하이퍼바이저의 사용 유무에 따른 RTOS의 성능 비교

심철*, 최민*
*충북대학교 정보통신공학부
e-mail:eisen@cbnu.ac.kr

Performance Comparison of RTOS with Hypervisor usage

Cheol Sim*, Min Choi*
*Dept. of Information and Communication, Chungbuk University

요 약

최근 ARM 프로세서의 가상화 확장 기술을 이용하는 임베디드 시스템에서 다종의 OS 작동을 지원하는 하이퍼바이저가 많이 개발되었다. 가상화 기술은 하드웨어 자원을 효과적으로 사용한다는 이점이 있지만, RTOS를 작동시킬 경우 하이퍼바이저의 오버헤드에 의해 RTOS의 성능이 저하될 수 있는 문제가 발생한다. 본 논문에서는 가상화 기술을 지원하는 ARMv7 Cortex-A15 프로세서를 탑재한 NVidia Jetson TK-1 임베디드 보드에서 RTOS가 단독으로 작동했을 때의 성능과 QPlus Hypervisor를 통해 Linux OS와 함께 RTOS가 작동했을 때의 성능을 측정하고 비교 분석 하였다.

1. 서론

가상화(Virtualization) 기술과 하이퍼바이저가 발전함에 따라 임베디드 시스템에서도 하나의 하드웨어 자원에서 다종의 OS를 동시에 작동시킬 수 있게 되었다.

이러한 기술은 고신뢰 OS로 손꼽히는 RTOS와 고성능 OS인 Linux 등을 하이퍼바이저를 통해서 동시에 작동시킬 수 있다. 실시간성이 필요한 통신 제어 작업은 RTOS, 다양한 유저 어플리케이션은 Linux가 수행함으로써 고신뢰와 고성능을 동시에 만족시킬 수 있다. 이는 다른 운영체제를 사용하는 장비 간의 통신 비용 등을 절감시켜주는 효과를 나타낸다.

이러한 이점에도 불구하고 고려해야할 점이 있다. RTOS의 특성 때문에 실시간성을 보장해줘야 한다. 여러 Guest OS가 동작 중일 때, 특정 시점에서 하나의 OS가 하드웨어 자원을 사용하므로 하이퍼바이저가 OS간의 문맥전환을 발생시킨다. 이때, 실시간성을 보장하지 않는 하이퍼바이저는 문맥전환을 하면서 발생하는 오버헤드에 의해 RTOS는 OS상의 시간이 아닌 실제 시간에 대해 지연시간을 가지게 되고 이는 RTOS의 성능 저하로 이어진다.

본 논문에서는 가상화 기술을 지원하는 ARMv7 Cortex-A15 프로세서가 탑재된 임베디드 보드에서 FreeRTOS가 단독으로 동작할 때와 하이퍼바이저 통해서 Linux OS와 함께 동작할 때의 성능을 비교해 본다.

2. 배경지식

하이퍼바이저란 하나의 물리적 하드웨어 계층과 여러 Guest OS 계층 사이에서 존재하는 미들웨어이며, Guest OS들이 자원을 사용하는 것을 관리하고 모니터링한다. 이를 통해 다수의 OS를 동작시키는 것을 도와주며. 각 OS는 혼자 동작하는 것과 같이 고립화를 보장한다.

ARMv7 Cortex-A15 프로세서는 가상화 기술을 지원하기 위해서 하드웨어 지원 가상화 기술인 Virtualization Extensions(VE)가 사용되었고, 인터럽트 가상화 기술(GICv2)을 지원한다. 이는 인터럽트가 발생했을 때 하이퍼바이저의 개입 없이 Guest OS가 인터럽트를 처리하여 오버헤드를 줄이는 기술이다.

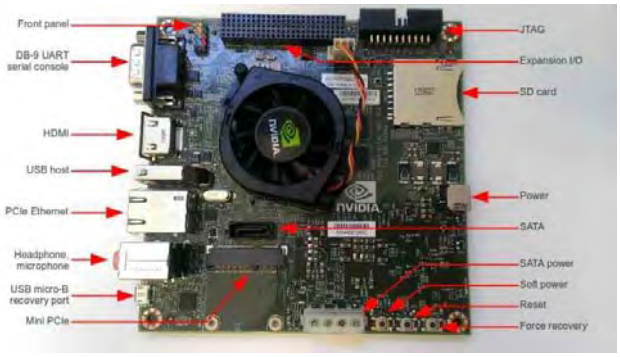
RTOS(Real-Time) OS는 작업을 처리하는 시간에 대해 엄격하거나 일정한 응답 시간을 요구하는 분야에 사용되는 실시간 운영체제이다. RTOS를 사용하는 분야에 따라 다양한 특징을 가진 RTOS들이 존재한다. 대표적으로 uC/OS-II와 본 논문의 실험 대상인 FreeRTOS 등이 있다.

