임베디드 시스템에서 CPU 선호도를 고려한 Pfair 실시간 멀티코어 스케줄러의 오버헤드 분석

이정인, 박상수
이화여자대학교 컴퓨터공학과
e-mail: jungin.lee@ewhain.net, sangsoo.park@ewha.ac.kr

An Overhead Analysis of Pfair Real-Time Multi-Core Scheduler with CPU Affinity on Embedded Systems

Jung-in Lee, Sangsoo Park
Dept of Computer Science & Engineering, Ewha Womans University

요 약

낮은 오버헤드를 갖는 실시간 스케줄링 알고리즘은 멀티코어 프로세서가 임베디드 시스템에서 사용되기 위한 가장 중요한 요소 중의 하나이다. 멀티코어 환경에서 스케줄링 오버헤드는 주로 메모리 성능을 저해시키는 코어간 태스크 이동에 의해 발생한다. 본 논문에서는 시스템 이용률 면에서 최적으로 알려진 Pfair 스케줄링 알고리즘을 스케줄링 시에 태스크의 CPU 코어 할당 방식에 대해 스케줄링 오버헤드를 측정하였다. 실험 결과 동일 코어 기반 태스크 할당 방식을 도입함으로 인해서 태스크 이동 횟수를 크게 줄일 수 있음을 보여주었다.

1. 서론

임베디드 시스템을 위한 응용 프로그램의 크기와 복잡도가 점점 증가하는 동시에 이를 저비용, 고성능을 만족하고 결함 허용성을 지원하도록 개발해야 하는 필요성이 대두됨에 따라 최근 임베디드 시스템에서 멀티코어 기반의 하드웨어 플랫폼이 크게 각광 받고 있다. 이러한 임베디드 시스템은 목표로하는 성능, 전력 소비, 제조 비용, 제품 출하시기 등 응용 프로그램에 종속적인 여러 제약점들을 만족해야 하는데, 예를 들어, 자동차 분야에서는 소비자에 의한 각종 편의 장비의 요구와 정부 규제에 따른 연비 강화 및 안전장치 추가 요구에 따라 자동차에 탑재되는 임베디드 시스템의 크기와 복잡도가 동시에 크게 증가하고 있다 [1,2].

멀티코어 프로세서는 두개 이상의 코어를 하나의 칩에 탑재하여 싱글코어 프로세서에 비해 높은 성능을 보다 낮은 전력으로 제공할 수 있다. 즉, 적절히 설계된 듀얼코어 프로세서는 같은 동작 주파수에 수행되는 싱글코어 프로세서에 비해 약 2 배의 성능을 제공할 수 있다. 반면 멀티코어 프로세서가 낮은 주파수에서 동작함에 따라서 같은 성능의 싱글코어 프로세서에 비해 소비 전력이 낮고 이에 따라 칩의 발열량도 줄어들게 된다. 이러한 장점들로 멀티코어 프로세서는 점점 임베디드 시스템을 위한 프로세서로 자리 잡고 있다.

많은 임베디드 시스템을 위한 응용 프로그램은 각각의 기능이 주어진 시간 내에 완료되어야하는 시간 제약성을 갖고 있다. 이러한 시간 제약성을 제공하기 위해서는 각각의 기능을 수행하는 응용 프로그램의 태스크를 위한 실시간 스케줄러가 필요하다. 싱글코어 프로세서 기반의 실시간 스케줄링 알고리즘은 지난 수년간 광범위 하게 연구되어 많은 수의 최적화된 알고리즘이 이미 개발되었으며 최근 이러한 싱글코어 기반의 실시간 스케줄링 알고리즘을 멀티코어 프로세서에 적용하려는 여러 노력이 있어 왔지만 아직 뚜렷한 성과를 내고 있지 못하고 있다 [3].

Pfair (Proportionate-fair) 스케줄링 알고리즘 [4]은 실제 구현이 가능하고 시스템 이용률 면에서 최적인 유일한 멀티코어를 위한 실시간 스케줄링 알고리즘으로 알려져 있다. Pfair 스케줄링 알고리즘은 시스템 수행 중에 각 태스크가 한 코어에서 다른 코어로 이동할 수 있게 하여 기존의 스케줄링 방식에 비해 높은 시스템 이용률에도 모든 태스크의 시간 제약성을 만족할 수 있다 [5]. 임베디드 시스템은 상대적으로 엄중한 자원 제약을 갖고 있기 때문에 이러한 Pfair 스케줄링 알고리즘이 실제 임베디드 시스템에 적용되기 위해서는 낮은 스케줄링 오버헤드로 구현이 가능해야 한다. 태스크가 빈번히 다른 코어로 이동할 경우 캐시와 TLB (Translation Lookaside Buffer) 등의 메모리 성능을 저해하여 실제 시스템에 적용할 때 Pfair 스케줄링 알고리즘의 유용성 낮출 수 있다 [6]. 본 논문에서는 위와 같은 태스크 수행 중 수행되는 코어의 이동에 따른 오버헤드를 분석하고, 이전에 태스크가 수행되었던 코어를 선호하여 수행이 지속되도록 하는 CPU 선호도를 고려한 스케줄링 방식의 오버헤드와 비교해 본다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문

