논문 2009-46SD-12-14

응용 프로그램 특성을 고려한 동적 전압 조절 기법

(Dynamic Voltage Scaling Technique Considering Application Characteristics)

조 영 진*, 장 래 혁**

(Youngjin Cho and Naehyuck Chang)

요 약

일반적인 동적 전압 조절(dynamic voltage scaling)의 가정과는 다르게 실제 시스템에 있어서는 응용 프로그램의 성능이 프로세서의 동작 속도에 정비례하지 않는다. 본 연구에서는 응용 프로그램의 성능과 동작 속도의 관계를 실측을 통하여 수치화하여 응용 프로그램의 특성을 모델링하고 각기 다른 응용 프로그램 특성 계수를 갖는 태스크 집합에 적합한 스케줄링 기법을 제시하였다. 또한, 모든 태스크의 단위 수행시간 변화에 따른 에너지의 변화량이 동일해야 에너지 최적이 된다는 해석적인 정리를 제시하였다. 본 연구에서 제시하는 스케줄링 기법은 이러한 해석적 정리에 기반을 두기 때문에 항상 각 태스크에 시스템 에너지 최적이 되는 조절비를 제시한다. 합성 태스크 집합을 이용한 실험결과에서 기존 연구 대비 약 7%의 추가적인 에너지 절감 효과를 얻을 수 있었다.

Abstract

In the real system environments, the performance of the application is not linearly proportional to the clock frequency of the microprocessor, in contrast to the general assumption of conventional dynamic voltage scaling. In this paper, we analytically model the relation between the performance of the application and the clock frequency of the microprocessor, and introduce the energy-optimal scheduling algorithm for a task set with distinct application characteristics. In addition, we present a theorem for the energy-optimal scheduling, which the derivative of the energy consumption with respect to the execution time should be the same for all the tasks. The proposed scheduling algorithm always generates the energy-optimal scaling factor thanks to the theorem for energy-optimal scheduling. We achieved about 7% additional energy reduction in the experiments using synthetic task sets.

Keywords: 동적 전압 조절, DVS, 스케줄링, 전력 소모 절감

I. 서 론

배터리로 동작하는 휴대용 내장형 시스템에 있어서 전력 소모의 효율적인 관리는 매우 중요한 문제이다. 동적 전압 조절(dynamic voltage scaling)은 가장 효과적인 전력 소모 절감 기법 중 하나이다. 이 동적 전압 조절에 관련하여 다수의 논문들이 발표되고 많은 연구가 수행되었으나 기존의 접근에는 몇 가지 근본적인 문제점이 있다.

먼저, 대부분의 연구가 프로세서 자체의 전력 소모 절감에 초점을 맞추고 있다. 동적 전압 조절을 프로세서의 동작 주파수와 공급 전압을 낮추어 전력소모를 절감하는 기법이기 때문에, 주변 장치에 있어서는 오히려 전력 소모가 증가하는 현상이 발생할 수 있다. 그에 몇 가지 기존 연구에서는 다양한 주변장치들, 예를 들어 메모리 시스템이나 DC-DC 컨버터 같은 전력 공급 장치의 효율을 고려한 최적의 동작 주파수를 찾고자 하는

* 정회원, 삼성전자
(Samsung Electronics)
** 정회원, 서울대학교 전기컴퓨터 공학부
(Seoul National University, Dept of EECS)
※ 이 논문은 2009년도 BK21과 한국학술진흥재단의 지원을 받아 연구되었음 (KRF-2009-013-D00099, 2009-0060054)
접수일자: 2009년10월27일, 수정완료일: 2009년12월3일

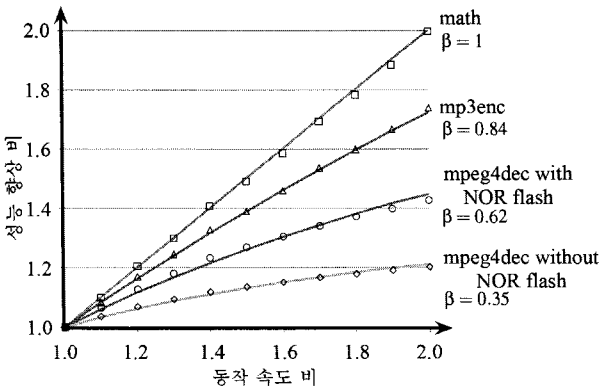

그림 1. 프로세서의 동작 속도 변경에 따른 성능의 변화(클럭 단위 정확도를 지닌 시뮬레이터를 통한 결과)

Fig. 1. Performance variation caused by the clock frequency changing of the microprocessor.

노력들이 있었고 이 연구들은 시스템 전체의 에너지를 고려하여 최적화를 수행하였을 때만 시스템 수준의 전력 소모 절감을 얻을 수 있음을 보여주었다[1~2].

이러한 시스템 수준의 기법들 또한 응용 프로그램의 특성은 고려하지 않고 시스템 특성만을 고려하여 최적화를 수행하였다. 하지만 그림 1에서 보다시피 동작 주파수에 따른 시스템의 에너지 소모는 응용 프로그램에 따라 같은 시스템 구성이라고 하여도 각기 다른 모양을 보이게 된다[3]. 그림 1의 결과는 TI OMAP OSK 플랫폼을 이용하여 실측한 결과임을 밝혀둔다. 이처럼 응용 프로그램에 따라 에너지 비용함수의 모양이 각기 달라지면 대표적인 동적 전압 조절 알고리즘인 Yao 알고리즘과 같이 각 태스크의 동작 주파수를 같은 값으로 가능한 한 낮게 만드는 방법은 더 이상 유효하지 않게 된다[4~5]. 따라서 새로운 스케줄링 방법이 필요하게 된다.

