Journal of The Korea Society of Computer and Information
Vol. 18, No. 2, February 2013
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2013.18.2.031

이산적 DVFS 멀티코어 프로세서 상에서 실시간 병렬 작업을 위한 확률적 저전력 스케쥴링

이 완 연*

Probabilistic Power-saving Scheduling of a Real-time Parallel Task on Discrete DVFS-enabled Multi-core Processors

Wan Yeon Lee*

요 약

본 논문에서는 멀티코어 프로세서에서 단일 실시간 병렬 작업의 데드라인을 만족하면서 전력 소모량의 확률적 기대 값을 최소화하는 스케쥴링 기법을 제안하였다. 제안된 기법에서는 단일 작업을 여러 개의 코어들 상에서 동시에 수행하는 병렬 처리 기법을 적용하였고, 전체 코어들 중에서 일부의 코어들만을 사용하고 나머지 코어들의 전원을 소등하여 전력 소모량을 줄였다. 또한 한정된 개수의 이산적 클락 주파수 값들을 가지는 DVFS 기반 멀티코어 프로세서에 대해서, 확률적 계산량 모델을 가진 실시간 병렬 작업의 데드라인을 만족하면서 전력 소모량의 확률적 기대 값을 최소화함을 수학적으로 증명하였다. 성능평가 실험에서, 제안된 기법이 기존 방법의 전력소모량을 최대 81%까지 감소시킴을 확인하였다.

▸ Keywords : 멀티코어 프로세서, 실시간 작업, 스케쥴링, 저전력 설계, DVFS

Abstract

In this paper, we propose a power-efficient scheduling scheme that stochastically minimizes the power consumption of a real-time parallel task while meeting the deadline on multicore processors. The proposed scheme applies the parallel processing that executes a task on multiple cores concurrently, and activates a part of all available cores with unused cores powered off, in order to save power consumption. It is proved that the proposed scheme minimizes the mean power consumption of a real-time parallel task with probabilistic computation amount on DVFS-enabled multicore processors with a finite set of discrete clock frequencies. Evaluation shows that the

• 제1저자 : 이완연 • 교신저자 : 이완연
• 투고일 : 2012. 12. 11, 심사일 : 2012. 12. 17, 게재확정일 : 2013. 1. 1.
* 동덕여자대학교 컴퓨터학과(Dept. of Computer Science, Dongduk Women's University)
1) 이 논문은 2012년도 동덕여자대학교 학술연구비 지원에 의해서 수행된 것임.

proposed scheme saves up to 81% power consumption of the previous method.

▸ Keywords : Multicore processor, Real-time task, Scheduling, Power-efficient design, DVFS

I. 서 론

유하나의 프로세서 칩 내의 프로세싱 코어(processing core)들의 개수는 프로세서 설계 기술 발전에 비례하여 계속 증가할 것으로 예상된다[1]. 배터리의 전원에 의지하여 동작하는 무선 컴퓨터 시스템에서는 프로세서가 소모하는 전력량을 줄이는 것은 매우 중요한 문제이다. 프로세서의 전력 소모량을 효율적으로 관리하기 위한 고전적인 방법으로, 작업을 수행하지 않는 유휴 시간(idle time) 동안 프로세서의 전원을 소등하는 power-down-when-idle 기법[2]이 있다. 프로세서의 전력 소모량을 더 효율적으로 관리하기 위해서 최근 들어 DVFS(dynamic voltage and frequency scaling) 기법[3]이 많이 사용되고 있다. DVFS 기법은 프로세서의 부하에 따라 프로세서에 입력되는 전압을 변화시켜서 프로세서의 클락 주파수(clock frequency)와 전력 소모량을 동적으로 조절하는 기법이다. 입력 전압을 증가시키면 프로세서의 연산 속도를 결정하는 클락 주파수 값도 증가하지만 더불어 전력 소모량도 같이 증가한다. 유휴 시간 동안 프로세서의 전원을 소등하는 power-down-when-idle 기법보다 클락 주파수 값을 감소시키는 DVFS 기법이 실시간 작업의 전력 소모량을 줄이는데 더 효과적이다[3,4].

본 논문에서는 DVFS 기법을 제공하는 멀티코어 프로세서 상에서 주어진 단일 실시간 작업의 데드라인을 만족하면서 전력 소모량을 최소화는 스케쥴링 기법을 제안하였다. 제안된 스케쥴링 기법은 한정된 개수의 이산적 클락 주파수 값들만을 제공하는 DVFS 기반 멀티코어 프로세서 상에서 동작 가능하도록 설계되었다. 제안된 기법에서는 단일 작업의 계산량을 여러 개의 코어들에 분배하여 동시에 수행하는 병렬 처리 기법을 적용하였고, 모든 프로세싱 코어들 중에서 일부의 코어들만을 사용하고 나머지 사용하지 않는 코어들의 전원은 소등하였다. 또한 작업의 변동하는 계산량을 과거의 계산량 분포도에 근거하여 확률적 계산량 모델로 표현하고, 도출된 확률적 계산량 모델에 근거하여 주어진 이산적 클락 주파수 값들을 적용하여 전력 소모량 기대 값을 최소화하였다.

본 논문에서는 제안된 기법이 확률적 계산량을 가진 실시간 병렬 작업의 데드라인을 만족하면서 전력 소모량의 확률적 기대 값을 최소화함을 수학적으로 증명하였다. 또한 성능 평가 실험을 통하여, 제안된 기법이 기존 스케쥴링 기법의 전력 소모량을 최대 81%까지 줄임을 보였다.

