Adaptive Rate Monotonic알고리즘을 이용한 멀티미디어 프로세스 스케쥴링 기법

이지숙, Felix M. Villarreal
서강대학교 전자계산학과

Multimedia Process Scheduling Mechanism with Adaptive Rate Monotonic Algorithm

JiSook Yi, Felix M. Villarreal
Dept. of Computer Science, Sogang University

요 약

본 논문에서는 멀티미디어 프로세스의 특성을 반영한 프로세스 스케쥴러를 설계하고 실제 구현을 통해 성능을 분석하였다. 제안한 프로세스 스케쥴링 기법은 주기가 짧은 프로세스에 높은 우선순위를 부여하지만 우선순위를 결정하기 위한 주기를 이전 작업의 주기에 의해 동적으로 계산하고 프로세스의 수행 중의 중단을 제한함으로써 Rate Monotonic 알고리즘을 동적이고 비중단적으로 수정하였다. 제안한 스케쥴링 기법은 BSD를 기초로 한 운영체제인 FreeBSD 상에서 구현하여 스케쥴러의 성능을 평가하였다. 제안한 스케쥴러에 대한 실험에서 FreeBSD 스케쥴러에 비해 멀티미디어의 실시간적인 특성을 만족하면서 수행 중의 동적인 상황 변화에 적응된 결과를 보인다

1. 서 론

하드웨어 및 데이터 저장 장치 기술의 발전으로 컴퓨터 성능이 향상되고 네트워크에서의 전송기술이 발전하면서 문자나 이미지 정보 이외에 비디오나 오디오 정보 등을 처리하고 전송하는 것이 가능해졌다 이에 따라 멀티미디어 시스템 및 그 응용 프로그램에 대한 연구가 활발히 진행되고 있다.

멀티미디어 데이터는 시간 독립적인 데이터와 시간 종속적인 데이터가 결합된 형태의 데이터로서 기존의 텍스트 기반의 데이터와는 달리 시간적인 연속성을 가지며 일정 시간 이내에 처리되어야만 의미를 갖는 실시간적인 제한을 갖고 있다 따라서 멀티미디어를 지원하기 위한 운영체제에서는 멀티미디어 데이터의 이러한 특성을 반영한 스케쥴링 기법을 필요로 한다 그러나 기존 운영체제는 시스템 전체의 처리율과 프로세스들 간의 자원의 공정한 분배를 목적으로 하며 이를 멀티미디어 프로세스와 일반 프로세스에 동일하게 적용하고 있어 멀티미디어 프로세스에 적합하지 않다

멀티미디어 프로세스의 실시간적인 특성을 만족시키기 위해 실시간 스케쥴링 알고리즘을 적용하는 연구가 진행되어 왔다[1] 그러나 실시간 스케쥴링 알고리즘을 멀티미디어 프로세스에 적용하여 운영체제 상에서 구현하는 데는 어려운 점들이 있다. 멀티미디어 프로세스를 위한 스케쥴링 알고리즘으로 널리 사용되는 Rate Monotonic 알고리즘은 스케쥴링 가능한 작업들의 조건으로 작업들이 완전히 주기적이고 중단 가능할 것을 기정하지만[3], 멀티미디어 프로세스의 경우 작업들의 주기는 일정하지 않으며 범용 운영체제는 비중단적으로 운영된다

본 논문에서는 멀티미디어 프로세스를 지원하기 위한 스케쥴링 방식을 제안하고자 한다. 범용 운영체제의 스케쥴링 알고리즘과 실시간 스케쥴링 알고리즘이 멀티미디어 프로세스를 스케쥴링하는 데 있어 어떠한 문제점이 있는지 제시하고, 그 문제들을 해결하기 위한 방법으로 Rate Monotonic 알고리즘을 동적이고 비중단적으로 개선한 스케쥴러를 제안한다.

2. 멀티미디어 프로세스 스케쥴링

2.1. 멀티미디어 프로세스의 특성

멀티미디어 데이터란 텍스트, 이미지 등의 시간 독립적인 데이터와 오디오, 동화상 비디오 등 시간 종속적인 데이터가 결합된 데이터이며 멀티미디어 프로세스는 멀티미디어 데이터를 처리하는 프로세스이다

멀티미디어 프로세스는 시간 종속적인 데이터들로 인해 실시간적인 특성을 갖고 있다. 즉, 멀티미디어 프로세스는 주기성을 갖고 있으며 주어진 마감시간(deadline) 내에 처리되어야한다. 멀티미디어 프로세스의 각 작업은 작업의 arrival time(a), execution time(e), 마감시간, 데이터 비율(r)에 의해 기술된다[그림 1][4].

