디바이스 특성을 고려한 저전력 스케줄링

양희백° 하 란
홍익대학교 컴퓨터공학과
{hbyang°, rhanha}@cs.hongik.ac.kr

Low Power Scheduling Based On Device Characteristics

Heabeck Yang° Rhan Ha
Dept. of Computer Engineering, Hongik University

요 약

현재 사용되는 PDA, 핸드폰 등의 이동기기는 보다 좋은 성능과 향상된 기능에 대한 시장의 지속적인 요구로 고성능을 요구하는 응용 프로그램이 점차 추가되고 있다. 이에 고성능 프로세서의 탑재가 일반화 되고 있으며, 그에 따른 전력 소비 또한 증가하고 있다. 시스템 전력 사용량의 증가 문제를 해결하고자 DVS기법, DPM기법 등이 제시되었으나 모바일 기기에 저전력 프로세서의 탑재가 일반화 되면서 전체 에너지 소비측면에서 디바이스의 비중이 상대적으로 증대되어 기존 스케줄링 기법은 하나의 시스템 요소만을 위한 최적화 방법을 제시할 뿐 전체 시스템의 에너지 소비를 최적화시키지는 못하게 되었다. 이에 본 논문에서는 이동기기에서 프로세서의 속도를 결정하는 과정과 스케줄러가 태스크의 우선순위를 결정하는 과정에 있어 단위 시간당 디바이스의 에너지 소비가 프로세서의 단위 시간당 에너지 소비보다 큰 현실을 반영하여, 태스크의 실행 중 필요한 디바이스의 전력 소모량을 기준으로 스케줄러가 프로세서 최적화 정책과 디바이스 최적화 정책 중 올바른 스케줄링 정책을 선택하여 프로세서의 속도를 결정하고 실행순서를 조절함으로써 시스템의 가용시간을 향상시키는 기법을 제안한다.

1. 서 론

배터리 파워로 동작하는 포터블 기기는 모바일 컴퓨팅, 무선통신, 워어러블 컴퓨팅(Wearable Computing)과 같은 산업분야뿐만 아니라 군사 분야까지 다양한 분야에서 폭넓게 쓰이고 있다. 이에 다양한 기능이 시스템에 첨가되었고, 좋은 성능과 많은 기능에 대한 시장의 지속적인 요구로 고성능을 요구하는 응용프로그램이 지속적으로 시스템에 추가되고 있다. 이를 지원하기 위해 고성능 프로세서의 탑재가 일반화되고 있으며, 그에 따른 전력소모도 증가하였다. 또한 모바일 컴퓨팅을 위한 저전력 프로세서의 등장으로 적은 전력량을 가지고 고성능을 발휘하게 됨으로써 상대적으로 시스템에서 차지하는 디바이스의 에너지 소비 비중이 증대되었다. 지금까지 실시간 시스템을 기반으로 external/internal slack을 활용하는 DVS(Dynamic Voltage Scaling)[1]과, 디바이스의 사용예측을 통하여 디바이스의 사용시간을 최대한 줄이는 DPM((Dynamic Power Management)[2] 등이 제시되었으나 시스템의 특정 요소에 초점을 맞추어 스케줄링 함으로써 전체 시스템의 에너지를 최적화하지 못하는 단점이 있다. 이에 본 논문은 저전력 프로세서 탑재로 프로세서의 단위 시간당 에너지 소비가 디바이스보다 상대적으로 작은 모바일 기기를 기반으로 하는 저전력 스케줄링 기법을 제안한다. 기존에 제시된 프로세서만을 고려하는 DVS 스케줄링 방법과 디바이스만을 고려하는 DPM 스케줄링 방법의 문제점을 극복하기 위해 서로 다른 전력 사용량을 가진 디바이스를 스케줄링 단계에서 고려한다. 태스크의 수행 시 사용되는 여러 개의 디바이스들 중에서 태스크의 전력 소모량에 큰 영향을 미치는 디바이스를 결정하고, 디바이스의 에너지양과 프로세서에서 소비되는 에너지양과의 관계를 이용하여 스케줄러는 디바이스 에너지 최적화 정책과 프로세서 에너지 최적화 정책 중 선택한다. 이렇게 결정된 정책에 따라 프로세서의 속도를 조절함으로써 시스템의 가용한 에너지를 최대로 활용한다. 이때 태스크가 실행되는데 필요한 디바이스들에 관한 정보는 태스크의 실행 전에 알 수 있다고 가정한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 저전력 스케줄링 기법에 대해 살펴본다. 3장에서 제안하는 기법을 설명하며, 4장에서 제안 기법을 평가하고 5장에서 결론을 맺는다.

