임베디드 응용프로그램의 동작 특성을 이용한 에너지 인식 스케쥴링 기법†

(Energy-Aware Scheduling Technique to Exploit Operational Characteristic of Embedded Applications)

한 창 혁*, 유 준 혁**
(Changhyeok Han and Joonhyuk Yoo)

요 약 효율적인 전력관리는 모바일 휴대용 제품 시장에서 중요한 경쟁력 중의 하나이다. 본 논문은 전력을 미리 예측하는 방법으로 실행중인 응용프로그램의 sleep 상태를 이용하는 Energy-Aware Scheduling policY(EASY)를 제안한다. 기존 대기 모드에서 전력소모를 줄이는 방법과의 차이점은 응용프로그램들이 얼마나 오랫동안 스케쥴러에서 sleep 상태에 있었는지를 검사하여 각 응용프로그램들의 동작 상태를 결정한다. EASY 기법은 측정된 sleep 시간을 기준으로 현재의 작업량에 맞는 적정한 CPU 클럭 주파수를 정하고, 다음 작업량의 적정한 CPU 클럭 주파수를 예측함으로서 동작 상태에서 전력 소모를 줄일 수 있다. 실험 결과 기존의 대기모드를 이용한 전력관리 기법과 비교하여 평균적으로 10-30%의 전력소모를 줄임으로써 제안된 기법의 우수성을 입증한다.

핵심주제어 : Energy-aware 스케쥴링 정책, sleep 시간, 클럭 주파수 예측

Abstract Efficient power management plays a crucial role to strengthen competitiveness in the market of portable mobile commodities. This paper presents a proactive power management technique, called by Energy-Aware Scheduling policY (EASY), to exploit the sleep time information of running applications. Different from previous power management approaches focusing on power conservation in standby mode, the proposed scheme characterizes each application program's operational characteristic in active mode by observing how long the task stays in sleep state of CPU scheduler. Based on the measured sleep time, the proposed EASY speculates an adequate CPU clock frequency according to the current CPU workload and scales the frequency directly to the predicted one. Experimental results show that the proposed scheme reduces the power consumption by 10-30% on average compared to traditional DPM approach, with a minimal impact on the performance overhead.

Key Words : Energy-aware scheduling, sleep time, clock frequency speculation

1. 서 론

최근 임베디드 하드웨어 기술의 발전과 무선네트워크 성능의 향상으로 스마트폰, 아이패드, e-리더 등과 같은 휴대용 모바일 제품 시장이 급격하게 증가하고 있다. 사용자에게 편리한 서비스를 제공하기 위한 다

† 이 논문은 2010년 한국연구재단의 신진과제 연구비 지원에 의해 연구되었음.
* 삼성전자(주) Global GSM 그룹, 제1저자
** 대구대학교 정보통신공학부, 교신저자

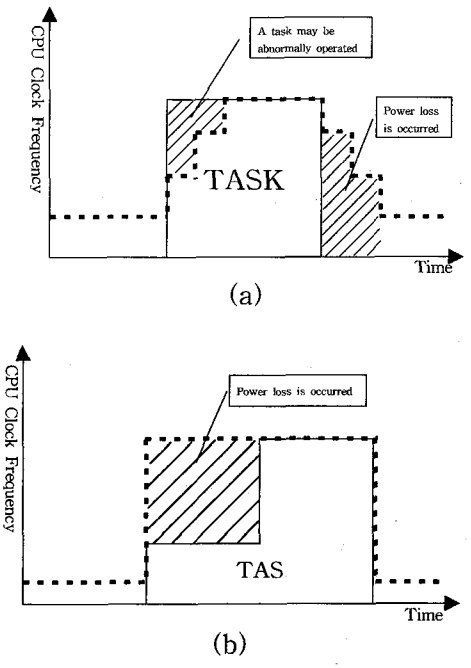
양한 기능들과 복잡한 응용프로그램들의 멀티태스킹 구동 등을 지원하면서 이들 제품의 전력소모도 빠르게 증가하고 있다. 그러므로 제한된 배터리 용량에 대처하기 위한 효율적인 전력관리 기술은 휴대용 모바일 기기의 시장경쟁력을 강화시키는 가장 중요한 요소가 되고 있다[1, 2, 3].

