리눅스에서의 Percentile Schedule 기법 구현

최동준[O] 이민석
한성대학교 컴퓨터신기술 대학원[O] 한성대학교 컴퓨터공학부
{dereek[O], mslee}@hansung.ac.kr

An Implementation of Percentile Scheduler on Linux

Dongjoon Choi[O] Minsuk Lee
Computer New Technology of the Hansung University Graduated School[O]
School of Computer Engineering, Hansung University

요 약

이 논문에서는 패킷 스케쥴에 사용된 Percentile 스케쥴 기법을 웹 서버에 적용하여 서버에 들어오는 다양한 요구를 클래스로 분류하여 차등적인 서비스를 제공하고자 하는 연구를 기술하였다. 논문에서는 범용 운영 체제를 실시간화 하기 위한 노력들을 살펴보았으며, Percentile에 기반한 스케쥴러를 리눅스 환경에 구성한 방법을 기술하였다. 제안된 기법의 성능을 살펴보기 위하여 실험을 수행하였으며 Percentile 스케쥴 기법을 적용한 경우 지정된 클래스에 따라 확연히 다른 응답 시간을 관찰할 수 있었다.

1. 서 론

범용 운영 체제를 사용하는 많은 시스템에서도 서비스 품질의 보장에 대한 요구가 커지고 있다. 특히 웹 서비스의 형태로 제공되는 멀티미디어 서비스, 차별화된 홈페이지 참조 등은 서비스 운영자의 수익에 많은 영향을 끼치도록 되어있다.

이 연구에서는 패킷 스케쥴에 사용되는 Percentile 스케쥴 기법을 웹 서버에 적용하여 서버에 들어오는 다양한 서비스 요구를 분류하여 유형 별로 차등적인 서비스를 제공하는데 이용하고자 한다. 이 연구는 서버 호스팅 업체, 전자 상거래 사이트 등에서 모든 고객이 평등하지 않다는 현실 인식에 그 바탕을 두고 있다. 서버 호스팅의 경우, 서로 다른 비용을 지불한 호스팅 고객에 대하여 같은 수준의 서비스를 제공하는 것은 공정하지 못하다. 많은 비용을 지불한 고객의 홈페이지에 대한 참조에 비하여 적은 비용을 지불한 고객의 홈페이지 보다 더 짧은 응답을 제공하여 호스팅 업체는 같은 서버 안에서 차별화된 서비스를 할 수 있게 된다. 또 전자 상거래 사이트의 경우, 사이트를 단순히 살펴보기 위하여 방문한 단발성 고객과 자주 물건을 구매한 고객, 또 그들 중에서도 물건을 구매하기 위하여 쇼핑 카트에 상품을 넣은 고객 등에 대하여 다른 품질의 서비스를 제공함으로써 더 많은 구매를 유도할 수 있다. 이러한 서비스 차별화 정책은 리눅스와 같은 범용 운영 체제가 주로 사용되는 환경에 특별한 프로세스 스케쥴 기법을 필요로 한다.

Percentile 스케쥴 기법은 워낙 경로 배정기에서 패킷 전송에 따른 우선 순위를 적용하기 위하여 개발된 스케쥴 방법이다 [1]. 이 논문에서는 그 방법을 웹 서버에 적용하여 위와 같이 차별화된 웹 서비스를 제공하는 용도로 사용하고자 한다.

2. 관련 연구

리눅스는 프로세스가 가지고 있는 시간 제약성을 엄격하게 맞춰준다는 것은 거의 불가능하다. 이를 개선하기 위한 여러 연구가 진행되어 왔다. 기존의 운영 체제를 수정하여 시간 제약성을 도모한 연구의 예는 RTAI[2], KURT[3], RTLinux[4], 비례 분할식 실행 방식[5], SMART[6], 리눅스 커널을 완전한 선점형 실시간 커널로 바꾸는 미국의 몬타비스타 사의 방식 [7] 등 여러 가지가 있다.