본 연구에서는 앞에 살펴본 바와 같이 응용 프로그램의 특성이 각기 달라지는 원인을 분석해서 수치화하고 이러한 환경에서의 최적 스케줄링 방법을 살펴본다. 본 연구 결과 응용 프로그램의 특성을 이용하여 최적화를 수행할 경우 기존의 동적 전압 조절 기법 대비 7%의 추가적인 전력 소모 절감을 얻을 수 있음을 확인하였다.

Ⅱ. 본 론

1. 응용 프로그램의 특성이 동적 전압 조절에 미치는 영향 분석

동적 전압 조절 기법에 있어서 가장 보편적인 가정 중에 하나는 응용 프로그램의 성능이 프로세서의 동작 속도에 정비례한다는 것이다[4~5]. 이 가정은 그럴듯해 보이지만 실제 시스템에서는 성립하지 않는다. 실제 시스템에 있어서는 응용 프로그램은 프로세서의 연산뿐만 아니라 메모리 시스템 같은 주변 장치의 접근 시간을 포함한다. 따라서 실제 응용 프로그램의 성능은 프로세서의 동작 주파수뿐만 아니라 메모리 시스템의 접근 빈도에도 영향을 받게 된다. Martin[6]과 Simunic[7]은 경험적 실험 결과를 통하여 응용 프로그램의 성능이 프로세서의 동작 주파수에 정비례하는 것이 아니라는 것을 보였다. 이러한 결과는 후에 이론적인 모델을 통해서도 입증되었다[1, 8]. 본 논문에서는 효과적인 내용전달을 위해서 시스템이 프로세서와 메인 메모리로 구성되어 있다고 가정한다. 여기서 메모리는 프로세서와는 다른 동작속도를 갖는 요소들을 대표하여 표현한 것으로 시스템의 구성에 따라서는 프로세서의 동작속도와 상관관계를 갖기도 하나 이러한 관계는 단순화를 위하여 무시하였다. 이때 응용 프로그램의 수행시간은 다음 수식 (1)과 같이 프로세서의 연산 시간과 메모리 접근 시간에 의해서 결정된다[1].

$$\tau_{exe} = \frac{N_c}{f_{cpu}} + \frac{N_m}{f_{mem}}. \quad (1)$$

여기서, N_c 와 N_m은 프로세서와 메모리에 의해서 소요되는 클럭 사이클이고, f_{cpu} 와 f_{mem}는 프로세서와 메모리의 동작 주파수이다.

그림 2는 동작 주파수에 따른 응용 프로그램의 성능 $(1/\tau_{exe})$을 표시한 것이다. 이 그래프의 각 점은 클럭 단위 정확도를 지닌 시스템 시뮬레이터[9]를 통해서 정밀하게 측정한 성능임을 밝혀둔다. 여기서 Math와 같이 연산 중심의 응용 프로그램은 선형적인 관계를 보여준다. 반면, MPEG 4 디코더와 같이 상당량의 메모리 접근을 가진 응용 프로그램들은 비선형적인 관계를 가짐을 알 수 있다. 특히나 NOR flash에 영상원본이 존재할 경우 비선형적인 관계는 더 심화됨을 알 수 있다. 이러한 응용 프로그램에 따른 동작 주파수와 실제 성능 사이의 변화는 일반적으로 다루어지는 입력 데이터에 따른 실행 시간 변이(execution time variation)와는 다른 특성이다. 실행 시간 변이는 입력 데이터의 차이에 의해서 프로그램의 수행 경로의 변화에 따라 발생하지만 본 논문에서 언급하는 성능의 변이는 연산과 메모리

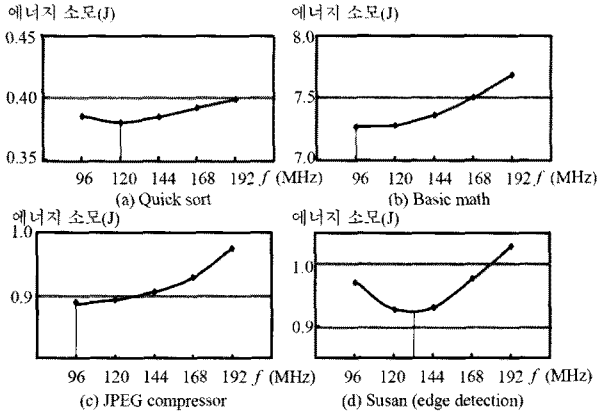

그림 2. 각 응용 프로그램에서의 동작 주파수에 따른 시스템 전력 소모의 비교(TI OMAP OSK 플랫폼을 이용한 실측값)

Fig. 2. Comparison of the system power consumption by the clock frequency variation of the microprocessor.

혹은 주변장치 접근량의 비에 의해서 결정된다.

