멀티프로세서 기반 리눅스에 실시간성 지원 방안 연구

A Study to support real-time on Multiprocessor Linux

이 상 길, 이 철 훈
충남대학교 컴퓨터공학과

Lee Sang-Gil, Lee Cheol-Hoon
Chungnam National University

요약

실시간성이란 연산의 정확성과 연산 수행시간이 예측 가능한 시간 내에 완료되는 것을 말한다. 범용 운영체제인 리눅스에서는 CFS(Complete Fair Scheduling)을 사용하여 높은 우선순위 태스크에 대한 빠른 응답성을 보장할 수 없기 때문에 실시간성을 제공하지 못한다. 이를 해결하기 위해 RTiK-Linux(Real-Time implanted Kernel for Linux)가 개발되어 리눅스 환경에서 실시간성을 제공하였지만 개발 당시 싱글프로세서 환경만을 고려하여 최근 사용되는 멀티프로세서 환경에서는 실시간성 제공에 어려움이 있다. 본 논문에서는 멀티프로세서 리눅스 상에서 발생하는 RTiK-Linux의 문제점에 대해 설명하고 이에 대한 해결방안을 기술하여 멀티프로세서 리눅스 상에서 실시간성을 제공하기 위한 방안을 연구하였다.

Ⅰ. 서론

최근 무기체계 모니터링 장비는 개발자의 편의와 라이선스 비용 절감을 위해 공개 소프트웨어로 구성된 운영체제인 리눅스 환경을 요구한다. 범용 운영체제인 리눅스의 경우 실시간성을 제공하기 위해 RTAI(Real-Time Application Interface)와 RT-Linux(Real-Time-Linux)를 이용하지만 RTAI의 경우 경성 실시간성을 제공하지 않는 단점이 있고 RT-Linux는 경성 실시간성을 지원하기 위한 어셈블리 코드 수정에 많은 노력이 필요하여 실시간 응용프로그램 개발에 어려움이 있다. 이를 개선하기 위해 RTiK-Linux 개발을 통한 리눅스에 실시간성을 제공하는 연구가 진행되었으나 이는 싱글프로세서 환경만을 고려한 설계로 최근 기술발전으로 대부분 사용하고 있는 멀티프로세서 환경에 맞도록 수정하는 것이 필요하다.

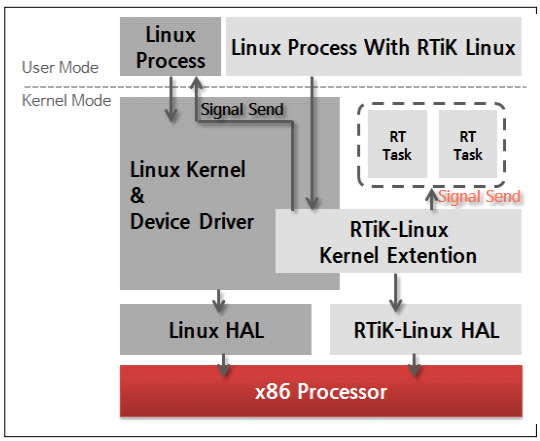
본 논문에서는 이를 해결하기 위해 RTiK-Linux를 확장한 RTiK-MP-Linux(Real-Time implanted Kernel for Multi Processor-Linux)를 설계하였다.

본 논문은 2장에서 관련 연구로 기존 RTiK-Linux에 대한 구조와 IO APIC, Local APIC에 대해서 설명하고 3장에서는 멀티프로세서 환경에서의 실시간성 제공을 위한 방법을 제시하고 4장에서는 실험 환경 및 결과를 기술하였으며, 마지막 5장에서는 결론을 맺는다.

Ⅱ. 관련연구

1. RTiK-Linux

RTAI와 RT-Linux를 대체하여 리눅스에 실시간성을 제공하기 위하여 연구개발된 RTiK-Linux는 [그림 1]과 같이 디바이스드라이버 형태로 커널에 이식된다. 리눅스의 커널 및 하드웨어 자원을 이용할 수 있도록 리눅스와는 별개의 HAL(Hardware Abstraction Layer)을 가지는 형태 커널 영역에 접근이 가능하다[1].



▶▶ 그림 1. RTiK-Linux의 구조

2. APIC(Advaned Programmable Interrupt Controller)

1.1 Local APIC

Local APIC는 인텔 아키텍처에서 사용하는 Interrupt Controller로 주변 하드웨어에서 발생하는 인터럽트를 프로세서에게 전달하거나 IPIs(Inter-Processor Interrupt) 메시지를 생성, 전달하는 역할을 한다[2][3].

1.2 IO APIC

각 프로세서마다 존재하는 Local APIC를 관리하기 위해 Intel Architecture에서는 IO APIC를 사용하여 각 프로세서에게 Interrupt를 전달한다. 부팅시 모든 프로세서에 독립적인 APIC ID를 부여하여 외부에서 발생하는 인터럽트나 IPIs를 해당 프로세서에게 전달한다[2].