본 논문은 휴대용 모바일 기기의 효율적인 전력관리를 위해 응용 프로그램의 동작 특성을 기반으로 하여 Energy-Aware Scheduling policY (EASY)에 기반한 전력관리 기법을 제안한다. 본 논문에서 제안하는 EASY는 응용 프로그램이 실행, 동작, 종료 될 때 각각의 응용 프로그램의 동작 특징을 분석하여 동적으로 활성모드에서 적절한 CPU 클럭 주파수를 설정함으로 전력낭비를 줄이도록 하였다. 이는 제한된 베터리 용량의 휴대용 모바일 기기에서 효과적으로 전력관리를 할 수 있는 방안이 될 수 있다.

본 논문의 구성은 서론에 이어 2장에서는 기존의 전력관리 기법인 DPM과 OPM에 대해서 알아보고 3장에서는 EASY의 주파수 예측 알고리즘과 어떻게 스케줄러에 적용/구현되어 있는지 설명하였다. 4장에서는 제안된 EASY의 평가를 위한 실험 방법을 설명하였고 마지막으로 5장과 6장에서는 제안된 EASY의 실험 결과를 분석, 결론을 논하였다.

2. 관련 연구

기존의 전통적인 전력관리 방식은 Dynamic Power Management(DPM)와 같이 대기상태(standby mode)의 소비전력을 줄이기 위하여 프로세서의 휴지시간(idle time)을 기준으로 DVS/DFS 방법을 통해 CPU의 전원전압이나 클럭 주파수를 조정한다[5, 6, 7]. DPM은 CPU의 부하량에 따라 단계적으로 클럭 주파수를 증가 또는 감소시키는 방식을 취하고 있다. 응용프로그램을 실행했을 때 충분히 빠른 클럭 주파수로 동작하지 않으면 비정상적인 동작이 발생할 가능성이 있고, 응용프로그램을 종료했을 때는 곧바로 원하는 클럭 주파수로 감소되는 것이 아니라 단계별로 조정되기 때문에 그림 1(a)에서와 같이 불필요한 전력소모가 발생한다[4]. 예를 들면, 스마트폰과 같이 응용프로그램의 생성과 종료가 자주 일어나는 멀티태스킹 환경에서 이와 같은 클럭 주파수의 단계별 조절로 인한 전력관리의 비효율성도 더욱 증가하게 된다.



<그림 1> 기존 방식의 문제점
((a) DPM, (b) OPM)

DPM이 CPU 대기상태(standby mode)의 전력소모를 줄이는 데 관심이 있다면, 최근 OPM (Operation characteristics based dynamic Power Management)과 같은 활성상태(active mode)에서의 전력관리 기법이 소개된 바 있다[3]. OPM은 전력소비를 줄이기 위해 Framework layer에서 각 응용 프로그램마다 요구되는 CPU 클럭 주파수를 정적으로 설정한 후, 해당 응용 프로그램이 실행될 때 마다, 미리 설정된 CPU 클럭 주파수를 설정하는 방식을 사용하고 있다. 해당 방식은 응용 프로그램의 상태와 상관없이 고정적인 CPU 클럭 주파수를 사용하기 때문에 그림 1(b)에 도식한 바와 같이 불필요한 전력손실이 발생하는 상황이 발생하며 사용자가 일일이 애플리케이션을 종료시켜야 하는 경우가 생긴다. 예를 들면, media player를 실행할 경우, 프로그램은 재생 상태와 정지 상태가 존재한다. 정지 상태에서는 많은 전력이 필요 없지만, OPM의 경우 정지 상태에서도 재생 상태와 동일한

CPU 클럭 주파수를 사용하도록 설정하기에 개선의 여지가 많다. 더불어 스마트폰과 같이 애플리케이션을 빈번히 설치하는 경우 OPM 방식으로는 적절히 대응할 수 없다는 단점이 있다.