의 분석 대상인 Pfair 실시간 멀티코어 스케줄러의 동작 원리를 설명하고 CPU 선호도에 따른 알고리즘 적용 시에 스케줄러의 동작을 기술한다. 3 장에서는 시뮬레이터를 통해 측정한 Pfair 스케줄러의 오버헤드를 제시하며 4 장에서는 본 논문의 결론을 내린다.

2. Pfair 스케줄링 알고리즘과 CPU 코어 할당 방식에 따른 동작 차이

Pfair 스케줄링 알고리즘은 M 개의 동일한 CPU 코어를 갖고 N 개의 주기적으로 수행되는 실시간 태스크를 갖는다고 가정한다. 스케줄러는 시스템 전역의 주기적인 타이머에 동기화 되어 CPU 의 수행시간을 $[t, t+1)$ 의 범위를 갖는 시간 유닛 t 단위로 나뉘어 각 태스크를 CPU 에 할당하는 알고리즘을 수행한다.

시스템의 각 태스크 T 는 시간 유닛 t 단위의 혹은 시간 유닛 t 의 배수의 태스크 주기 T_p 와 수행시간 T_e 로 정의되어, 매 주기 T_p 마다, 수행시간 T_e 를 갖는 태스크 T 의 새로운 인스턴스가 생성되게 된다. 태스크의 시간 제약성을 만족하기 위해서는 릴리즈(Release)된 태스크의 인스턴스는 반드시 제한시간(Deadline) 혹은 태스크의 주기인 T_p 내에 수행이 완료되어야 한다.

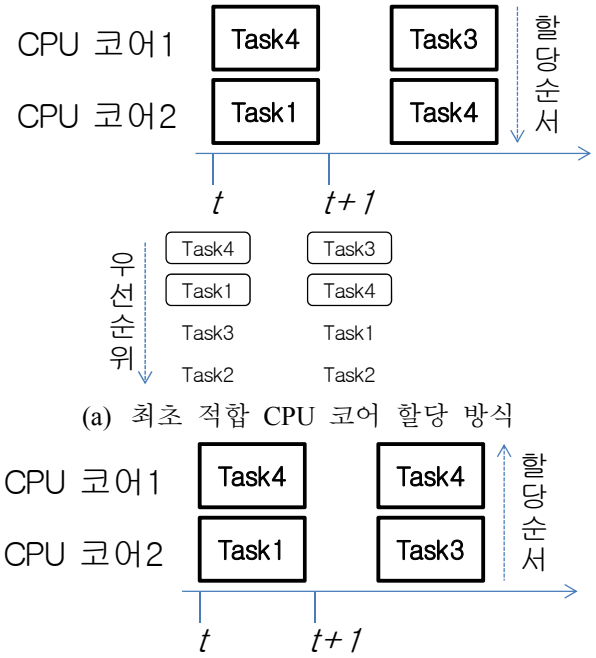
Pfair 스케줄링 알고리즘의 동작 원리는 위의 정의를 기반으로 매 스케줄링 지점 혹은 매 시간 유닛 t 마다 시간 제약성을 만족하도록 코어의 수인 M 개의 태스크를 선택하는 문제로 정의될 수 있다. 이때, Pfair 스케줄링 알고리즘의 기본 아이디어는 시간이 흘러감에 따라 각 태스크의 시간 제약을 만족하기 위해 반드시 해당 태스크의 비중만큼 수행되어야 함을 의미하며 이러한 비율은 [7] 에서 Pfairness 라고 정의되었다. 즉, 임의의 시간 t 에 모든 태스크가 Pfairness 를 만족하면 전체 시스템은 시간 제약성을 만족한다고 정의된다.

이때 선택된 M 개의 태스크를 M 개의 CPU 코어에서 수행되도록 할당할 때 선택된 서브 태스크들은 어떠한 코어에서 수행되어도 여전히 시간 제약성을 만족할 수 있다. 그러나 다음과 같이 할당하는 방식에 따라 Pfair 스케줄링 알고리즘의 오버헤드가 크게 달라질 수 있다.