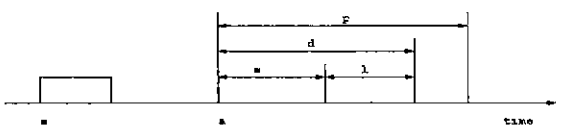

그림 1 멀티미디어 프로세스

2.2. 전통적인 스케줄링 방식의 문제점

범용 운영체제는 시분할(time sharing) 프로세스에 적절하게 설계되어 시분할 프로세스에 높은 우선순위를 할당하고 CPU 이용률이 높은 프로세스의 우선순위를 감소시킴으로써 직업들간에 공정한 지원 사용과 CPU의 이용률(utilization)을 극대화시키는 것을 목적으로 한다[2]. 그러나 멀티미디어 프로세스의 주기적인 특성과 마감시간을 고려하지 않기 때문에 마감시간 내에 작업을 마치지 못하는 결과를 갖게 된다. 또 이미 생성된 멀티미디어 프로세스에 일정한 서비스의 질(Quality of Service)를 보장하기 위한 admission control에 대한 고려가 없다.

이러한 문제점들을 살펴보기 위해 다음과 같은 실험을 하였다. 실험 환경은 Pentium 166MHz의 CPU, 48M의 메모리를 가진 시스템에서 BSD에 기초해 80386이상의 기종을 위해 구현된 FreeBSD 버전 3.0을 사용하였으며 프로그램은 MPEG Player와 fibonacci 39번 째 수를 계산하는 CPU-orient 프로세스, file에 read/write를 하는 프로세스, CC 컴파일러를 사용하였다

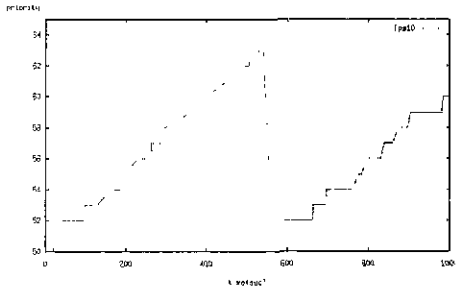

그림 2 멀티미디어 프로세스의 우선순위의 변화

[그림2]은 MPEG player를 수행했을 때의 우선순위 변화를 나타낸 그래프이다. 그림에서 보는 바와 같이 멀티미디어 프로세스는 주기적으로 CPU를 사용하며 CPU를 사용함에 따라 우선순위가 급격히 떨어지게 되는 것을 볼 수 있다. 이러한 이유는 CPU를 많이 사용하는 프로세스의 우선순위를 감소시키는 FreeBSD의 스케줄링 방식 때문이다.

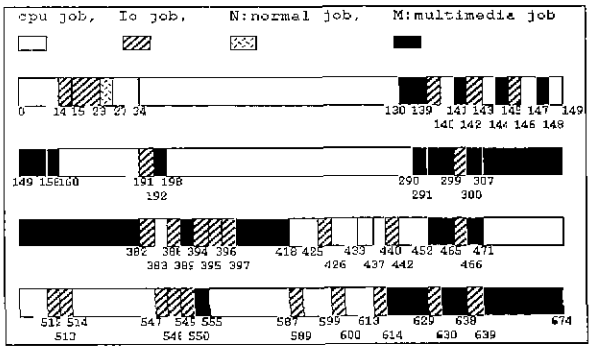

그림 3 프로세스들의 CPU 점유

또 [그림3]은 멀티미디어 프로세스와 일반적인 작업들을 동시에 수행했을 때의 CPU 점유를 나타낸 그림이다. 멀티미디어 프로세스가 갖고 있는 주기적인 특성과 달리 시스템이 과부하된 상황에서는 비주기적으로 수행되게 되는 것을 알 수 있다

2.3. 멀티미디어 프로세스 스케줄러

기존 운영체제의 단점을 개선하여 멀티미디어 프로세스를 지원하기 위한 프로세스 스케줄링 방식은 다음 사항을 고려하여야 한다.

- 멀티미디어 프로세스는 다른 시분할 프로세스보다 높은 우선순위를 가지며 CPU 사용량에 의해 변경되지 않아야 한다
- 이미 생성된 프로세스에 QoS를 보장하기 위한 admission control이 수반되어야 한다.