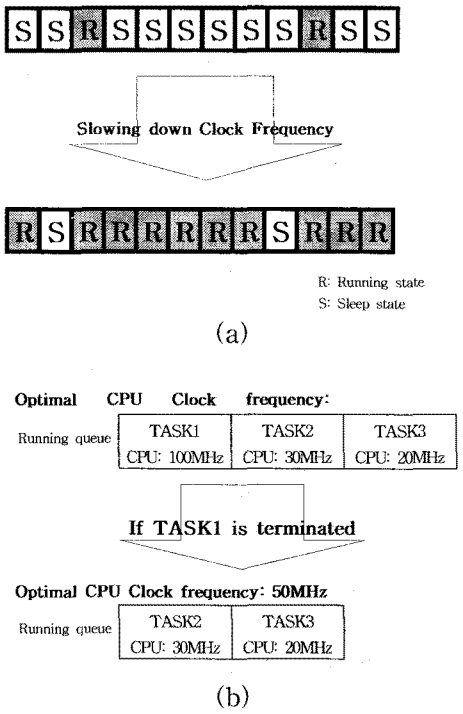
배터리 기반으로 동작하는 대부분의 임베디드 기기는 DPM 또는 OPM을 사용하고 있다. 하지만 DPM의 전력관리 방법에서는 대기상태에서 소비전력을 줄이기 때문에 멀티미디어 스트리밍 서비스, Text mesasge, 이메일, 웹 브라우징 등과 같은 응용 프로그램의 경우 동작 상태와 대기상태가 반복적으로 나타나는 경우에 추가적인 전력소비를 줄일 수 있는 여력이 존재한다. 또한 OPM의 전력관리 방법에서는 응용 프로그램의 동작 상태의 고려 없이 일정한 CPU 클럭 주파수를 사용하므로, music player, video player와 같이 파일 재생, 정지로 전력을 사용하지 않는 구간과 전력을 소모하는 구간이 나누어져 있는 응용 프로그램에서 추가적인 전력소비를 줄일 수 있는 여력이 존재한다.

3. 에너지 인식 스케쥴러

본 논문에서 제안하는 EASY는 효율적인 전력관리를 위해서 스케쥴러에서 응용 프로그램의 sleep 상태를 줄이는데 초점이 맞추어져 있다. CPU 클럭 주파수를 설정하기 위해 발생한 인터럽트의 시간과, 각 인터럽트간의 간격동안 running 상태에서 머문 시간, ready 상태에 머문 시간 그리고 sleep 상태에 머문 시간을 기준으로 다음 주기의 CPU 클럭 주파수를 설정한다. 그림 2(a)에서 볼 수 있듯이 응용 프로그램이 running 상태의 시간이 짧고 sleep 상태가 길면 CPU 클럭 주파수를 낮추어 running 상태의 시간을 늘림으로써 동작 상태에서의 전력 소모를 줄일 수 있다.

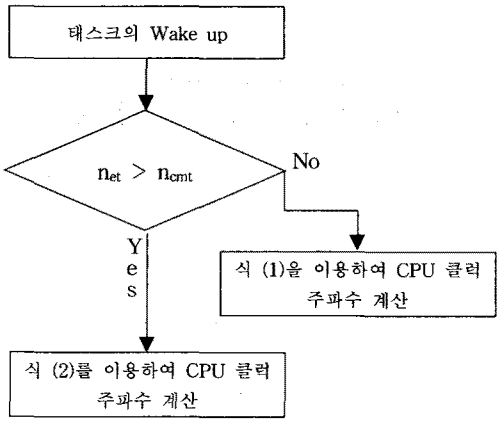
무엇보다도 EASY는 멀티태스킹 환경에서도 전력관리를 효율적으로 할 수 있는 장점이 있다. 기존의 DPM은 각각의 응용 프로그램이 요구하는 적정한 CPU 클럭 주파수 없이 전체적인 CPU 부하량을 기준으로 CPU 클럭 주파수를 설정하지만, EASY는 각각의 응용 프로그램에서 필요로 하는 CPU 클럭 주파수를 이전의 동작 특성을 근거로 미리 예측함으로써 그림 3에서 볼 수 있듯이 멀티태스킹 환경에서 특정 응용 프로그램이 생성/수행/종료되었을 때 적절한 CPU 클럭 주파수로 바로 설정할 수 있다.