리눅스를 실시간 시스템으로 만들기 위한 노력인 RTAI[2], RTLinux[4] 기법에서는 리눅스 하부에 실시간 커널이 있어 그 스케쥴러가 선점형 스케쥴을 하며 리눅스 커널과 응용 프로그램을 그 실시간 커널 상의 한 비실시간 태스크로 실행한다. 실시간 커널의 스케쥴러는 철저하게 우선 순위에 기반한 스케쥴을 하기 때문에 낮은 지연시간 그리고 매우 예측이 가능한 시스템 환경을 제공한다. 또 정확한 프로세스 동작 시점을 유지하기 위하여, 커널 소스를 수정하여 인터럽트 처리 방법을 변경하였다. RTAI, RTLinux 방식에서는 리눅스 상에서 실행되는 비실시간 태스크와는 달리 실시간 커널 위에서 실행되는 태스크는 모두 같은 메모리 공간에서 실행되어 메모리 보호 기능이 취약하며, 리눅스가 제공하는 서비스들을 이용할 수 없고, 실시간 비실시간 태스크들 사이에는 특별한 IPC (Interprocess Communication) 방법이 제공되어 그를 통해 서로 정보를 교환한다.

KURT[3]는 리눅스에서 태스크를 주기적으로 시행하기 위한 방법을 제공한다. KURT는 새로운 운영 체제가 아니고 어떠한 실시간 이벤트가 발생할 때 프로세스를 기동시켜 주는 스케쥴링 기법이다. KURT는 일반적인 리눅스 프로세스들도 주기성을 가지고 실행될 수 있도록 만들기 때문에 RTLinux나 RTAI와 같은 환경에서 문제가 되는 응용 프로그램에서의 시스템 자원이나 기능의 사용 제한이 없다. KURT는 오로지 타이머를 조작하여 주기적인 프로세스 기동을 하기 때문에 인터럽트에 의한 프로세스의 기동 지연 문제를 해결하기 위하여 실제 기동 시점 이전에서부터 프로세스를 점유하는 비효율적인 방법을 사용하고 있다. 또 원천적으로 우선 순위에 기반한 스케쥴을 하지 않기 때문에 완벽한 시간 제약성 만족을 보장하지 않고 통계적으로만 시간 제약성을 만족한다.

비례 분할식 CPU 할당 방식은[5,8,9,10] 직관적으로는 주어진 가중치에 따라 CPU 자원을 배분하는 장점이 있으나 퀀텀, 즉 자원 배분의 최소 시간 단위의 정밀도가 자원 배분을 정확도를 결정하기 때문에 정밀도를 높이기 위하여 퀀텀을 짧게 잡은 경우 상당한 스케쥴러 오버헤드가 생기게 된다. 또 네트워크 처리를 뒤로 미룸으로써 우선 순위가 높은 데이터의 경우에도 처리가 지연되는 또 다른 문제를 가지고 있다.

3. Percentile Scheduler 구현

이 논문에서는 리눅스의 스케줄러를 Percentile에 기반한 스케줄러로 바꾸고자 한다. 따라서 먼저 리눅스의 스케줄러가 어떤 방법으로 스케줄을 하는지 살펴볼 필요가 있다.
원래의 리눅스는 스케줄러의 시스템 호출을 통해 선택될 수 있는 FIFO (first in first out), RR (round robin), OTHER 방식의 세 가지의 스케줄링 클래스를 유지한다. 리눅스 스케줄러는 FIFO, RR, OTHER의 순서로 프로세스를 우선 스케줄한다. 리눅스 스케줄러는 매번 스케줄러에 의해 계산된 가중치에 기반한 스케줄을 한다. 스케줄러의 가중치 계산에는 스케줄 클래스, CPU 점유 시간, 시스템 호출에 의해 결정되는 nice 값, 캐쉬와 멀티 프로세스를 고려한 CPU 친근도 등이 고려된다. 리눅스는 자신에게 할당된 퀀텀을 다 사용하거나 프로세스 자체가 입출력이나 다른 시스템 호출에 의하여 블록되는 경우를 제외하고는 중단되지 않는다. 따라서 모든 프로세스가 가중치에 기반하여 스케줄링 되기 때문에 데드락 상태에 빠지지 않는 이상 모든 프로세스가 비교적 공정하게 CPU 서비스를 받게 된다. 이 말은 리눅스가 철저한 우선 순위에 의한 서비스 제공을 바탕으로 한 실시간 응용에 적합하지 않다는 것을 의미한다. 그럼에도 불구하고 상당히 많은 웹 서버나 멀티미디어 서비스가 리눅스 상에서 이루어지고 있는 것은 CPU의 성능에 의존한 빠른 응답을 할 수 있기 때문이다. 하지만 이 방법은 서비스 요구의 증가에 따라 시스템 사용량이 100%에 가까워지면 점수록 적절한 응답 시간을 어떤 요구에 대해서도 보장할 수 없는 문제가 생긴다. 따라서 서비스 요구가 아주 많은 상황에서도 적정 수준의 서비스 응답 시간을 통계적으로 보장하고, 오버로드 상황에서도 차별된 서비스를 일부 요구에 대하여 제공할 수 있는 방법이 필요하다.

