논문 2016-11-01

에너지 효율적인 멀티코어 임베디드 시스템을 위한 부하 불균형 스케줄링 방법

(Load Unbalancing Scheduling Method for Energy-Efficient Multi-core Embedded Systems)

최 영 진*
(YoungJin Choi)

Abstract : We proposed a load unbalancing scheduling method for energy-efficient multi-core embedded systems considering DVFS (Dynamic Voltage/Frequency Scaling) power consumption and task characteristics. It is a new kind of scheduler which combines load balancing and load unbalancing technique. The purpose of the method is to effectively utilize energy without much effect in performance. In this paper, we conduct experiments on energy consumption and performance using the previous load balancing and unbalancing techniques and our proposed technique. The proposed technique reduced energy consumption more than 13.7% when compared to other algorithms. As a result, the proposed technique shows low energy consumption without much decline in the performance and is adequate for energy-efficient multi-core embedded systems.

Keywords : Load unbalancing, Load balancing, Real-time scheduler, Multi-core embedded system

Ⅰ. 서 론

1. 연구배경

임베디드 디바이스에서 실행되는 애플리케이션은 사용자 요구에 의해 많은 기능들이 포함되고 있고 복잡도와 코드 크기가 급격히 증가하고 있다 [1]. 배터리를 이용하는 임베디드 디바이스는 성능과 저전력이 매우 중요하다. 이러한 요구사항으로 인해 임베디드 디바이스를 저비용으로 고성능과 저전력을 만족하는 시스템을 개발해야 하는 필요성이 대두되었다. 임베디드 시스템으로 구성되는 장치들 중에서 프로세서가 소비하는 에너지가 많은 부분을 차지한다. 기존 연구에 따르면 전체 시스템에서 18~30%이상을 소비하는 것으로 알려져 있다 [2]. 따라서 프로세서가 차지하는 에너지 소모율을 줄이기 위해 다방면으로 연구되고 있다.

멀티코어 프로세서는 두 개 이상의 코어를 하나의 칩에 탑재하여 싱글코어 프로세서보다 높은 성능과 낮은 전력 소모를 보이는 장점이 있다. 멀티코어 프로세서가 낮은 주파수로 동작할 때 같은 성능의 싱글 코어 프로세서에 비해 소비 전력이 낮아지고 발열량이 줄어들게 된다. 이러한 장점으로 인해서 멀티코어 프로세서는 저전력을 요구하는 모바일 임베디드 시스템뿐만 아니라 서버, 데스크톱, 멀티미디어 시스템에 이르기까지 여러 분야에서 폭넓게 사용되고 있다. 프로세서의 효율적인 전력 관리를 위해 다양한 전력 절감 기술들이 제안되었다. 이러한 기법들 중 대표적인 것이 DPM (Dynamic Power Management)과 DVFS (Dynamic Voltage/Frequency Scaling)이다. DPM과 DVFS 관한 연구는 오래전부터 많은 주목을 받아왔다 [2-8]. DPM은 코어를 동적으로 전력모드를 변환하는 기술이다. 즉, 사용하지 않는 코어의 전원을 차단하고 필요할 때 전원을 공급한다. DVFS는 동적으로 코어의 전압/주파수를 변화시킬 수 있는 기술이고 전압/주파수에 따라서 태스크의 실행시간이 달라진다. 많은 프로세서들이 DPM과 DVFS 기능을 탑재하고 있다. 멀티

* Corresponding Author (youngjin.choi@lignex1.com)
Received: 13 Apr. 2015, Revised: 2 June 2015, Accepted: 15 Sep. 2015.
Y.J. Choi : LIG Nex1
※ 본 논문은 LIG Nex1과 과학기술연합대학원대학교의 학술연구비에서 지원하여 연구하였음.

 2016 Feb. 11(1) 1-8
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2016.11.1.1

코어 플랫폼은 DPM과 DVFS를 적용하는 방식에 따라서 Full-Chip Platform (또는 Global DVFS/DPM)와 Per-Core Platform (또는 Per-Core DVFS/DPM)으로 나눈다 [6, 9]. Full-Chip Platform은 모든 코어를 동일한 주파수와 전력모드로 설정이 가능하다. 모든 코어가 같은 주파수로 동작되기 때문에 전력 관리 측면에서 융통성이 부족하다 [6]. 이러한 특성을 갖는 프로세서는 ARM11 MPCore와 Intel Core 2 Quad 등이 있다. 반면에 Per-Core Platform은 개별적으로 주파수와 전력모드를 설정할 수 있다. 즉, 각 코어가 다른 주파수로 동작이 가능하다. Montecito/Foxton Multi-Core와 AMD Phenom Quad-Core 프로세서가 이러한 특성을 갖는다. 제조과정에서 비용 문제가 있지만[6] 전력관리 측면에서 Per-Core Platform이 Full-Chip Platform 보다 효율적이다.

