Journal of The Korea Society of Computer and Information
Vol. 19, No. 4, April 2014

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.4.025

멀티코어 프로세서의 누수 전력을 고려한 실시간 작업들의 확률적 저전력 DVFS 스케쥴링

이 관우*

Stochastic Power-efficient DVFS Scheduling of Real-time Tasks on Multicore Processors with Leakage Power Awareness

Kwanwoo Lee*

요 약

본 논문에서는 멀티코어 프로세서 상에서 실시간 작업들의 데드라인들을 만족하면서 전력 소모량의 확률적 기대값을 최소화하는 문제를 해결하는 스케쥴링 기법을 제시하였다. 제시된 기법에서는 주어진 작업들의 불확실한 계산량을 과거의 계산량 분포에 기반하여 확률적 계산량으로 변환하고, 한정된 개수의 이산적 클락 주파수 값들을 이용하여 변환된 확률적 계산량의 전력 소모 기대 값을 최소화한다. 또한 시스템의 부하량이 적을 때에는 누수 전력을 고려하여 전체 코어들 중에서 일부의 코어들만을 사용하고 나머지 코어들의 전원을 소등시켜서 전력 소모량을 줄인다. 성능평가 실험에서 제시된 기법이 기존 방법의 전력 소모량을 최대 69%까지 감소시킴을 확인하였다.

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

Abstract

This paper proposes a power-efficient scheduling scheme that stochastically minimizes the power consumption of real-time tasks while meeting their deadlines on multicore processors. In the proposed scheme, uncertain computation amounts of given tasks are translated into probabilistic computation amounts based on their past completion amounts, and the mean power consumption of the translated probabilistic computation amounts is minimized with a finite set of discrete clock frequencies. Also, when system load is low, the proposed scheme activates a part of all available

•제1저자 : 이관우 •교신저자 : 이관우
•투고일 : 2014. 2. 3, 심사일 : 2014. 3. 5, 게재확정일 : 2014. 3. 11.
* 한성대학교 정보시스템공학과(Dept. of Information Systems Engineering, Hansung University)
※ 본 연구는 한성대학교 교내연구비 지원 과제임.

cores with unused cores powered off, considering the leakage power consumption of cores. Evaluation shows that the scheme saves up to 69% power consumption of the previous method.

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

I. 서 론

배터리 전원에 의존하여 동작하는 무선 컴퓨팅 기기들에게는 저전력 관리가 매우 중요한 문제이다. 실시간 작업들을 수행하는 프로세서의 전력 소모량을 줄이기 위해서 과거에는 사용하지 않는 프로세싱 코어(processing core)의 전원을 동적으로 소등시키는 DPM(dynamic power management) 기법[1]을 적용하였다면, 최근에는 프로세싱 코어에 입력되는 전압을 변화시켜서 프로세싱 코어의 연산 속도를 결정하는 클락 주파수(clock frequency)와 전력 소모량을 동적으로 조절하는 DVFS(dynamic voltage and frequency scaling) 기법[2,3]을 적용하여 DPM 기법보다 전력 소모 효율을 개선시켰다.

본 논문에서는 DVFS 기법을 제공하는 멀티코어 프로세서 상에서 주어진 실시간 작업들의 데드라인을 만족하면서 전력 소모량을 최소화는 스케쥴링 기법을 다룬다. DVFS 기법을 활용한 기존의 실시간 스케쥴링 연구들[2-6]은 프로세싱 코어들을 모두 사용한다는 전제하에서 전력 소모량을 줄이는 방법을 다루었다. 그러나 최근 연구[7,8]에서 시스템 부하량이 적을 때에는 사용 빈도가 낮은 프로세싱 코어들에 대해서 할당된 작업들을 다른 코어에게 전가하고 전원을 소등시키는 것이 오히려 프로세서 전체 전력 소모량을 줄일 수 있다는 사실이 입증되었다. 따라서 본 논문에서 제시된 스케쥴링 기법은 시스템 부하량이 적을 때에는 모든 프로세싱 코어들을 사용하지 않고 일부의 프로세싱 코어들만을 사용하고 나머지 사용하지 않는 코어들의 전원은 소등하도록 설계되었다.

또한 제시된 기법은 작업의 불확실한 계산량을 과거의 계산량 분포에 근거하여 확률적 계산량 모델로 표현하고, 도출된 확률적 계산량 모델에 근거하여 전력 소모 확률적 기대 값을 최소화하였다. 그리고 한정된 개수의 이산적 클락 주파수 값들만을 제공하는 DVFS 기반 멀티코어 프로세서 상에서 동작 가능하도록 설계되었다. 성능 평가 실험을 통하여, 제시된 기법이 기존 스케쥴링 기법의 전력 소모량을 최대 69%까지 줄임을 보였다.

