높은 우선순위의 비주기적 태스크 그룹을 위한 리눅스 스케줄러 확장

김영승*, 조현철*, 진현욱*, 이상일**
*건국대학교 컴퓨터공학부
**국방과학연구소
e-mail: *{truth50, gamja474, jinh}@konkuk.ac.kr, **happyjoy@add.re.kr

An Extension of Linux Scheduler for High-Priority Aperiodic Task Groups

Young-Seung Kim*, Hyun-Chul Jo*, Hyun-Wook Jin*, Sang-Il Lee**
*Dept of Computer Science and Engineering, Konkuk University
**Agency for Defense Development

요 약

임베디드 소프트웨어는 국방, 항공 우주, 자동차와 같이 다양한 응용분야에서 그 중요성이 부각되고 있다. 이와 함께 시스템 자원의 효율성을 높이고 응용 소프트웨어 간 안전한 실행환경을 제공하기 위해서 자원 파티셔닝의 필요성이 강조되고 있다. 최근 임베디드 시스템의 응용 분야가 다양해지면서 주기적인 파티션과 함께 비주기적인 파티션들에 대한 요구가 증가하고 있다. 하지만 기존 시스템들은 비주기적인 파티션은 고려하고 있지 않거나, 주기적인 파티션에 비해서 우선순위가 낮게 취급하고 있다. 이러한 문제를 해결하기 위해서 본 논문은 높은 우선순위의 비주기적인 태스크 그룹을 지원하기 위한 구조를 제안하고, 리눅스의 cgroup 프레임워크를 확장하여 구현한다.

1. 서론

임베디드 소프트웨어는 국방, 항공 우주, 자동차와 같이 다양한 응용분야에서 그 중요성이 부각되고 있다. 이와 함께 시스템 자원의 효율성을 높이고 응용 소프트웨어 간 안전한 실행환경을 제공하기 위해서 자원 파티셔닝의 필요성이 강조되고 있다 [1]. 자원 파티셔닝은 하나의 응용 소프트웨어를 구성하는 태스크들 단위로 시스템 자원(예, CPU, 메모리 등)을 할당하고, 자원 사용 관점에서 서로 다른 응용 소프트웨어에 속한 태스크 간 간섭이 발생하는 것을 차단하는 기법이다. 리눅스는 최근에 태스크 그룹 단위의 자원 할당과 스케줄링이 가능하도록 Control Group (cgroup) 프레임워크를 구현하였으며 [2], 이러한 cgroup은 자원 파티셔닝을 구현하기에 적합한 기능들을 제공하고 있다.

최근 임베디드 시스템의 응용 분야가 다양해지면서 주기적인 파티션과 함께 비주기적인 파티션들에 대한 요구가 증가할 것으로 전망된다. 예를 들어서 기존의 기계 제어 소프트웨어는 센서 값을 읽고 액추에이터를 제어하기 위해서 태스크들은 주기적으로 수행되도록 구현되었으며, 이들을 포함하고 있는 파티션도 역시 주기적으로 수행되는 것이 적합하다. 하지만 사용자와 상호작용하거나 네트워크 메시지에 의해서 임무를 수행하는 태스크들은 높은 응답성과 자원의 효율적 사용을 위해서 비주기적으로 수행되는 특성을 가지며, 이들을 위한 파티션은 이벤트 기반의 비주기적 실행이 적합할 것이다.

하지만 기존 시스템들은 비주기적인 파티션은 고려하고 있지 않거나, 주기적인 파티션에 비해서 우선순위가 낮게 취급하고 있다. 파티셔닝 기능을 정의하고 있는 ARINC-653과 같은 표준들은 파티션들이 주기적으로 시스템 자원을 사용하는 환경만을 고려하고 있다 [3]. 앞에서 언급한 리눅스 cgroup은 파티셔닝을 구현하기에 적합하지만 주기적인 태스크 그룹이 비주기적인 태스크 그룹에 비해서 항상 높은 우선순위를 갖도록 되어 있다. 이러한 문제를 해결하기 위해서 본 논문은 높은 우선순위의 비주기적인 태스크 그룹을 지원하기 위한 구조를 제안하고, 리눅스의 cgroup 프레임워크를 확장하여 구현한다.

