이동형 개인 컴퓨팅 환경의 에너지 효율 증가를 위한 빠른 전압 조절을 고려한 가변 성능 알고리즘

서의성° 이준원
한국과학기술원 전자전산학과
{ses°, joon}@calab.kaist.ac.kr

Dynamic voltage scaling policy for processors with fast voltage transition on personal computing environment

Euiseong Seo° Joonwon Lee
CS Dept. Korea Advanced Institute of Science and Technology

요 약

DVS(dynamic voltage scaling)은 이동형 프로세서에서 에너지 효율을 높이기 위한 필수 요소로 자리 잡고 있다. DVS를 효과적으로 사용하기 위해선 대상 태스크의 특성과 하드웨어 특성에 맞는 DVS 알고리즘이 필요하다. 상품화 수준의 많은 운영체계들이 일정한 인터벌(interval)을 바탕으로 시스템 사용 상황을 분석하여 목표 성능을 결정하는 방식을 사용하고 있다. 이러한 방식은 태스크의 특성이 갑자기 변하여 성능을 요구할 경우 인터벌만큼의 시간이 진행된 후에야 반응 한다는 단점이 있다. 또한, 태스크 별 특성이 아닌 시스템 전체의 특성을 따르므로 이질적인 성격의 태스크들이 동시에 실행 되는 환경에는 적합하지 않다. 최근의 모바일 프로세서들은 수 마이크로초 수준의 성능 전환 시간을 제공하고 있으며 이 속도는 계속 줄어들고 있다. 프로세서의 고성능화로 인해 I/O 작업의 경우 프로세서 성능에 따른 실행 시간의 차이가 존재 하지 않는다. 이러한 두 가지 특성을 바탕으로 우리는 TIB(timer interrupt based) 알고리즘을 제안한다. TIB 알고리즘은 일정한 길이의 인터벌 대신 타임 슬라이스(time slice)를 성능 결정의 단위로 삼는다. 성능의 결정은 태스크 별로 이루어지며 각 태스크가 사용했던 이전 타임 슬라이스가 타이머 인터럽트(timer interrupt)에 의해 끝났다면 최대의 성능을 그 외의 경우는 최저의 성능으로 실행하게 된다. 이러한 접근 방식을 통해 I/O 작업이나 이벤트를 기다리는 태스크에 대해 최저 성능을 제공함으로써 실행 시간의 적은 손해를 대가로 많은 에너지 절감을 이룰 수 있다. 또한, 태스크의 속성이 변한 경우 타임 슬라이스 길이 만큼의 지체만을 허용하게 된다. 이러한 TIB 인터벌에 기반한 알고리즘에 비해 개별 태스크의 특성에 따른 성능 조절과 태스크의 변화에 따른 빠른 반응을 자랑으로 한다. 본 논문에선 TIB 알고리즘을 리눅스 커널에 구현하여 성능을 평가하였고 그 결과 리눅스에서 사용되는 기존 인터벌 기반의 알고리즘들에 비해 좋은 전력 절감 효과를 얻을 수 있었다.

1. 서 론

모바일(mobile) 기기의 활용도가 높아지고 성능 역시 증가하면서 기기의 전력 소모는 중요한 이슈가 되고 있다. 특히 프로세서의 성능이 크게 증가하면서 프로세서는 최대의 전력을 소모하는 부품이 되었다.

프로세서의 전력 소모는 입력 전압의 제곱과 비례하며 클럭에 비례한다. 또한 클럭은 입력 전압에 비례하므로 결국 프로세서의 소비 전력은 입력 전압의 세제곱에 비례하게 된다. DVS(dynamic voltage scaling)는 전압과 클럭을 떨어뜨려 성능을 희생하는 대가로 절전 효과를 얻는다. 이 기능은 현재 대부분의 모바일 환경을 위한 프로세서에서 필수 요소가 되어 있다.

하지만, 성능 전환 과정에서 전압을 조정하기 위해 잠시 프로세서가 멈추는 시간이 생긴다. 따라서 불필요한 성능 전환은 오히려 에너지 및 성능의 낭비를 초래한다. 이러한 오버헤드를 줄이기 위해 효율적인 성능 결정 알고리즘이 필요하다. 성능 결정 알고리즘은 태스크의 시간성(timeliness)을 지키거나 태스크의 성능 피해를 최소화 하면서 프로세서의 성능을 최소로 유지하는 것을 목표로 한다. 태스크의 시간성 또는 실행 시간에 대한 요구는 태스크의 속성 및 시스템의 상황에 따라 다르다. 따라서, 바람직한 DVS 알고리즘은 이러한 변수를 고려하여 설계 되어야 한다.