멀티코어 시스템 환경에서 고성능 또는 저전력을 위해 태스크를 적절한 코어에 할당하는 기법이 있는데, 부하 균형과 부하 불균형 기법 [3, 10]이 존재한다. 최근 멀티코어 기반 임베디드 환경에서 성능향상 또는 낮은 전력 소모를 달성하기 위해 운영체제 레벨에서 부하 균형/불균형 기법들이 중요시되고 있다 [1]. 본 논문에서는 멀티코어 프로세서의 DVFS 전압/주파수 단계별 전력 소모 특성과 태스크 특성을 동시에 고려한 에너지 효율적인 부하 불균형 스케줄링 알고리즘에 대해 연구하였다.

2. 연구의 목적 및 필요성

멀티코어 프로세서 기반 운영체제에서 부하를 분산하는 알고리즘은 부하 균형 기법과 부하 불균형 기법으로, 두 가지로 나눌 수 있다. 부하 균형 기법은 시스템에 주어진 모든 태스크를 모든 코어에 공평하게 할당하는 방법이다. 그러나 기존의 부하 균형 기법에 대해서 전력 소모 측면에서 문제가 제기되었다. 심지어 한 코어에서도 충분히 실행될 수 있는 적은 수의 태스크를 모든 코어에 균등하게 분산 할당하면 불필요한 전력 소모가 발생한다. 부하 불균형 기법은 시스템에 주어진 모든 태스크들을 한 코어에 이용률의 최대 임계값까지 할당하고 그 임계치가 넘으면 다른 코어에 이용률의 최대 임계값까지 할당하는 방법이다. 모든 코어에 이와 같은 방식으로 태스크들이 할당이 된다. 작업이 없는 코어는 DPM을 이용하여 저전력 모드로 전환하여 에너지를 절감한다. 그러나 DPM만 사용할 경우에는 코어에 작업이 있을 때 항상 최대 주파수로 동작하기 때문에 비효율적인 에너지 소모를 보이고 전력 절감을 위해서 추가적으로 DVFS을 적용하여 부하 불균형 기법을 사용할 경우 특정 코어에 최대 임계값까지 태스크가 할당이 되면 높은 주파수로 동작되기 때문에 비효율적인 에너지 소모를 보인다.

본 논문에서는 멀티코어 프로세서의 DVFS의 전압/주파수 단계별 전력 소모 특성과 태스크 특성을 동시에 고려한 에너지 효율적인 부하 불균형 기법을 제안한다. 제안 기술은 성능저하가 크지 않고 효율적인 에너지 소모를 목표로 한다.

Ⅱ. 에너지 효율적인 부하 불균형 스케줄링 방법

1. 시스템 모델

본 논문에서 사용하는 실시간 태스크 스케줄링 모델과 멀티코어 플랫폼 모델은 기존의 관련 연구에서 널리 사용되는 모델을 따른다. 실시간 태스크 스케줄러에서 실행의 기본단위는 하나의 태스크이다. 태스크의 집합 (Task Set)인 Ψ는 아래 수식과 같이 정의 된다. τ 는 각각의 태스크, T는 태스크의 주기 또는 마감시간, C는 최악의 실행시간 (Worst Case Execution Time, WCET)을 의미한다.

$$\Psi = \{\tau_1(T_1, C_1), \tau_2(T_2, C_2), \ldots, \tau_n(T_n, C_n)\} \quad (1)$$

주기 태스크의 마감시간 (Deadline)은 태스크의 주기와 같다고 가정하였고 최악의 실행시간 (WCET) 기반으로 태스크 스케줄링 한다. 따라서 태스크의 다음 주기가 되기 전에 현재 실행하고 있는 태스크가 끝나야 한다. 태스크는 서로 의존성 (Dependency)이 없다고 가정하였고 스케줄링과 선점에 필요한 비용도 없다고 가정하였다. 또한, 분할 스케줄링 방법을 사용하였고 각 코어는 EDF 스케줄링 알고리즘 [11]을 사용하였다. 아래 수식은 EDF 알고리즘이고 이용률이 1이하이면 스케줄링 가능하다. 다시 말해서, 각 코어에 할당된 태스크들의 이용률 합이 1이하이면 모든 태스크가 마감시간 전에 실행을 마치는 것을 보장한다.