① 최초 적합 CPU 코어 할당 방식: Pfair 스케줄러를 구현하기 위한 가장 단순한 방법으로 태스크를 순차적으로 최초로 사용 가능한 코어에 할당하는 최초 적합 방식이다. 구현이 매우 단순하고 각 CPU 코어에서 이전 스케줄링 알고리즘에 대한 선택을 유지할 필요가 없다.

② 동일 CPU 코어 선호 할당 방식: Pfair 스케줄러가 태스크를 스케줄링 할 때 한 태스크가 이전에 할당되었던 코어에 먼저 할당을 시도하는 할당 방식이다. 이의 구현을 위해서는 각 코어에서 이전 스케줄링 알고리즘에 대한 선택을 유지해야 하며 Pfair 스케줄러가 시스템 전역의 상태를 확인해야하는 추가적인 과정이 필요하다.

예를 들어, (그림 1)과 같이 2 개의 CPU 코어를 탑재한 ($M=2$) 시스템의 임의의 시점 t 에 4 개의 태스크 Task1-4 의 태스크 중에서 가장 높은 우선순위를 갖는 태스크가 Task4, 다음 높은 우선순위를 갖는 태스크가 Task1 이고 각각 CPU 코어 1, 2 에 할당이 되었다고 하자.



(a) 최초 적합 CPU 코어 할당 방식

(b) 동일 CPU 코어 선호 할당 방식

(그림 1) CPU 코어 할당 방식에 따른 Pfair 스케줄링 알고리즘의 동작 예제

이때 (그림 1a)의 최초 적합 CPU 코어 할당 방식에서는 t+1 시점에 가장 높은 우선순위를 갖는 태스크가 Task3, 다음 높은 우선순위를 갖는 태스크가 Task4 일 경우 가장 높은 우선순위를 갖는 태스크인 Task3 은 가장 먼저 가용한 CPU 코어인 코어 1 에 할당이 되고 Task 4 는 남는 CPU 코어인 코어 2 에 할당되게 된다.

즉, Task4 는 시점 t 와 t+1 에 모두 수행이 되지만 스케줄러에 의해서 CPU 코어 1 에서 코어 2 로 이동하게 되어 수행되게 되는 것이다.

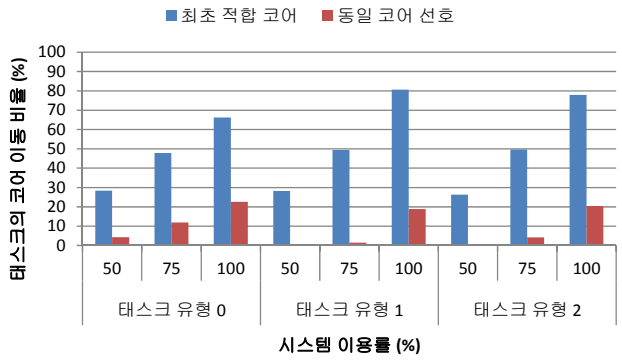
반면, (그림 1b)의 동일 CPU 코어 선호 할당방식에서는 t+1 시점의 스케줄링을 수행할 때 먼저 가장 높은 우선순위를 갖는 M 개의 태스크 중에서 t 시점에 스케줄링 되었던 태스크가 동일한 CPU 코어에서 수행되도록 할당하며 남은 태스크들을 순서대로 가용한 태스크에 할당되도록 한다. 즉, Task4 는 시점 t 와 t+1 에 모두 CPU 코어 1 에서 수행되고 CPU 코어 2 는 시점 t 에서는 Task1 을 시점 t+2 에서는 Task3 을 수행한다.

따라서, (그림 1)의 예제에서는 동일 CPU 코어 선호 할당 방식을 사용함으로 인해서 Task4 의 태스크 이동 횟수를 1 회 줄일 수 있어 이에 따른 스케줄링 오버헤드를 감소시킬 수 있다.

3. 실험 결과

본 논문에서는 성능 평가를 위해서 Pfair 스케줄링 알고리즘을 이산 시간 기반 시뮬레이터로 구현하였다. 각각의 CPU 코어 할당 방식에 대해 스케줄링 오버헤드를 태스크 이동 횟수로 비교하였으며 실험 결과는 (그림 2)와 같다.