멀티코어 프로세서의 전력소모량을 줄이기 위해서 기존에도 DVFS 기법을 활용한 스케쥴링 기법에 관한 연구들이 많이 진행되어 왔다. 그러나 대부분의 기존 스케쥴링 기법들[4-8]은 프로세싱 코어들의 개수보다 작업들의 개수가 더 많은 시스템 환경만을 다루었고, 반대의 경우인 코어 개수보다 작업 개수가 더 적은 환경은 고려하지 않았다. 최근 들어 프로세싱 코어 개수가 작업 개수보다 더 많은 멀티코어 프로세서 환경("저부하 멀티코어 프로세서"라 명명)에서, 일부의 프로세싱 코어들만을 사용하여 주어진 실시간 작업들의 데드라인을 만족하면서 전력 소모량을 최소화하는 스케쥴링 기법들[9-11]이 연구되었다. 최근의 스케쥴링 기법들 [9]와 [10]은 고정된 계산량을 가지는 작업들에 대해서는 전력 소모량을 효과적으로 줄이지만, 변동하는 계산량을 가지는 작업들에 대해서 비효율적으로 동작하는 방법이다. 반면 본 논문에서 제안된 스케쥴링 기법은 변동하는 계산량을 가지고 일정 간격으로 반복적으로 도착하는 주기적 작업(periodic task)에 대해서 전력 소모량을 효과적으로 줄이는 방법을 다루었다.

본 논문의 선행 연구인 [11]에서도 변동하는 계산량을 가진 단일 실시간 병렬 작업의 전력 소모량 확률적 기대 값을 최소화하는 스케쥴링 기법을 다루었다. 선행 연구에서는 무한 개수의 연속된 클락 주파수 값들을 적용한 비현실적 DVFS 기법 환경에서 전력 소모량의 확률적 기대 값을 최소화하는 상대적으로 단순한 문제의 해법을 다루었고, 본 논문의 확장 연구에서는 한정된 개수의 이산적 클락 주파수 값들만을 적용하는 현실적 DVFS 기법 환경에서 전력 소모량의 확률적 기대 값을 최소화하는 상대적으로 더 복잡한 문제의 최적 해법을 다루었다. 실제 생활에서 사용하는 DVFS 기반 프로세서는 한정된 개수의 이산적 클락 주파수 값들만을 제공하므로[3], 본 논문에서 제안된 기법이 실생활의 프로세서들에게 적용 가능한 실용적인 방법이다.

본 논문의 나머지 부분은 다음과 같이 구성된다. II 장에서

는 DVFS 기반의 멀티코어 프로세서 모델과 확률적 계산량을 가진 주기적 실시간 병렬 작업 모델에 대해서 정의한다. III 장에서는 제안된 스케쥴링 기법의 동작 과정을 상세히 설명하고, 제안된 스케쥴링 기법이 확률적 계산량을 가진 실시간 작업의 전력 소모량 확률적 기대 값을 최소화함을 증명한다. IV 장에서는 제안된 스케쥴링 기법과 기존의 스케쥴링 기법의 성능 비교 실험을 다룬다. 마지막으로 V 장에서는 본 논문의 내용을 요약하고 정리한다.

II. 시스템 환경 모델

1. 프로세서 모델

멀티코어 프로세서에서 사용 가능한 프로세싱 코어들의 개수를 N으로 나타낸다. 멀티코어 프로세서는 DVFS 기법을 제공하고, 사용되지 않는 프로세싱 코어들의 전원은 별도로 소등하여 전력소모량을 줄인다[2]. 사용 가능한 K개의 주파수들을 오름차순으로 정렬하여 $F_1, ..., F_K$으로 나타낸다.

프로세싱 코어들에게 적용되는 클락 주파수는 코어가 단위 시간당 처리하는 계산량(연산 사이클 개수)을 나타내고, 따라서 클락 주파수가 증가하면 코어의 실행 속도가 증가하고 또한 코어가 소모하는 전력량도 증가한다. 각 클락 주파수 F_k가 단위 시간당 소모하는 전력량을 E_k로 나타낸다. 정의에 따라 $F_i < F_j$ 이면 $E_i < E_j$이다. 프로세싱 코어들에게 적용되는 클락 주파수는 언제든지 변경할 수 있고, 같은 순간에 코어들에게 적용되는 클락 주파수는 동일하다.

2. 주기적 실시간 작업 모델

주어진 단일 작업은 일정 시간마다 반복적으로 새로운 작업이 도착하는 주기적 작업이고, 이전에 도착한 작업은 새로운 작업이 도착하기 이전에 수행을 완료해야 한다. 따라서 작업의 도착 주기가 작업 수행을 완료해야 하는 데드라인이 되며, 작업의 데드라인(즉, 도착 주기)을 D로 나타낸다. 주기적으로 도착하는 작업들이 필요로 하는 계산량(연산 사이클 개수)은 사전에 미리 알지 못하고, 도착하는 작업마다 다른 계산량을 가지고 있다. 그리고 변동하는 계산량에도 최대 한계점은 주어지고, 최대 한계 계산량을 W로 나타낸다.

병렬 처리 기법은 작업이 요구하는 전체 계산량을 여러 개의 부분들로 분리하여 다수의 코어들에게 분배하고, 분배된 계산량을 다수의 코어들이 동시에 실행하는 방법이다. 일반적으로 병렬 처리 기법을 적용한 작업의 실행 완료 시간은, 병렬 처리에 투입된 코어들의 개수로 비례하여 실행 시간이 단축된다.