※ 본 연구는 한국과학재단 목적기초 연구사업(R04-2002-000-00039)과 정보통신부 ITRC사업 HY-SDR 연구센터의 후원으로 연구되었습니다.

2. 관련연구

Slack time은 시스템에서 제공하는 시작시간과 데드라인을 근거로 하여 태스크의 수행이 데드라인보다 일찍 완료되어 프로세서가 idle상태로 있는 시간을 말하며 external, internal slack으로 구분된다. External slack이란 특정 태스크의 Worst Case Execution Time(WECT)를 기준으로 데드라인과 비교하여 계산된 idle time이다. External slack은 태스크의 수행시간 이전에 알 수 있음으로 이 값을 기준으로 starting voltage를 조절한다. Internal slack은 WCET와 AET(Actual Execution Time)과의 차이로, 코드상의 여러 분기점에 의한 다양한 실행 경로의 존재로 인한 전체 실행시간의 다양함에 근거한다. 하지만 태스크의 시작 시간에 Internal slack 산출이 불가능함으로 starting voltage 조절을 위한 기준으로 이용될 수 없다. 따라서 external slack을 통한 속도 결정 후 수행이 끝난 시점에 다음 태스크의 external slack과 함께 처리된다. Internal slack을 이

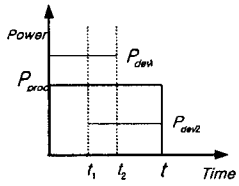
용하기 위한 방법은 [3]에서 제안되었으며 컴파일 타임에 알 수 있는 태스크 각각의 WCET를 기반으로 실행 시 내부적으로 발생하는 slack을 이용하는 방법을 제시했다.

DPM은 시스템의 성능감소 없이 전력소비를 줄이는 효과적인 접근방법으로, 서비스 요구가 없는 디바이스는 idle mode로 동작하다가, 태스크로부터 디바이스 사용요청이 들어왔을 때 요구한 디바이스를 가동시키는 방식이다. DPM을 적용하기 위해서는 태스크가 수행되는 동안 다양한 작업량을 가지며, 작업량의 변동이 예측 가능하다는 가정이 필요하다. 따라서 작업량 관찰을 기반으로 하는 제어정책을 가지고 있다. 가장 간단한 정책으로는 데스크 탑에서 사용되고 있는 timeout policy가 있다. 이 정책은 일정시간 동안 디바이스 사용 요청이 없는 경우 디바이스를 가동하지 않다가, 서비스 요청이 있을 때 가동되는 방식을 취하고 있다. DPM 알고리즘은 예측 가능 기법[4][5]과 확률 기법[6]으로 분류되며, 실질적인 측정보다 시뮬레이션을 통하여 계산된다. 이는 하드웨어와 소프트웨어간의 복잡한 상호작용 때문이다. 이러한 이유로 Microsoft와 Toshiba에서 Advanced Configuration and Power Interface(ACPI)[7]가 주장되었다. ACPI는 전력관리를 위한 HW/SW간에 동일한 인터페이스를 제공함으로써 하드웨어 개발자나, Operating system(OS) 디자이너, 디바이스 드라이버 프로그래머가 보다 쉽게 제품을 개발할 수 있도록 돕는다.