실험에서 사용된 태스크 집합은 난수 발생기를 사용하여 무작위로 생성하였다. 실험에서 태스크 집합은 크게 3 종류로 분류되며 태스크 집합 유형 0 은 낮은 비중을 갖는 태스크 만으로 구성되고, 태스크 집합 유형 1 은 높은 비중을 갖는 태스크 만으로 구성되며, 마지막으로 태스크 유형 2 는 낮은 비중의 태스크와 높은 비중의 태스크를 혼합한 태스크 집합으로 구성된다. 태스크 유형 0 은 태스크의 비중이 [0.1, 0.5) 범위에, 태스크 유형 1 은 [0.5, 0.9) 범위에 균일하게 분포하도록 하고, 태스크 유형 2 는 낮은 비중의 태스크와 높은 비중의 태스크가 0.2 대 0.8 의 비율로 분포하도록 하였다. 실험에서는 태스크 집합의 전체 시스템 이용률이 50% 에서 100% 사이가 되도록 유형별로 100 개의 서로 다른 태스크 집합을 GSL 을 이용하여 생성하였다.



(그림 2) 실험 결과

(그림 2)의 실험 결과를 보면 태스크의 시스템 이용률이 증가됨에 따라 스케줄링 시에 태스크가 다른 코어로 이동하게 되는 비율이 증가함을 알 수 있다. 특히 최초 적합 코어 할당 방식의 경우 시스템의 이용률이 75% 이상인 경우 태스크 이동 비율이 태스크의 유형에 상관 없이 50%에 달하여 임의의 시점에 2 번 중의 한번은 태스크가 이전에 수행되던 CPU 코어와 다른 코어에서 수행됨을 알 수 있다. 즉, 최초 적합 코어 할당 방식의 경우 임베디드 시스템 적용될 시 수행 시간 중에 매우 높은 수준의 스케줄링 오버헤드가 소요되어 그 효율성이 떨어지게 될 것으로 예상된다. 반면 동일 코어 선호 방식의 경우 최초 적합 할당 방식에 비해 태스크의 코어 이동 비율이 획기적으로 감소된 것을 알 수 있다. 태스크 유형에 따라 다르기는 하지만 시스템 이용율 75% 이하에서는 태스크의 코어 이동 비율이 10% 이하이며 시스템 이용률이 100%인 경우에도 태스크 유형에 관계 없이 약 20% 의 수준을 유지하는 것을 알 수 있다.

4. 결론

본 논문에서는 멀티코어 프로세서 기반의 하드웨어 플랫폼에서 최적으로 알려져 있는 실시간 스케줄링 알고리즘인 Pfair 스케줄링 알고리즘을 전체 시스템 성능에 큰 영향을 미칠 수 있는 태스크 이동 횟수라는 면에서 스케줄링 오버헤드를 분석하였다.

실험 결과는 Pfair 스케줄러의 스케줄링 시마다 각 태스크가 이전에 수행되었던 상태를 기억하고 이를 다음 스케줄링 시에 활용하는 추가적인 구현으로 스케줄링 시의 태스크 이동 횟수를 크게 감소시킬 수 있음을 확인하였다. 시스템 오버헤드에 민감한 임베디드 시스템을 위해 이와 같은 스케줄링 오버헤드를 분석함으로써 본 논문의 결과는 향후 Pfair 알고리즘이 실제 적용될 때 도움이 될 수 있을 것으로 예상된다.

Acknowledgement

본 논문은 한국연구재단의 기초연구사업 지원(과제번호: 2011-0013422), 세계수준의 연구중심대학육성사업(과제번호: R33-10085)의 지원을 받아 수행된 것임.

참고문헌

[1] W. Wolf, Computers as Components: Principles of Embedded Computing System Design. Morgan Kaufmann, 2001.

[2] P. Leteinturier, "Multi-core processors: driving the evolution of automotive electronics architectures (http://embedded.com/)," 2007.

[3] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, A Categorization of Real-time Multiprocessor Scheduling Problems and Algorithms in Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Chapman and Hall/CRC, 2004.

[4] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: a notion of fairness in resource allocation," Algorithmica, vol. 15, no. 6, pp. 600-625, 1996.

[5] P. Holman and J. H. Anderson, "Using supertasks to improve processor utilization in multiprocessor real-time systems," Proceedings of Euromicro Conference on Real-Time Systems, 2003, pp. 41-50.

[6] K. Hirata and J. Goodacre, "ARM MPCore; the streamlined and scalable ARM11 processor core," Proceedings of Asia and South Pacific Design Automation Conference, 2007, pp. 747-748.

[7] J. H. Anderson and A. Srinivasan, "Mixed Pfair/ERfair scheduling of asynchronous periodic tasks," *Proceedings of Euromicro Conference on Real-Time Systems*, 2001, pp. 76 - 85.