Percentile 스케줄링 기법은 패킷 별 우선 순위에 기반한 네트워크 라우터에 사용된 스케줄링 기법이다[1]. Percentile 스케줄에서 클래스 i 에 속하는 패킷의 스케줄 우선 순위는 다음과 같다.

$$\Pr(i) = \frac{Si}{Ni * P} \qquad (1)$$

식에서 Si 는 주어진 마감 시간 안에 처리된 패킷의 개수이며 Ni 는 현재까지 클래스 i 로 분류된 패킷의 개수, P 는 주어진 Percentile 이다. 이 $\Pr(i)$ 값이 낮을수록 높은 우선 순위를 가지게 된다. 즉 마감시간 안에 처리된 패킷의 수가 Percentile에 비하여 상대적으로 적은 클래스의 패킷이 더 우선하여 처리된다는 것이다. 이 식의 유도 과정은 [1]에서 볼 수 있다. 이 스케줄 방법은 직관적으로 모든 클래스의 패킷이 주어진 Percentile에 해당하는 비율로 마감 시간을 지키게 된다는 것을 알 수 있다.

패킷 전송에 사용되는 Percentile 스케줄 방식은 한 패킷의 최대 길이가 제한되어 있고 패킷이 전송되고 있는 동안에는 다른 패킷이 중간에 끼어들 수 없다는 점이 가정된다. 반면에 비 주기적으로 실행되는 일반적인 프로세스 스케줄의 경우에는 한 태스크의 실행 시간을 특정 값으로 한정하는 것이 적당하지 않기 때문에 프로세스의 실행 시간과 스케줄 단위에 관한 특별한 고려가 필요하다. 이 고려 사항은 실제 스케줄러가 구현되는 응용 환경에 따라 각각 정의될 수 있으며 이 논문에서는 서버에 도착하는 각 페이지 요구 (http 서버 입장에서의 하나의 페이지 요구)는 하나의 태스크로 간주하고 모든 태스크는 같은 실행 시간과 마감 시간을 가진다고 가정하였다.

그림 1의 스케줄러는 위의 우선 순위 계산식 (1)에 기반하여 가장 우선 순위가 큰 요구를 꺼내어 스케줄 한다. 이 과정에서 실제로는 위 우선 순위 계산식을 적용한 결과, 발생할 수 있는 아주 작은 우선 순위 차이를 무시하고 스케줄 대상이 될 후보 집합을 만든 뒤에 그 집합에 있는 태스크 가운데 EDF (Earliest Deadline First) 기준에 따라 가장 먼저 도착한 태스크를 골라 수행한다. 사용한 서버는 실시간 시스템이 아닌 일반적인 서버로서 이러한 방법은 시스템의 사용도가 아주 높을 때 대화형 사용자(주로 X 윈도우 기반 응용 프로그램을 사용하는)의 응답 시간을 매우 길게 할 수 있는 단점이 있다.