응용 프로그램은 CPU의 성능에 의존적인 CPU-bound와 인터럽트와 이벤트에 의존적인 I/O-bound로 분류 할 수 있다. 전자의 경우 복잡한 수학연산, 데이터베이스 처리 등 CPU 의존적인 응용 프로그램이고, 후자의 경우 데이터 패킷, 키보드 입력 등 인터럽트를 발생시키는 외부 입력에 의존적이다. CPU-bound 응용 프로그램의 경우 CPU 클럭 주파수를 높여서 가능한 한 해당 응용 프로그램을 빨리 실행하고 종료하는 것이 전력소모를 줄이지만, I/O-bound 응용 프로그램의 경우 CPU 클럭 주파수를 응용 프로그램의 인터럽트 발생 속도에 맞추어 조정하는 것이 전력 소모를 줄일 수 있는 방법이다. 예를 들어 사용자가 워드 프로세서 프로그램을 동작시킬 경우 워드 프로세서는 대부분의 스케쥴러의 상태가 sleep 상태에 놓이게 될 것이다. 이 경우 CPU 클럭 주파수를 낮추더라도 워드 프로세서의 성능 저하 없이 전력 소모를 줄일 수 있다.



<그림 2> EASY의 기본 개념

본 논문에서 제안하는 EASY는 스케줄러에서 태스크가 Wakeup 하여 sleep 상태에서 ready/running 상태로 변화될 때, 이전에 수행되었던 태스크의 CPU 클럭 주파수, running/ready/sleep 상태 그리고 발생한 인터럽트의 시간을 이용하여 다음 태스크를 위한 최적의 CPU 클럭 주파수를 예측한다. 본 논문에서는 단순화를 위해 한 개의 응용 프로그램은 한 개의 태스크를 가지고 있고, 태스크는 스케줄러에서 ready, running, sleep 상태 중 한 곳에 있다고 가정한다. 그림 3과 같이 태스크가 Wake up 될 때, 전력소모의 최적화를 위한 CPU 클럭 주파수 예측과 성능 저하를 최소화하기 위해서 두 가지의 CPU 클럭 주파수를 정하는 알고리즘을 제안한다. 태스크의 종료시간과 새로운 인터럽트가 발생한 시간을 비교하여, 태스크의 종료시간이 더 크면, CPU 클럭 주파수를 높이는 주파수 예측 알고리즘을 선택하고, 인터럽트 발생시간이 더 크면 CPU 클럭 주파수를 낮추는 주파수 보정 예측 알고리즘을 선택한다. 각각 태스크의 적정한 CPU 클럭 주파수가 예측 된 후, 스케줄러에서 다음 태스크들의 작업이 실행될 때의 적정한 CPU 클럭을 예측하기 위해, ready/running 상태에 있는 태스크들에게 설정된 CPU 클럭 주파수의 합을 구해서 OS에서 사용되어야 할 전체 적정 CPU 클럭 주파수를 설정한다.