작업이 요구하는 변동하는 계산량은 사전에 정확하게 알 수는 없지만, 과거에 도착하여 수행을 완료한 작업들의 계산량 분포 정보를 이용하여 미래에 도착할 작업의 계산량을 확률적으로 예측할 수 있다[8,11]. 그림 1은 과거의 계산량 분포 정보를 이용하여 미래 작업의 변동하는 계산량을 확률적 계산량 모델로 표현하는 방법을 설명하고 있다.

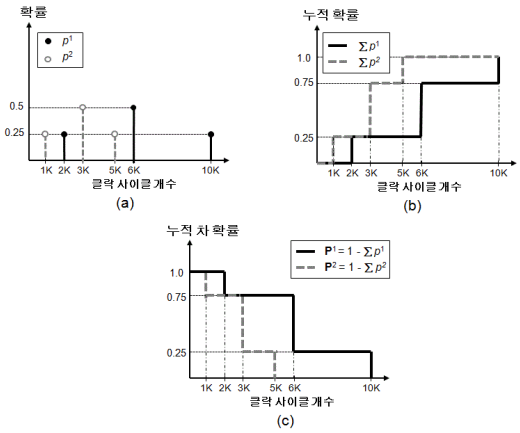

그림 1. 작업의 확률적 계산량 모델
Fig. 1. Probabilistic computation model of task

그림 1 (a)은 과거에 도착하여 수행을 완료한 네 개의 작업들에 대해서 하나의 코어에서 소요된 클락 사이클 개수의 확률 분포도와 그리고 두 개의 코어에서 소요된 클락 사이클 개수의 확률 분포도를 보여주고 있다. 하나의 코어에서 실행을 완료하는데 소요된 클락 사이클 개수는 2K, 6K, 6K, 10K이고, 두 개의 코어에서 병렬로 실행을 완료하는데 소요된 클락 사이클 개수는 1K, 3K, 3K, 5K이다. 그림 1 (a)에서 p_c^1는 하나의 코어에서 c 개의 클락 사이클 개수가 소요될 확률을 나타내고, p_c^2는 두 개의 코어에서 병렬로 c 개의 클락 사이클 개수가 소요될 확률을 나타낸다. 그림 1 (b)는 그림 1 (a)의 확률 분포(probability distribution)에 대한 누적 확률 분포(cumulative probability distribution)를 보여주고 있다. 하나의 코어에서 작업 계산량이 c 개 이하의 클락 사이클 개수를 가질 확률은 $\sum_{i=1}^{c} p_i^1$이고, 두 개의 코어에서 작업의 병렬 계산량이 c 개 이하의 클락 사이클 개수를 가질 확률은 $\sum_{i=1}^{c} p_i^2$이다. 그림 1 (c)는 그림 1 (b)의 누적 확률

분포에 대한 누적 차 확률 분포(tail cumulative probability distribution)를 보여주고 있다. 하나의 코어에서 작업 계산량이 c 개 이상의 클락 사이클 개수를 가질 확률은 $P_c^1 = 1 - \sum_{i=1}^{c} p_i^1$이고, 두 개의 코어에서 작업의 병렬 계산량이 c 개 이상의 클락 사이클 개수를 가질 확률은 $P_c^2 = 1 - \sum_{i=1}^{c} p_i^2$ 이다. 본 논문에서는 작업이 n개의 코어들에서 병렬로 실행될 때, 작업의 계산량이 c 개 이상의 클락 사이클 개수를 요구할 확률을 P_c^n로 표기하였다. P_c^n는 작업이 n개의 코어들에서 병렬로 실행될 때, c 개의 클락 사이클 수행한 이후에도 실행이 완료되지 않고 잔여 작업량이 남아 있을 확률을 의미한다. 그림 1 (b)에서 보여주는 누적 확률 분포는 항상 증가 함수를 가지므로, 그림 1 (c)에서 보여준 누적 차 확률 분포는 항상 감수 함수로 표현된다. 즉, $c1 < c2$ 이면 $P_{c1}^n \geq P_{c2}^n$이다.

III. 제안된 스케쥴링 기법

1. 스케쥴링 정형화

주어진 단일 작업을 수행하는데 할당된 코어들의 개수를 n으로 나타낸다. 이 경우에 나머지 $(N-n)$ 개의 코어들 전원은 소등된다. 주어진 작업을 n개의 코어들 상에서 동시에 수행할 때, c번째 연산 클락 사이클에 적용된 클락 주파수 값을 f_c^n로 표시한다. f_c^n는 한정된 개수의 이산적 값을 가진 집합에 소속되므로 $f_c^n \in \{F_1, ..., F_K\}$이다. 주어진 작업의 c번째 연산 클락 사이클이 소요하는 단위 시간당 전력 소비량을 $E(f_c^n)$로 표현하고, 이 경우에 $f_c^n \in \{F_1, ..., F_K\}$이므로 $E(F_1) = E_1$, …, $E(F_K) = E_K$이다.