본 논문은 다음과 같이 구성되어 있다. 서론에 이어 2장에서는 리눅스에서 지원하는 그룹 스케줄링 프레임워크인 cgroup에 대해서 소개하고 관련 연구에 대해 논의한다. 3장에서는 높은 우선순위의 비주기적 태스크 그룹을 지원하기 위한 구조를 제안하고, 초기 구현 결과물의 성능을 보인다. 마지막으로 4장에서는 본 논문의 결론과 향후 계획에 대하여 기술한다.

2. 배경지식 및 관련연구

2.1 그룹 스케줄링

리눅스 커널은 효율적인 시스템 자원 관리를 위해서 cgroup 프레임워크를 제공한다 [2]. 이를 통하여 프로세스

본 연구는 민군겸용기술사업(Dual Use Technology Program) 지원을 받아 수행되었습니다 (UM13018RD1).

들을 그룹화하고, 해당 그룹에 CPU, 메모리, 네트워크 I/O와 같은 시스템 자원을 일정한 기준에 따라 분배하여 사용할 수 있다. 예약된 각 시스템 자원은 서브시스템(subsystem)으로 정의되며, cgroup은 서로 다른 시스템 자원 종류에 대해서 여러 서브시스템을 포함할 수 있다.

예로서 태스크 그룹 스케줄링을 위해서는 CPU 서브시스템이 이용된다. CPU 서브시스템은 특정 태스크 그룹에 할당될 CPU 시간을 정의하고, 태스크 그룹 간 CPU 사용 시간을 일정한 비율로 보장한다. CPU 서브시스템은 매개변수를 통하여 CPU 자원을 할당하며, 리눅스에서 제공하는 스케줄러 별로 매개변수를 관리한다. CFS(Completely Fair Scheduler)에 속한 태스크 그룹의 CPU 할당시간은 cpu.shares 매개변수에 CPU 할당시간에 대한 비율을 지정함으로써 정의할 수 있다. RT(Real-Time) 스케줄러에 속한 태스크 그룹은 cpu.rt_period_us와 cpu.rt_runtime_us를 통해 CPU 자원을 할당한다. cpu.rt_period_us는 CPU 자원이 정기적으로 할당되는 주기를 의미하며, cpu.rt_runtime_us는 주어진 주기 내에서 CPU 자원을 최대한 연속적으로 사용할 수 있는 시간을 정의한다. 이러한 내용은 표1에 요약되어 있다.

매개변수	설 명
cpu.shares	태스크 그룹의 상대적인 CPU 시간 할당 비율을 지정
cpu.rt_runtime_us	태스크 그룹이 CPU 자원을 연속적으로 점유할 수 있는 시간
cpu.rt_period_us	태스크 그룹에 CPU 자원이 할당되는 주기

(표 1) CPU 서브시스템의 매개변수

리눅스의 cgroup을 이용하면 다양한 특성의 프로세스를 그룹 스케줄링 할 수 있으며 프로세스들을 공통의 특성으로 분류하여 하나의 그룹으로 묶고, 해당 그룹의 특성에 필요한 자원을 분배하거나 적절한 스케줄링 알고리즘을 적용할 수 있다.

2.2 관련연구

그룹 스케줄링을 위해 파티셔닝 기법을 이용할 수 있다. 파티셔닝이란 단일 컴퓨팅 노드에서 공간 및 시간 자원에 대한 분리를 통해 태스크를 그룹화 할 수 있는 기술이다. 이러한 파티셔닝을 위한 계층적 스케줄러에 대한 연구도 활발히 진행되고 있다 [1, 4]. 하지만 이러한 연구들은 주기적 파티션를 대상으로 하는 경성 실시간 시스템을 고려하고 있으며, 비주기적 파티션과 주기적 파티션이 혼재된 시스템을 대상으로 하고 있지 않다.