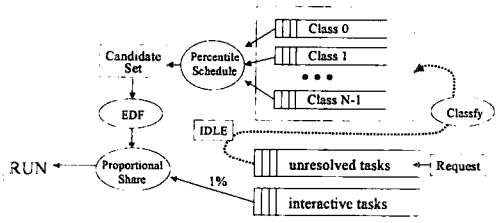

그림 1. Percentile 스케줄러의 구성

4. 실험

4.1 실험 환경

실험은 서버 PC와 페이지 요구를 발생하기 위한 클라이언트 PC를 100Mbps 스위치 기반 이더넷으로 구성되었다. 서버는 256M 바이트의 메인 메모리를 가진 Pentium II 400MHz의 PC이며 아파치 서버가 http 서버로 사용되었다. 클라이언트는 동급의 PC 두 대와 166MHz Ultra Sparc 컴퓨터 두 대가 사용되었으며 이 네 대의 클라이언트가 끊임없이 서버에 패킷을 요구하도록 실험 환경을 만들었다. 클라이언트 시스템에는 페이지 요구를 지속적으로 발생시키는 가상적인 브라우저 프로그램이 실행되며 실제 페이지 요구는 한 이더넷 패킷에 담을 수 있는 작은 정적인 페이지에 대하여 이루어졌다.

그림 2에서 네트워크 프로세싱 결과 추출된 요구에는 http 서버가 스케줄되어 수신한 이후에야 우선 순위가 할당된다. 패킷의 수신 후에 우선 순위가 할당되기까지는 모든 클래스에 대한 요구와 다른 네트워크 데이터에 대하여 모두 같은 우선 순위가 적용되어 도착순으로 처리된다는 것을 의미하며, 과도한 패킷이 수신되어 버퍼 부족으로 인하여 패킷이 버려질 때에도 http 서버가 분류하는 우선 순위와는 상관없이 버려질 패킷이 선택된다. 즉 서버에 의해 우선 순위가 결정되기까지는 우선 순위의 무시 내지는 역전을 겪으면서 스케줄되어 완전한 우선 순위 기반 스케줄 결과와는 약간 차이를 보이게 된다. 다시 말해 좀더 무딘 결과를 얻게 된다.

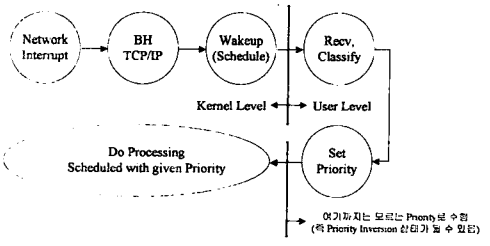

그림 2. 리눅스에서의 네트워크 데이터 처리

4.2 실험 결과

실험에서는 위 실험 환경에서 Percentile 스케줄을 적용한 경우와 적용하지 않은 경우 클라이언트의 페이지 요구에 대한 응답 시간을 비교하였다. 먼저 그림 3은 페이지 요구를 다섯 개의 클래스로 분류하여 차등적인 서비스를 한 결과 클라이언트에서 관찰된 응답 시간을 그래프로 표현한 것이다. 이 응답 시간은 클라이언트 시스템에서 관찰된 시간이기 때문에 서버로의 TCP 연결 요구를 한 시점부터, 페이지 요청, 결과 수신까지의 모든 시간을 더한 것으로 서버가 관찰하는 시간과는 다른 시간이다. 그림에서 X 축은 응답 시간이며 Y 축은 누적된 패킷 비율이다. 그림에서 Percentile이 높을수록 응답 시간이 빠르다는 것을 확연히 볼 수 있다.

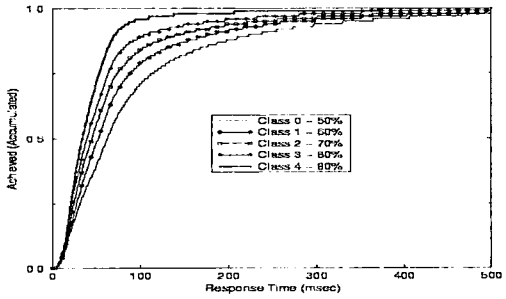

그림 3. Percentile 스케줄 결과

그림 4는 같은 실험 환경에서 http 서버의 수정 없이, 또 원래의 리눅스 스케줄러를 그대로 이용하여 요구를 처리한 경우의 그래프이다. 이 경우에는 각 클래스 별 우선 순위가 하나도 적용되지 않았기 때문에 모든 클래스에 대하여 같은 응답 시간을 보이고 있다.