멀티미디어 프로세스가 갖는 실시간적인 특성으로 인해 멀티미디어 프로세스 스케줄러로 실시간 스케줄링 알고리즘을 적용한 스케줄러가 연구되어 왔다. 실시간 스케줄링 알고리즘으로는 Earliest Deadline First(EDF)[3], Rate Monotonic(RM)[3] 등이 있다

- **Rate Monotonic** 프로세스가 시스템에 요청하는 데이터 비율을 스케줄링의 선택기준으로 이용하여 데이터 비율이 가장 큰 프로세스 혹은 주기가 가장 짧은 프로세스에 높은 우선순위를 할당한다 프로세스들의 이용률의 합이 ln(2)를 초과하지 않을 때 스케줄이 가능하며[3] 이를 admission control에 사용한다.
- **Earliest Deadline First** 큐에 도착된 요청들 중에서 마감시간이 가장 가까운 프로세스에 높은 우선순위를 할당한다. 프로세스들의 이용률의 합이 1을 초과하지 않을 때 모든 프로세스에 대해 마감시간을 놓치지 않는 스케줄링을 제공한다[3].

이 알고리즘들은 모두 우선순위가 높은 프로세스가 우선순위가 낮은 프로세스의 수행을 중단시킬 수 있다.

2.4. Rate Monotonic 알고리즘의 문제점

본 논문에서는 Rate Monotonic 알고리즘을 개선한 스케줄링 알고리즘을 제안하고자 함에 있어 Rate Monotonic 알고리즘이 멀티미디어 프로세스를 지원하는데 있어서 문제점들을 살펴보고자 한다.

Rate Monotonic 알고리즘은 스케줄링 가능한 작업의 조건으로 작업들이 완전 주기적이고 중단 가능할 것을 가정한다[3]. 이러한 가정은 다음과 같은 문제점을 발생시킨다

- **정적인 우선순위 할당** Rate Monotonic 알고리즘은 프로세스의 주기에 따라 정적으로 우선순위를 할당하며 한번 결정된 프로세스의 우선순위는 수행 중 변경되지 않는다. 그러나 멀티미디어 프로세스의 주기는 실시간 프로세스의 주기와 달리 일정하지 않다 프로세스의 우선순위를 정적으로 할당함으로써 CPU에 대한 over/under reservation이 발생한다
- **중단적인 스케줄링** Rate Monotonic 알고리즘은 수행 중인 프로세스보다 높은 우선순위를 가진 프로세스가 도착했을 때 수행을 중단하고 높은 우선순위의 프로세스를 선택하여 수행한다 그러나 일반적인 운영체제는 프로세스가 커널 모드에서 수행 중일 경우에는 임의적인 중단을 허용하지 않는다

3. 제안하는 멀티미디어 프로세스 스케줄러

제안하는 멀티미디어 스케줄러는 BSD 프로세스 스케줄러와 앞절에서 기술한 단점들을 개선한 비중단적이고 동적인 Rate Monotonic 알고리즘을 통합한 스케줄러이다

3.1. 스케줄러의 동작방식

시분할 프로세스를 위한 스케줄러와 멀티미디어 프로세스를 위

한 스케줄러를 분할하여 적용한다. 시분할 프로세스는 기존의 Multilevel Feedback Queue를 사용한 스케줄러를 이용하고 멀티미디어 프로세스에는 Rate Monotonic 알고리즘에 의해 우선순위를 할당하고 프로세스가 자원을 기다리며 sleep 상태로 들어가는 경우 즉 작업의 주기가 끝나는 경우에만 CPU를 다른 프로세스에 넘겨준다[그림 4].

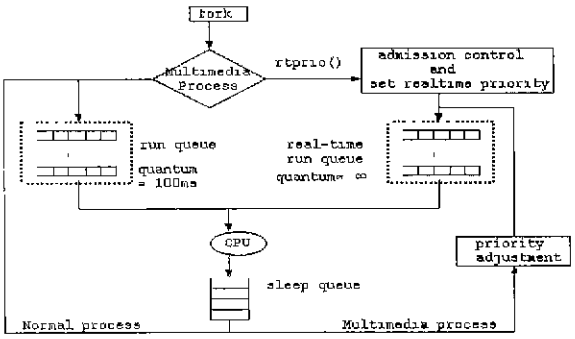

그림 4 제안하는 스케줄러의 동작방식

3.2.채택제어 및 우선순위 할당

멀티미디어 프로세스의 생성은 우선 시분할 프로세스로 생성되었다가 채택제어와 우선순위를 초기화를 거친 후 실시간 큐에 삽입된다

· **채택제어** 시스템의 과부하를 방지하기 위해 CPU 이용률이 다음과 같은 조건을 만족시킬 경우에만 멀티미디어 프로세스를 생성한다 (U CPU 이용률, N: 프로세스의 수, c. quantum, T: 프로세스의 주기)

$$U \le \ln(2) - \max_{0 \le r \le n} c(1/T_1 - 1/T_r) \quad [5]$$

· **우선순위 할당** Rate Monotonic 알고리즘은 정적인 알고리즘이지만 멀티미디어 프로세스의 주기는 데이터에 따라 그 주기가 일정치 않으므로 정적인 우선순위 방식은 적절치 않다. 이를 해결하기 위해 본 논문에서는 멀티미디어 프로세스의 주기를 동적으로 계산하기 위한 방안을 제시한다.