일반적인 동적 전압 조절 문제에 있어서, 각 태스크 T_i는 도착시간 a_i, 데드라인 d_i, 최대 동작 속도에서의 수행시간(worst case execution time) τ_i 그리고 결정되어야할 동작 속도 조절 비(scaling factor) s_i의 네 가지 요소에 의해서 (a_i, d_i, τ_i, s_i)으로 정의 될 수 있다. 기존의 문제 정의에 있어서는 실제 수행 시간이 조절 비에 비례한다고 가정하기 때문에 추가적인 응용 프로그램의 특성은 기술 하지 않는다. 하지만 실제 시스템에 있어서 수행 시간은 앞서 그림 2와 수식 (1)에서 보는 바와 같이 실행시간의 일부만이 프로세서의 동작 주파수 조절에 영향을 받기 때문에, 실제 태스크의 수행시간 $\tau_{exe,i}$는 다음과 같다:

$$\tau_{exe,i} = \tau(\beta_i s_i + (1-\beta_i)), \tag{2}$$

여기서 β_i는 응용 프로그램의 특성 계수로서 전체 실행 시간 중에 프로세서의 동작 속도에 비례하는 부분의 비이다. 수식 (2)를 통하여 그림 2에 소개된 각 프로그램의 실측 수행 시간에 맞춰 보면 각각의 곡선을 얻을 수 있고 그것은 실측 결과와 아주 잘 맞아 떨어짐을 알 수 있다. 따라서 앞서 말한 동작 주파수에 따른 실행시간의 변이는 응용 프로그램 특성 계수 β를 통해서 잘 수치화 할 수 있으며, 그것은 실측 결과와도 잘 일치함을 알 수 있다.

2. 응용 프로그램의 특성을 고려한 최적의 동작 속도 도출

본 절에서는 각기 다른 β계수를 갖는 응용 프로그램에서 에너지 최적의 조절 비를 찾는 방법을 보이도록 하겠다. 앞서 말한 바와 같이 기존의 동적 전압 조절에서 가장 기본적인 가정은 응용 프로그램의 수행 시간이 동작 속도에 비례한다는 것이다. 이러한 가정 하에서 태스크의 스케줄링의 기본은 Yao 알고리즘[4]에 기반을 둔 방법이 널리 사용된다. 따라서 스케줄링의 결과는 잔여 시간(slack time)을 최소화하며 모든 태스크의 조절 비를 동일하게 하는 것이 최적의 해가 된다. 하지만 각 태스크가 각기 다른 β_i를 갖는 경우에는 이러한 접근은 더 이상 유효하지 않게 된다.

먼저 N개의 태스크의 집합 $\Gamma = \{T_1, ..., T_N\}$이 주어졌다고 가정하자. 그리고 각 태스크 T_i가 이제 β를 추가하여 다섯 개의 요소 $(a_i, d_i, \tau_i, \beta_i, s_i)$로 구성될 때 전체 에너지 소모는 다음과 같이 주어진다.

$$\begin{aligned} E_i(s_i) &= P(s_i) \times \tau_{exe,i} \\ &= P(s_i) \times \tau_i(\beta_i s_i + (1-\beta_i)) \end{aligned} \tag{3}$$

여기서 $P(s)$는 시스템의 조절 비에 대한 전력 소모 함수이다. 단, $P(s)$는 오목 함수(convex function)여야 하고, 이것은 그림 1에서 보다시피 일반적인 시스템에 있어서 자연스럽게 성립하는 가정이다.

정리하면 최적 동작 속도 문제는 전체 에너지 소모 $\sum_1^N E_i(s_i)$를 최소화는 조절비의 집합 $S = (s_1, ..., s_N)$을 결정하는 문제가 된다. 이 문제에서 일단 각 태스크의 도착 시간과 데드라인이 동일하다고 가정하면 전체 수행시간에 대한 제약조건은 다음과 같다.

$$\sum_{i=1}^{N} t_{exe,i} < d_i - a_i = D, \forall\ T_i \in \Gamma \tag{4}$$

여기서 D는 전체 수행 시간에 대한 데드라인이다.

정리 1. 태스크들의 전체 수행에 대한 데드라인만이 존재한다고 할 경우, 만약 $S = (s_1, ..., s_N)$이 에너지 최적 조절 비라고 하면, $\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i \beta_i} = \lambda, \forall\ T_i \in \Gamma$ 을 만족해야 한다. 그리고 에너지 최적 조절 비는 다음 중 한 가지가 된다.

$$\sum_{i=1}^{N}\tau_{exe,i} < D\text{이면}, \frac{dE_i(s_i)}{ds_i}=0, \forall\ T_i\in\Gamma$$
$$\sum_{i=1}^{N}\tau_{exe,i} = D\text{이면}, \frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}=0, \forall\ T_i\in\Gamma \quad (5)$$

증명: 정리 1은 Lagrangian Multiplier를 이용하여 증명이 가능하다. 목적 함수와 제약 조건을 Lagrangian 함수로 표현하면 아래와 같다.

$$L(S,\lambda,\gamma)=\sum_{i=1}^{N}E_i(s_i)+\lambda(\sum_{i=1}^{N}\tau_{exe,i}-D+\gamma^2) \quad (6)$$

여기서 γ는 여유 변수(slack variable)로 부등식을 등식으로 전환하여 주는 역할을 하며, $\gamma\geq 0$이여야 함으로 제곱식을 이용하였다.

