리눅스에서 하반부처리 스케줄링을 이용한 사용자 프로세스의 실행시간 안정화에 관한 연구

정경조[O] 정석간 박찬익
포항공과대학교
{braiden[O], javamaze, cipark}@postech.ac.kr

Stabilizing Execution Time of User Processes by Bottom Half Scheduling in Linux

Kyong Jo Jung[O] Seok Gan Jung, Chanik Park
System Software Laboratory, Pohang University of Science and Technology

요 약

예측할 수 없이 빈번하게 발생하는 인터럽트와 인터럽트 처리시간의 대부분을 차지하는 하반부 처리시간에 의해서 스케줄러는 사용자 프로세스에게 정상적으로 CPU를 할당해 줄 수 없는 이른바 " 빼앗긴 시간 문제" 가 발생하게 된다. 본 논문에서는 이러한 문제를 해결하기 위해서, 하반부들이 사용할 수 있는 최대시간을 동적으로 계산하고, 처리시간을 제한하는 " 하반부 스케줄링" 방법을 제안하고, 제안한 구조를 리눅스에서 구현하고 제안된 구조에 의해서 사용자 프로세스에게 할당된 CPU 시간을 안정화 시킬 수 있음을 멀티미디어 응용을 사용한 실험을 통해서 보이고자 한다.

1. 서 론

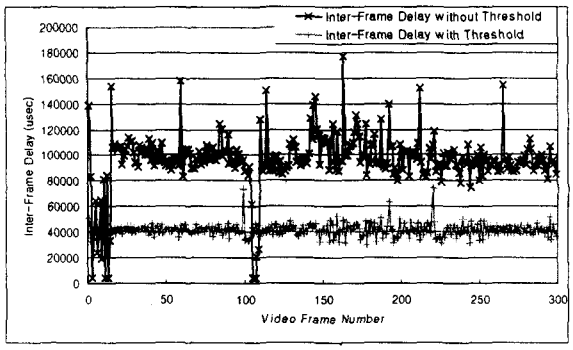
범용 운영체제상에서 인터럽트 처리시간에 의해 발생하는 " 빼앗긴 시간 문제" 는 멀티미디어 응용과 같은 연성 실시간 응용들을 지원하기 위해서 넘어야 할 가장 큰 문제 중 하나이다. 이러한 문제에 의해서 사용자 프로세스에게 안정적으로 CPU를 할당해 줄 수 없게 되며, 따라서, 연성 실시간 응용들의 수행품질을 저하시키게 된다. 리눅스에서는 인터럽트 처리를 두 부분으로 나누어 처리한다. 하나는 " 상반부" (Top Half)로써, PIC(Programmable Interrupt Controller)에게 응답하는 것과 같은 긴급한 처리를 담당하게 되며, 나머지 하나는 " 하반부" (Bottom Half)로써, 나머지 처리들을 담당하게 된다. 인터럽트 처리의 대부분은 하반부에서 이루어지게 되며, 이러한 하반부에서의 인터럽트 처리시간에 의해서 현재 수행되고 있던 사용자 프로세스의 실행시간을 빼앗게 된다. 고속의 네트워크 장치에서 유입되는 데이터가 많은 경우 인터럽트 신호가 매우 빈번하게 발생하게 되며, 하반부를 처리하는데 소요되는 시간 역시 기복이 심하게 생기게 된다. 이러한 예측이 불가능한 하반부 처리시간의 기복은 현재 수행되고 있는 사용자 프로세스의 실행시간에 영향을 미치게 되고, 사용자 프로세스의 실행시간 역시 기복이 발생하게 된다. 또한, " 빼앗긴 시간 문제" 에 의해서 스케줄러가 정상적으로 CPU 시간을 사용자 프로세스에게 할당해 주지 못하는 이른바 " 스케줄링 이상(異狀) 현상" 이 발생하게 된다.