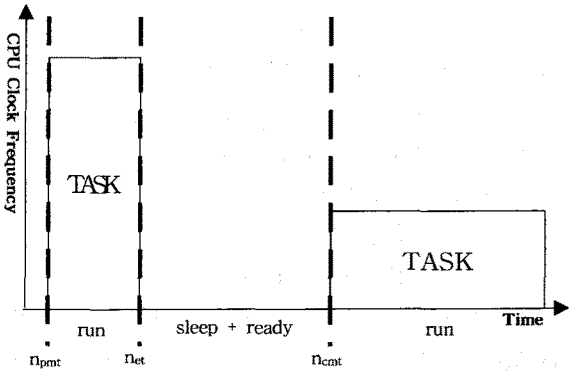


<그림 3> 태스크 Wake up의 순서도

3.1 주파수 예측 알고리즘

주기적인 태스크의 작업이 반복될 때 CPU 클럭 주파수는 이전의 작업량을 통해 다음 작업의 적정 CPU 클럭 주파수를 예측할 수 있다. 예를 들어 멀티미디어 스트리밍 서비스의 경우, 주기적으로 네트워크로부터 데이터 패킷이 들어오게 된다. 이 때, 들어온 데이터 패킷은 다음 데이터 패킷이 도착하기 전까지 처리를 하면 성능 저하 없이 동작이 가능하다. 이 예시에서와 같은 경우, 주기적인 인터럽트가 발생하기 때문에 스케줄러에서 CPU 클럭 주파수를 설정할 때, 이전 태스크의 작업량과 인터럽트의 **시간간격을 이용해서 적정**한 CPU 클럭 주파수의 예측이 가능하다. 그 결과 새로운 인터럽트가 발생하기 전까지 이전의 태스크가 해야 할 일을 끝낼 수 있다.

그림 4는 제안된 주파수 예측 알고리즘을 위한 파라미터를 나타내는데, 새로운 인터럽트가 발생한 시간을 n_{cmt}, 이전의 인터럽트가 발생한 시간을 n_{pmt}, 이전의 태스크가 실행 종료된 시간을 n_{et}, 예측할 CPU 클럭 주파수를 $f(n_{cmt})$라고 정의한다. 새로운 인터럽트가 발생한 시간이 이전의 태스크가 종료된 시간보다 크면, 식 (1)을 사용하여 현재 태스크에 대한 적정한 CPU 클럭 주파수를 예측한다. 태스크가 running 상태에 있을 때 사용된 CPU 클럭 주파수의 합($\sum_{i=n_{pmt}}^{n_{et}} f(i)$)을 구한 뒤, running 상태에 머문 시간, ready 상태에 머문 시간, sleep 상태에 머문 시간의 합으로 나눈다. 이 때 running 상태, ready 상태, sleep 상태의 합은 새롭게 들어온 인터럽트 시간에서 이전 인터럽트 시간을 뺀 값($n_{cmt} - n_{pmt}$)과 동일하다.



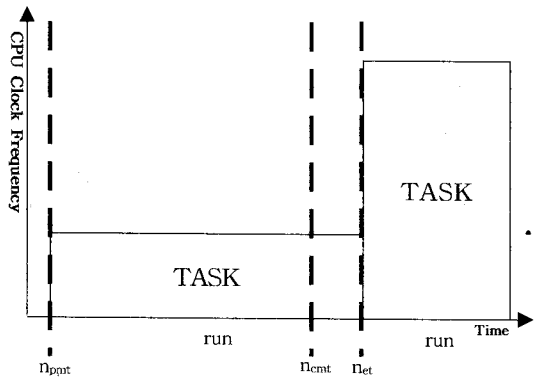
<그림 4> CPU Clock Frequency 설정법

$$f(n_{cmt}) = \frac{\sum_{i=n_{pmt}}^{n_{et}} f(i)}{n_{cmt} - n_{pmt}} \qquad (1)$$

3.2 주파수 보정 예측 알고리즘

태스크에 들어오는 인터럽트가 항상 주기적이지는 않다. 예를 들어, 워드 프로세스 프로그램의 경우, 사용자가 키 입력을 빨리 할 수도 있고, 천천히 할 수도 있다. 사용자가 키 입력을 갑자기 빨리 하는 것처럼, 인터럽트가 비주기적으로 발생을 하게 되는 상황에서, 식 (1)로 CPU 클럭 주파수를 계산하게 되면, 이전 태스크가 종료되기 전에 새로운 인터럽트가 발생하게 된다. 그림 5는 태스크 종료 시간보다 새로운 인터럽트가 더 빨리 와서 CPU 클럭 주파수를 높이는 상태를 보여준다. 이 상황이 반복되면, running queue에 인터럽트가 처리되지 못하고 적체가 되며 성능저하가 발생한다.