s_i, γ 그리고 λ에 대한 L의 각 편미분이 최소 점에서 0이 되어야 한다. 따라서 해는 아래 세 개의 연립방정식의 해로 결정된다.

$$\frac{\partial L}{\partial s_i}=\frac{\partial E_i(s_i)}{\partial s_i}+\lambda\tau_i\beta_i=0, \forall\ i \quad (7)$$

$$\frac{\partial L}{\partial \gamma}=2\lambda\gamma=0 \quad (8)$$

$$\frac{\partial L}{\partial \lambda}=\sum_{i=1}^{N}\tau_{exe,i}-D+\gamma^2=0 \quad (9)$$

수식 (8)의 해는 $\lambda=0$ 이나 $\gamma=0$ 이여야 한다. 먼저 $\lambda=0$ 이라고 하면, 이 연립방정식의 해는 데드라인 제약 조건에 무관한 해를 얻게 되고, 그해는 수식 (7)에 의해서 $\frac{dE_i(s_i)}{ds_i}=0, \forall\ i$가 된다. 반면, $\lambda\neq 0$ 이고 $\gamma=0$이라고 가정하면, 해는 수식 (7)과 수식 (9)에 의해서 다음과 같다.

$$\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}=\lambda, \forall\ T_i\in\Gamma,$$
$$\sum_{i=1}^{N}\tau_{exe,i}=D \quad (10)$$

어떠한 경우든 $\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}$은 모든 태스크에 대해서 동일한 값이어야 한다.

■

위 정리는 단위 실행시간 변화에 따른 에너지 소모의 변화가 모든 태스크에 대해서 동일해야 한다는 것을 의미한다. 따라서 메모리 접근이 많은 β값이 낮은 태스크의 경우는 다른 태스크에 비해서 더욱더 동작 주파수를 낮추어 주어야 함을 의미한다. 즉, 메모리 접근에 의해서 소모되는 시간은 프로세서의 동작 속도와는 무관하기 때문에 프로세서의 동작 속도를 많이 낮추어 전력 소모를 줄여도 상대적으로 적은 성능 저하를 유발한다는 것을 알 수 있다.

3. 응용 프로그램의 특성을 고려한 최적의 동작 속도 결정의 예

본 연구에서는 에너지 모델로 잘 알려진 동적 전력 모델(dynamic power model)을 사용 하였다.

$$P(s)=aC_{eff}V^2f=\frac{\alpha}{s^3} \quad (11)$$

위 수식에서 두 번째 식은 $f=1/s$와 $V\propto f$를 가정하여 얻을 결과이다. 이 모델에서 메모리 시스템의 전력 소모는 고려되지 않았다. 하지만, 메모리 시스템이 임베디드 시스템에서 가장 널리 사용되는 active-power down 정책을 사용한다고 할 경우 메모리의 전력 소모는 단지 메모리 접근 횟수에 거의 비례하기 때문에 특별히 고려하지 않아도 정확한 결과를 얻을 수 있음을 밝혀 둔다.

그림 3은 4개의 태스크를 가진 태스크 집합에서의 실험결과를 보여준다. 각 태스크는 표 1과 그림 3.(a)에 나온 바와 같이 각기 다른 최대 동작 속도에서의 수행 시간과 β 값을 가지고 있다. 이 태스크 집합에 대해 EDF(earliest deadline first)를 이용하여 스케줄링을 할 경우 그림 3.(b)와 같은 결과를 얻게 된다. 그리고 일반적인 기존 동적 전압 조절 기법을 사용할 경우 그림 3.(c)와 같은 결과를 얻게 된다. 기존의 동적 전압 조절 기법은 잔여 시간을 남기지 않는 선에서 최대한 동작

표 1. 태스크 집합의 구성 (단위: 초)
Table 1. Task set composition (unit: second).

	도착시간(a)	데드라인(d)	τ	β
T_1	0.0	1.4	0.6	0.4
T_2	0.2	1.1	0.2	0.8
T_3	0.4	1.7	0.5	1.0
T_4	1.0	1.8	0.1	0.6

속도를 낮추고자 하여 조절 비를 $\sum_{i=1}^{N}\tau_i/D$로 결정한다[4~5]. 하지만 실제로는 태스크의 일부 수행 시간은 프로세서의 동작 속도에 영향을 받지 않기 때문에 태스크 집합은 기대보다 먼저 수행을 마치게 되고 그림 3.(c)와 같이 잔여 시간을 남기게 된다.

이러한 문제를 해결하기 위하여 몇 가지 기존 연구[8, 10]는 β을 고려하여 잔여시간을 완전히 제거하고자 노력하였다. 만약 기존에 동적 전압 조절의 가정과 같이 결정되어야할 모든 조절 비가 동일하여야 한다면, 잔여 시간을 완전히 제거할 수 있는 조절 비는 다음과 같다.

$$s = \frac{D-\tau}{\beta\tau} + 1$$
$$\tau = \sum_{i=1}^{N}\tau_i$$
$$\beta = \frac{\sum_{i=1}^{N}\tau_i\beta_i}{\sum_{i=1}^{N}\tau_i} \quad (12)$$

이 경우 표 2에 나온 바와 같이 기존의 동적 전압 조절이 1.29의 조절 비를 사용하는데 비해 1.42로 더욱 동작 속도를 낮추어 완전히 잔여 시간을 제거하고 그를 통하여 에너지 소모를 20%가량 추가적으로 절감함을 알 수 있다. 하지만 진정한 최적 동작 속도는 정리 1에서 알 수 있다시피 모든 태스크에 대해서 $\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}$을 동일하게 맞추어야 한다. 이 예의 경우 최적 조절 비는 표 2에 나온 바와 같이 각기 다르게 결정된다.