멀티코어 프로세서 상에서 연구된 대부분의 기존 저전력 스케쥴링 기법[2-6]들은 일부 코어의 전원을 소등하여 전력 소모 효율성을 개선시키는 가능성을 배제하고 설계되었다. 최근의 저전력 스케쥴링 기법[7,8]에서 일부 코어의 전원 소등을 고려하였지만, 고정된 계산량에 적합하도록 설계되어 불확실한 계산량을 가지는 작업들에 대해서는 동작 효율성이 많이 떨어진다. 기존 연구 [9]에서는 일부 코어의 전원 소등을 고려하였고 불확실한 계산량에 적합하도록 설계되었지만, 무한 개수의 연속된 클락 주파수 값들을 적용한 비현실적 DVFS 기법 환경에서 동작하도록 설계되었다. 반면 제시된 기법은 한정된 개수의 이산적 클락 주파수 값들만을 적용하는 현실적 DVFS 기법 환경에서 전력 소모량의 확률적 기대 값을 최소화하도록 동작한다. 실제 생활에서 사용하는 DVFS 기반 프로세서는 한정된 개수의 이산적 클락 주파수 값들만을 제공하므로[3,10], 본 논문에서 제시된 기법이 실생활의 프로세서들에게 적용 가능한 실용적인 방법이다.

본 논문의 나머지 부분은 다음과 같이 구성된다. II 장에서는 DVFS 기반의 멀티코어 프로세서 모델과 확률적 계산량을 가진 실시간 작업 모델에 대해서 설명한다. III 장에서는 제시된 스케쥴링 기법의 동작 과정을 상세히 설명하고, IV 장에서는 제시된 스케쥴링 기법과 기존 스케쥴링 기법의 성능 비교 실험을 다룬다. 마지막으로 V 장에서는 본 논문의 내용을 요약하고 정리한다.

II. 시스템 환경

1. 실시간 작업 모델

서로 독립적으로 수행되는 주기적 실시간 작업들의 개수를 M으로 나타내고, 이들 M개의 작업들을 $T^1, ..., T^M$로 표시한다. 주기적 실시간 작업들의 도착 주기가 실행을 완료해

야 하는 데드라인(deadline)이 되며, 각각의 작업 T^m의 데드라인을 D^m로 나타낸다. 작업들은 다른 D^m 값들을 가질 수 있다. 또한 각각의 작업 T^m이 데드라인까지 수행을 완료해야하는 계산량(수행에 소요되는 전체 클락 사이클 개수)은 사전에 알려지지 않고 수행을 완료해야만 알 수 있다. 다만 작업들의 불확실한 계산량에 대해서도 최대 값은 주어지고, T^m의 최대 계산량을 W^m로 나타낸다.

작업이 요구하는 변동하는 불확실한 계산량은 사전에 정확하게 알 수는 없지만, 과거에 도착하여 수행을 완료한 작업들의 계산량 분포 정보를 이용하여 미래에 도착할 작업의 계산량을 확률적으로 예측할 수 있다[5,9,10,11]. 그림 1은 과거의 계산량 분포 정보를 이용하여 미래 작업의 불확실한 계산량을 확률적 계산량 모델로 표현하는 방법을 설명하고 있다.

그림 1(a)은 과거에 도착하여 수행을 완료한 작업들에 대해서 소요된 클락 사이클 개수의 확률 분포(probability distribution)를 보여주고 있다. p_c는 작업이 수행을 완료하였을 때, c 개의 클락 사이클 개수가 소요된 확률을 나타낸다. 그림 1(b)는 그림 1(a)의 확률 분포에 대한 누적 확률 분포(cumulative probability distribution)를 보여주고 있다. 하나의 코어에서 작업 계산량이 c 개 이하의 클락 사이클 개수를 가질 확률은 $\sum_{i=1}^{c} p_i$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 '이상'의 클락 사이클 개수를 가질 확률은 $\Phi_c = 1 - \sum_{i=1}^{c} p_i$이다. 본 논문에서는 작업의 계산량이 c 개 '이상'의 클락 사이클 개수를 요구할 확률을 Φ_c로 표기하였다. Φ_c는 작업이 c 개의 클락 사이클을 수행한 이후에도 실행이 완료되지 않고 잔여 계산량이 남아 있을 확률을 의미한다. 그림 1(b)의 누적 확률 분포는 항상 증가 함수를 가지므로, 그림 1(c)에서 보여준 누적 차 확률 분포는 항상 감소 함수로 표현된다. 즉, $c1 < c2$ 이면 $\Phi_{c1} \geq \Phi_{c2}$이다.

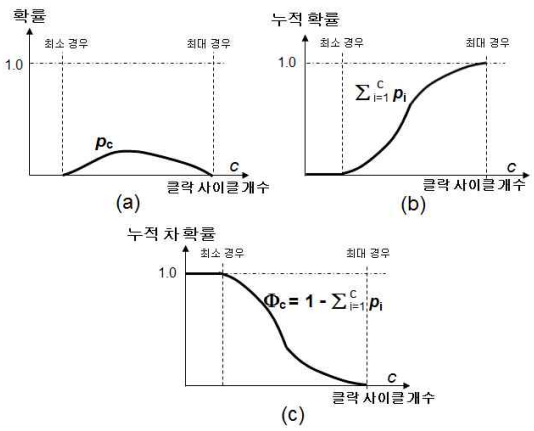

그림 1. 작업의 확률적 계산량 모델
Fig. 1. Probabilistic computation model of task

2. 프로세서 모델

멀티코어 프로세서에서 사용 가능한 프로세싱 코어들의 개수를 N으로 나타낸다. 멀티코어 프로세서는 DVFS 기법을 제공하고, 사용되지 않는 프로세싱 코어들의 전원은 별도로 소등하여 전력 소모량을 줄인다. 사용 가능한 K개의 클락 주파수(clock frequency)들을 오름차순으로 정렬하여 $F_1, ..., F_K$으로 나타낸다.

