 위의 알고리즘들은 주기적으로 태스크 당 일정한 수준의 대역폭의 서비스를 제공하지 못하여 태스크 당 대역폭의 할당이 불규칙적이다.

본 논문에서는 태스크간 공정한 디스크 서비스를 제공하고 주기 당 일정 대역폭을 요구하는 멀티미디어 서비스를 지원하기 위해 디스크 대역폭 예약이 가능한 디스크 스케줄링기법을 제시한다.

실시간 태스크는 시간 제약성에 따라서 마감시간 상실이 전체 시스템 성능에 치명적인 영향을 미치는 경성 실시간 태스크와 시스템 성능 저하가 완만한 연성 실시간 태스크로 나눌 수 있다. 이 두 가지 분류 중에서 본 논문에서는 연성 실시간 태스크에 속하는 멀티미디어 태스크를 사용한다.

다음 장에서는 본 연구와 연관된 관련 연구를 살펴보고, 3장에서는 본 논문에서 제시하고 있는 파일시스템의 개략적인 소개와 새로운 디스크 스케줄링 방법과 디스크 대역폭 예약 기법에 대해서 기술한다. 그리고 4장에서는 제시된 기법을 시뮬레이션하여 측정결과를 제시하였고, 5장에서는 이 논문의 결론 및 향후 연구과제에 대해서 언급한다.

2. 관련연구

비 주기적인 실시간 태스크를 스케줄링하기 위한 효과적인 방법으로 마감시간을 고려한 우선순위 할당 정책이 있다[1]. 그러나 우선순위를 할당함에 있어서 마감시간만을 고려 하였기 때문에 과부하 상황에서 대처 하기 어려운

* 본 논문은 1999년도 정보 통신부의 대학 기초 연구 지원 사업에 의하여 연구되었음

단점이 있다.

과부하 상황을 고려한 주기적 실시간 데이터의 스케줄링에 관한 연구가 있다[4]. 그러나 주기 당 다른 대역폭을 요구하는 데이터와는 적합하지 않으므로 새로운 형태의 실시간 스케줄링이 필요하다.

실시간 태스크를 스케줄링 하는데 있어서 태스크의 마감시간 뿐만 아니라 Weighted Fairness를 고려한 SMART (A Scheduler for Multimedia Applications)가 있다[5]. SMART 또한 데이터에 대한 표현 방법이 없고, 스케줄러의 각 요소간 상호관계가 명확하지 못한 단점이 있다.

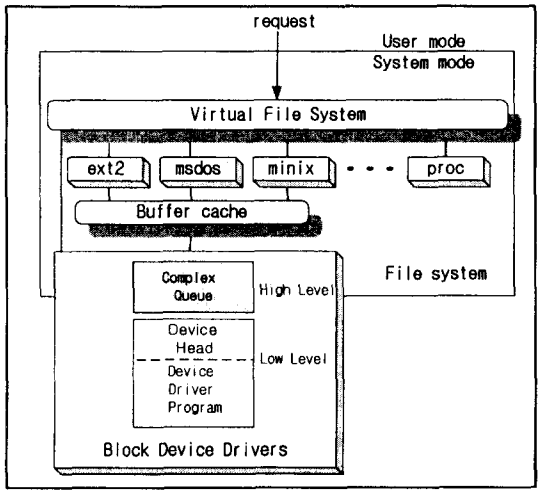
실시간 태스크에 대한 디스크 대역폭에 대한 예약하는 방법으로 통계적인 멀티플랙싱 방법이 있다[6]. 이 방법은 VBR 스트림을 위한 결정적인 보장을 제공하는 방법이다. 단점으로는 디스크 대역폭에 대해서 최적의 예약방법을 제공하지 못한다.

리눅스와 같은 범용 운영체제의 파일 시스템에서는 디스크 스케줄링 방법으로 C-SCAN방법을 사용하고 있다. 본 논문에서는 태스크간 공정한 디스크 서비스를 제공할 수 있는 디스크 스케줄링 기법을 제시하고 멀티미디어 서비스의 주기 당 요구 대역폭을 보장하기 위하여 디스크 대역폭 예약을 지원한다.