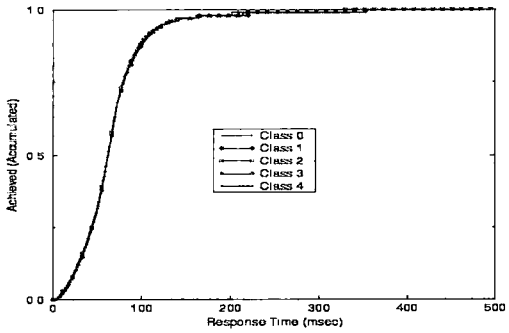

그림 4. 리눅스 시분할 스케줄 결과

5. 결론

이 논문에서는 패킷 스케줄에 사용된 Percentile 스케줄 기법을 웹 서버에 적용하여 서버에 들어오는 다양한 요구를 클래스로 분류하여 차등적인 서비스를 제공하고자 하는 연구를 기술하였다. 논문에서는 범용 운영 체제를 실시간화 하기 위한 노력들을 살펴보았으며, Percentile에 기반한 스케줄러를 리눅스 환경에 구성한 방법을 기술하였다. 제안된 기법의 성능을 살펴보기 위하여 실험을 수행하였으며 Percentile 스케줄 기법을 적용한 경우 지정된 클래스에 따라 확연히 다른 응답 시간을 관찰할 수 있었다. Percentile 스케줄 기법은 기존의 시분할 스케줄로는 할 수 없는 새로운 서비스를 가능하게 하며, 이를 바탕으로 서버가 사용되는 많은 시스템에서 더 많은 수익을 창출할 수 있는 기회를 제공할 수 있다.

이 연구의 후속 연구로는 스케줄러를 더 개선하여 프로토콜 프로세싱 과정에서도 우선 순위를 지정하여 확실한 우선 순위 정책을 구현하는 일과, 하드 디스크, 데이터 베이스 시스템 등 일반적으로 서버에서 이용되는 다른 서비스가 연계되었을 경우에도 클래스 별 우선 순위를 지정할 수 있도록 만드는 일이다.

참고 문헌

[1] Agarwal, N. "Percentile Goal As A Performance Criterion In A Multiclass GI/G/1 Queuing System", Ph.D. thesis, North Carolina State University, Raleigh, NC, USA, 1995.
[2] http://www.aero.polimi.it/projects/rtai/
[3] http://www.ittc.ku.edu/kurt/
[4] http//www.rtlinux.org
[5] K. Jeffay, F. Donelson Smith, A. Moorthy, J., Anderson, Proportional Share Scheduling of Operating System Services for Real-Time Application", proceeding of the 19th IEEE Real-Time Systems Symposium, Dec. 1998, pp. 480-491.
[6] J. Nieh, M.S. Lam, The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications, Proc., 16th ACM Symp. on Operating System Principles, Saint-Malo, France, Oct. 1997, pp. 184-197.
[7] http://www.hardhatlinux.com
[8] P. Goyal, X. Guo, H. M. Vin, A Hierarchical CPU Scheduler for Multimedia Operating Systems, Proc., USENIX Symp. On Operating Systems Design and Implementation, Seattle, WA, Oct, 1996, pp. 107-121.
[9] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. Baruah, J. gehrke, C. Plaxton, A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems, Proc. 17th IEEE Real-Time Systems Symposium, Dec. 1996, pp. 288-299.
[10] C. Waldspurger, W. Weihl. Lottery Scheduling: Flexible Proportional Share Resource Management, Proc. USENIX Symp. On Operating System Design and Implementation, Nov. 1994, pp. 1-12.