[그림1]은 하반부 처리에 의해 빼앗긴 시간의 영향을 보이기 위해서 프레임간 지연시간을 측정한 실험한 결과이다. 실험에서는 mplayer[7]를 사용했으며, 초당 25프레임으로 인코딩되어 있는 영화파일을 매우 많은 네트워크 트래픽이 유입되는 상태에서 재생했다. 만약, CPU 자원이 충분하다면, 재생되는 프레임사이의 지연시간은 40msec가 될 것이다. 그러나, 실험에서는 하반부 처리에 의해서 빼앗긴 시간에 의해서 평균 지연시간은 약100msec 이며, 지연시간의 기복도 심하다. 두 번째 실험은 첫 번째 실험과 같은 방법으로 진행하되, 하반부에서 사용할 수 있는 CPU 시간을 첫 번째 실험에서 측정된 평균 하반부 처리시간으로 제한한 상태에서 진행되었다. 두 번째 실험결과를 보면, 평균지연시간 및 지연시간의 기복이 줄어든 것을 볼 수 있다. 이 실험에서 중요시 해야 할 점은 두 실험에서 하반부에서 사용한 평균 CPU 시간은 거의 동일한데도 불구하고, 프레임간 지연시간의 기복이 줄어들 뿐 아니라, 평균지연시간 역시 현저히 줄어든 것이다. 이러한 결과는 커널모드에서 소비되는 CPU 시간에 심한 기복이 발생할 경우에 커널의 스케줄러가 CPU 자원을 정상적으로 프로세스들에게 분배하지 못하기 때문이다. 본 논문에서는 이러한 현상을 " 스케줄링 이상(異狀) 현상" 이라 정의하고 이후 실험결과를 통해서 다시 설명하고자 한다.

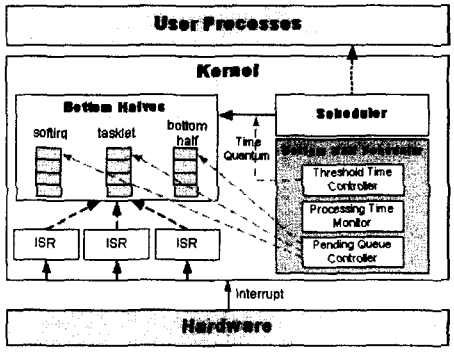
본 서론에 이어서 2장에서는 본 논문에서 제시하는 하반부 처리 스케줄링을 위한 구조에 대해서 기술하고 3장과 4장에서는 실험결과와 결론에 대해서 각각 기술하도록 한다.

2. 전체 구조

리눅스에는 기능에 따라서 softirq, tasklet, bottom half와 같은 세가지 종류의 하반부가 존재한다. (본 논문에서 " 하반부" 는



[그림1] 하반부의 소비시간이 프로세스에 미치는 영향



[그림2] 하반부 스케줄러의 전체구조

세가지 종류의 하반부들을 통칭한다.) 디바이스 드라이버나 커널 서비스는 그 목적에 따라서, 세가지 종류 중 하나의 하반부를 사용하게 된다.

인터럽트 신호가 도착하면, ISR (Interrupt Service Routine)에서는 인터럽트를 처리하는 동안 다른 인터럽트를 막아놓게 된다. 따라서, ISR에서는 긴급한 처리만을 수행한 후에 하반부를 활성화 시킨다. 이 후에, 각 CPU에 대해서 존재하는 *ksoftirq_CPUn*라는 커널 쓰레드에서는 처리해야할 하반부들이 존재하는지 검사하고 각 하반부들을 실행시키게 된다. 기존 하반부 처리 구조에서는 활성화된 하반부는 처리되기를 기다리는 모든 하반부들을 실행시키게 되므로, 하반부에 의해서 소비되는 CPU시간의 기복이 크게 된다. 이러한, 심한 기복에 의한 스케줄러의 비정상적인 CPU시간 분배문제를 해결하기 위해서 본 논문에서는 다음의 세가지 모듈로 이루어진 하반부 스케줄러를 제안한다.

- Processing Time Monitor
- Threshold Time Controller
- Pending Queue Controller

Processing Time Monitor는 각 하반부에서 소비한 CPU 시간을 측정한다. 하반부들이 사용하는 CPU시간이 지나치게 크거나 소비시간의 심한 기복을 줄이기 위해서, Threshold Time Controller에서는 하반부들이 사용할 수 있는 최대 CPU시간을 계산한다. Pending Queue Controller에서는 하반부들이 소비한 시간이 Threshold Time Controller에 의해서 계산된 시간에 도달했을 때, 대기하고 있는 작업들을 제거한 후, 다음 활성화 시점에 다시 대기큐에 삽입하게 된다.