작업의 주기는 이전 작업의 주기의 지수 평균을 구함으로써 추측할 수 있다. CPU 스케줄링에 있어 작업의 주기는 실행 큐에 들어가서 자발적인 환경변환(voluntary context switching) 이후 다시 실행 큐에 들어오기까지의 시간으로 볼 수 있다. C_n을 n번째 작업의 주기, P_{n+1}을 다음 작업의 주기에 대한 추정값이라 할 때, $0 \le \alpha \le 1$인 α에 대해

$$P_{n+1} = \alpha P_n + (1-\alpha) C_n$$

로 정의된다 C_n은 n번째 작업이 sleep queue를 빠져 나온 시간에서 실행 큐에 들어갈 때의 시간을 제함으로써 얻어지며 P_0 와 α 값은 상수로 주어진다 우선순위는 위 식에 의해 얻은 주기에 따라 부여된다.

3.3. 제안한 스케줄러의 성능 평가

제안한 스케줄러의 성능을 평가하기 위한 실험 환경으로 Pentium 166MHz의 CPU, 48M의 메모리를 가진 시스템에서 BSD에 기초해 80386이상의 기종을 위해 구현된 FreeBSD 버전 3.0이 사용되었으며 프로그램은 MPEG Player와 fibonacci 39번째 수를 계산하는 프로그램이 사용되었다

멀티미디어 프로세스와 시분할 프로세스를 동시에 수행했을 때 각 프로세스의 수행 결과인 표 1,2를 살펴보면 본 논문에서 제안한 스케줄러는 멀티미디어 프로세스에 대해 BSD 스케줄러에 비해 수행된 fps가 증가하며 Rate Monotonic 알고리즘과는 비슷한 성능을 보인다[표1]. 시분할 프로세스에 대해서 제안한 스케줄러는 Rate Monotonic에 비해 Turnaround time이나 Waiting time이 감소되어 시분할 프로세스에 대해서도 적절한 서비스를 제공함을 알 수 있다[표2].

	CPU time	WaitingTime	TurnaroundTime	frame/sec
BSD	10883 ms	1107 ms	26060 ms	29.89 fps
RM	10718 ms	0 ms	14900 ms	52.28 fps
Proposed	10776 ms	0 ms	14690 ms	53.03 fps

표 1 멀티미디어 프로세스의 수행 결과

	CPU time	WaitingTime	TurnaroundTime
BSD	33678 ms	6162 ms	63350 ms
RM	33938 ms	7109 ms	61070 ms
Proposed	33373 ms	4603 ms	49063 ms

표 2 시분할 프로세스의 수행 결과

4. 결 론

멀티미디어 프로세스가 갖는 실시간적이고 주기적인 특성을 만족시키기 위해 주기가 짧은 프로세스에 높은 우선순위를 부여하면서 수행 중의 동적인 변화에 적절히 대응하도록 우선순위를 조정하는 방안을 제시하였다. 동적인 주기 계산을 함으로써 CPU 자원이 프로세스에 under/over reservation되는 것을 방지하고 따라서 멀티미디어의 실시간적인 특성을 만족시키는 동시에 시분할 프로세스의 요구 사항을 만족시키게 되었다. 그러나 admission control이나 동적으로 주기를 계산하는 부분에 있어 시간적인 오버헤드가 발생하는 단점을 갖고 있다.

참고 문헌

[1] C. W. Mercer and S. Savage and H. Tokuda, Operating System Support for Multimedia Applications *Proc. of the International Conference on Mulumedia Computing and Systems*, 90-99. May 1994

[2] A Silberschatz and J Perterson and P.Galvin, *Operating System concepts*, Adisson Wesley, 1994

[3] C. L. Liu and J. W. Layland, Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment, *Journal of the ACM*, 20-(1):46-61, Jan 1973

[4] J. Y. T Leung and M L Merrill, A Note on Preemptive Scheduling of Periodic Real-Time Tasks, *Information Processing Letters*, 11(3):115-118, Nov 1980

[5] R. Gopalakrishnan and Gurudatta M Parukar, Bringing Real-time scheduling Theory and Practice Closer for Multimedia Computing, *SIGMETRICS*. 1996