프로세싱 코어들에게 적용되는 클락 주파수는 코어가 단위 시간당 처리하는 계산량(클락 사이클 개수)을 의미하고, 따라서 클락 주파수가 증가하면 코어의 처리 속도가 증가하며 또한 코어가 소모하는 전력량도 같이 증가한다. 각 클락 주파수 F_k가 단위 시간당 소모하는 전력량을 P_k로 나타낸다. 정의에 따라 $F_i < F_j$ 이면 $P_i < P_j$ 이다. 프로세싱 코어들에게 적용되는 클락 주파수는 언제든지 변경할 수 있고, 같은 순간에 코어들에게 다른 클락 주파수 값을 적용할 수 있다. 프로세싱 코어가 유휴시간(idle time)에 소모하는 누수전력(leakage power)을 P_0로 나타내고, 유휴시간의 누수전력에 대칭하는 가상의 클락 주파수 F_0를 새롭게 가정하여 사용한다. 유휴시간에는 아무런 작업도 수행하지 않으므로, $F_0 = 0$의 의미를 가진다. 즉, $P_0 < P_1$ 이면서 $F_0 < F_1$ 이다. 주어진 작업의 c번째 연산 클락 사이클이 소요하는 단위 시간당 전력 소모량을 $E(f_c)$로 표현하고, 이 경우에 $f_c \in \{F_1, ..., F_K\}$ 이므로 $E(F_1) = P_1$, ..., $E(F_K) = P_K$ 이다.

III. 제시된 스케쥴링 기법

1. 단일 코어 상에서 준최적 스케쥴링 기법

기존에 알려진 무한 개수의 연속적 클락 주파수 값들을 이용한 최적 해법[9]을 활용하여 한정된 개수의 이산적 클락 주파수 값들에 대한 준최적 해법을 도출한다. 기존 연구[9]에서는 클락 주파수 f_c가 소모하는 전력량을 $(f_c)^3$라고 가정하였

다. 먼저 주어진 이산적 클락 주파수 값들로부터 연속적 클락 주파수에 대한 전력 소모량 수식 $(\alpha \cdot f_c^3 + P_0)$을 도출한다. 여기서 f_c는 c번째 연산 클락 사이클에 적용된 클락 주파수 값이고, α는 프로세서 특성을 나타내는 상수 계수이며, 프로세싱 코어의 P_0는 누수 전력을 나타내는 상수이다. 연속적 클락 주파수에 대한 전력 소모량 수식을 도출하는 방법은, 매트랩(MATLAB) 프로그램을 사용하여 주어진 이산적 클락 주파수 값들로부터 최소 거리 합을 가지도록 α 값과 P_0 값을 결정한다.

α와 P_0 값이 결정되면, 단일 코어 상에서 주어진 작업들의 전력 소모량 확률적 기대 값을 최소화하도록 f_c를 다음과 같이 결정할 수 있다. 작업 T^m의 c번째 연산 클락 사이클에 적용되는 클락 주파수 값을 f_c^m로 표기하면, c번째 연산 클락 사이클을 실행하는데 소요되는 시간은 $1/f_c^m$이다. 따라서 단일 코어 상에서 주어진 X개 작업들의 최대 계산량을 항상 데드라인 이전에 실행을 완료하기 위해서는 다음 수식 (1)을 만족하여야 한다.

$$\sum_{m=1}^{X} \frac{\sum_{c=1}^{W^m} 1/f_c^m}{D^m} \leq 1 \qquad \text{수식 (1)}$$

또한 단일 코어 상에서 확률적 계산량 Φ_c^m를 가진 X개 작업들의 전력 소모 확률적 기대 값은 다음의 수식 (2)와 같이 정의될 수 있다.

$$\sum_{m=1}^{X} \frac{\sum_{c=1}^{W^m} \alpha \cdot (f_c^m)^3 \cdot \Phi_c^m}{D^m} + P_0 \qquad \text{수식 (2)}$$

기존 연구 [9]에서는 연속적 클락 주파수$(\alpha \cdot f_c^3 + P_0)$에 대해서, 단일 코어 상에서 다중 작업들에 대해서 수식 (1)을 만족하면서 수식 (2)를 최소화하는 스케쥴링 방법을 도출하였다. 단일 코어에서 X개 작업들을 스케쥴링할 때, 작업 T^m의 c번째 연산 사이클에 적용되는 클락 주파수 f_c^m는

$$f_c^m = (\frac{1}{\Phi_c^m})^{1/3} \cdot \sum_{m=1}^{X} \frac{\sum_{c=1}^{W^m} (\Phi_c^m)^{1/3}}{D^m} \qquad \text{수식 (3)}$$

이다. $\frac{\sum_{c=1}^{W^m} (\Phi_c^m)^{1/3}}{D^m} = \Pi^m$로 간략화하여 표현하면, 수식 (3)의 f_c^m을 수식 (2)에 적용한 최소 전력 소모 기대 값은

$$\alpha \cdot (\sum_{m=1}^{X} \Pi^m)^3 + P_0 \qquad \text{수식 (4)}$$

이다.