3. 디바이스 특성을 고려한 스케줄링

본 논문은 프로세서의 에너지 소비가 상대적으로 작은 모바일 시스템을 기반으로, 실행될 태스크에 필요한 디바이스 에너지 소비량과 현재 프로세서의 속도로 수행되었을 때의 에너지양을 가지고 프로세서 최적화 기법과 디바이스 최적화 기법 중 상황에 맞는 방법을 결정하여 프로세서의 속도를 조절하는 방법을 제안한다. 이 기법은 기존의 알고리즘이 프로세서가 디바이스에 비해 전력을 많이 소모한다고 가정하여 기인된 에너지 낭비를 막고, 프로세서의 속도를 현재 프로세서의 속도를 기준으로 디바이스의 에너지 소비량을 고려하여 가장 적절한 속도로 결정함으로써 시스템의 전체 에너지를 줄이는 장점이 있다. 제안된 방법은 세 단계로 나누어진다. 본 논문에서 사용된 기호는 표1에서 설명한다.

첫 번째는 현재 프로세서의 속도를 기준으로 실행할 태스크가 사용할 디바이스의 에너지양을 고려하여 스케줄링 정책을 결정하기 위해 프로세서가 사용하는 디바이스들 중 가장 높은 에너지 소비 비중을 차지하는 디바이스를 결정하는 단계이다



[그림1] 프로세서와 디바이스간의 에너지 소비

태스크의 실행 시 다양한 디바이스가 사용된다고 가정하고, 태스크의 전력소비량에 영향을 미치는 디바이스를 결정하기 위해 두 개의 서로 다른 디바이스를 사용하는 경우를 살펴보자. 그림1에서 시간 t는 현재의 프로세서 속도로 태스크의 수행이 완료되는 시점을 나타낸다. t_1는 디바이스2를 사용하기 시작하는 시점을 나타내며, t_2는 디바이스1의 사용이 완료된 시점을 나타낸다. 이때 태스크가 시간 t동안 디바이스에서 소비된 에너지는 $P_{dev1}t_2$, $P_{dev2}(t-t_1)$가 되고, 프로세서에서 소비된 에너지는 $P_{proc}t$가 된다. 따라서 태스크가 사용한 전체 에너지 E_{sys}는 $P_{dev1}t_2+P_{dev2}(t-t_1)+P_{proc}t$으로 표현할 수 있으며, 각각의 디바이스에 대한 태스크가 사용한 전체 에너지의 비율은 다음과 같다.

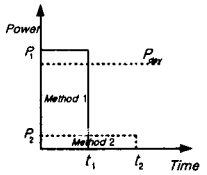
변수	설명
P_{proc}^{now}	현재시점에서의 프로세서의 단위시간당 전력 소비량
P_{proc}^{fix}	속도를 조절했을 때의 프로세서의 단위시간당 전력 소비량
P_{dev}	디바이스의 단위시간당 전력소모량
t'	프로세서의 속도를 변경하였을 때의 수행시간
f_{fix}	프로세서의 속도를 조절했을 때의 frequency
f_{now}	현재시점에서의 프로세서의 frequency
E_{sys}	시스템 전체의 에너지 소모량
E_{gain}	에너지 이익
E_{loss}	에너지 손실

[표 1] 변수 정의

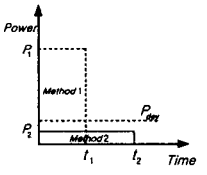
Device1이 차지하는 비중:$P_{dev1}t_2/E_{sys}$

Device2이 차지하는 비중:$P_{dev2}(t-t_1)/E_{sys}$

이를 이용하여 시스템에서 가장 큰 에너지 비중을 차지하는 디바이스를 결정한다.



[그림2] 디바이스의 에너지 소비량의 큰 경우



[그림3] 디바이스의 에너지 소비량의 작은 경우

두 번째는 스케줄링 정책 결정 단계이다. 그림 2,3에서 디바이스의 단위 시간당 전력량이 차이로 인해 시스템 전체 에너지 소비는 스케줄링 방식에 좌우된다. 그림2처럼 P_{dev}가 큰 경우에는 높은 프로세서 속도로 작업을 수행한 방식이 에너지를 적게 소모한다. 즉 디바이스의 소비 에너지를 줄이는 기법이 시스템 전체 에너지 소비 측면에서 에너지 소비를 줄인다. 그림3처럼 P_{dev}가 작은 경우에는 낮은 프로세서 속도로 작업을 수행한 방식이 에너지를 적게 소모한다. 즉 프로세서의 에너지 소비를 최적화하는 방식이 전체 시스템의 에너지 소비를 줄인다.