이렇게 최적 조절 비를 이용하여 스케줄링 할 경우 기존 연구에 비하여 약 5%의 추가적인 전력 소모 절감 효과를 얻을 수 있다. 이 예를 살펴보면, β값이 낮은 T_1과 같은 경우 조절 비를 늘리고 동작 속도를 더욱 낮추어도 실행 시간은 그다지 늘어나지 않기 때문에 전력 소모에 있어서 큰 이익을 얻을 수 있다. 즉 β값이 작은 태스크들은 수행 시간이 프로세서의 동작 속도에 덜 민감하다는 것을 의미함으로 상대적으로 β값이 큰 태스크들의 동작 속도를 빠르게 하여 많은 잔여시간을 확보하고 그것을 β값이 작은 태스크들에 투자하는 것이 전체적으로는 더 최적이라는 것을 의미한다.

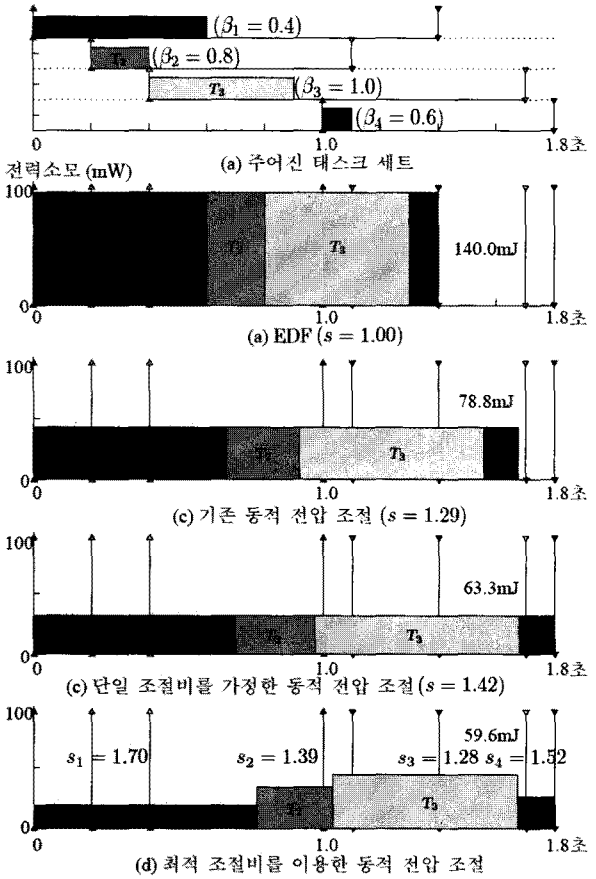

그림 3. 동적 전압 조절 기법을 통한 스케줄링 결과 비교

Fig. 3. Comparison of the result of the dynamic voltage scaling techniques.

표 2. 동적 전압 조절 기법에 따른 전력 소모 비교 ($s^{\dagger} = \{1.70, 1.39, 1.28, 1.52\}$)

Table 2. Comparison of the power consumption by the dynamic voltage scaling techniques.

방법	조절 비	총 수행 시간 (초)	에너지 소모 (mJ)					
			T_1	T_2	T_3	T_4	합계	비율 (%)
EDF	1.00	1.40	60.0	20.0	50.0	10.0	140.0	-
기존 DVS	1.29	1.68	31.5	11.6	30.2	5.5	78.8	100.0%
단일 조절비	1.42	1.80	24.6	9.4	24.9	4.4	63.3	80.3%
최적 조절비	$s^{\dagger}$	1.80	15.6	9.8	30.5	3.7	59.6	75.6%

4. 응용 프로그램 특성 계수 β를 고려한 최적 조절 비 결정 휴리스틱 알고리즘

본 연구의 기존 연구[3]에서 각 태스크가 각기 다른 에너지 모델을 가질 경우 최적의 동작 속도 결정 문제는 NP-Hard 임이 알려져 있다. 본 연구와의 차이는 기존 연구에서는 각 태스크의 에너지 모델이 각기 다른 임의의 오목 함수임을 가정한 것이 전부였으나 본 논문에서는 태스크를 특성 계수 β를 통하여 수치화 하고 그것을 이용했다는데서 가장 큰 차이를 가진다는 것을 먼저 밝혀둔다. 그리고 해공간이 연속적이지 않은 이산 공간(discrete space)을 가질 경우, 최적 동작 속도 결정 알고리즘은 잘 알려진 NP-Hard 문제인 0-1 Knapsack 문제로 모델링 될 수 있다. 본 문제의 해는 Integer Linear Programming을 이용하여 구할 수 도 있으나 본 논문에서는 같은 결과를 제시하는 단순한 휴리스틱 알고리즘을 제시하겠다. 제시하는 휴리스틱 알고리즘은 알고리즘 1에 기술되어 있다.