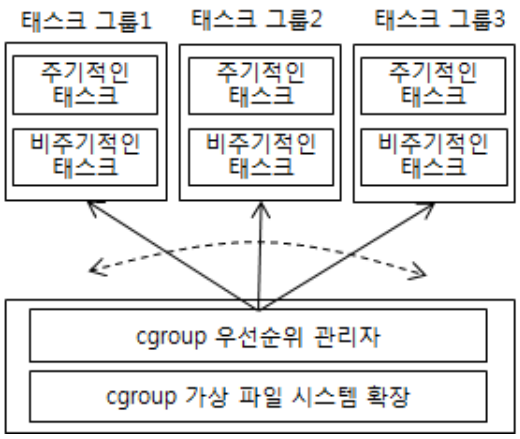
그리고 리눅스 시스템에서 실시간성을 지원하기 위한 cgroup 기반의 스케줄링 기법도 연구가 되고 있다 [5]. 그러나 cgroup을 이용한 스케줄링의 가능성을 보여주고는 있으나, 리눅스의 스케줄링 구조가 가지는 근본적인 문제를 해결하지는 못하고 있다.

3. 그룹 스케줄링을 위한 스케줄러

본 장에서는 주기적인 태스크 그룹과 비주기적인 태스크 그룹의 우선순위를 유연하게 결정할 수 있는 구조를 제안한다.

3.1 스케줄러 구조 설계

본 연구에서 제안하는 스케줄러의 구조는 그림 1과 같다. 그림 1에서 하나의 태스크 그룹은 주기적인 태스크와 비주기적인 태스크가 함께 동작될 수 있다. 리눅스 시스템은 주기적 그룹 스케줄러가 비주기적 그룹 스케줄러보다 높은 우선순위를 가진다. 따라서 비주기적인 태스크 그룹이 주기적인 태스크 그룹보다 높은 우선순위로 동작할 수 없다. 높은 응답성이 필요한 비주기적 태스크 그룹의 요구사항을 만족시키기 위해서는 높은 우선순위의 비주기적 태스크 그룹을 지원할 수 있어야 한다. 이러한 문제를 해결하기 위해 태스크 그룹에 우선순위를 할당하고 이러한 우선순위를 기반으로 스케줄링 할 수 있는 구조가 필요하다.



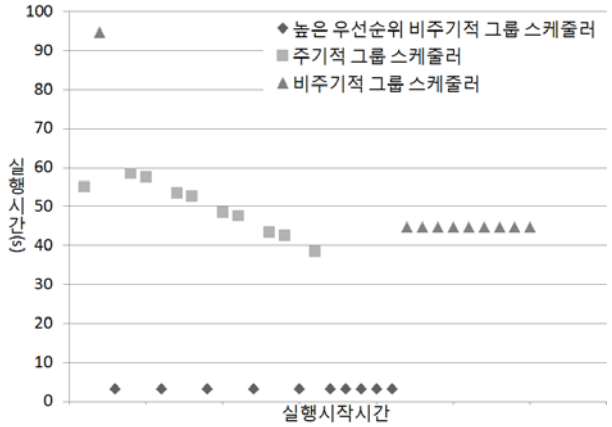
(그림 1) 전체 스케줄러 구조

기존의 cgroup 프레임워크는 태스크 그룹에 대한 CPU 자원의 할당 비율만을 제어할 수 있기 때문에 절대적인 그룹 간 우선순위를 할당해 줄 수는 없다. 따라서, cgroup 가상 파일 시스템을 확장하여 그룹 간의 우선순위를 지원할 수 있도록 하고 cgroup 프레임워크를 수정하여 태스크 그룹에 우선순위를 할당하고 관리할 수 있는 cgroup 우선순위 관리자를 추가한다.