하반부들이 소비하는 시간은 유입되는 작업량이 일정함에도 불구하고 기복을 갖게된다 ([그림 5] 실험결과 참조). 다시 말해서, 하반부 스케줄러에서는 하반부 처리시간이 급격히 증가할 경우 대기중인 작업들을 하반부의 처리시간이 적은 시점으로 연기함으로써 하반부에서 사용하는 CPU시간의 기복을 줄이게 된다. 또한, 커널모드에서 소비되는 시간의 급격한 증감에 의해서 발생하는 " 스케줄링 이상 현상" 을 막음으로써 스케줄러가 CPU를 사용자 프로세스에게 안정적으로 배분할 수 있도록 한다.

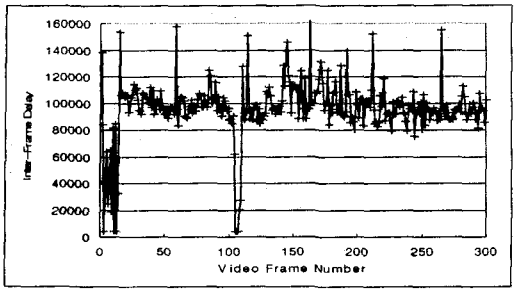
3. 실험결과

본 장에서는 리눅스상에서 제안된 구조에 대한 실험결과를 기술한다. 하반부 스케줄러의 구현은 커널버전 2.4.20을 기반으로 이루어졌으며, 네트워크 장치가 가장 높은 대역폭을 갖는 장치 중 하나이기 때문에 네트워크 패킷 처리에 대해서만 구현되었다.

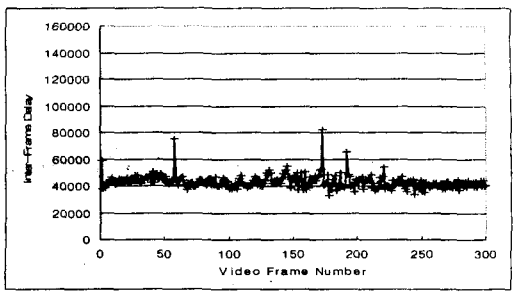
실험환경은 두 대의 2GHz Pentium-4 PC를 사용했으며, 각각 512MB 메모리와 100Mbps 네트워크 카드가 장착되었다. 또한, 네트워크에 로드를 발생시키기 위해서, 한 개의 PC에서는 ttcp[6]를 사용해서 50 Byte 크기의 UDP 패킷을 상대 PC에게 지속적으로 전송하는 상태에서 실험이 진행되었으며, 실험을 위해서 수정된 mplayer [7]를 사용해서 각 프레임 사이의 지연시간을 측정했다.

실험에 사용된 비디오 데이터는 초당 25프레임으로 인코딩된 MPEG-4 데이터이며, 정상적으로 재생된다면, 40 msec간격으로 디코딩되어야 한다. [그림3] 에서 볼수 있듯이, 첫 번째 실험에서는 기존의 리눅스상에서 실험한 결과로써, 하반부 처리에서 소비하는 CPU시간 때문에 mplayer가 필요한 만큼의 CPU를 할당받지 못하기 때문에 프레임간 지연시간이 크게 높아졌다.(평균 지연시간 96.2 msec) 두 번째 실험은 하반부 스케줄러를 사용한 결과로써, [그림4] 와 같이 평균 지연시간이 43.2 msec로 낮아진다.

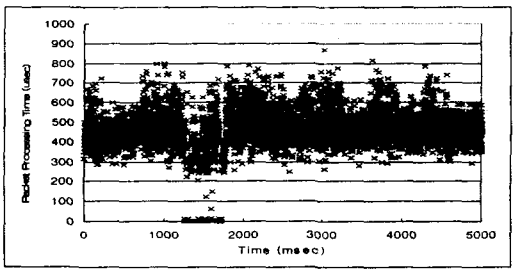
[그림5]와 [그림6]은 하반부에 의해서 소비된 시간을 측정한 결과로, 각각 첫 번째와 두 번째 실험에서의 1 msec당 하반부에서 소비한 시간을 나타낸다. [그림5]에서는 하반부에서 소비한 CPU시간이 심한 기복을 보이는 반면, [그림6]에서 볼 수 있듯이, 하반부 스케줄러를 사용한 경우에는 초기 300 msec 이후에는 안정된 소비시간을 유지하 게 된다. [표1]에서 볼 수 있듯이, 두 실험에서의 하반부에서 사용한 CPU시간은 거의 동일하다. 그러나, 사용자 영역에서의 CPU 사용비율은 하반부 스케줄러를 사용한 경우가 그렇지 않은 경우의 3배정도 크다. 다시 말해서, 실제로 하반부에서 소비하는 시간은 유사함에도 불구하고, 커널 모드에서의 지나치게 심한 소비시간의 기복에 의해서 스케줄러가 사용자 프로세스에게 정상적으로 CPU시간을 할당하지 못하기 때문에 이와 같은 결과가 야기된다.