알고리즘 1. 응용 프로그램 특성을 고려한 최적 조절 비 휴리스틱 알고리즘

입력: 태스크 집합 Γ

출력: 조절 비 $s_1,\ldots,s_N$

$\forall s_i \Leftarrow \left|\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}\right| \le 0$을 만족하는 최대 조절 비

while not feasible do

다음 조건을 만족하는 태스크를 T_i를 찾는다.

(1) $s_i > 1$

(2) 도착시간과 데드라인이 연속적으로 인접한 태스크 중 어느 하나라도 시간 조건을 위배한다.

(3) (1)과 (2)를 만족하는 태스크 중 $\left|\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}\right|$이 최솟값을 가지는 태스크

s_i를 한 단계 감소시킨다.

end while

본 휴리스틱 알고리즘은 가장 먼저 모든 s_i를 $\left|\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}\right| \le 0$ 을 만족하는 최댓값으로 설정한다. 이것은 Jejurikar의 연구[11]에서 밝혀진 바와 같이 크리티컬 스피드 이하에서는 에너지 소모가 오히려 증가하기 때문에 해당 속도 이하로 과도하게 조절할 이유는 전혀 없기 때문이다. 이렇게 조절 비를 최대로 할 경우 상당량의 데드라인 초과를 유발할 것이다. 따라서 조절 비를 줄여 속도를 빠르게 할 필요가 있는데 이때 어떤 태스크의 조절 비를 줄일 것인가가 알고리즘의 핵심이다. 바로 $\left|\frac{dE_i(s_i)}{ds_i}\frac{1}{\tau_i\beta_i}\right|$이 단위 실행시간 변화 당 에너지의 변화량이기 때문에 속도를 빠르게 하여 실행시간을 줄였을 때 유발되는 에너지 증가를 가장 적게 만드는 태스크의 속도를 증가시키는 것이 바로 최적의 방법이다. 정리 1에서 증명한 것도 바로 이 값을 최대한 동일하게 만들어야 한다는 것으로 정리 1에 따라서 단위 실행시간 변화 당 에너지의 변화량이 가장 적은 태스크의 속도를 조금 더 빠르게 만드는 것이다. 이 작업을 모든 태스크가 데드라인 초과를 유발하지 않을 때까지 반복한다.

III. 성 능 평 가

1. 실험 환경

본 논문에서 제시한 알고리즘의 성능평가를 위하여 합성 태스크 집합(synthetic task set)을 이용하여 실험을 수행하였다. 합성 태스크 집합은 여섯 개의 요소 $(N,\lambda,\mu_d,\sigma_d,\mu_\tau,\sigma_\tau,\beta_{\min},\beta_{\max})$로 기술된다. 여기서 N은 태스크의 수이고, λ는 도착시간 간격(inter-arrival time)의 평균, μ_d와 σ_d는 데드라인의 평균과 표준편차, μ_τ와 σ_τ는 최대 동작 속도에서의 실행시간의 평균과 표준 편차, 그리고 $\beta_{\min}$과 $\beta_{\max}$는 β의 최솟값 최댓값이다. 일반적인 태스크 모델과 같이 본 연구에서도 태스크의 도착시간의 간격은 exponential 분포를 따르고, 데드라인과 실행시간은 gaussian 분포를 따른다고 가정하였다. 그리고 β는 uniform 분포를 따른다. 본 실험에서 사용된 합성 태스크 집합은 표 3에 기술되어 있다.

합성 태스크 집합은 크게 세 집단으로 나뉘어져 있다. 첫째 집단($T_{1\_1},\ldots,T_{1\_4}$)은 최대 동작 속도에서의 수행시간의 평균값을 조절하여 점유율 변화에 따른 성능 차이를 보여준다. 두 번째 집단($T_{2\_1},\ldots,T_{2\_4}$)과 세 번째 집단($T_{3\_1},\ldots,T_{3\_4}$)은 β의 평균과 편차의 변화에 따른 성능 차이를 보여준다.

실험은 Matlab의 TORSCHE Scheduling Toolbox에

표 3. 휴리스틱 알고리즘의 검증을 위한 합성 태스크 집합 (단위:ms)
Table 3. Synthetic task set for the verification of the heuristic algorithm (unit: ms).

태스크 집합		N	λ	μ_d	σ_d	μ_τ	σ_τ	β_{min}	β_{max}
점유율 변화 (utilization)	T_{1_1}	20	10.0	30.0	10.0	9.0	2.0	0.4	1.0
	T_{1_2}	20	10.0	30.0	10.0	8.0	2.0	0.4	1.0
	T_{1_3}	20	10.0	30.0	10.0	7.0	2.0	0.4	1.0
	T_{1_4}	20	10.0	30.0	10.0	6.0	2.0	0.4	1.0
β의 평균 변화	T_{2_1}	20	10.0	30.0	10.0	7.5	2.0	0.5	1.0
	T_{2_2}	20	10.0	30.0	10.0	7.5	2.0	0.4	0.9
	T_{2_3}	20	10.0	30.0	10.0	7.5	2.0	0.3	0.8
	T_{2_4}	20	10.0	30.0	10.0	7.5	2.0	0.2	0.7
β의 편차 변화	T_{3_1}	20	10.0	30.0	10.0	7.5	2.0	0.6	0.7
	T_{3_2}	20	10.0	30.0	10.0	7.5	2.0	0.5	0.8
	T_{3_3}	20	10.0	30.0	10.0	7.5	2.0	0.4	0.9
	T_{3_4}	20	10.0	30.0	10.0	7.5	2.0	0.3	1.0

일반 동적 전압 조절을 하지 않은 EDF와 기존의 일반적인 DVS[4] 그리고 Hsu의 방법[10]과 제안된 방법을 모두 구현하여 비교 실험 하였다. Hsu의 방법은 β을 고려하여 최저 잔여 시간을 맞추는 방법이며, 반면 본 연구의 방법은 정리 1에 입각하여 각 태스크의 단위 수행 시간당 에너지의 변화를 최대한 동일하게 하는 방법임을 다시 한 번 밝혀 둔다.