주어진 작업의 c번째 연산 클락 사이클에 f_c^n의 주파수를 적용하면, c번째 연산 클락 사이클을 실행하는데 소요되는 시간은 $1/f_c^n$이다. 따라서 주어진 작업의 최대 계산량인 W를 항상 데드라인 이전에 실행을 완료하기 위해서는 아래 수식 (1)을 만족하여야 한다.

$$\sum_{c=1}^{W} \frac{1}{f_c^n} \leq D \qquad \text{수식 (1).}$$

또한 주어진 작업을 n개의 코어들 상에서 수행할 때, 확률적 계산량 P_c^n를 가진 실시간 작업의 전력 소모량 확률적 기대치 값은 아래의 수식 (2)와 같이 정의 될 수 있다[11].

$$n \cdot \sum_{c=1}^{W} P_c^n \cdot E(f_c^n) \qquad \text{수식 (2).}$$

즉 데드라인을 만족하면서 전력 소모량의 확률적 기대치 값을 최소화하는 문제는, 각각의 n에 대해서$(1 \leq n \leq N)$ 수식 (1)은 만족하면서 수식 (2)를 최소화하고 문제와 동일하다[11]. 본 논문에서는 주어진 작업의 데드라인을 만족하면서 전력 소모량의 확률적 기대치 값을 최소화하는 스케쥴을 '최적 스케쥴'이라고 명한다.

2. 제안된 스케쥴링 기법

수식 (1)을 만족하면서 수식 (2)를 최소화하는 최적 스케쥴은 아래의 정리(theorem)들에서 보여주는 특성들을 가진다.

정리 1: 최적 스케쥴은 $F_x < F_y < F_z$에 대해서 $\frac{E_y - E_x}{F_y - F_x} > \frac{E_z - E_y}{F_z - F_y}$인 '전력 비효율적' 클락 주파수 F_y는 사용하지 않는다.

(증명): 최적 스케쥴이 전력 비효율적 클락 주파수 F_y를 사용하여 $C_y = (C_x + C_z)$개의 연산 사이클들을 수행한다고 가정한다. '새로운 스케쥴'로 명명된 스케쥴은 F_x를 사용하여 C_x개의 연산 사이클들을 수행하고, F_z를 사용하여 C_z개의 연산 사이클들을 수행한다. $\frac{C_x + C_z}{F_y} = (\frac{C_x}{F_x} + \frac{C_z}{F_z})$인 경우에 가정된 최적 스케쥴과 '새로운 스케쥴'은 동일한 실행 시간을 가지고, 이 수식을 변경하면 $C_z \cdot \frac{F_x}{F_x - F_y} = C_x \cdot \frac{F_z}{F_y - F_z}$이다. 이 수식을 전력 비효율적 클락 주파수의 조건인 $\frac{E_y - E_x}{F_y - F_x} > \frac{E_z - E_y}{F_z - F_y}$에 대입하면, $E_y \cdot \frac{C_x + C_z}{F_y} > (E_x \cdot \frac{C_x}{F_x} + E_z \cdot \frac{C_z}{F_z})$이다. F_y를 사용하여 $(C_x + C_z)$개의 연산 사이클들을 수행할 때의 전력 소모량은 $E_y \cdot \frac{C_x + C_z}{F_y}$이고, F_x를 사용하여 C_x개의 연산 사이클들을 수행하고 F_z를 사용하여 C_z개의 연산 사이클들을 수행할 때의 전력 소모량은

$(E_x \cdot \frac{C_x}{F_x} + E_z \cdot \frac{C_z}{F_z})$이다. 이 사실은 가정된 F_y를 사용하는 최적 스케줄보다 전력 소모량이 적은 스케줄이 존재함을 의미하는 것으로, 최적 스케줄 정의에 모순된다. 모순 증명법에 근거하여 최적 스케줄은 F_y를 사용하지 않는다. ▣

위의 정리 1에 근거하여 최적 스케줄은 전력 비효율적인 클락 주파수를 사용하지 않으므로, 본 논문에서는 전력 비효율적인 클락 주파수는 코어가 사용하는 클락 주파수 집합 $\{F_1, ..., F_K\}$ 구성에서 제외시킨다. 클락 주파수 값을 입력으로 하고 클락 주파수의 단위 시간당 전력 소모량을 출력으로 하는 함수에서, 전력 비효율적인 클락 주파수는 증가 함수의 특성($F_x < F_y < F_z$에 대해서 $\frac{E_y - E_x}{F_y - F_x} \leq \frac{E_z - E_y}{F_z - F_y}$)을 위배하는 입력 값이다. 따라서 남겨진 클락 주파수들은 그림 2와 같이 증가 함수의 특성을 가진다.

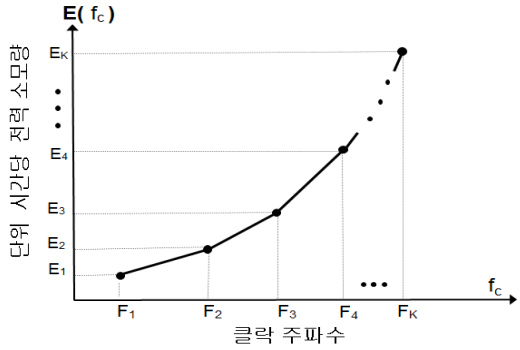

그림 2. 클락 주파수의 전력 소모량 함수
Fig. 2. Power consumption function of clock frequencies

정리 2: 최적 스케줄에서 $1 \leq c1 < c2 \leq W$이면 $f^n_{c1} \leq f^n_{c2}$이다.