본 연구에선 노트북이나 PDA와 같은 개인 컴퓨팅 환경을 고려한다. 목표로 하는 환경은 멀티미디어 응용 프로그램과 같은 소프트 리얼타임(soft real-time) 태스크 와 대부분의 태스크들에 대해 빠르게 실행 될수록 좋은(best-effort) 속성을 갖는 것을 가정한다. 즉, 본 연구에서 목표로 하는 DVS 알고리즘은 DVS를 사용하지 않는 경우와 대등한 실행 속도를 보장하며 태스크의 속성이 변하여 적절하지 못한 성능을 제공하는 경우 빠른 반응을

본 연구는 ITRC 프로그램을 통해 이루어졌습니다.

통해 적절한 성능을 제공하여야 한다.

본 논문에선 최근 프로세서의 성능 전환 속도가 빨라지고 있고 프로세서가 제공하는 최저 성능에서도 I/O 처리 속도는 변화가 적다는 사실에 착안하여 새로운 알고리즘을 제안한다. 본 논문에서 제안하는 TIB(timer interrupt based) 알고리즘은 태스크가 사용하는 타임 슬라이스가 어떤 형태로 끝냈는지를 태스크의 속성에 대한 판단 근거로 사용하여 태스크 별로 프로세서 성능을 결정한다. 제안하는 알고리즘은 Linux 커널에 구현되어 속성이 다른 태스크들에 대해 성능 평가를 하였다. 평가 결과 기존 Linux에서 사용되던 인터벌 기반에 비해 절전 능력이 더 우수하였으며 실행 시간에선 큰 차이가 나지 않았다.

2. 관련 연구

하드 리얼타임 환경이 아닌 개인용 컴퓨팅 환경에서 DVS의 활용에 대한 연구는 태스크가 필요한 프로세서 자원에 대해 커널에 정보를 제공하는 원시적인 형태의 접근과 운영체계에서 투명하게 성능 조절을 하는 형태의 연구로 구분할 수 있다.

이 중 투명한 성능 조절은 크게 태스크 별로 필요한 성능을 분석하거나[1][2][3] 태스크의 형태를 파악하여 태스크 별로 목표 성능을 결정하는 형태와 일정한 인터벌 동안 시스템의 프로세서 자원 사용 상태를 파악하여 성능을 결정하는[4][5][6] 두 가지 방식이 사용되고 있다.

전자의 경우 태스크의 성격을 파악하기 위한 오버헤드가 크며 태스크의 실시간성을 중시하여 최적의 성능을 찾아내기 위해 노력하므로 실시간성이 없는 태스크에 대해서는 에너지 효율을 높이는 방법으로 사용할 수 없다는 단점이 있다.

후자의 경우 Linux와 Windows 등 널리 사용되는 운영체계에서 채택하여 사용하고 있다[7][6]. 이것은 인터벌 기반의 접근 방식이 투명하고, 구현 및 실행에 오버헤드가 없으며, 성능 전환의 빈도가 적으므로 오버헤드가 적기 때문이다. 하지만, 시스템 전체에 대한 분석을 통해 시스템 전체에 대한 성능을 결정하므로 일정 주기로 실행되는 프로세서 바운드(bound) 작업이거나 I/O 바운드 작업과 프로세서 바운드 작업이 섞여 있는 등의 상황에선 어느 태스크에게도 만족스럽지 못한 성능을 제공하게 된다. 또한, 장기간 휴지기를 지난 후 일정 기간 실행 되는 이벤트 기반의 태스크의 경우 한 인터벌이 지난 후에야 성능이 변화됨으로 반응이 느리고 실제로 성능이 변화되었을 때에는 이미 실행이 끝나 다시 휴지기에 있는 경우 등의 문제가 생길 수 있다.

3. TIB 알고리즘

우리가 제안하는 TIB 알고리즘은 프로세서의 성능 전환에 필요한 시간이 운영체계의 타임 슬라이스 길이에 비해 상당히 짧은 경우를 가정한다. 실제로 Pentium M 프로세서의 경우 성능 변화에 필요한 프로세서 작동 정지 시간이 10 마이크로초 이내이지만[8] Linux의 경우 타임 슬라이스는 1 밀리 초, Windows는 10 밀리 초로 그 차이가 크다.

빨라진 성능 전환 속도는 태스크의 속성이 변화하였을 경우 빠른 대응을 가능하게 해준다. 또한 시스템이 휴지(idle) 상태가 되었을 경우 성능을 최저 수준으로 낮추었을 때 성능 전환 비용을 상쇄하고 에너지 상의 이익을 얻을 수 있는 휴지 상태의 길이를 짧게 만든다. 표 1은 Pentium M 1400 Mhz 프로세서와 512 Mbytes의 램을 장착한 삼성 SX-10 랩탑의 전력 소모를 측정한 결과이다.