2. 실험 환경

표 4와 그림 4는 합성 태스크 집합을 이용한 실험결과를 보여 준다. 먼저 표 4의 점유율 변화에 따른 성능 변화를 살펴보면, 점유율이 감소함에 따라서 본 연구에서 제시하는 최적 동적 전압 조절 기법의 효율이 감소함을 알 수 있다. 이것은 점유율이 감소함에 따라서 대체적으로 더 낮은 동작 속도에서 동작하게 되기 때문이다. 통상의 에너지 함수는 오목 함수이기 때문에 동작 속도가 낮은 지역에서는 에너지 함수의 기울기가 감소한다. 즉, 동작 속도 변화에 따른 에너지의 변화가 적어지기 때문에 최적 기법과 기존 Hsu의 기법과 차이가 감소하는 것이다. 두 번째로 β의 평균 변화에 따른 효율을 비교하면, β의 평균이 낮아짐에 따라 기존 DVS 대비 Hsu의 기법과 최적 기법 모두 효율이 좋아진다. 이것은 기존 DVS에서는 β를 1로 가정하게 되고 실제 태스크의 수행이 기대이상 빠르게 되어 필요 없이 많은 잔여 시간을 남기기 때문에 좋은 성능을 얻지 못한다. 반면 Hsu의 기법이나 최적 기법은 정확한 β을 기반으로 잔여시간을 남기지 않고 충분히 동작 속도를 낮추어

표 4. 합성 태스크 집합을 이용한 실험결과 (mJ)
Fig. 4. The experimental results of the heuristic algorithm using the synthetic task set.

점유율 변화					
태스크	EDF	DVS	Hsu[10]	최적	%
T_{1_1}	17.97	10.35	8.76	8.29	5.41
T_{1_2}	15.65	9.18	8.13	7.85	3.43
T_{1_3}	15.04	7.29	5.79	5.62	2.90
T_{1_4}	11.76	5.58	4.60	4.50	2.30
β의 평균 변화					
태스크	EDF	DVS	Hsu[10]	최적	%
T_{2_1}	15.28	7.97	6.99	6.80	2.73
T_{2_2}	14.83	8.12	6.49	6.27	3.42
T_{2_3}	14.56	7.91	6.14	5.79	5.83
T_{2_4}	14.96	8.12	5.91	5.61	5.06
β의 편차 변화					
태스크	EDF	DVS	Hsu[10]	최적	%
T_{3_1}	14.75	8.34	7.05	6.91	1.91
T_{3_2}	14.55	7.82	6.61	6.45	2.42
T_{3_3}	14.67	7.49	6.05	5.86	3.19
T_{3_4}	14.81	4.56	5.92	5.50	7.19

더 많은 에너지 절감을 얻게 된다. 마지막으로, β의 편차 변화에 따른 실험결과를 살펴보자. 앞서 말한 바와 같이 Hsu의 기법에 있어서는 모든 태스크를 동일한 조절비로 동작 시키는 반면 최적 기법은 각 태스크의 β를 고려하여 각 태스크에 적합한 조절 비를 제시한다. 따라서 β의 변이가 클 경우 최적 기법은 더욱 좋은 효율을 보이게 된다. 정리하면, 본 연구에서 제시하는 최

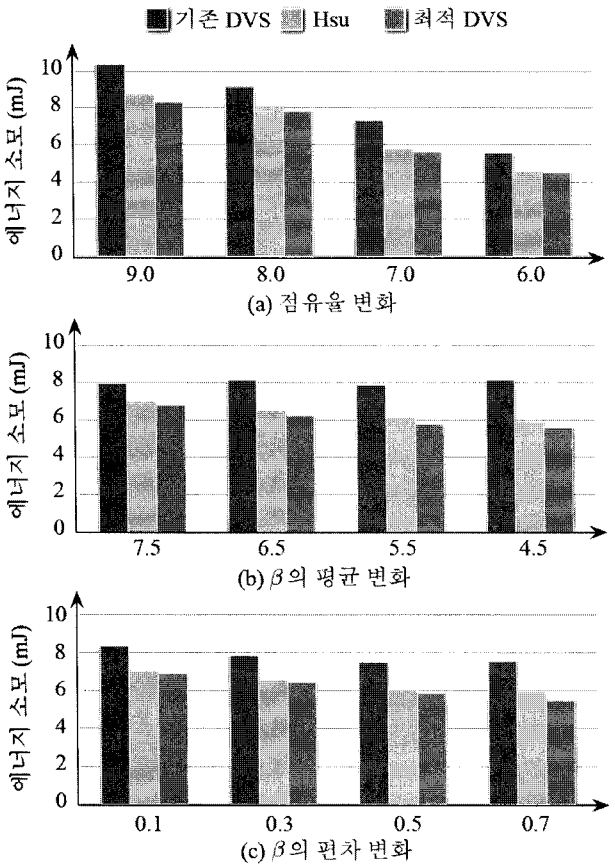

그림 4. 합성 태스크를 이용한 실험결과 그래프
Fig. 4. Graph of the experimental results using synthetic task set.