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \quad (2)$$

멀티코어 플랫폼은 기존 연구에서 많이 사용한 Intel XScale [12]을 사용하였다 [5, 13-15].

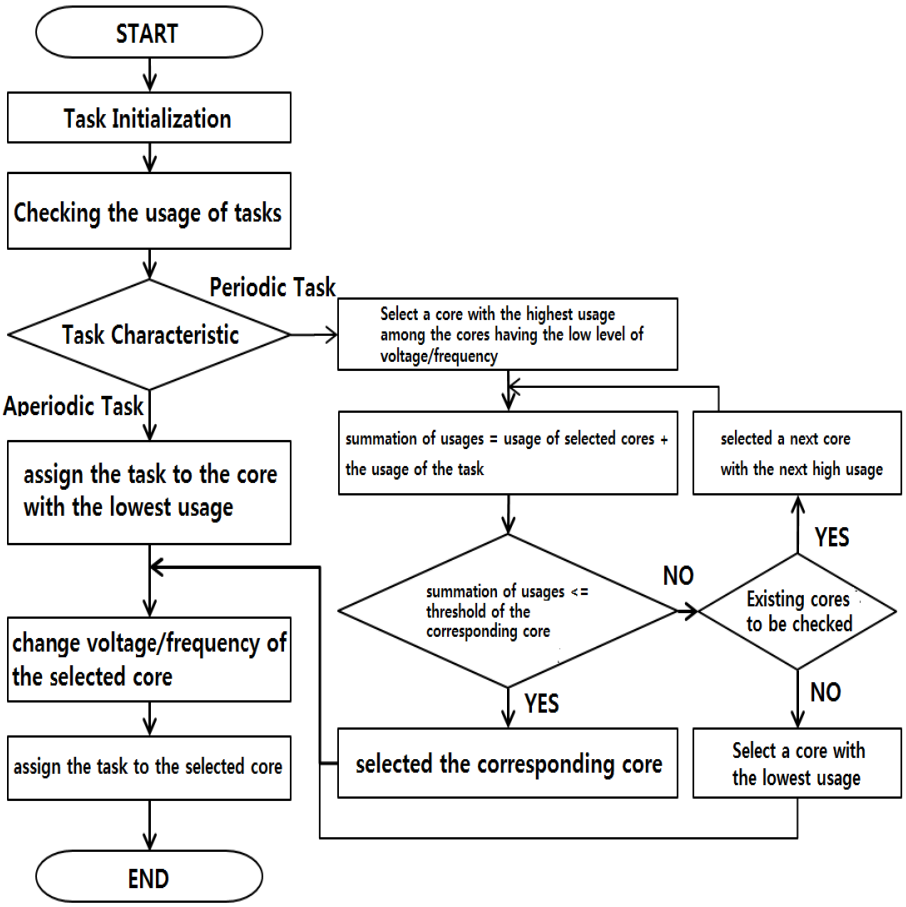

그림 1. 스케줄러 알고리즘 순서도
Fig. 1 Flow Chart of Scheduler Algorithm

표 1. XScale 전력 소비 모델 [5]
Table 1. XScale Power Consumption Model

Frequency (MHz)	150	400	600	800	1000
Voltage (V)	0.75	1.0	1.3	1.6	1.8
Power Consumption (W)	0.08	0.17	0.4	0.9	1.6
Execution Time (e)	3	2	1.5	1.2	1

이 모델은 호모지니어스 (Homogeneous) 멀티 코어 프로세서이다. DVFS와 DPM 기능을 사용할 수 있고, DVFS 기능은 5 단계 전압/주파수 특성을 갖는다. 각 코어는 개별적으로 DVFS와 DPM이 적용 가능하다고 가정하였다.

특정 코어의 부하가 0.33 미만일 때 해당 코어는 150Mhz로 동작하고 0.33 ~ 0.5 에서는 400Mhz, 0.5 ~ 0.66 에서 600Mhz로 동작, 0.66 ~ 0.83에서 800Mhz로 동작, 0.83 ~ 1에서 1000Mhz로 동작한다. 프로세서의 DVFS 특성을 고려하여 임계값이 결정되었다. 본 논문은 임계값을 애플리케이션(또는 태스크)에 따라 다르게 설정하는 연구 [8]와는 달리 태스크와 하드웨어 특성을 고려한 것이다. DVFS 관련 기존연구 [16]에 따르면 공급전압과 태스크 실행시간은 역전관계 (Inverse Relationship) 이다. 즉, 공급전압을 낮추면 태스크의 실행시간이 증가된다. 표 1은 XScale 모델의 전압/주파수별 전압(V), 전력 소모(W)와 늘어난 실행시간 비율(e)을 보여준다. 인스트럭션 실행이 없을 때는 최소 전압/주파수 레벨에서 유휴 (Idle) 상태로

전환된다. 유휴 상태의 전력소모는 0.04 Watt로 가정하였다. 본 논문에서는 변환 오버헤드 (Transition Overhead)는 적은 전력을 소모하므로 고려하지 않았고, DPM 기능을 이용하여 특정코어에 전원을 차단하였을 때 전력소모는 없다고 가정하였다.