(증명): $1 \leq c1 < c2 \leq W$에 대해서 $f^n_{c1} > f^n_{c2}$이라고 가정한다. $f^n_{c1}, f^n_{c2} \in \{F_1, ..., F_K\}$이므로 $F_a = f^n_{c1}$이고 $F_b = f^n_{c2}$이라고 하면 $F_a > F_b$이다. II장 2절의 마지막 문장에서 보여주듯이 $P^n_{c1} \geq P^n_{c2}$이므로, $(F_a \cdot P^n_{c1} + F_b \cdot P^n_{c2}) > (F_b \cdot P^n_{c1} + F_a \cdot P^n_{c2})$이다. 즉 $c1$번째 사이클에 F_a를 사용하고 $c2$번째 사이클에 F_b를 사용하는 최적 스케줄보다, $c1$번째 사이클에 F_b를 사용하고 $c2$번째 사이클에 F_a를 사용하는 스케줄이 더 적은 전력 소모량을 보인다. 이 사실은 가정된 최적 스케줄보다 전력 소모량이 적은 스케줄이 존재함을 의미하는 것으로, 모순 증명법에 근거하여 최적 스케줄에서는 $f^n_{c1} \leq f^n_{c2}$이다. ▣

위의 정리 2에 근거하여 최적 스케줄에서는 낮은 클락 주파수에서 높은 클락 주파수로의 전환만 이루어지고, 반대의 경우는 발생하지 않는다. 낮은 클락 주파수에서 높은 클락 주파수의 전환 지점을 표시하기 위해서, $\pi_1, ..., \pi_K$를 다음과 같이 정의하였다. π_k는 $F_{(k-1)}$에서 F_k로 전환된 시점(F_k가 적용된 최초 연산 사이클 지점)을 나타낸다. F_k는 사용되지 않고 $F_{(k-1)}$에서 $F_{(k+1)}$로 전환된다면, $\pi_k = \pi_{(k+1)}$이다. 정리 2에 따르면 최적 스케줄에서는 $\pi_k \leq \pi_{(k+1)}$이다. 또한 F_1이 가장 낮은 클락 주파수이므로 $\pi_1 = 1$이다. $\pi_k \leq \pi_{(k+1)}$ 특성을 이용하고 또한 실행시간이 데드라인에 종료될 때 수식 (2)가 최소화된다는 사실[11]을 반영하면, 수식 (1)은 다음과 같이 변경될 수 있다.

$$\frac{\pi_2 - 1}{F_1} + \frac{\pi_3 - \pi_2}{F_2} + ... + \frac{W - \pi_K}{F_K} = D \quad \text{수식 (3).}$$

또한 수식 (2)는 다음과 같이 변경될 수 있다.

$$E_1 \cdot \sum_{c=1}^{\pi_2 - 1} P^n_c + E_2 \cdot \sum_{c=\pi_2}^{\pi_3 - 1} P^n_c + ... + E_K \cdot \sum_{c=\pi_K}^{W} P^n_c \quad \text{수식 (4).}$$

최적 스케줄을 찾는 문제는 수식 (3)의 조건을 만족하면서 수식 (4)를 최소화하는 π_k 값들을 찾는 문제로 귀결된다. π_k 값들은 데드라인 D 값과 할당된 코어 개수 n의 값에 영향을 받고, 또한 아래 정리 3에서 보여주듯이 상호간에도 서로 연관성을 가진다.

정리 3: 최적 스케줄에서 $\pi_1, ..., \pi_K$의 값들은 다음의 관계식을 따른다: $2 \leq x < y \leq K$에 대해서

$$P^n_{\pi_x} \cdot \frac{E_x - E_{x-1}}{1/F_{x-1} - 1/F_x} = P^n_{\pi_y} \cdot \frac{E_y - E_{y-1}}{1/F_{y-1} - 1/F_y} \quad \text{수식 (5).}$$

(증명): 수식 (3)의 좌측 식을 변경하면 다음과 같다.

$$\frac{\pi_2-1}{F_1}+\frac{\pi_3-\pi_2}{F_2}+...+\frac{W-\pi_K}{F_K}=$$

$$-\frac{1}{F_1}+\pi_2\bullet(\frac{1}{F_1}-\frac{1}{F_2})+...+\pi_K\bullet(\frac{1}{F_{K-1}}-\frac{1}{F_K})+\frac{W}{F_K}.$$

또한 수식 (4)을 변경하면 다음과 같다.

$$E_1\bullet\sum_{c=1}^{\pi_2-1}P_c^n+E_2\bullet\sum_{c=\pi_2}^{\pi_3-1}P_c^n+...+E_K\bullet\sum_{c=\pi_K}^{W}P_c^n=$$

$$(E_1-0)\bullet\sum_{c=1}^{W}P_c^n+(E_2-E_1)\bullet\sum_{c=\pi_2}^{W}P_c^n+...$$

$$+(E_K-E_{K-1})\bullet\sum_{c=\pi_K}^{W}P_c^n.$$

위의 변경된 수식들을 이용하여 Lagrange Multiplier Method[12]를 적용한다. $L(\pi_2,...,\pi_K,\lambda)$ 함수를 다음과 같이 정의한다.

$$L(\pi_2,...,\pi_K,\lambda)=\sum_{k=2}^{K}\{(\mathrm{E_k}-\mathrm{E_{k-1}})\bullet\sum_{c=\pi_k}^{W}\mathrm{P_c^n}\}+\mathrm{E_1}\bullet\sum_{c=1}^{W}\mathrm{P_c^n}$$

$$+\lambda\bullet\{\mathrm{D}-(\sum_{k=2}^{K}\pi_k\bullet(\frac{1}{\mathrm{F_{k-1}}}-\frac{1}{\mathrm{F_k}})+(\frac{\mathrm{W}}{\mathrm{F_K}}-\frac{1}{\mathrm{F_1}}))\}.$$

위의 수식 $L(\pi_2,...,\pi_K,\lambda)$을 각 π_k에 대해서 미분하면,

$(E_k-E_{k-1})\bullet(-P_{\pi_k}^n)-\lambda\bullet(\frac{1}{F_{k-1}}-\frac{1}{F_k})=0$이고,

즉 $-\lambda=P_{\pi_k}^n\bullet\frac{E_k-E_{k-1}}{1/F_{k-1}-1/F_k}$ 이다.