Ⅲ. 본론

1. Process Affinity를 이용한 코드 구현

멀티프로세서에서는 각각의 프로세서가 스레드를 통해 태스크를 동작킨다. 이때 실시간 처리를 위해 중지 상태의 태스크가 특정 프로세서에 종속적이지 않으면 태스크가 다시 실행될 때, 다른 프로세서로 변경될 수 있다. 싱글 프로세서에서는 한 개의 CPU 캐시를 이용하나 멀티프로세서 환경에서는 Context Switching시 프로세서가 변경되면 캐시에 등록하는 과정에서 지연 시간이 발생하여 응답성을 낮추는 것으로 실시간성에 영향을 미친다. 이를 해결하기 위하여 Process Affinity 코드를 이용한다. 리눅스에서 제공하는 CPU 구조체를 이용하여 1번 CPU를 제외한 모든 CPU번호를 마스크한 다음 sched_setaffinity 함수를 통해 1번 CPU는 항상 RTiK-MP-Linux로 관리되는 실시간 태스크를 실행하도록 지정하여 멀티프로세서 환경에서의 응답성을 높여 안정적인 실시간 태스크의 수행이 가능하다.

```
cpu_set_t set;
int ret;
CPU_ZERO (&set);    /* initialized CPU Structure */
CPU_CLR (1, &set);  /* CPU #1 Set */
ret = sched setaffinity (0, sizeof (cpu set t), &set);
```

▶▶ 그림 3. Process Affinity 설정 코드

Ⅳ. 실험환경 및 결과

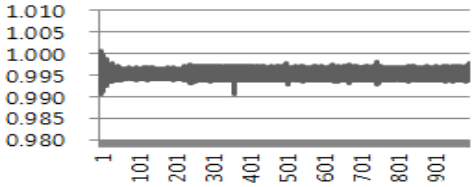
1. 실험환경

멀티프로세서 리눅스 환경에서 실시간성을 제공하기 위한 RTiK-MP-Linux의 실험환경은 [표 1]과 같다.

표 1. 실험 환경

	HOST
CPU	Intel Core2 Duo E8400 @ 3.00GHz
OS	Ubuntu Desktop 10.04 LTS
Kernel Version	2.6.32.63

멀티프로세서 리눅스 환경에서 RTiK-MP-Linux의 실시간성 제공 성능을 확인하기 위해 HOST에 RTiK-MP-Linux이식한 다음 1ms의 주기를 설정 후 발생하는 주기를 측정하였다. 아래 [그림 4]의 그래프는 1천회 발생 기록으로 주기가 일정하게 1ms 범위 이내로 발생하는 것을 볼 수 있으며, [표 2]를 통해 최소 주기 값과 최대 주기 값이 오차범위 1% 내의 결과를 보이는 것을 확인할 수 있다.



▶▶ 그림 4. 1ms 주기 측정 그래프

표 2. 1ms 주기 실험 결과

	1ms 주기 실험 결과
최소 주기 값	0.9905 ms
최대 주기 값	1.0006 ms
평균 주기 값	0.9999 ms
오 차 율	0.50 %

Ⅴ. 결론

본 논문에서는 멀티프로세서 리눅스 환경에서 실시간성 지원 방안을 연구하기 위하여 기존 RTiK-Linux를 개선한 RTiK-MP-Linux를 설계하였다. 멀티프로세서 환경에서 특정 프로세서가 실시간 태스크만을 처리할 수 있도록 CPU구조체와 sched_setaffinity 함수를 사용하여 항상 1번 프로세서가 그 역할을 수행하도록 설정하였다. 이를 통해 프로세서간 Context Switching 과정이 줄어들어 실시간 태스크의 응답 속도가 좋아짐으로 인한 안정적인 실시간성 제공을 확인할 수 있었다.

향후 연구과제로는 프로토타입으로 개발된 RTiK-MP-Linux를 사용자 영역의 일반 응용프로그램에서 쉽게 사용할 수 있도록 미들웨어 API 형태로 설계 및 구현하는 방안이 필요하며, 리눅스 스케줄러 상에서 RTiK-MP-Linux의 실시간 태스크와 일반 리눅스 태스크간의 효율적인 스케줄링 방안을 연구할 필요가 있다.

■ 참 고 문 헌 ■

[1] 송창인, 김종진, 이철훈 “리눅스상의 실시간성 지원을 위한 RTiK-Linux의 설계 및 구현”, 한국콘텐츠학회논문지, 제9권, 제1호, pp.17-18, 2011.

[2] Intel, “Intel 64 and IA-32 Architecture Software Developer’s Manual Volume 3B : Systems Programming Guide”, 2012.

[3] 송창인, 이승훈, 이철훈, “멀티프로세서 윈도우즈 XP 상에서 실시간성 지원”, 한국컴퓨터정보학회, 제20권, 제1호, pp.21-24, 2012.