수식 (3)에서 도출된 클락 주파수 값은 무한 개수의 연속적 클락 주파수들을 제공하는 비현실적 프로세서 환경에서만 사용 가능하고, 한정된 개수의 이산적 클락 주파수들로만 구성된 프로세서 환경에서는 수식 (3)에서 도출된 f_c^m를 주어진 이산 클락 주파수들 $F_1, ..., F_K$에서 유사한 하나를 선택하여 교체하여야 한다. 수식 (4)의 최적 해법에 최대한 근접하기 위해서, $F_{k-1} < f_c \leq F_k$인 경우에 f_c^m에 F_k 값을 적용한다. 만약 f_c^m에 F_k 대신에 상대적으로 느린 클락 주파수 F_{k-1}를 적용하면, 실행 시간이 증가하여 수식 (1)의 조건인 데드라인 이내에 수행을 완료하지 못하는 문제가 발생하게 된다.

예외적으로 $f_c^m > F_K$인 경우에는, 최대 클락 주파수 F_K보다 큰 클락 주파수가 제공되지 않기 때문에 f_c^m에 F_K 값을 적용해야 한다. 이 경우에는 실행 시간이 증가하여 수식 (1)의 조건인 데드라인을 만족하지 못하는 문제가 발생하며, 이를 해결하기 위해서 다른 곳에 할당된 클락 주파수 값을 증가시켜서 실행 시간을 단축시켜야만 한다. 전력 소모량 확률적 기대 값을 낮추기 위해서는 낮은 Φ_c 값을 가진 계산 부분에 적용되는 클락 주파수 값을 증가시켜야 한다. II절 1장에서 설명하였듯이 주어진 계산량의 후반부 클락 사이클일수록 낮은 Φ_c 값을 가지므로, 후반부 사이클에서부터 적용될 클락 주파수 값을 증가시켜서 실행 시간을 단축시킨다. 정리하자면, c의 값을 W^m부터 적용하여 점진적으로 감소시키면서 수식 (3)의 f_c^m을 계산한다. $W^m \geq c \geq \beta$에 대해서 $f_c^m > F_K$라면, 증가될 실행 시간 $\sum_{c=\beta}^{W^m} (1/F_K - 1/f_c^m)$을 계산한 이후에 f_c^m에 F_K 값을 적용한다. $c < \beta$에 대해서는 $F_{k-1} < f_c \leq F_k$ 이므로, 감소될 실행 시간 $\sum_{c=1}^{\beta} (1/f_c^m - 1/F_k)$을 계산하고 f_c^m에 F_k 값을 적용한다. 최종적으로 증가될 실행 시간 $\sum_{c=\beta}^{W^m} (1/F_K - 1/f_c^m)$과 감소될 실행 시간 $\sum_{c=1}^{\beta-1} (1/f_c^m - 1/F_k)$을 비교하여, 감소

될 실행 시간이 증가될 실행 시간 보다 크거나 같아질 때까지 $\beta \geq c \geq 1$에 대해서 f_c^m에 F_k 대신에 F_{k+1}을 적용한다.

2. 단일 코어상에서 최적 스케쥴링 기법

본 절에서는 한정된 개수의 이산적 클락 주파수들을 사용하여 수식 (1)을 만족하면서 수식 (2)를 최소화하는 최적 해법을 도출한다. 단일 코어 상에서 단일 작업에 대해서 수식 (1)을 만족하면서 수식 (2)를 최소화하는 최적 해법은 기존 연구 [10]에서 도출되었다. 최적 해법에서는 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 클락 사이클 지점)을 나타낸다. $\pi_k \leq \pi_{(k+1)}$ 특성을 이용하면 단일 작업의 실행 시간이 데드라인 이내에 완료되는 조건을 다음과 같이 표현할 수 있다.

$$\frac{\sum_{c=1}^{W} 1/f_c}{D} \Rightarrow$$

$$(\frac{\pi_2 - 1}{F_1} + \frac{\pi_3 - \pi_2}{F_2} + ... + \frac{W - \pi_K}{F_K}) \bullet \frac{1}{D} \leq 1$$

수식 (5)

또한 누수 전력을 제외한 단일 작업의 전력 소모 확률적 기대 값은 다음과 같다.

$$\frac{\sum_{c=1}^{W} \alpha \bullet (f_c)^3 \bullet \Phi_c^m}{D^m} \Rightarrow$$

$$P_1 \bullet \sum_{c=1}^{\pi_2 - 1} \Phi_c + P_2 \bullet \sum_{c=\pi_2}^{\pi_3 - 1} \Phi_c ... + P_K \bullet \sum_{c=\pi_K}^{W} \Phi_c$$

수식 (6)

최적 스케쥴을 찾는 문제는 수식 (5)의 조건을 만족하면서 수식 (6)을 최소화하는 π_k 값들을 찾는 문제로 귀결된다. π_k 값들은 상호간에 다음과 같은 연관성을 가진다[10].

$$\Phi_{\pi_k} \bullet \frac{P_k - P_{k-1}}{1/F_{k-1} - 1/F_k} = \Phi_{\pi_{k+1}} \bullet \frac{P_{k+1} - P_k}{1/F_k - 1/F_{k+1}}$$

수식 (7)

π_K의 값을 고정시키면 수식 (7)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 (7)의 관계식을 만족시키는 $\pi_{K-1},...,\pi_2$의 값들을 찾으면, 수식 (5)을 만족하면서 수식 (6)를 최소화하는 $\pi_1,...,\pi_K$의 값들은 결정할 수 있다.