λ 값은 상수이므로, 각 π_k에 대해서

$P_{\pi_k}^n\bullet\frac{E_k-E_{k-1}}{1/F_{k-1}-1/F_k}$의 값은 서로 동일하고, 따라서

$P_{\pi_x}^n\bullet\frac{E_x-E_{x-1}}{1/F_{x-1}-1/F_x}=P_{\pi_y}^n\bullet\frac{E_y-E_{y-1}}{1/F_{y-1}-1/F_y}$이다. ▣

위의 정리 3에 근거하여, π_K의 값을 고정시키면 수식 (5)의 관계식을 만족시키는 $\pi_{K-1},...,\pi_2$의 값들을 찾을 수 있다. 고정된 π_K의 값을 $\pi_K=1$부터 $\pi_K=W$까지 변경시키면서 수식 (5)의 관계식을 만족시키는 $\pi_{K-1},...,\pi_2$의 값들을 찾으면, 수식 (3)을 만족시키는 $\pi_{1,}...,\pi_K$의 값들은 결정할 수 있다. n의 값의 따라서 $\pi_{1,}...,\pi_K$의 값들이 달라지기 때문에, 모든 n의 값들($1\le n\le N$)에 대해서 $\pi_{1,}...,\pi_K$의 값들을 찾는다. 그리고 각각의 n에 대해서 찾아진 $\pi_{1,}...,\pi_K$를 수식 (4)에 적용하여, 수식 (4)를 최소화하는 n을 선택한다.

각각의 n에 대해서 $\pi_{K-1},...,\pi_2$의 값들을 찾는 상세 과정은 다음과 같다. π_K의 값을 $\pi_K=1$로 초기화하고, 수식 (5)에서 도출된 아래의 수식 (6)을 만족시키는 π_x의 값을 $x=K-1,...,2$에 대해서 확률적 계산량 모델 P_c^n의 분포도에서 찾는다.

$$P_{\pi_x}^n=P_{\pi_K}^n\bullet\frac{E_K-E_{K-1}}{1/F_{K-1}-1/F_K}\bullet\frac{E_x-E_{x-1}}{1/F_{x-1}-1/F_x}\quad \text{수식 (6)}.$$

확률적 계산량 모델 $P_{\pi_x}^n$의 값은 항상 1 이하이므로, $P_{\pi_x}^n>1$이면 π_x의 값을 1로 설정한다.

결정된 $\pi_K,...,\pi_2$의 값들을 수식 (3)에 적용하여 데드라인 D 값과 동일한지 비교한다. 데드라인 D 값보다 작다면, π_K의 값을 1만큼 증가시키고 수식 (6)을 이용한 $\pi_{(K-1)},...,\pi_2$ 값들을 찾는 과정을 다시 수행한다. $\pi_K=W$로 고정하고 찾아진 $\pi_{(K-1)},...,\pi_2$ 값들을 수식 (3)에 적용하여도 데드라인 D 값보다 작다면, $\pi_K=W$로 고정하고 $\pi_{(K-1)}$의 값을 1만큼 증가시켜서 나머지 $\pi_{(K-2)},...,\pi_2$ 값들을 수식 (6)을 이용하여 찾는다. 비슷한 과정으로 $\pi_y=W$로 고정하고 찾아진 $\pi_{(y-1)},...,\pi_2$ 값들을 수식 (3)에 적용하여도 데드라인 D 값보다 작다면, $\pi_y=W$로 고정하고 $\pi_{(y-1)}$의 값을 1만큼 증가시켜서 나머지 $\pi_{(y-2)},...,\pi_2$ 값들을 수식 (6)을 이용하여 찾는다. 이 과정을 결정된 $\pi_K,...,\pi_2$의 값들을 수식 (3)에 적용하여 데드라인 D 값과 동일할 때까지 반복한다.

위에서 설명한 수식 (3)을 만족하는 $\pi_K,...,\pi_2$의 값들을 찾는 과정의 평균 계산 복잡도는 $O(N\bullet K\bullet(\log_2 W)^2)$이다. $\pi_y=W$로 고정하고 찾아진 $\pi_{(y-1)},...,\pi_2$ 값들을 수식 (3)에 적용하여도 데드라인 D 값보다 작아서, 고정 값을 π_y 값 대신 $\pi_{(y-1)}$로 교체하는 과정은 최대 $(K-2)$번 발생한다. 고정된 π_y 값을 기준으로 수식 (6)을 만족하는 $\pi_{(y-1)},...,\pi_2$ 값들을 찾는 과정은, 이분법 탐색(bisectional search) 기법을 적용하면 각각의 π에 대해서 $O(\log_2 W)$