3. Complex Queue

3.1 파일 시스템 개관

본 논문에서는 범용 운영 체제인 리눅스 커널 상에서 디스크 대역폭 예약을 통한 연성 실시간 디스크 스케줄링을 위한 파일 시스템을 위하여 아래 그림<그림 1>과 같은 구조를 제안한다.



<그림 1> 연성 실시간 서버를 위한 리눅스 파일시스템

리눅스의 파일 시스템이 지원하는 범위는 리눅스 커널에 의한 단일화된 인터페이스로써 가상 파일 시스템(VFS)에 의해서 가능하다. 가상 파일 시스템은 응용 프로그램을 지원하기 위한 시스템 호출을 제공하고, 내부 구조를 유지하며, 적당한 파일 시스템으로 작업을 전달한다[7].

기존의 블록 디바이스 드라이브에서는 디스크 스케줄링 알고리즘으로 SCAN 방법을 사용한다. 본 논문에서는 커널 소스 /linux/drivers/block/ll_rw_blk.c(블록 디바이스에 대한 모든 read/write을 다루는 파일)을 수정하여 디스크 대역폭 예약을 지원하는 2계층 디스크 스케줄링 기법을 구성하여 멀티미디어 서비스를 지원하도록 한다.

3.2 Complex Queue

연성 실시간 태스크의 요구 디스크 대역폭 예약 및 디스크 스케줄링을 지원하기 위하여 Complex Queue 방법을 사용한다. Complex Queue는 Switching Queue와 C-SCAN Queue로 구성되어 있다. 이 방법은 기존의 디스크 스케줄링 방법에 새로운 계층인 Switching Queue를 새롭게 첨가 하여 각각의 태스크에서 주기별 요구가 있을 때 마다 Switching Queue에 있는 Service table을 참조한다. Service table에는 각각의 태스크 식별자 및 태스크 블록들에 관한 정보들이 있다. Service table를 참조한 결과 서비스가 가능한 태스크는 Switching Queue의 Service Queue로 들어간 다음 그 아래 계층인 C-SCAN Queue로 들어간다. 서비스가 가능하지 못한 태스크는 Switching Queue에 있는 Waiting Queue에 들어가서 다음주기 까지 기다렸다가 위의 과정을 반복한다. C-SCAN Queue에 들어온 요구는 탐색 시간을 최소화 하기 위하여 C-SCAN 디스크 스케줄링 알고리즘을 사용하여 요구들을 서비스 한다.

본 논문에서 제시한 Complex Queue 방법의 디자인 목적은 다음과 같다.

① 태스크들간의 공정한 서비스 기회 부여
② 멀티미디어 태스크를 위해 디스크 대역폭 할당
③ 디스크 대역폭의 동적 할당을 지원

3.2.1 Switching Queue (1 layer)

연성 실시간 태스크는 주기별로 일정량의 디스크 대역폭을 요구한다. <그림 2> 같이 위의 요구를 만족 시키기 위해 Switching Queue에서는 이 요구량을 미리 Service table에 만들어 요구가 있을 때 마다 서로 비교 한다.

각 태스크의 주기 당 보장된 블록 수가 서비스 블록 수 보다 크거나 같으면 Service Queue로 간다. Service Queue에 있는 요구들은 C-SCAN Queue로 가서 C-

SCAN 디스크 스케줄링 알고리즘을 사용하여 태스크 간 공정한 디스크 서비스를 제공한다

반대로 각 태스크의 주기 당 보장된 블록 수가 서비스 블록 수 보다 작으면 Waiting Queue로 들어 가서 대기 상태에서 다음 주기까지 기다린 다음 다시 Service table을 참조하여 위의 과정을 반복한다.

아래의 수식은 위의 과정을 Service table에 있는 정보를 이용하여 나타낸다.

① $SB_i \geq GBPR_i$: Waiting Queue로 들어간다.

② $SB_i < GBPR_i$: Service Queue로 들어간다.