다음으로 다중 작업들에 대해서 수식 (1)을 만족하면서 수식 (2)를 최소화하는 최적 해법을 다음과 같이 도출할 수 있다. 수식 (1)을 단순화하기 위해서 γ^m을 다음과 같이 정의한다.

$$\gamma^m = \frac{\sum_{c=1}^{W^m} 1/f_c^m}{D^m} \leq 1$$

단일 코어 상에서 X개 작업들을 스케쥴링할 때, 모든 작업들의 데드라인들을 만족시키기 위해서는 수식 (1)로부터 $\sum_{m=1}^{X} \gamma^m \leq 1$ 이다. 각각의 작업 T^m에 대해서 단축된 데드라인 $\gamma^m \bullet D^m$과 Φ_c^m의 값에 근거하여 최적 해법을 찾는 과정, 다시 말해서 $\pi_1,...,\pi_K$의 값들을 결정하는 과정은 위의 단락에서 설명하였다. 주어진 $\gamma^m \bullet D^m$과 Φ_c^m의 값에 근거하여 T^m에 대해서 도출된 수식 (6)의 최소값을 Ψ^m로 표시하면, 최적 해법은 $\sum_{m=1}^{X} \Psi^m$가 최소가 되도록 $\sum_{m=1}^{X} \gamma^m \leq 1$의 조건에 맞는 한도에서 γ^m의 값들을 결정하는 것이다.

γ^m의 값들을 결정하는 과정은 다음과 같다. 먼저 γ^m의 값이 커지면 Ψ^m도 줄어들기 때문에, $\sum_{m=1}^{X} \Psi^m$을 최소화하기 위해서는 $\sum_{m=1}^{X} \gamma^m = 1$이다. 어떤 경우에도 최대 클락 주파수를 적용하면 항상 데드라인에 작업 수행을 완료할 수 있도록, 초기 값으로 $\gamma^m \bullet D^m = \frac{W^m}{F_K}$ 즉 $\gamma^m = \frac{W^m}{F_K \bullet D^m}$을 할당한다. 그리고 잔여 할당 가능한 시간 $(1 - \sum_{m=1}^{X} \gamma^m)$을 X개 작업들에 추가로 균등 분배할 경우, 추가 분배된 시간에 대한 Ψ^m의 감소량을 각각의 작업에

대해서 계산한다. 계산된 Ψ^m의 감소량을 비교하여, 가장 높은 감소량을 가진 작업에게만 실제로 추가 분배된 시간을 할당하고, 나머지 작업들에 대해서는 분배된 시간을 할당하지 않는다. 추가로 시간을 분배받은 해당 작업의 γ^m 값은 증가되고, 분배 이후에는 잔여 할당 가능한 시간 $(1-\sum_{m=1}^{X}\gamma^m)$은 줄어든다. 이처럼 잔여 할당 가능한 시간을 균등 분배하여 X개 작업들의 Ψ^m의 감소량을 계산하고, 가장 높은 감소량을 가진 작업에게만 실제로 분배된 시간을 추가 할당하는 과정을 $(1-\sum_{m=1}^{X}\gamma^m)\approx 0$가 될 때까지 반복한다.

윗 문단에서 설명한 γ^m 값들에 대한 결정 과정이 완료되면, $\sum_{m=1}^{X}\gamma^m = 1$ 이므로 수식 (1)을 만족한다. 또한 주어진 γ^m 값에 대해서 수식 (6)을 최소화하도록 $\pi_1,\ldots,\pi_K$의 값들을 결정하고, 단일 코어에 주어진 X개 작업들에 대해서 수식 (6)의 합들이 최소화되도록 γ^m 값들을 결정하였으므로 결론적으로 수식 (2)는 한정된 개수의 이산적 클락 주파수들만을 사용하여 최소화된다.

3. 작업 배치(task assignment) 기법

앞의 III장 1절과 2절에서는 단일 코어 상에서 다중 작업들을 배치하였을 때에 전력 소모 기대 값을 최소화하도록 스케쥴링하는 방법을 다루었다. 해결해야하는 남은 문제는 주어진 N개의 프로세싱 코어들 중에서 몇 개까지 활성화하여 사용하고 나머지는 전원을 소등할지 결정하는 문제와, 주어진 M개의 작업들을 활성화된 코어들에게 어떻게 배치할지 결정하는 문제이다.