Clock (Mhz)	Voltage (V)	Idle (W)	Peak (W)
600	0.956	17.77	21.92
800	1.180	19.00	24.65
1000	1.308	20.09	28.79
1200	1.436	21.56	32.94
1400	1.484	22.59	37.34

표 1 SX-10의 성능과 상태에 따른 전력 소모

이 결과를 바탕으로 성능 전환 시에 최대 전력을 소모한다고 가정하면 시스템이 80.90 마이크로초 이상 휴지 상태에 있을 경우 최저 수준으로 낮추는 것이 이익이라는 결론을 얻을 수 있다. 이것은 기존 운영체계에서 사용하는 타이머의 정밀도보다 짧은 시간으로서 이 결론으로부터 만약 운영체계가 휴지 상태에 들어가게 될 경우 성능을 최저로 유지하는 것이 좋다는 결론을 얻을 수 있다.

I/O 작업의 경우 프로세서에서 처리하는 부분에 비해 I/O 대기 시간이 월등히 길기 때문에 표2에서 보듯이 I/O 작업의 경우 프로세서의 성능은 전체 실행 시간에 큰 영향을 끼치지 못한다. 따라서, I/O 바운드 작업의 경우 최저 성능으로 실행한다면 적은 실행 시간의 손실로 에너지 소비를 줄일 수 있다.

Clock (Mhz)	Copy 512Mbytes (seconds)	10^9 while-loops (seconds)
600	52.871	11.84
800	52.028	8.91
1000	51.034	7.19
1200	50.765	5.99
1400	50.339	5.12

표 2 프로세서 성능과 태스크 속성에 따른 실행 시간

제안하는 알고리즘은 이상의 두 가지 특성을 바탕으로 각 태스크별로 타임 슬라이스가 타이머 인터룹트에의해 강제적으로 끝났을 경우 프로세서 바운드로 정하여 다음 타임 슬라이스를 최대의 성능으로 실행하고 그 외의 경우는 I/O 대기 또는 휴지기의 태스크로 가정하여 다음 타임 슬라이스에서도 최저의 성능으로 실행한다.

본 알고리즘을 사용할 때 성능 전환 명령이 실행된 후 성능 전환이 이루어지기까지 지체가 없다고 가정하면 최악의 경우 0.43 밀리 초의 지연이 발생하게 되며 이것은 인터벌 기반의 알고리즘에 비해 빠른 값으로서 일반적인 멀티미디어 응용 프로그램에서 수용할만한 값이다.

4. 구현 및 성능 평가

TIB 알고리즘은 Linux 커널 2.6.11에 수십 라인의 코드 변경으로 구현이 가능하였다.

성능 평가를 위해 SX-10 랩탑을 사용하였으며 MPEG 재생을 제외한 모든 실험은 싱글 유저 모드에서 실행하였다. 무선랜 및 LCD 등은 전력 소모를 최소화 하도록 설정하였다. 전력 측정에 사용한 장비는 Yokogawa WT-210으로서 매 20 마이크로 초 마다 전력 소모를 측정하며 100 밀리 초에 한 번씩 평균한 값을 출력한다.

비교를 위해 프로세서가 제공하는 모든 성능 레벨을 포함하여 고정된 인터벌을 사용하는 CPUSpeed와 가변 인터벌을 사용하며 Pentium M에 최적화된 Ondemand[7] 알고리즘에 대해서 실시하였다.

평가 대상 태스크는 여러 속성을 대표하기 위해 512메가의 파일을 카피하는 " File Copy", 1000x1000의 매트릭스 연산을 하는 " Matrix Multiplication", 커널 2.6.11를 컴파일 하는 작업으로서 I/O와 프로세서 바운드가 섞여 있는 " Kernel Compilation", 640x480 30fps의 MPEG 4를 재생하는 " MPEG Play"로 구성하였다. 이 중 MPEG Play를 제외한 나머지 태스크는 알고리즘들 간의 공정성을 위해 가장 낮은 성능에서 실행이 끝나는 시간을 기준으로 휴지 상태를 포함하여 그 시간 동안 소모하는 전력을 측정하였다[9].