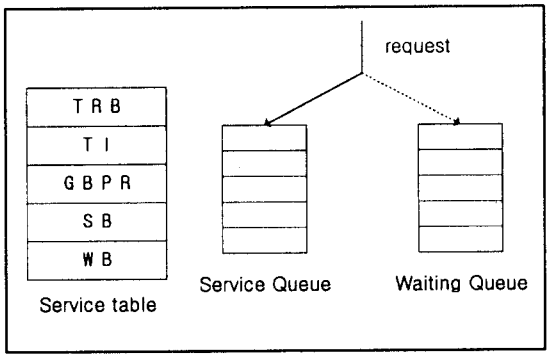
(단, SB_i 는 i 번째 태스크의 서비스 블록 수, $GBPR_i$는 i번째 태스크의 주기 당 보장된 블록 수, [표 1] 참조)

대역폭 할당 기준이 되는 Service table의 정보는 [표 1]에 있다.

[표 1] Service table

TRB (Total Request Blocks)	전체 요구 블록 수 (대역폭)
TI_i (Task Identifier)	태스크 식별자
$GBPR_i$ (Guaranteed Blocks per Round)	주기 당 보장된 블록 수
SB_i Service Blocks of each task	각 태스크의 서비스 블록 수
WB_i Waiting Blocks of each task	각 태스크의 waiting 블록 수

아래 그림이 연성 실시간 태스크의 요구 디스크 대역폭을 예약하는 Switching Queue 모델을 종합적으로 나타내었다.



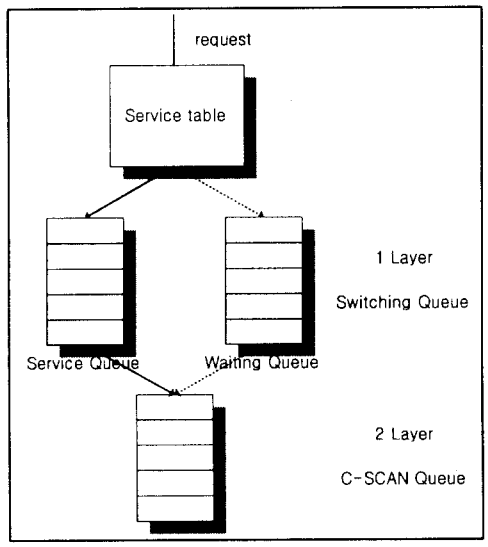
<그림2>Switching Queue

Switching Queue에서는 연성 실시간 태스크 서비스를 위한 태스크 당 주기별 할당된 디스크 대역폭의 요구를 처리하는 큐잉 모델이다.

위 과정을 통해 연성 실시간 태스크의 주기내 디스크 서비스를 보장하게 된다. 따라서 주기내 일정 대역폭을 요구하는 멀티미디어 서비스의 요구를 항상 보장 할 수 있게 된다.

3.2.2 C-SCAN Queue (2 layer)

Switching Queue에서 디스크 대역폭이 예약된 요구들의 블록들의 탐색 시간을 최소화하기 위하여 Switching Queue 아래에 C-SCAN Queue를 만들어서 C-SCAN 디스크 스케줄링 알고리즘을 사용한다. <그림 3> 은 앞에서 본 Switching Queue와 C-SCAN Queue를 합친 Complex Queue을 종합적으로 나타내었다.



<그림 3>Complex Queue

4. 실험 및 결과

3장에서 제시한 Complex Queue에 대해서 다음 2가지로 나누어서 실험을 했다.

첫번째로 C-SCAN 디스크 스케줄링과 본 논문에서 제안한 Complex Queue 방법을 사용했을 때 만족되는 request의 수를 실험을 했다.

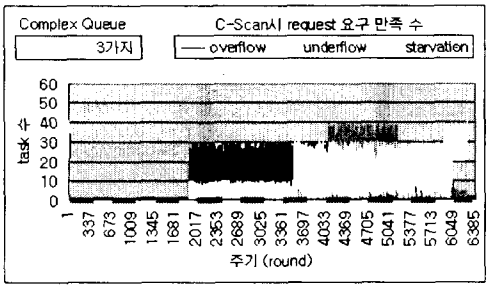
두 번째로 연성 실시간 태스크에서는 응답 시간이 중요하므로 C-SCAN 디스크 스케줄링과 본 논문에서 제안한 Complex Queue 방법 사이의 주기별 소요시간의 차이를 실험을 했다.