$(P_{proc}^{now}+P_{dev})t \geq (P_{proc}^{fix}+P_{dev})t'$을 이용하여 프로세서의 속도를 변경하였을 때 전체 시스템 측면에서 에너지 이익은 식(1),(2)을 만족하는 경우에 발생한다. 식(1),(2)에서 f_{now}, P_{dev} P_{proc}^{now}은 상수이므로, 대입하여 f_{fix}의 범위를 구할 수 있고 이를 이용하여 최적의 정책을 결정하게 된다. 식(1)은 디바이스 최적화 정책을 나타내고 식(2)은 프로세서 최적화 정책을 나타낸다.

$$\frac{f_{fix}(f_{now}+f_{fix})}{(f_{now})^2}P_{proc}^{now} \leq P_{dev} \quad \text{if } f_{now} < f_{fix} \qquad (1)$$

$$\frac{f_{fix}(f_{now}+f_{fix})}{(f_{now})^2} P_{proc}^{now} \geq P_{dev} \quad \text{if } f_{now} > f_{fix} \qquad (2)$$

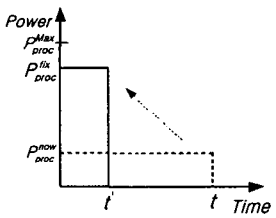
세 번째 단계는 프로세서 속도 결정단계이다. 앞 절에서 정한 정책에 따라 프로세서의 속도를 결정한다.

디바이스 최적화 정책은 그림4에서 보듯이 디바이스의 사용시간을 줄임으로써 에너지 소비를 줄이는 방식이다. 이때 프로세서의 속도의 증가로 에너지 손실과 이익이 발생함으로 식(3),(4)을 통하여 에너지 이익과 손실을 구한다.

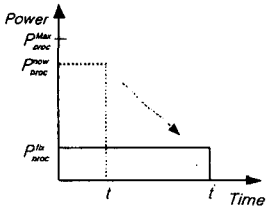
$$E_{gain} = P_{dev}(t - t') = \frac{f_{fix} - f_{now}}{f_{fix}} P_{dev} t \qquad (3)$$

$$E_{loss} = P_{proc}^{fix} t' - P_{proc}^{now} t = \frac{(f_{fix})^2 - (f_{now})^2}{(f_{now})^2} P_{proc}^{now} t \qquad (4)$$

식(3)은 디바이스의 작동시간이 단축됨으로써 발생하는 에너지 이익을 나타내며, 식(4)는 프로세서의 속도 변화로 인해 발생하는 에너지 손실을 나타낸다. 이를 바탕으로 $\Delta E = E_{gain} - E_{loss}$을 구한다. 이 때 P_{dev}, P_{now}, f_{now}, t값은 이미 존재하므로, 시스템이 지원하는 f_{fix}값들을 대입함으로써 ΔE를 최대로 하는 f_{fix}을 정한다.



[그림4]디바이스 최적화 정책



[그림5] 프로세서 최적화 정책

프로세서 최적화 정책은 그림5처럼 프로세서의 속도를 낮춤으로써 에너지 이익을 얻는 방식이다. 속도를 변화시켰을 경우의 에너지 이익과 손실을 구하여, 에너지 이익이 최대가 되는 프로세서의 속도를 결정한다.

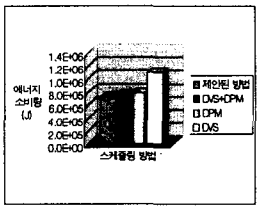
$$E_{gain} = P_{proc}^{now} t - P_{proc}^{fix} t' = \frac{(f_{now})^2 - (f_{fix})^2}{(f_{now})^2} P_{proc}^{now} t \qquad (5)$$

$$E_{loss} = P_{dev}(t' - t) = \frac{f_{now} - f_{fix}}{f_{fix}} P_{dev} t \qquad (6)$$