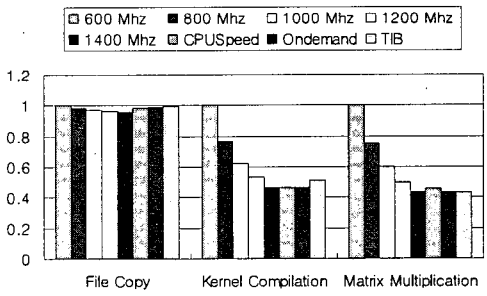

그림 1 태스크에 따른 실행 시간 비교

그림 1에서 볼 수 있듯이 TIB를 포함한 전력 관리 알고리즘들은 대부분 최대 성능과 유사한 실행 속도를 보이고 있다. Kernel Compilation의 경우 TIB가 근소하게 떨어지는 성능을 보이고 있다. 이것은 태스크의 속성이 자주 변함으로써 성능 전환이 빈번하게 발생하는데, 실제 Pentium M의 경우 성능 전환 명령이 입력 된 상태에서 성능 전환 작업이 발생하기까지 수백 마이크로 초의 지연이 있는 것으로 추측 되며 이로 인해 발생하는 부적절한 상태로 인한 것으로 보인다. 반대로 태스크의 속성이 변하지 않는 Matrix Multiplication의 경우 근소하게 TIB가 다른 알고리즘에 비해 빠른 것을 알 수 있다.

전력 소모의 경우 그림 2에서 볼 수 있듯이 TIB 알고리즘은 좋은 결과를 보여주고 있다. CPUSpeed의 경우 File Copy에서, Ondemand의 경우 MPEG Play에서 절전 효과를 보여주지 못하고 있음을 고려할 때 TIB 알고리즘이 매우 짧은 태스크 속성 판단 기준을 통해 보다 정확한 성능 레벨을 결정하고 있음을 확인할 수 있다.

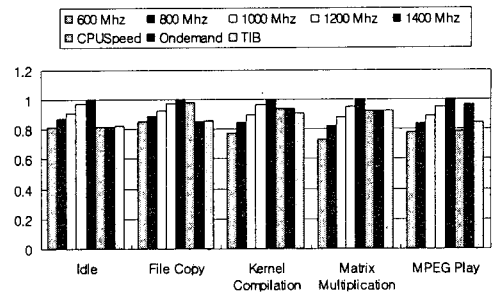

그림 2 태스크에 따른 전력 소모 비교

5. 결론

본 논문에서 제안하는 TIB 알고리즘은 태스크의 실행 시간을 최소화 하며 절전 효과를 거둬야 하는 랩탑, PDA 등의 환경에서 최근 개발되고 있는 빠른 성능 전환이 가능한 프로세서의 장점을 이용하였다. TIB 알고리즘은 인터벌 기반의 알고리즘의 장점인 투명성, 적은 오버헤드, 쉬운 구현과 태스크 속성 분석 방식의 장점인 태스크 별 차별적인 성능 제공 및 정확성이라는 장점을 모두 갖추고 있다.

향후 프로세서의 성능 전환 명령과 실제 성능 전환 사이의 시간이 단축된다면 TIB 알고리즘은 대부분의 태스크들에 대해 성능의 손실 없이 전력 절감 효과를 얻을 수 있을 것이다.

[1] Flautner, K., Mudge, T. "Vertigo: Automatic performance-setting for Linux", Proceedings of The 5th Symposium on Operating System Design and Implementation, pp. 105-116, 2002

[2] Freeh, V. W., Minerick, R. J., Kogge, P. M. "Dynamic power management using feedback", Workshop on Compilers and Operating Systems for Low Power, pp. 6.1-6.10, 2002

[3] Soma, R., Choi, K., Pedram, M. "Fine-grained voltage and frequency scaling for precise energy and performance trade-off based on off-chip access to on-chip computation times", Proceedings on Design, Automation and Test in Europe Conference and Exhibition Volume I, pp. 4-10, 2004

[4] Burd, T., Pering, T., Brodersen, R. "The simulation and evaluation of dynamic voltage scaling algorithms", Proceedings of The International Symposium on Low Power Electronics and Design, pp. 76-81, 1998

[5] Demers, A., Weiser, M., Welch, B., Shenker, S. "Scheduling for reduced cpu energy", Proceedings of the First Symposium on Operating Systems Design and Implementation, pp. 13-23, 1994

[6] Microsoft Corporation, "Microsoft windows native processor performance control", http://www.microsoft.com/hwdev/tech/onnow/ProcPerfCtrl.asp, 2002

[7] Pallipadi, V. "Enhanced Intel SpeedStep Technology and Demand-Based Switching on Linux", Intel Developer Service, http://www.intel.com/cd/ids/developer/asmo-na/eng/microprocessors/195910.htm?page=1

[8] Intel Corporation, "Intel Pentium M Processor Datasheed", 2003, chapter 2.2. p.14

[9] Hensbergen, E. V., Rajamony, R., Miyoshi, A., Lefurgy, C., Rajkumar, R. "Critical power slope: Understanding the runtime effects of frequency scaling", Proceedings of The 16th Annual International Conference on Supercomputing, pp. 35-44, 2002