외부 인터럽트가 갑자기 빨리 입력 될 경우 새로운 인터럽트가 발생하는 시간이 이전 태스크가 종료되는 시간 보다 빠르게 된다. 이 경우 식 (2)를 사용하여 CPU 클럭 주파수를 높이게 된다. 새롭게 들어온 인터럽트의 시간에서 이전의 인터럽트의 시간을 뺀 값이 각각의 태스크가 수행되어야 할 시간이다. 하지만 running 상태에서 동작한 시간이 각 인터럽트 간의 시간 $n_{cmt} - n_{pmt}$ 보다 크므로, 각 인터럽트 간의 시간의 차와 태스크가 종료된 시간에서 새롭게 들어온 인터럽트 시간의 차를 뺀 값으로 보정해 준다. 이렇게



<그림 5> CPU Clock Frequency 보정법

함으로써 태스크가 동작을 끝내야 하는 시간$n_{et} - n_{cmt}$ 만큼 줄었으므로 CPU 클럭 주파수가 높아지게 된다.

$$f(n_{cmt}) = \frac{\sum_{i=n_{pmt}}^{n_{et}} f(i)}{(n_{cmt} - n_{pmt}) - (n_{et} - n_{cmt})} \qquad (2)$$

3.3 EASY 스케쥴링 정책

<표 1> EASY 스케쥴러 Pseudo 코드

```
If there is a ready/running task then
   While(ready task is not null)
      Find a next task having highest priority  --------①
   Exit
Else
   Select idle task
End if

If selected task is not idle task then
   Task->activeStateCount++  --------②
      If task has an additional message or event then
         All the task clock frequencies are set as one step
higher clock frequency
      End if
   Record CPU clock frequency of selected task
End if

While(ready task is not null)
   sum speculated CPU Clock frequency of all tasks
Exit  --------③
```

본 논문에서 제안하는 EASY의 스케쥴러 정책에 대한 구현은 표1과 같다. ready, running 상태에 존재하는 task 대기 열에서 가장 높은 우선순위를 가진 task를 선택한 뒤(①), 해당 태스크가 얼마나 running 상태에 있었는지 Task Control Block(TCB)의 activeStateCount 변수에 기록한다(②). activeStateCount 변수는 태스크가 wake up 할 때 n_{et}의 값으로 이용되며 이 후 running 상태에서 사용한 CPU 클럭 주파수의 합 $\sum_{i=n_{pmt}}^{n_{et}} f(i)$를 구하는 데 사용된다. 그 후 현재 ready, running 상태에 있는 각 task들의 예측된 CPU 클럭 주파수를 합하여(③) 적정한 CPU 클럭 주파수를 구하게 된다.

4. 실험 환경

제안된 EASY 기법을 평가하기 위해 EZ-X5의 H/W platform, serial cable의 연결을 위해 Desktop 컴퓨터 그리고 소모 전력 측정을 위해 multimeter 기기를 사용하였다(그림 6).

<그림 6> 테스트 환경

EZ-X5는 FALINUX에서 만든 임베디드 보드로 400MHz PXA255 ARM-core CPU, 64MB SDRAM, 512KB Boot Flash, 64MB NAND Flash를 가지고 있고 전압으로는 5V를 사용한다. EZ-X5에서 지원하는 CPU clock table은 표 2와 같다[8].