적 동작 속도 조절 기법은 점유율이 높고 β의 평균은 낮은 상태에서 변이가 클 때 상대적으로 가장 높은 에너지 절감 효과를 얻을 수 있음을 알 수 있다.

Ⅳ. 결 론

본 연구는 동적 전압 조절(dynamic voltage scaling)에서 동작 속도에 태스크의 수행시간이 비례한다는 가장 기본적인 가정이 실제 환경에서는 적합하지 않음을 실험 결과를 통하여 보이고 이러한 수행속도에 따른 성능 변화를 수치화하고 그것을 이용한 최적 동작 속도 조절 기법을 제시하였다. 특히, 수행 속도에 따른 성능이 비례 관계를 갖지 않을 때 최적의 동작 속도 결정 방법을 해석적으로 정리하였으며, 그것을 기반으로 하여 최적의 동작 속도를 얻는 휴리스틱 알고리즘을 제시하였다.

본 연구에 따르면, 동적 전압 조절에 있어서 동작 속도를 Yao 알고리즘에 따라 같은 값을 유지하며 최대한 낮추기 보다는, 단위 수행시간 변화 당 에너지의 변화가 최대한 동일하게 하는 것이 최적임을 증명하였다. 각 태스크의 수행 속도의 변화에 따른 성능 변화가 다를 때 이러한 정리에 입각하여 최적 스케줄링을 한 결과 기존 연구대비 최대 약 7%의 추가적인 에너지 절감 효과를 얻을 수 있었다.

참 고 문 헌

[1] Youngjin Cho and Naehyuck Chang. Energy-aware clock frequency assignment in microprocessors and memory devices for dynamic voltage scaling. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 26(6):1030-1040, 2007.

[2] 최용석(Yongseok Choi), 장래혁(Naehyuck Chang), 김태환(Taewhan Kim) 대한전자공학회, 전자공학회논문지-SD 電子工學會論文誌 第44卷 SD編 第2號, 2007. 2, pp. 95 ~ 103

[3] Youngjin Cho, Naehyuck Chang, Chaitali Chakrabarti, and Sarma Vrudhula. "High-level power management of embedded systems with application-specific energy cost functions." in proceedings of Design Automation Conference (DAC), pp 568-573, 2006.

[4] F. Yao, A. Demers, and S. Shenker. "A scheduling model for reduced cpu energy." in proceedings of the Annual Symposium on Foundations of Computer Science (FOCS), pp 374, 1995.

[5] Tohru Ishihara and Hiroto Yasuura. "Voltage scheduling problem for dynamically variable voltage processors." in proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), pp. 197-202, 1998

[6] Thomas L. Martin and Daniel P. Siewiorek. Nonideal battery and main memory effects on cpu speed-setting for low power. IEEE Transaction on Very Large Scale Integration (VLSI) Systems, 9(1):29.34, 2001.

[7] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. D. Micheli, "Dynamic voltage scaling and power management for portable systems," in proceedings of Design Automation Conference, 2001, pp. 524-529.

[8] Kihwan Choi, R. Soma, and M. Pedram. "Dynamic voltage and frequency scaling based on workload decomposition." in proceedings of

the International Symposium on Low Power Electronics and Design (ISLPED), pages 174.179, 2004.
[9] Yongsoo Joo, Yongseok Choi, Jaehyun Park, Chanik park, Sung Woo Chung, Eui-Young Chung and Naehyuck Chang, Energy and Performance Optimization of Demand Paging with OneNAND Flash, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 27(11), pp. 1969-1982, Nov., 2008.
[10] Chung-Hsing Hsu and Wu-Chun Feng. "Effective dynamic voltage scaling through cpu-boundedness detection." in Workshop on Power Aware Computing Systems, 2004.
[11] R. Jejurikar and R. Gupta. "Dynamic voltage scaling for systemwide energy minimization in real-time embedded systems." in proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), pp. 78-81, 2004.

저 자 소 개

조 영 진(정회원)
2003년 서울대학교 조선해양공학, 컴퓨터공학과 학사 졸업.
2005년 Arizona State University, visiting scholar.
2009년 서울대학교 전기컴퓨터 공학부 박사 졸업.
2009년~현재 삼성전자 DS부문 책임연구원.
<주관심분야 : System-on-chip, 저전력 설계, 메모리 제어기 설계>

장 래 혁(정회원)
1989년 서울대학교 제어계측 공학과 학사 졸업.
1992년 서울대학교 제어계측 공학과 석사 졸업.
1996년 서울대학교 제어계측 공학과 박사 졸업.
1997년 미국 University of Michigan 연구원.
2005년 미국 Arizona State University 방문교수.
2009년 미국 University of Southern California 방문교수.
1997년~현재 서울대학교 컴퓨터공학부 교수.
<주관심분야: 저전력 시스템, 내장형 시스템, 연료전지 배터리 하이브리드 시스템, 그린컴퓨팅>