2. 부하 불균형 알고리즘

임베디드 멀티코어 환경에서 성능 저하가 크지 않고 효율적인 에너지 소모를 위해 본 논문에서는 DVFS 전압/주파수 단계별 전력소모 특성과 태스크 특성을 고려한 에너지 효율적인 부하 불균형 스케줄러를 제안한다. 이 기법은 태스크를 분산 할당하는 부하 균형 기법과 태스크를 집중 할당하는 부하 불균형 기법을 혼합한 새로운 형태의 스케줄러이다.

태스크 특성을 고려하여 태스크를 코어에 집중 또는 분산 할당하는 것이다. 태스크 종류는 크게 주기 태스크와 비주기 태스크가 있다. 비주기 태스크의 경우는 성능 측정기준 (Metric)이 마감시간을 만족하는 것보다는 짧은 대기시간이다 [3]. 마감시간이 있지만 대기시간을 줄이기 위해 모든 코어에 분산 할당하는 전략을 사용한다. 즉, 모든 코어 중에서 이용률이 가장 낮은 코어에 비주기 태스크를 우선적으로 할당하여 코어의 최대 주파수로 스케줄링한다. 주기 태스크의 경우 중요한 성능 측정기준이 빠른 실행보다는 마감시간을 지키는 것이다. 마감시간을 지킬 수 있다면 긴 대기시간은 문제가 되지 않는다. 그래서 주기 태스크는 특정 코어에 임계값까지 집중 할당하는 전략을 사용한다. 이용률이 가장 높은 코어에 많은 태스크를 집중 할당을 함으로써 나머지 코어들은 저전력 모드로 전환하여 낮은 전력 소모를 기대할 수 있다 [3].

효율적인 에너지 소모를 위해, DVFS 전압/주파수 단계별 전력 소모 특성을 고려하여 주기 태스크를 멀티코어에 할당해야 한다. 표 1과 같이 XScale 모델은 각 코어를 5 단계 전압/주파수로 조정 가능하다. 각 전압/주파수 단계는 태스크 마감시간을 지키면서 스케줄링이 가능한 이용률의 임계값이 있다.

그림 1은 전력 소모 특성과 태스크 특성을 고려한 부하 불균형 스케줄러에 대한 순서도이다. 새로운 태스크가 도착하면 태스크 이용률을 계산하고 태스크 종류를 분류한다. 주기태스크의 이용률은 실행시간÷주기로 계산이 되고, 비주기 태스크의 이용률은 실행시간÷마감시간으로 계산된다. 비주기 태스크가 특정코어에 할당되면 태스크 이용률이 코어 이용률에 합산하고, 실행이 끝나면 비주기 태스크 이용률이 코어이용률에서 차감된다.