N개의 프로세싱 코어들 중에서 활성화되어 사용되는 코어들의 개수를 η으로 표시하면, III장 1절에서 언급된 수식 (4)를 이용하여 η 값을 결정한다. 수식 (4)는 오목 증가 함수이므로, M개 작업들의 전체 Π 합을 활성화된 η개 코어들에게 균등 분배하여야 활성화된 코어들의 전체 전력 소모량이 최소화된다[2]. 간결성을 위해서 i번째 코어에 배치된 작업들의 Π 합에 대해서 수식 (4)를 적용한 값을 아래와 같이 표기한다.

$$CP_i(\sum\Pi) = \alpha \bullet (\sum\Pi)^3 + P_0$$

그리고 그림 2에서 보여주듯이 $\sum\Pi$ 값을 x축으로 했을 때, $CP_i(\sum\Pi)$ 함수와 임의의 직선 함수 $s \bullet \sum\Pi$와 단일 지점에서만 인접하도록 s의 값을 결정한다.

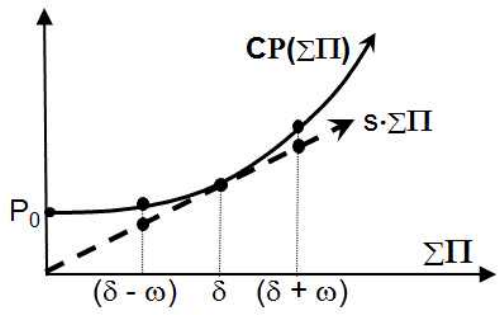

그림 2. δ값 도출 과정
Fig. 2. Derivation of δ value

두 개의 함수가 인접하는 단일 지점을 δ로 표기할 때, 활성화될 코어들의 개수 η의 값은 $\eta = \frac{\sum_{m=1}^{M}\Pi^m}{\delta}$ 로 결정된다. 만약 η의 값이 정수가 아니고 소수이면, 소수 η에 내림으로 인접한 정수와 올림으로 인접한 정수에 대해서 $\eta \bullet CP_i(\frac{\sum_{m=1}^{M}\Pi}{\eta})$의 값을 비교하여 작은 값을 가지는 정수를 선택한다.

$\eta = \frac{\sum_{m=1}^{M}\Pi^m}{\delta}$로 결정하는 것이 $\eta \bullet CP_i(\frac{\sum_{m=1}^{M}\Pi}{\eta})$의 값을 최소화한다. 그 이유는 다음과 같다. 양의 값 x에 대해서 $(\eta+x)$개의 코어들을 활성화하면, $(\eta+x)$개의 코어들에게 작업들의 전체 Π 합을 균등 분배하면 각 코어들에게 분배된 값은 δ보다 줄어들게 된다. 각 코어들에게 균등 분배된 값을 $(\delta-\omega)$로 표시하면, 그림 2에서 보여주듯이 $(\eta+x) \bullet CP(\delta-\omega) > (\eta+x) \bullet s \bullet (\delta-\omega)$ 이고 $(\eta+x) \bullet s \bullet (\delta-\omega) = \eta \bullet s \bullet \delta = \eta \bullet CP(\delta)$ 이므로, $(\eta+x) \bullet CP(\delta-\omega) > \eta \bullet CP(\delta)$ 이다. 즉, $(\eta+x)$개의 코어들에게 작업들의 전체 Π 합을 균등 분배하였을 경우의 $(\eta+x)$개 코어들의 최소 전력 소모량 $(\eta+x) \bullet CP(\delta-\omega)$은 η개의 코어들에게 전체 Π 합을 균등 분배하였을 경우의 η개 코어들의 최소 전력 소모량 $\eta \bullet CP(\delta)$보다 크다. 비슷한 이유로, $(\eta-x)$개의 코어들에게 작업들의 전체 Π 합을 균등 분배하면 $(\eta-x)$개 코어들의 최소 전력 소모량 $(\eta-x) \bullet CP(\delta+\omega)$은 η개 코어들에게 전체 Π 합을 균등 분배하였을 경우의 η개 코어들의 최소 전력 소모량 $\eta \bullet CP(\delta)$보다 크다.

마지막으로 M개의 작업들을 η개의 코어들에게 어떻게 배치할지 결정한다. 이상적으로는 M개 작업들이 배치를 완료한 이후에 각 코어에 배치된 작업들의 Π 합을 균등하게 하는 것이 전체 코어들의 전력 소모량을 최소화하지만, 각 코어에 배치된 작업들의 Π 합을 최대한 균등하게 되도록 작업들을 코어에게 배치하는 문제는 NP-hard 문제로 알려져 있다[2]. 따라서 기존 연구[2]에서 휴리스틱 배치 기법들 중에서 전력 소모량을 줄이는데 가장 좋은 성능을 보이는 것으로 밝혀진 WFD(Worst-Fit Decreasing) 휴리스틱 배치 기법을 본 논문에서 선택하여 적용한다. WFD 휴리스틱 배치 기법은 전체 작업들의 Π 값을 내림차순으로 정렬하여 Π 값이 큰 작업부터 Π 값이 작은 작업 순서로 배치한다. 정렬된 각 작업이 배치될 때, η개의 코어들에게 기존에 할당된 작업들의 Π 합을 계산하여 가장 적은 Π 합을 가진 코어에게 주어진 작업이 추가로 배치된다.