실험 환경 및 디스크 파라미터 값은 [표 2]에 있다. 실험은 실제 리눅스 디스크 서비스 구조와 유사한 시뮬레이션 환경을 조성하여 실행 하였다.

[표 2] 실험 환경 및 디스크 파라미터

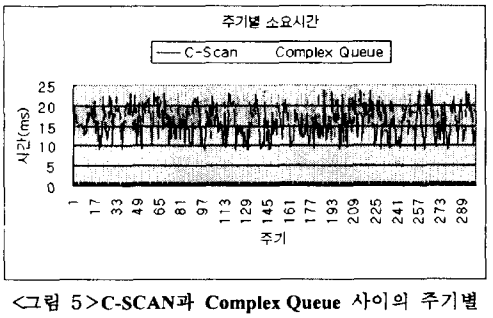
실험 환경	
Session service thread	각각 50, 100
비디오 파일	50개의 50Mbyte
배치	Random block 배치 Clustered block 배치
블록 크기	1Kbyte
주기별 요구 블록 수	4개로 고정
디스크 파라미터	
탐구 시간	5.3 ms
회전 지연 시간	2.99ms
데이터 전송 시간	17.5 to 23.3 MB/sec

<그림 4>는 첫 번째 실험 결과이다. C-SCAN 방법을 사용 하였을 경우, 일정 시간 지나고 나면 overflow, underflow, starvation의 태스크 수가 갑자기 증가하는 것을 볼 수가 있다. 반면 본 논문에서 제시한 Complex Queue 방법은 실험 결과 overflow, underflow, starvation 모두 다 0에 가깝다.



<그림 4>C-SCAN시 request 만족 수

<그림 5>는 두 번째 실험 결과이다. 주기별 소요시간의 차이가 거의 없으므로 기존의 C-SCAN 방법과 본 논문에서 제안한 Complex Queue 방법 각각의 반응 시간 또한 거의 비슷하다.



<그림 5>C-SCAN과 Complex Queue 사이의 주기별 소요시간

5. 결론 및 향후 연구과제

본 논문에서는 주기 당 일정 대역폭을 요구하는 멀티미디어 서비스를 위해 디스크 대역폭의 예약이 가능한 디스크 스케줄링 기법인 Complex Queue을 제안했다. 실험 결과에서 알 수 있듯이 계속해서 주기가 반복 되어도 태스크 간 공정한 디스크 서비스를 제공한다. 그리고 응답 시간 또한 기존의 방법과 거의 차이가 없다.

향후 계획으로는 태스크의 속성에 따라 부호화 속도를 변경할 수 있는 VBR(Variable Bit Rate) 태스크에 대해서도 고려할 것이다. 각 태스크별 응답 시간에 대한 실험 및 다양한 주기별 요구 블록 수를 가지는 태스크들에 대한 실험을 할 계획이다. 또한 리눅스 커널에 포팅하여 실측에 의한 실험을 할 계획이다.

참고 논문

[1] B.Adelberg, H.Garcia-Molina, B.Kao, "Emulating Soft Real-Time Scheduling Using Traditional Operating System Scheduler", April 1994

[2] A. L. N. Reddy and J. Wyllie, "Disk Scheduling in a Multimedia I/O System", In Proceedings of the first ACM International Conference on Multimedia, Pages 225-233, Anheim, CA, 1993

[3] M.S.Chen, D.D. Kandlur, and P. S. Yu , "Optimization of the Group Sweeping Scheduling(GSS) with Heterogeneous Multimedia Streams", In Proceedings of the First ACM International Conference on Multimedia, page 235-241, Anaheim, CA, 1993

[4] Teik Guan Tan, Wynne Hsu, "Scheduling Multimedia Applications under Overload and Indeterministic Conditions"

[5] Jason Nieh, Monica S.Lam, "The Design of SMART : A Scheduler for Multimedia Application", June 1996

[6] Ravi Wijayarane, Narasimha Reddy, "Providing QOS quarantees for disk I/O", Department of Computer Science, Texas A & M University, 2000

[7] R Maguns, U Kunitz, M Dziadzka, D Verworner, M Beck, H Bohme, "LINUX KERNEL INTERNALS", Second Edition