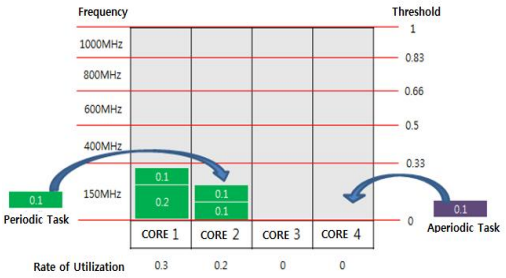

그림 2. 태스크 할당 전
Fig. 2 Before Assigning Tasks

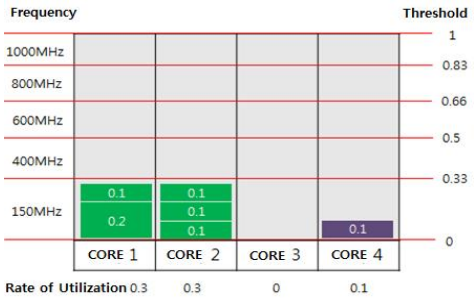

그림 3. 태스크 할당 결과
Fig. 3 Result of Assigned Tasks

비주기 태스크와 주기 태스크는 다른 알고리즘을 적용한다. 비주기 태스크가 도착하면 이용률이 가장 낮은 코어를 선택 후, 해당 코어의 전압/주파수를 조정한다. 마지막으로 비주기 태스크가 선택된 코어에 할당된다. 반면에 주기 태스크가 도착하면 전압/주파수 레벨이 낮은 코어들 중에서 이용률이 큰 코어를 선택한 후, 선택된 코어의 이용률과 할당 예정인 주기 태스크의 이용률을 더한다. 그 결과 값이 선택된 코어의 현재 전압/주파수 레벨의 임계값 미만이면 선택된 코어에 바로 주기 태스크를 할당한다. 그러나 그 결과 값이 선택된 코어의 현재 전압/주파수 레벨의 임계값을 넘으면 동작 중인 코어들 중에서 다음으로 이용률이 큰 코어를 선택한 후, 선택된 코어의 이용률과 할당 예정인 주기 태스크의 이용률을 합하고 그 결과 값이 선택된 코어의 현재 전압/주파수 레벨에서의 임계값이 미만일 때까지 반복하여 최종 선택된 코어에 주기 태스크를 할당한다. 모든 코어를 조사를 해본 후에, 각 코어의 이용률과 할당 예정인 주기 태스크 이용률의 합들이 모든 코어의 현재 전압/주파수 단계에서의 임계값 이상일 경우에는 이용률이 가장 낮은 코어를

선택되고 그 코어의 동작 전압/주파수를 상향 조정하여 주기 태스크를 할당한다.

그림 2는 주파수별 전력 소모 특성과 태스크 특성을 고려하여 각 코어에 태스크를 할당하는 방법을 보여준다. 주기 태스크는 마감시간을 만족할 수 있고 동작 주파수가 낮은 코어들 중 이용률이 가장 높은 코어에 할당을 한다. 비주기 태스크는 이용률이 가장 낮은 코어에 우선적으로 할당한다.

그림 3은 알고리즘을 적용한 결과이다. 코어에 태스크가 없으면 DPM을 이용하여 코어의 전원을 차단한다. 다시 설명하면, 코어 1에서 150MHz 동작할 때 주기 태스크 이용률이 0.33까지만 집중 할당한다. 이후 새로운 태스크가 도착하면 150MHz로 동작하는 코어 2에 할당한다. 코어 2에서도 태스크 이용률이 0.33까지만 집중 할당을 한다. 코어 2의 이용률이 0.33 도달했고 다른 태스크가 들어오면 코어 3에 할당한다. 코어 4까지 이용률 0.33까지만 집중 할당을 한다. 모든 코어가 150MHz로 동작을 하고 있다. 또 새로운 태스크가 도착을 하면 특정 코어를 주파수를 높여야 한다. 주파수를 높이면 이용률의 임계값이 높아진다. 특정 코어가 400MHz로 동작해야 하고 코어의 이용률이 0.5까지 태스크를 집중 할당을 한다. 이와 같은 방식으로 반복된다. 정리를 해보면, 멀티코어 프로세서에는 각 주파수 단계의 임계값들이 있는데, 각 주파수 단계별로 집중 할당과 분산 할당하는 방식이다. 비주기 태스크는 이용률이 가장 낮은 코어에 우선적으로 할당을 한다. 이와 같은 전략을 사용함으로써 더욱 낮은 전력 소모와 비주기 태스크는 짧은 대기 시간을 기대할 수 있다.

Ⅲ. 실험 및 평가

1. 실험환경

시뮬레이션을 통해서 에너지 효율적인 멀티코어 임베디드 시스템을 위한 부하 불균형 스케줄러의 성능을 측정하였다. 에너지 소모, 프로세서 이용률, 비주기 태스크의 평균 대기시간에 대한 성능 평가를 위해서 기존 연구인 태스크 특성에 따른 부하 불균형 전략[3]과 비교하였다. 멀티코어 실시간 스케줄러 시뮬레이터에서 각 코어의 스케줄링 방법은 EDF을 사용하였고 멀티코어 스케줄링 방법은 분할 스케줄러 (Partitioned Scheduler)을 사용하였다.

본 논문에서는 옥타코어 (8코어) 프로세서를 사용하였고 실험시간은 12초로 설정하였다.