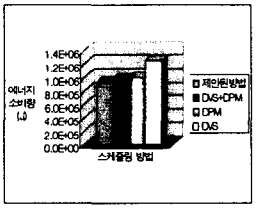
식(5)은 프로세서의 속도 감소로 얻는 에너지 이익을 나타내며, 식(6)은 디바이스 사용시간 증가로 인한 에너지 손실을 나타낸다. 이를 바탕으로 $\Delta E = E_{gain} - E_{loss}$을 최대로 하는 수행속도를 결정한다. 이때 결정된 속도는 태스크의 시간제약 조건을 만족시켜야 한다. 시간제약 조건이 만족되지 경우 속도를 다시 결정한다. 이렇게 두 가지의 정책에 따라 프로세서의 속도를 결정함으로써 상황에 맞는 프로세서의 속도를 결정할 수 있다. 뿐만 아니라 디바이스를 사용하지 않은 태스크는 slack time을 이용하여 프로세서 속도를 낮출 수 있게 된다.

4. 성능평가

제안된 알고리즘의 효율성은 Intel Xscale PXA250을 탑재한 임베이드 보드를 사용하여 검증하였다. 보드에 탑재된 운영체제는 BCET와 WCET값을 태스크에 전달할 수 있도록 실시간 운영체제인 eCos를 사용하였다. 이러한 기반위에 BCET/WCET 비율을 변화시켜 가면서 사용된 에너지양을 비교하였다. 이때 태스크의 마감시간과 다음 주기는 프로세서를 최대 공급 전압으로 하였을 때 소모되는 최악 수행시간을 기준으로 하였다.



[그림6] BCET/WCET이 0.5



[그림7] BECT/WCET이 0.7

그림 6,7은 BECT/WCET 비율이 0.5와 0.7일 때 에너지 소비량을 나타내었다. 제안된 기법은 디바이스를 고려하면서 slack time을 이용하여 스케줄링 함으로써 BCET/WCET이 0.5과 0.7일 때 최악의 경우인 DVS보다 26~30%의 에너지를 절약한다. DVS는 디바이스 소비 에너지를 고려하지 않고 프로세서 속도를 낮추어 스케줄링 함으로써 전체 에너지 소비가 증가되었고, DPM은 시스템에서 에너지 비중이 낮은 디바이스를 고려하지 않음으로써 에너지 소비가 증가되었다. 이 실험은 임베이드 보드를 기반으로 하기 때문에 다양한 디바이스를 고려하지 못했다. 이에 리눅스를 기반으로 ACPI을 활용하여 다양한 디바이스를 사용하는 경우를 성능평가하기 위해 연구가 진행 중이다.

5. 결론

본 논문에서는 모바일 기기에서 유한한 전력량을 가지고 효율적으로 스케줄링 함으로써 가용시간을 최대화 시키는 스케줄링 기법을 제안했다. 그리고 실험을 통해 시스템의 특정 요소에 초점을 맞추어 스케줄링 하는 기존의 방법보다 디바이스의 사용 전력량과 현재 프로세서 속도에 따라 상황에 맞게 스케줄링 함으로써 보다 효과적으로 가용한 에너지를 활용함을 확인할 수 있다.

참고문헌

[1] Padamnabhan Pillai,Kang G. Shin, "Real time Dynamic voltage scaling for low power embedded operating systems", In Proceedings of the 18th ACM symposium on Operating System Principles(SOSP-01), pp. 89-102, 2001

[2] Benini, et al., "A survey of Design Techniques for System-Level Dynamic Power Management", IEEE Transactions on VLSI, pp. 219-316, Jun. 2000

[3] Johan Pouwelse, Koen Langendoen, Henk Sips, "Energy priority scheduling for variable voltage processors", In Proceeding of Low Power Electronics and Design International Symposium, pp. 8-33, 2001

[4] M. Srivastava, A. Chandrakasan, and R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation", IEEE Transactions on VLSI Systems, pp. 42-55, Mar. 1996

[5] K. Li, R. Kumpf, P. Horton and T. Anderson. "A Quantitative Analysis of Disk Drive Power Management in Protable Computers", In Proceeding of USENIX Winter Conference, pp. 279-292, 1994

[6] G. A. Paleolgo, L. Benini, A Bogliolo, and G. D. Micheli, "Policy Optimization for Dynamic Power Management", In Proceedings of Design Automation Conference, pp. 182-187, 1998

[7] HTTP://www.acpi.info