계산 복잡도를 가진다. 또한 고정된 π_y 값을 기준으로 찾아진 $\pi_{(y-2)},\ldots,\pi_2$ 값들을 수식 (3)을 적용하여 데드라인 D 값과 동일하게 만들기 위해서, 고정된 π_y 값을 1만큼 증가시키는 대신 이등분법(탐색영역의 최소 값과 최대 값의 중간 값을 적용)을 적용하면 계산 복잡도를 줄일 수 있다. 이등분법을 적용하였을 때, 수식 (3)을 적용하여 데드라인 D 값과 동일한 결과를 도출하는 고정된 π_y 값을 결정하는 과정은 $O(\log_2 W)$ 계산 복잡도를 가진다. 따라서 각각의 n에 대해서 $\pi_K,\ldots,\pi_2$의 값들을 찾는 전체 과정의 계산 복잡도는 $O((K-2)\bullet\log_2 W+K\bullet\log_2 W\bullet\log_2 W)=O(K\bullet(\log_2 W)^2)$이고, 따라서 모든 n의 값에 대한 전체 과정의 계산 복잡도는 $O(N\bullet K\bullet(\log_2 W)^2)$이다.

IV. 성능 평가

성능 평가를 위하여 제시된 기법과 기존 방법[8]의 전력소모량을 비교하였다. 제시된 기법이 전력 소모량을 줄이기 위해서 병렬 처리 기법을 활용하였다면, 기존 방법에서는 작업을 하나의 코어에서만 실행하였다. 기존 방법에서는 사용되지 않는 코어들의 전원을 소등하는 것을 고려하지 않았지만, 본 실험에서는 최대한 엄격한 조건에서 성능 비교가 되도록 사용되지 않는 코어들의 전원을 소등하도록 기존 방법을 개선하여 적용하였다. 또한 기존 방법은 DVFS 기법이 무한 개으로 연속된 클락 주파수들을 사용하는 비현실적 DVFS 기법을 가정하였지만, 본 실험에서는 유한개의 이산적 클락 주파수 값들을 사용하여 최소 전력 소모량의 가진 스케쥴을 찾도록 기존 방법을 개선하여 적용하였다. 평가 지표로는 '기존 방법이 소모하는 단위 시간당 전력량' 대비 '제시된 방법이 소모하는 단위 시간당 전력량' 비율을 '상대적 전력 소모량(Normalized Power Consumption: NPC)'이라고 정의하고, 이를 제시된 방법의 성능 지표로 사용하였다.

표 1. 프로세서 모델
Table 1. Processor Model

k	1	2	3	4	5
F_k (MHz)	150	400	600	800	1000
E_k (mW)	80	170	400	900	1600

실용적인 DVFS 프로세서 실험을 위해서, 실제로 널리 사용되는 인텔 XScale 프로세서[3]의 DVFS 기법 데이터를 사용하였다. 표 1은 인텔 XScale 프로세서에서 제공하는 5개의 클락 주파수 값들과 단위 시간당 전력소모량 값을 보여주고 있다. 5개의 주파수 값들의 단위 시간당 전력소모량 함수 $E(f_c)$는 그림 2와 같은 오목 증가 함수이다.

모의실험에서는 작업들이 요구하는 연산 사이클 개수는 1K와 1000K 사이에서 정규 분포 또는 지수 분포를 따르도록 인위적으로 생성하였다. 천개의 작업들을 우선적으로 생성하여, 생성된 과거의 계산량 정보를 이용하여 II장 2절에서 설명한 확률적 계산량 모델을 도출하였다. 도출된 확률적 계산량 모델을 기반으로 실시간 작업의 데드라인을 만족하면서 전력 소모량을 최소화하는 스케쥴을 찾았고, 평가 신뢰도를 높이기 위하여 작업을 십만번 생성하여 찾아던 스케쥴들의 전력 소모량 평균 값을 실험 결과 지표로 사용하였다.

병렬 처리 기법은 일반적으로 할당된 코어 개수에 비례하여 계산량이 줄어들지만, 병렬 처리 기법의 속도 향상 효과는 작업의 분할성 특성에 따라 달라진다. 다양한 작업들의 병렬 처리 기법 효율성을 평가하기 위해, C개의 연산 사이클 계산량을 n개의 코어들에서 병렬 처리 기법을 적용할 때의 효율성 모델들을 다음과 같이 정의하였다.

선형 속도증가: 병렬 처리시 각 코어 계산량은 C/n
반선형 속도증가: 병렬 처리시 각 코어 계산량은 $2C/n$
제곱근 속도증가: 병렬 처리시 각 코어 계산량은 $C/\sqrt{n}$

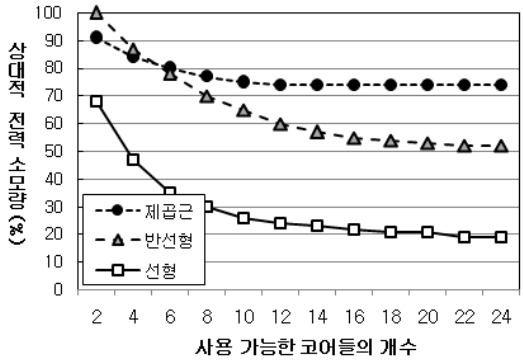

그림 3. 정규 분포상에서 상대적 전력소모량
Fig. 3. NPC over normal distribution

멀티코어 프로세서에서 사용 가능한 프로세싱 코어들의 개수 N의 값과 병렬 처리 기법의 효율성 모델에 대한 제시된 방법의 상대적 전력 소모량을 측정하였다. 그림 3은 작업의 계산량 부하가 정규 분포를 이용하여 생성되었을 때의 결과를 보여주고 있다. N의 값이 커지면 병렬 처리 기법에 적용할 수 있는 코어들의 개수도 같이 증가하여, 병렬 처리 기법을 이용한 전력 소모량 감소 효과도 커짐을 그림 3에서 보여주고

있다. 또한 병렬 처리 기법의 효율성면에서 '선형 속도증가' 모델이 전력 소모량 감소 효과가 가장 좋음을 보여주고 있다. '반선형 속도증가' 모델과 '제곱근 속도증가' 모델을 비교하면, N의 값이 4 이하일 때는 '제곱근 속도증가' 모델의 전력 소모량 감소 효과가 우위를 보이고, N의 값이 4를 초과하면 '반선형 속도증가' 모델이 우위를 보인다. $N=24$일 때, '선형 속도증가' 모델의 상대적 전력 소모량은 약 19%이다.