표 2. 비주기 태스크 셋
Table 2. Aperiodic Task Set

Number (Aperiodic Task)	Release Time	Execution Time
1	6.51s	0.43s
2	6.51s	0.11s
3	5.41s	0.31s
4	5.37s	0.13s
5	6.25s	0.14s
6	6.59s	0.20s
7	5.76s	0.36s
8	5.85s	0.11s
9	7.64s	0.18s
10	6.30s	0.11s
11	6.21s	0.11s
12	4.37s	0.18s
13	3.86s	0.14s
14	6.74s	0.47s
15	5.78s	0.62s
16	4.40s	0.18s
17	4.90s	0.11s
18	6.43s	0.10s
19	6.03s	0.15s
20	5.91s	0.16s

2. 실험결과 및 분석

본 논문에서 제안하는 에너지 효율적인 멀티코어 임베디드 시스템을 위한 부하 불균형 스케줄링 기법을 기존 연구인 부하 불균형 기법 [4], 기존 기술인 부하 균형 기법과 성능을 비교하였다.

옥타 코어에서 주기 태스크와 비주기 태스크를 구성하여 12초 동안 실험을 하였다. 주기 태스크의 실행시간은 100ms, 주기는 1000ms, 릴리즈 시간은 0s 이다. 0.1 이용률을 갖는 주기 태스크를 3~20개 사용하였다. 비주기 태스크는 다른 릴리즈 시간과 실행시간을 갖는 20개를 사용하였다.

$$f(x) = \frac{ak^{\alpha}}{1-(k/p)^{\alpha}} x^{-\alpha-1} \quad k \le x \le p \qquad (3)$$

비주기 태스크의 실행시간은 Bounded Pareto 분포[4, 17]을 사용하여 100~1000ms 범위에서 생성을 하였다. Bounded Pareto 분포에서 최소값(k)은 100, 최대값(p)은 1000, 변화성 (Variability, α)은 1.1로 설정하였다. 표 2는 Bounded Pareto분포를 이용한 테스트 셋의 한 예이다. Bounded Pareto 분포를 사용한 이유는 실시간시스템에서 비슷한 실행시간을 갖는 비주기 태스크의 수가 많을 것이라 가정하였다.

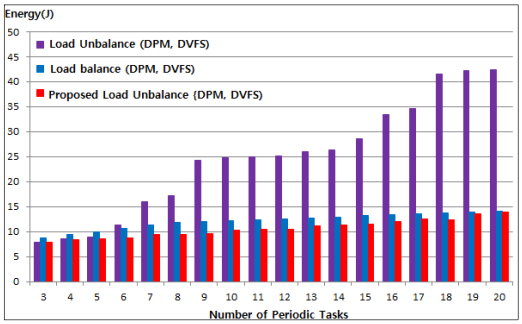

그림 4. 주기/비주기 태스크 실험의 에너지 소모
Fig. 4 Energy Consumption of Experiment including Periodic/Aperiodic tasks

표 3. 에너지 절감률 비교
Table 3. Comparison of Energy Saving-rate

Algorithm	Energy Saving-rate
Load Balance (DPM, DVFS)	13.7%
Load Unbalance (DPM, DVFS)	126.8%

또한, 릴리즈 시간은 Normal 분포 [18]를 사용하였고 평균값(m)은 6, 표준편차(σ)를 1.5로 설정하였다. 생성된 비주기 태스크들의 마감시간은 1000ms로 설정하였다. 비주기 태스크가 특정시점에 집중적으로 할당되는 것을 실험하기 위해서 평균값(m)과 표준편차(σ) 값을 위와 같이 설정하였다.

예를 들면, 실시간시스템이 가동 중일 때 주기 태스크는 주기적으로 실행 중일 것이고, 특정 이벤트가 들어오면 그 시점에 집중적으로 비주기 태스크들이 실행되는 환경을 가정하였다.

그림 4는 주기 태스크와 비주기 태스크 실험에서 에너지 소모 결과이다. 0.1 이용률을 갖는 주기 태스크를 3~20개 사용하였고, 하나씩 늘려가면서 실험을 하였다. 제안한 부하 불균형 기법이 항상 낮은 에너지 소모를 보이고 있다.

표 3은 제안한 알고리즘을 다른 알고리즘들과 비교했을 때 에너지 절감 비율을 나타낸 것이다. 실험 결과에서 제안한 알고리즘을 부하 균형 기법 (DPM, DVFS)과 비교해볼 때 13.7%, 부하 불균형 기법 (DPM, DVFS)과 비교해 볼 때 126.8% 에너지 절감하였다.