3.2 초기 구현 및 성능 분석

3.1절에서 제안한 구조의 초기 구현을 위해 리눅스에서 제공되는 스케줄러에 RT 그룹 스케줄러보다 높은 우선순위를 가지는 CFS 그룹 스케줄러를 추가하여 높은 우선순

위 비주기적 그룹 스케줄러를 구현하였다. 그리고 추가된 스케줄러의 CPU 할당 비율을 정의할 수 있도록 CPU 서브시스템에 cpu.shares_high 매개변수를 추가하였다. 초기 구현에서는 하나의 태스크 그룹에 한 종류의 스케줄러만 사용하는 것으로 가정한다.



(그림 2) 성능 측정 결과

성능 측정을 위해 그룹 별 태스크를 동시에 실행시켜 우선순위에 따라 프로세스가 실행되는 시간의 차이를 살펴보았다. 새로 추가된 높은 우선순위의 CFS 그룹 스케줄러가 RT 그룹 스케줄러의 프로세스를 선점하고 실행할 수 있어 기존의 CFS 그룹 스케줄러의 프로세스보다 빠른 실행시간을 가질 것이라 예상할 수 있다.

성능 측정은 Intel i3-2330M 프로세서의 싱글코어 환경에서 진행되었다. 그룹 스케줄링을 위해 세 개의 태스크 그룹을 만들고 태스크 그룹 별로 태스크를 10개씩 포함하도록 하였다. 그리고 세 개의 태스크 그룹을 높은 우선순위 비주기적 그룹 스케줄러, 주기적 그룹 스케줄러, 비주기적 그룹 스케줄러로 각각 동작하도록 하였다. 실험에서 사용된 태스크는 실수 연산을 수행하는 프로그램이며 5백만 번의 계산을 수행 후 종료한다. 그림 2는 모든 태스크를 동시에 실행시킨 결과를 보여주고 있다. 가로축은 각 프로세스가 실행을 시작한 시간을 나타내며, 세로축은 프로세스 하나가 5백만 번의 연산을 끝내는데 걸린 시간을 나타낸다. 측정 결과 우선순위가 높은 프로세스는 우선순위가 낮은 프로세스를 선점하여 실행되어 실행시간이 짧은 것을 확인할 수 있으며, 새로 추가된 CFS 그룹 스케줄러가 RT 그룹 스케줄러보다 빠른 실행시간을 보이고 있다. 이것은 제안된 구조가 비주기적 태스크 그룹의 응답성을 높일 수 있는 가능성을 보여준다.

4. 결론 및 향후 계획

본 논문에서는 주기적인 태스크 그룹보다 높은 우선순위를 가지는 비주기적인 태스크 그룹을 지원하기 위한 구조를 제안하였다. 또한, 리눅스 스케줄링 구조를 확장하여 높은 우선순위를 가지는 CFS 그룹 스케줄러를 추가하고 실험을 통하여 제안된 구조가 높은 우선순위의 비주기적 태스크 그룹을 지원할 수 있음을 보였다.

향후 계획으로는 태스크 그룹에 우선순위를 할당하여 그룹 간 우선순위를 지원할 수 있는 구조를 구현한다.

참고문헌

[1] Han, S., & Jin, H. W. (2013). Resource partitioning for Integrated Modular Avionics: comparative study of implementation alternatives. Software: Practice and Experience.

[2] cgroups [Online]. Available : http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt

[3] Aeronautical Radio Inc., “Avionics Application Software Standard Interface (Part 1): Required Services”, ARINC Specification 653 P1-2, December 2005.

[4] Checconi, F., Cucinotta, T., Faggioli, D., & Lipari, G. (2009, June). Hierarchical multiprocessor CPU reservations for the linux kernel. In Proceedings of the 5th international workshop on operating systems platforms for embedded real-time applications (OSPERT 2009), Dublin, Ireland (pp. 15-22).

[5] Joo, Y., Lee, D., Kim, J., & Eom, Y. I. (2013, January). Cgroups-based scheduling scheme for heterogeneous workloads in smart TV systems. In Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication (p. 96). ACM.