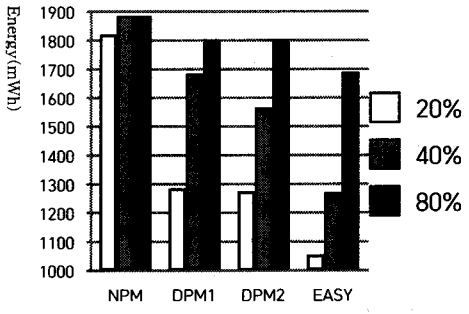
<표 2> EZ-X5에서 제공하는 CPU clock frequency

Clock levels	Clock frequency
Level 0	99.5MHz
Level 1	132.7MHz
Level 2	199.1MHz
Level 3	265.4MHz
Level 4	331.8MHz
Level 5	398.1MHz

기존의 상용 OS에서는 DPM, 회로전력 차단 기법, deep sleep mode 기법 등 여러 가지 전력관리 정책이 사용되고 있고, 여러 전력관리 정책이 상호 영향을 줄 수 있기 때문에 EASY 기법만의 효과적인 측정이 힘들다. 따라서 본 논문에서는 EZOS라는 자체 제작한 OS를 만들어 EASY와 DPM 그리고 전력관리를 하지 않았을 때를 비교하여 실험하였다. EZOS는 32개의 멀티태스킹이 가능하고, 비선점형 커널과 multi-level round-robin EASY scheduler를 커널 내부에 구현하였다. 평가를 위해 채택된 EZOS는 POSIX 인터페이스를 따르고 있기 때문에 다른 운영체제에서 작성한 프로그램을 이식하기가 용이하고, 복합적인 전력관리 기법을 채용하고 있지 않기 때문에 본 실험에 영향을 줄 수 있는 여러 가지 전력관리 기법의 상호 의존성을 제거할 수 있어 정확한 각 전력관리 기법간의 실험 및 비교가 가능하다는 장점이 있다.

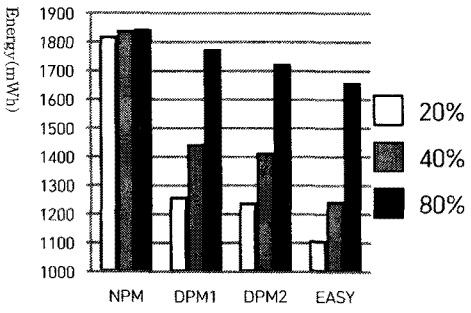
5. 실험 결과

본 논문에서는 single-tasking 환경에서 CPU 부하량을 20%, 40%, 80%를 주었을 때와, multi-tasking 환경에서 CPU 부하량을 20%, 40%, 80% 주었을 때로 분류하여 EASY의 전력관리 정도를 측정하였다.



<그림 7> Single-tasking 환경에서 전력 소비의 비교

single-tasking 환경에서는 한 개의 태스크가 주기적으로 sleep 상태와 running 상태가 바뀌는 것을 2분 동안 동작하도록 하여 측정하였고, multi-tasking 환경에서는 3개의 태스크가 비주기적으로 2분 동안 동작하도록 하여 소모된 전력과 태스크의 작업이 종료된 시간을 측정하였다. 본 논문에서는 EASY의 성능 비교를 위해, EASY(EASY), CPU 클럭 주파수의 단계 주기를 1초로 하는 DPM(DPM1)과 0.5초로 하는 DPM(DPM2), 그리고 전력관리를 하지 않고 CPU 클럭 주파수를 최고 높게 설정한 상황(NPM)과 비교하여 실험하였다. 본 논문은 CPU의 에너지 소비량과 각각의 전력관리 방식에서 태스크가 종료됐을 때의 시간을 mWh와 second 단위로 비교하였다.