그림 5는 주기 태스크와 비주기 태스크 실험에서 비주기 태스크의 평균 대기시간 결과이다. 실험에서 사용된 모든 알고리즘은 비주기 태스크를

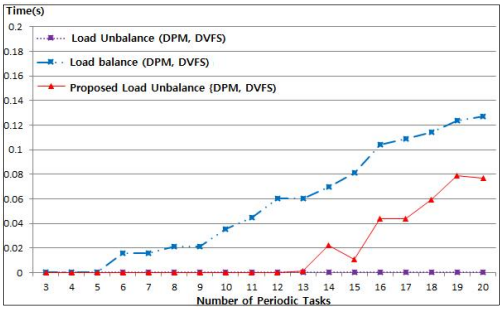

그림 5. 비주기 태스크의 평균 대기시간
Fig. 5 Mean Waiting Time of Aperiodic Tasks

이용률이 가장 낮은 코어에 할당된다. 주기 태스크를 하나씩 늘려가면서 실험을 하여 평균 대기시간을 측정하였다. 이전 실험에서는 제안한 알고리즘이 에너지 소모측면에서 낮은 전력소모를 보이고 있지만 비주기 태스크의 평균 대기시간은 증가하였다.

실험에서 가장 짧은 평균 대기시간을 보여준 부하 불균형 기법 (DPM, DVFS)이다. 부하 불균형 기법이 좋은 성능을 보인다. 이것은 특정 코어에 주기 태스크를 집중할당을 하고 나머지 코어에 비주기 태스크를 분산할당하기 때문에 짧은 대기시간을 얻을 수 있지만 에너지 소모가 많다. 부하 균형 기법 (DPM, DVFS)는 긴 대기시간을 보이고 있다. 이것은 DVFS를 사용하고 있고 주기 태스크들과 비주기 태스크들이 모든 코어에 분산할당 되어서 비주기 태스크가 주기 태스크 때문에 바로 실행될 수 없는 상황이 발생한다.

제안한 부하 불균형 기법 (DPM, DVFS) 경우는 주기 태스크 수가 3~12개에서 짧은 대기 시간을 보였지만 이 후부터 비교적 긴 대기시간을 갖는다. 주기 태스크 14개부터 평균대기시간이 증가하고 있다. 그림 4와 그림 5의 결과를 종합해보면, 멀티코어 시스템의 부하가 크지 않을 때는 성능저하가 크지 않으면서 낮은 전력 소모를 보이고 있다.

그림 6은 각 스케줄링 알고리즘들의 멀티코어 프로세서 이용률 결과이다. 이용률이 높다는 것은 멀티 코어의 유휴 (idle) 시간이 많지 않다는 것을 의미한다. 제안한 부하 불균형 기법와 기존 기술인 부하 균형 기법 (DPM, DVFS)는 높은 프로세서 이용률을 보이고 있다. 이 두 개의 알고리즘을 비교해 볼 때, 실험에서 제안한 부하 불균형 기법이 최대 2.6% 더 높은 이용률을 보이고 있다. 프로세서 이용률 측면에서 부하 불균형 기법 (DPM, DVFS)는 좋은 결과를 기대할 수 없다.

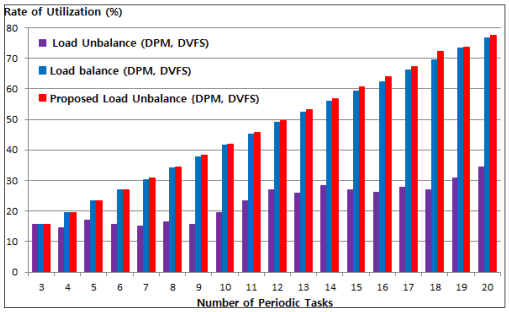

그림 6. 멀티코어 프로세서 이용률

Fig. 6 Utilization Rate of Multi-Core Processor

실험을 통해 성능, 에너지 소모 그리고 프로세서 이용률의 결과를 종합적으로 고려해보면, 본 논문에서 제안한 기술인 태스크특성과 전력소모특성을 고려한 에너지 효율적인 부하 불균형 스케줄링 알고리즘이 멀티코어 시스템의 부하가 비교적 작을 때는 성능저하가 크지 않고 저전력 소모를 보인다.

Ⅳ. 결 론

본 논문에서는 멀티코어 프로세서의 DVFS 전압/주파수 단계별 전력 소모 특성과 태스크의 특성을 동시에 고려한 부하 불균형 기법을 제안하였다.

실험을 통해서 기존 알고리즘들과 제안 기술의 전력소모, 성능, 이용률을 비교하였다. 전력 소모 측면에서 기존 알고리즘인 부하 균형 기법과 부하 불균형 기법과 비교했을 때, 제안 기술이 약 13%이상 에너지를 절감했다. 그리고 비주기 태스크의 평균 대기시간을 비교해 보면, 시스템에 부하가 작을 때는 평균 대기시간이 차이가 없었지만 부하가 커졌을 때 평균 대기시간이 다소 길어졌다. 또한 실험에서 제안 기술이 대부분의 경우 높은 프로세서 이용률을 보이고 있다.