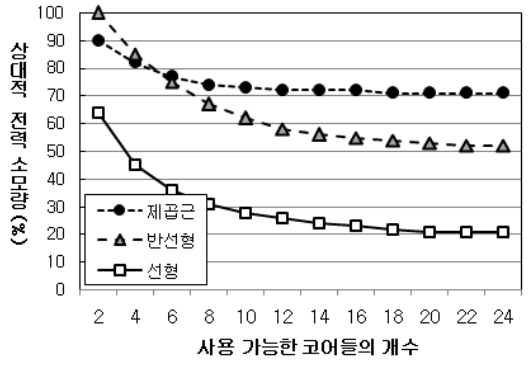

그림 4. 지수 분포상에서 상대적 전력소모량
Fig. 4. NPC over exponential distribution

그림 4는 작업들의 계산량 부하 값들이 지수 분포를 따를 때의 상대적 전력소모량을 보여주고 있다. 그림 3와 유사하게 N의 값이 커질수록 전력 소모량 감소 효과가 좋고 또한 '선형 속도증가' 모델이 전력 소모량 감소 효과가 가장 좋음을 보여주고 있다. $N=24$일 때, '선형 속도증가' 모델의 상대적 전력 소모량은 약 21%이다.

IV. 결론

본 논문에서는 멀티코어 프로세서에 적합한 실시간 작업용 저전력 스케쥴링 기법을 제안하였다. 제안된 기법은 단일 작업을 여러 개의 코어들상에서 병렬로 수행하는 방법을 적용하였다. 또한 한정된 개수의 이산적 클락 주파수 값들만을 제공하는 환경에서, 제안된 기법이 확률적 계산량을 가진 실시간 병렬 작업의 데드라인을 만족하면서 전력 소모량의 확률적 기대 값을 최소화함을 수학적으로 증명하였다. 성능 평가 실험에서, 제안된 기법과 기존 스케쥴링 기법의 전력 소모량을 최대 81%까지 감소시킴을 확인하였다.

참고문헌

[1] Semiconductor Industry Association (SIA), International Technology Roadmap for Semiconductors: 2005 Edition, http://www.itrs.net.

[2] L. Benini, A. Bogliolo, and G. Micheli, ``A survey of design techniques for system-level dynamic power management,'' IEEE Trans. VLSI Syst., vol. 8, no. 3, pp. 299-316, 2000.

[3] R. Xu, C. Xi, R. Melhem, and D. Moss, ``Practical PACE for embedded systems,'' ACM Int'l Conf. Embedded Software, 2005, pp. 54-63.

[4] C. Yang, J. Chen, and T. Kuo, ``An approximation algorithm for energy-efficient scheduling on a chip multiprocessor,'' Design, Automation and Test in Europe Conf., 2005, pp. 468-473.

[5] H. Aydin and Q. Yang, ``Energy-aware partitioning for multiprocessor real-time systems,'' Int'l Parallel Distributed Processing Symp., 2003, p. 113.2.

[6] A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. Hashimi, ``Energy optimization of multiprocessor systems on chip by voltage selection,'' IEEE Trans. VLSI Syst., vol. 15, no. 3, pp. 262-275, 2007.

[7] E. Seo, J. Jeong, S. Park, and J. Lee, ``Energy efficient scheduling of real-time tasks on multicore processors,'' IEEE Trans. Parallel Distrib. Syst., vol. 19, no. 11, pp. 1540-1552, 2008.

[8] C. Xian, Y. Lu, and Z. Li, ``Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time,'' Design Automation Conf., pp. 664-669, 2007.

[9] H. Pack, J. Yeo and W. Lee, ``Energy-efficient multi-core scheduling for real-time video processing,'' Journal of the Korea Society of Computer and Information, vol. 16, no. 6, pp. 11-20, 2011.

〔10〕 W. Lee, ``Power-efficient scheduling of periodic real-time tasks on lightly loaded multcore processors,'' Journal of the Korea Society of Computer and Information, vol. 17, no. 8, pp. 11-19, 2012.
〔11〕 W. Lee and K. Kim, ``Energy-saving stochastic scheduling of a real-time parallel task with varying computation amount on multi-core processors,'' IEICE Trans. Fundamentals, vol. E94-A, no. 2, pp. 842-845, 2011.
〔12〕 D. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, 1984.

저 자 소 개

이 완 연
1994: POSTECH
컴퓨터공학 공학사.
1996: POSTECH
컴퓨터공학 공학석사.
2000: POSTECH
컴퓨터공학과 공학박사
현 재: 동덕여자대학교
컴퓨터학과 교수
관심분야: 내장형 컴퓨터,
시스템소프트웨어,
바이오IT, 모바일 컴퓨팅
Email : wanlee@dongduk.ac.kr