<그림 8> Multi-tasking 환경에서 전력 소비의 비교

본 논문에서 제안된 EASY는 그림 7의 결과처럼 single-tasking 환경에서 기존의 DPM1, DPM2, NPM과 비교하여 전력 소비에 대해 평균적으로 각각 16.4%, 14%, 28.3%의 감소가 있었다. 또한, multi-tasking 환경에서는 그림 8과 같이 EASY는 기존의 대기모드를 이용한 전력관리 기법에 비해 각각 14.1%, 8.8%, 27.1%의 전력소모가 감소하였다. 한편, EASY는 태스크의 실행 종료 시간을 기준으로 약 2-5% 정도의 소폭의 시간 지연이 관측되었지만, 이는 전반적인 전력 소모의 감소량에 비해 무시할 만한 수준이다. 그러므로 제안된 EASY가 응용프로그램의 동작 특성을 이용한 클럭 주파수 예측 기법을 채용하여 저전력 임베디드시스템의 전력관리 효율성을 대폭 개선했음을 확인할 수 있다.

6. 결 론

본 논문은 동작 모드에서 전력 관리를 위해서 새로운 Energy-Aware Scheduling policY (EASY)를 제안하였다. EASY 기법은 각각의 태스크가 각각의 인터럽트 간격동안 스케쥴러에서 ready, running, sleep 상태에 있는지를 측정하여 적절한 CPU 클럭 주파수를 설정하고, 다음 작업량에 대한 클럭 주파수를 예측함으로서 동작모드에서 전력 소모를 줄일 수 있다. 실험결과는 제안된 EASY가 기존의 대기모드를 이용한 전력관리 기법과 비교하여 평균적으로 10-30%의 전력 소모를 줄임으로써 제안된 기법의 우수성을 입증하였다.

참 고 문 헌

[1] Y. Fei, L. Zhong and N. K. Jha, "An Energy-Aware Framework for Dynamic Software Management in Mobile Computing systems", ACM Transactions on Embedded Computing Systems, Vol. 7, No.3, Article 27, Publication date: April 2008.

[2] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications", School of Computer Science, Carnegie Mellon University, SOSP, Kiawah Island, SC, 1999.

[3] S. Lee, "A Power Management Method Using Application Operating Characteristics in Mobile Phones", KUGT program in Korea University, Master Thesis, 2009.

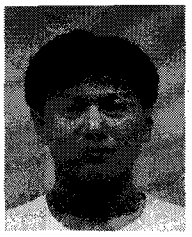
[4] P. Rong and M. Pedram, "Energy-Aware Task Scheduling and Dynamic Voltage Scaling in a Real-Time System", Int'l Journal of Low Power Electronics, American Scientific Publishers, Vol. 4, No. 1, Apr. 2008.

[5] T. Simunic, L. Benini, A. Acquaviva, P. Glynn and G. De Micheli, "Dynamic Voltage Scaling and Power Management for Portable Systems", in Proc. Of Design Automation Conference, June 2001.

[6] F. Yao, A. Demers and S. Shenker, "A Scheduling Model for Reduced CPU Energy" in Proc. USENIX Symposium on Operating System Design and Implementation, pp. 13-23, 1994.

[7] W. Yuan and K. Nahrstedt, "Energy-efficient CPU Scheduling for Multimedia Applications", ACM Transactions on Computer Systems, June 2005.

[8] http://forum.falinux.com/



한 창 혁 (Changhyeok Han)

- 비회원
- 대구대학교 컴퓨터IT공학부 전산공학전공 공학사
- 삼성전자 Global GSM 개발팀 선임연구원
- 관심분야 : 운영체제, 임베디드 시스템

유 준 혁 (Joonhyuk Yoo)

- 정회원
- 포항공과대학교 전자전기공학과 공학사
- 포항공과대학교 전자전기공학과 공학석사
- 미국 매릴랜드대학교 컴퓨터공학과 공학박사
- 대구대학교 정보통신대학 정보통신공학부 임베디드 시스템전공 교수
- 관심분야 : 임베디드 소프트웨어, 컴퓨터구조, 사이버-물리 시스템

논 문 접 수 일 : 2010년 09월 18일
1차수정완료일 : 2010년 11월 22일
2차수정완료일 : 2011년 01월 10일
게 재 확 정 일 : 2011년 01월 24일