본 논문에서는 활성화될 코어의 개수를 결정할 때는 III장 1절에서 기술한 무한 개수의 연속적 클락 주파수들을 위한 최적 해법을 사용하고, WFD 휴리스틱 배치 기법을 적용하여 활성화된 코어들에게 작업들을 배치한 이후에는 III장 2절에서 기술한 한정된 개수의 이산적 클락 주파수들만을 사용하여 각 코어에게 배치된 다중 작업들의 전력 소모량 기대 값을 최소화하는 해법을 사용한다. 활성화될 코어의 개수를 결정할 때 III장 2절에서 기술한 방법을 사용하지 않는 이유는, III장 2절의 방법은 작업 배치를 완료한 이후에 각 코어의 전력 소모 확률적 기대 값을 정량적으로 계산하기가 어려워서, 상대적으로 전력 소모 확률적 기대 값을 정량적으로 계산하기가 용이한 III장 1절의 방법을 선택하였다.

IV. 성능 평가

성능 평가를 위하여 제시된 기법과 기존 방법[7,8]의 전력 소모량을 비교하였다. 제시된 기법이 확률적 계산량을 기반으로 활성화될 코어 개수와 클락 주파수 값을 결정하였다면, 기존 방법에서는 고정된 최대 계산량을 기반으로 활성화될 코어 개수와 클락 주파수 값을 결정하였다. 작업들을 활성화된 코어들에게 배치할 때는 제시된 기법과 기존 방법에서 동일하게 WDF 휴리스틱 기법을 적용하였다. 평가 지표로는 '기존 방법의 단위 시간당 전력 소모량' 대비 '제시된 방법의 단위 시간당 전력 소모량' 비율을 '상대적 전력 소모비율'이라고 정의하고, 이를 성능 지표로 사용하였다. DVFS 프로세서 실험을 위해서 실제로 널리 사용되고 있는 인텔 XScale 프로세서[3]의 DVFS 데이터를 사용하였다. 표 1은 인텔 XScale 프로세서에서 제공하는 5개의 클락 주파수 값들과 단위 시간당 전력 소모량 값을 보여주고 있다.

표 1. 프로세서 모델
Table 1. Processor Model

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

모의실험에서는 프로세서 내에 8개의 프로세싱 코어가 사용 가능하고, 스케쥴링 대상인 실시간 작업들의 개수는 16개, 24개, 32개로 변화를 주었다. 각 작업의 데드라인 값은 0.01초와 1초 사이에서 균등하게 선택하였고, 최대 계산량은 100K 연산 사이클과 100,000K 연산 사이클 사이에서 균등하게 선택하였다. 실제 계산량은 100K 연산 사이클과 100,000K 연산 사이클 사이에서 정규 분포를 가지도록 선택하였다. 동일한 실험 조건에 대해서 10만개의 작업 집합들을 생성하여 성능을 측정하고, 측정된 성능들의 평균을 실험 결과로 표시하였다.

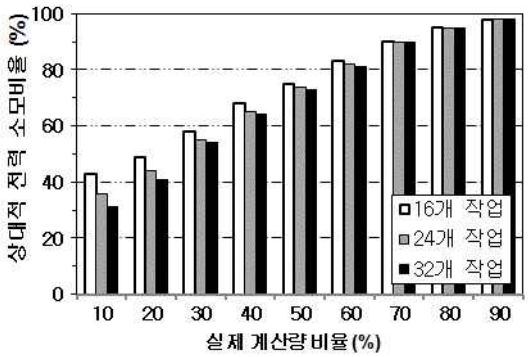

그림 3. 상대적 전력 소모비율
Fig. 3. Relative power consumption

그림 3은 제시된 방법의 상대적 전력 소모비율을 보여주고 있다. x축의 '실제 계산량 비율'은 작업들의 최대 계산량 대비 작업을 완료하는데 필요한 실제 계산량의 비율을 나타낸다. 실제 계산량 비율 값이 낮아질수록 제시된 기법의 상대적 전력 소모비율은 낮아진다. 그리고 작업들의 개수가 많아질수록 상대적 전력 소모비율이 다소간 낮아짐을 확인하였다. 즉, 제시된 방법은 작업들의 개수가 많고 실제 계산량이 최대 계산량보다 적을수록 전력 소모 절감 효과가 커진다. 실제 계산량

비율이 10%이고 작업들의 개수가 32개일 때, 상대적 전력 소모 비율이 31%이므로 제시된 방법이 기존 방법의 전력 소모량을 69% 감소시켰다.