결과를 종합해보면, 본 논문에서 제안한 에너지 효율적인 멀티코어 임베디드 시스템을 위한 부하 불균형 스케줄링 방법이 멀티코어 시스템의 부하가 크지 않을 때는 성능저하가 크지 않으면서 낮은 전력 소모를 보이는 장점이 있다. 비교적 부하가 크지 않은 임베디드 시스템 환경에서 큰 성능저하 없이 저전력 멀티코어에 적합한 스케줄링 알고리즘이다.

References

[1] G.S. Lim, C.W. Min, Y.G. Eom, “Load-balancing for improving user responsiveness on multicore embedded systems,” Proceedings of Linux Symposium, pp. 25-33, 2012.

[2] H. Aydin, R. Melhem, D. Mosse, P. Mejia-Alvarez, “Power-aware scheduling for periodic real-time tasks,” IEEE Transactions on Computers, Vol. 53, No. 5, pp. 584-600, 2004.

[3] H.R. Jeon, W.H. Lee, S.W. Chung, “Load unbalancing strategy for multicore embedded processors,” IEEE Transactions on Computers, Vol. 59, No. 10, pp. 1434-1440, 2010.

[4] C.H. Lee, K.G. Shin, “On-line dynamic voltage scaling for hard real-time systems using the EDF algorithm,” Proceedings of 25th IEEE International Real-Time Systems Symposium, pp. 319-327, 2004.

[5] W.Y. Shieh, C.C. Pong, “Energy and transition-aware runtime task scheduling for multicore processor,” Journal of Parallel and Distributed Computing, Vol. 73, No. 9, pp. 1225-1238, 2013.

[6] D. He, W. Mueller, “A heuristic energy-aware approach for hard real-time systems on multi-core platforms,” Microprocessors and Microsystems, Vol. 37, No. 8, pp. 858-870, 2013.

[7] Y.S. Hwang, K.S. Chung, “Dynamic power management technique for multicore based embedded mobile devices,” IEEE Transactions on Industrial Informatics, Vol. 9, No. 3, pp. 1601-1612, 2013.

[8] J.M. Kim, M. Kim, S.W. Chung, “ Application-aware scaling governor for wearable devices,” Proceedings of IEEE The 24th International Workshop on Power And Timing Modeling, Optimization and Simulation, pp. 1-8, 2014.

[9] D.S. Zhang, F. Chen, S. Jin, “Global EDF-based online, energy-efficient real-time scheduling in multi-core platform,” Proceedings of IEEE International Conference on Computer Science and Automation

Engineering, pp. 666-670, 2011.
[10] V. Srinivasan, G.R. Shenoy, S. Vaddagiri, D. Sarma, V. Pallipadi, "Energy-aware task and interrupt management in linux," Proceedings of Linux Symposium, Vol. 2, pp. 187-198, 2008.
[11] C.L. Liu, "Scheduling algorithms for multiprogramming in a hard real-time environment," Journal of the ACM, Vol. 20, No. 1, pp. 46-61, 1973.
[12] XScale, http://ko.wikipedia.org/wiki/XScale
[13] W.Y. Lee, "Energy-saving DVFS scheduling of multiple periodic real-time tasks on multi-core processors," Proceedings of 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, pp. 216-223, 2009.
[14] G. Chen, K. Huang, J. Huang, C. Buckl, A. Knoll, "Effective online power management with adaptive interplay of DVS and DPM for embedded real-time system," Proceedings of IEEE Euromicro Conference on Digital System Design, pp. 881-889, 2013.
[15] B. Xu, C. Xi, R. Melhem, D. Mosse, "Practical PACE for embedded systems," Proceedings of 4th ACM International Conference on Embedded Software, pp. 54-63, 2004.
[16] Y. Ding, W. Zhang, "Multicore real-time scheduling to reduce inter-thread cache interferences," Journal of Computing Science and Engineering", Vol. 7, No. 1, pp. 67-80, 2013.
[17] Bounded Pareto Distribution, https://en.wikipedia.org/wiki/Pareto_distribution
[18] Normal Distribution, https://en.wikipedia.org/wiki/Normal_distribution

YoungJin Choi (최 영 진)

He received his M.S degree in Computer Software and Engineering from University of Science and Technology (UST), Daejeon Korea, in 2014. He is currently working as a researcher for LIG Nex1. His current research interests includes Real-Time System, Embedded Software, Physical Computing Platforms and Internet of Things

Email : youngjin.choi@lignex1.com
youngjin.choi@outlook.com