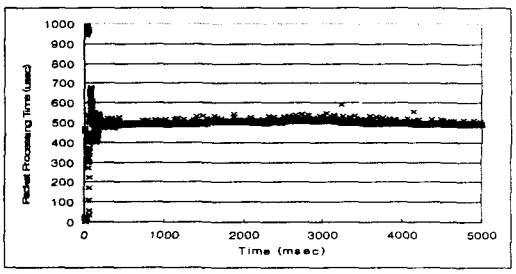
[그림3] 기존 리눅스에서의 프레임간 지연시간



[그림4] 하반부 스케줄러 사용시 프레임간 지연시간



[그림5] 기존 리눅스에서의 하반부에서의 소비시간



[그림6] 하반부 스케줄러 사용시 하반부에서의 소비시간

[표1] 네트워크 처리량 및 CPU 할당비율의 비교

		기존 리눅스	제안된 구조
재생 시간		29.03 초	14.52 초
네트워크 처리량		10.006 MB/초	10.046 MB/초
msec당 평균 패킷 처리시간		454.489 usec	497.697 usec
CPU 할당 비율(%)	사용자 영역	7.5	24.8
	커널 영역	92.4	75.1

4. 결론 및 향후과제

본 논문에서는 하반부 처리에 의해서 빼앗긴 CPU 시간에 의해서 사용자 프로세스에게 안정적으로 CPU를 할당해 줄 수 없는 문제가 발생하며, 이로 인해서 멀티미디어 응용과 같은 실시간 응용들이 일정한 품질의 서비스를 제공할 수 없는 문제가 발생함을 보였다. 또한, 커널모드에서의 실행시간의 심한 기복에 의해서 스케줄러가 유효한 CPU 자원을 사용자 프로세스에게 정상적으로 분배하지 못하는 상태를 " 스케줄링 이상(異狀) 현상" 을 정의하고, 이러한 문제를 해결하기 위해서, 하반부 스케줄러 구조를 제안하고, 하반부 스케줄러를 통해서 사용자 프로세스에게 할당되는 CPU시간을 안정적으로 제공할 수 있음을 실험을 통해서 확인했다.

향후에는 네트워크 처리에 관련된 하반부 처리뿐 아니라, 모든 하반부 처리를 스케줄링 할 수 있는 통합적인 구조로 확장하고, 유입되는 작업량에 빠르게 반응할 수 있는 스케줄링 알고리즘에 대해서 연구할 계획이다.

5. 참고문헌

[1] Luca Abeni, *Luigi Palopoli, Giuseppe Lipari,* and Jonathan Walpole, "Analysis of a Reservation-Based Feedback Scheduler,"Proceedings of the IEEE Real-Time Systems Symposium, 2002

[2] John Regehr and John A. Stankovic, "Augmented CPU *Reservations: Towards Predictable Execution on* General-Purpose Operating Systems," Proceedings of *the Real-Time Technology and Applications* Symposium, 2001

[3] Kevin Jeffay, F. Donelson Smith, Arun Moorthy, and James Anderson, "Proportional Share Scheduling of Operating System Services for Real-Time Applications," *Proceedings of the IEEE Real-Time Systems* Symposium, 1998

[4] Luca Abeni, "Coping With Interrupt Execution Time in RT Kernels: a Non-Intrusive Approach", IEEE Real-Time Systems Symposium Work in Progress, 2001

[5] A. *Indiresan*, A. Mehra, and K.G.Shin, "Receive Livelock Elimination via Dynamic Interrupt Rate Control," Technical report, University of Michigan, 1997

[6] Test TCP(TTCP) benchmark tool, http://www.ccci.com/tools/ttcp/

[7] Mplayer - Movie player for Linux, http://www.mplayerhq.hu