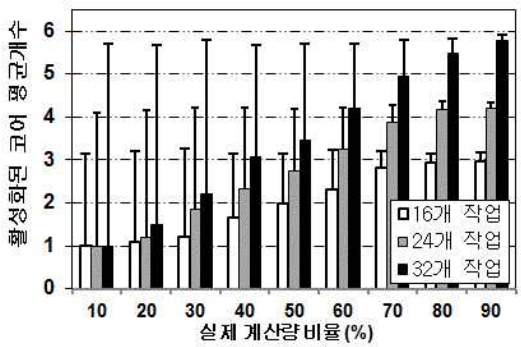

그림 4. 활성화된 코어 개수
Fig. 4. Number of activated cores

그림 4는 주어진 8개의 프로세싱 코어들 중에서 제시된 방법이 활성화하여 사용한 코어들의 평균 개수를 보여주고 있다. 직사각형 박스가 제시된 방법에서 활성화된 코어들의 개수를 나타내고, 박스 위의 선형 바는 기존 방법에서 활성화한 코어 개수와 제시된 방법에서 활성화된 코어 개수와의 차이를 나타낸다. 실제 계산량 비율이 낮아질수록 제시된 기법에서 활성화된 코어 개수는 줄어들고, 작업들의 개수가 많아질수록 활성화된 코어 개수는 증가한다. 반면 기존 방법에서는 실제 계산량 비율에는 영향을 받지 않고, 작업들의 개수가 많아질수록 활성화된 코어 개수가 증가한다. 실제 계산량 비율이 10%이고 작업들의 개수가 32개일 때, 제시된 방법이 기존 방법보다 활성화된 코어 개수를 약 83% 줄였다. 정리하자면, 실제 계산량 비율이 낮고 작업들의 개수가 많을수록 제시된 방법은 적은 수의 코어들만을 활성화하고 또한 전력 소모 절감 효과가 증대된다.

M개의 작업들과 N개의 코어들, 그리고 K개의 클락 주파수들이 주어진 환경에서, 기존 방법[7,8]의 계산 복잡도는 $O(M \cdot \log M + M \cdot N + N \cdot K)$로 상대적으로 낮은 반면, 제시된 기법의 계산 복잡도는 W가 작업들의 최대 계산량을 나타낼 때 $O(N \cdot M^2 \cdot K \cdot W^2)$로 상대적으로 높다. 그러나 제시된 스케줄링 기법은 오프라인(off-line)에 적용되는 정적(static) 방법으로, 프로세서가 작업들의 실행을 시작하기 이전에 스케줄링 계산이 한번 실시된다. 따라서 제시된 기법의 증가된 계산 복잡도가 미치는 영향은 전체적으로 매우 미미하고, 프로세서가 작업들의 실행을 시작한 이후에 미치는 영향은 전혀 없다.

제시된 기법과 무한 개수의 연속된 클락 주파수들을 사용한 저전력 스케줄링 방법[9]의 실험 결과를 비교하면, 제시된 기법의 상대적 전력 소모 비율이 다소 높다. 즉 제시된 기법의 전력 소모량 감소 비율이 상대적으로 낮다. 그 이유는 무한 개수의 연속된 클락 주파수들이 제공되는 환경에서는 최적의 클락 주파수를 자유롭게 선택하는 반면, 제시된 기법에서는 주어진 한정된 개수의 클락 주파수들 중에서만 최선의 클락 주파수를 제한적으로 선택하기 때문이다. 그리고 활성화되어 사용된 코어 개수는 동일하다. 그 이유는 III장 2절에서 설명하였듯이, 제시된 기법과 무한 개수의 연속된 클락 주파수들을 사용한 저전력 스케줄링 방법[9]이 동일한 과정을 사용하여 활성화될 코어 개수를 결정하기 때문이다.

V. 결 론

본 논문에서는 멀티코어 프로세서 상에서 실시간 작업들의 데드라인을 만족하면서 전력 소모량을 줄이는 스케줄링 기법을 제시하였다. 한정된 개수의 이산적 클락 주파수 값들만을 제공하는 DVFS 기반 멀티코어 프로세서 환경에서, 시스템의 부하량에 따라서 일부 코어의 전원을 소등할 수 있고, 불확실한 계산량을 확률적 계산량으로 변환하여 전력 소모 확률적 기대 값을 최소화하는 스케줄링 문제를 제시된 기법이 처음으로 해결하였다. 성능 평가 실험에서, 제시된 기법이 기존 스케줄링 기법의 전력 소모량을 최대 69%까지 감소시키고, 활성화된 코어 개수를 최대 83%까지 감소시킴을 확인하였다.

참고문헌

[1] 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.

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

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

[4] 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.

[5] 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.

[6] 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.

[7] W. Lee, ''Power-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors,'' Journal of the Korea Society of Computer and Information, Vol. 17, No. 8, pp. 11-19, 2012.

[8] E. Seo, J. Jeong, S. Park, and J. Lee, ''Energy efficient scheduling of real-time tasks on multicore processors,'' IEEE Trans. Parallel Distributed Syst., Vol. 19, No. 11, pp. 1540-1552, 2008.

[9] W. Lee, ''Stochastically power-minimum scheduling of real-time multicore systems with leakage power awareness,'' Electronics Letters, Vol. 49, No. 13, pp. 791-793, 2013.

[10] Z. Lu, Y. Zhang, M. Stan, J. Lach, and K. Skadron ''Procrastinating voltage scheduling with discrete frequency sets,'' Design, Automation and Test in Europe Conf., pp. 456-461, 2006.

[11] J.R. Lorch and A.J. Smith, ''PACE: a new approach to dynamic voltage scaling,'' IEEE Trans. Computers, Vol. 53, No. 7, pp. 856-869, 2004.

[12] D. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, 1984.

저 자 소 개

이 관 우

1994: POSTECH
컴퓨터공학 공학사.

1996: POSTECH
컴퓨터공학과 공학석사.

2003: POSTECH
컴퓨터공학과 공학박사

현 재: 한성대학교
정보시스템공학과 부교수

관심분야: 실시간 시스템,소프트웨어공학

Email : kwlee@hansung.ac.kr