3. 실험 설계

실험을 위해 사용된 환경은 ARMv7 Cortex-A15 프로세서를 탑재한 NVidia 社의 Jetson TK-1 개발 보드를 사용하였다(그림 1). 하이퍼바이저로는 한국전자통신연구원에서 개발한 QPlus Hypervisor를 사용하였고. Guest OS로는 Jetson TK1에 포팅된 Linux(Ubuntu 14.04)와 포팅된 FreeRTOS를 사용 하였다.

실험 설계는 FreeRTOS가 단독으로 동작할 때, QPlus Hypervisor에서 Linux와 함께 동작할 때의 두 가지 상황에 대하여 FreeRTOS의 Task 전환 지연시간과 인터럽트

지연시간을 각각 측정하였다. QPlus Hypervisor를 통해 실험할 경우, Linux는 부팅이 완료된 후 백그라운드 프로세스를 제외하고 추가적인 부하를 만들지 않았다.



(그림 1) NVidia Jetson TK-1 개발 보드

성능 비교를 위한 실험 방법은 Rhealstone benchmark를 참고하여 진행하였다[8]. Jetson TK-1 개발 보드의 GPIO 포트들을 활용하여 Tektronix 오실로스코프로 실험 결과를 측정하였다.

Task 전환 지연시간 측정은 2개의 Task(T1, T2)를 대상으로 하였다. T1이 자신의 GPIO 포트에 HIGH를 출력하고 Task 전환을 한 뒤 T2가 자신의 GPIO 포트에 HIGH를 출력하면 이때의 시간차를 Task 전환 지연시간으로 볼 수 있다.

인터럽트 지연시간 측정은 두 개의 Task(T1, T2)와 인터럽트 핸들러를 대상으로 하였다. 하나의 GPIO 포트에 대해 T1과 T2가 각각 HIGH와 LOW를 출력하면서 Task를 전환한다. Task를 전환하는 시점부터 인터럽트 핸들러가 제어하는 GPIO 포트에 HIGH가 출력될 때까지의 시간을 인터럽트 지연시간으로 볼 수 있다.

4. 실험 결과 및 분석

실험 결과는 앞서 설계한 것과 같이 아래의 4가지 결과를 도출하였는데 QPlus Hypervisor를 통해 Linux와 함께 동작하는 FreeRTOS는 단독으로 수행될 때보다 Task 전환 지연시간이 약 1100배 이상, 인터럽트 지연시간은 약 50배 이상 느려지는 것을 확인하였다.

실험 결과를 토대로 유추할 수 있는 점은 다음과 같다. 단독으로 FreeRTOS가 동작할 때 Task 전환 지연시간과 인터럽트 지연시간이 크게 차이 나지 않지만, QPlus Hypervisor를 통해 동작하는 FreeRTOS는 Task 전환 지연시간이 많은 차이를 보이며 인터럽트 지연시간보다 느려졌다는 점이다.

여기서 주목할 점은 ARMv7 Cortex-A15 프로세서는 인터럽트 가상화(GICv2)를 지원하는 점이다. 이것은 인터럽트가 발생했을 때, Guest OS로 직접 인터럽트를 처리할 수 있게 해줌으로써 하이퍼바이저의 인터럽트 처리 오버헤드를 감소시켜주는 기술이다. 이러한 이점으로 인해 앞서 보인 하이퍼바이저를 통한 Linux 및 FreeRTOS 동작 시 인터럽트 지연시간이 더 빠르게 측정되었다고 볼 수 있다.

5. 결론

본 논문은 ARMv7 Cortex-A15를 탑재한 Jetson TK-1 개발 보드에서 FreeRTOS를 단독으로 작동시킬 때와 하이퍼바이저를 통해 작동시킬 때의 성능을 비교 분석하였다. 실시간성을 보장하지 않는 하이퍼바이저는 실제 동작되는 시간과 시스템 상에서의 시간이 일치하지 않아지게 되고, RTOS의 특성을 살리지 못하는 결과를 초래하게 된다.

앞으로의 과제로는 이러한 하이퍼바이저에서도 RTOS가 실시간성을 갖도록 하는 시스템 설계 방안의 문제를 남겨둔다.

참고문헌

[1] Cheol Sim, Min Choi. “Implementation of Porting RTOS to ARM Cortex-A15.” Proceedings of the Korean Society of Computer Information Conference , 25.1 (2017.1): 3-4.

[2] Kihong Lee, Dongwoo Lee, Young Ik Eom. "A Study on Trend for Hardware-assisted Virtualization Technology." Proceeding of the Korean Information Science Society, (2013.6): 1316-1318.

[3] Young Gyo Jung, Byung Jun Lee, Hee Yong Youn. "The comparative analysis of Hypervisor according to Virtualization Method." Proceedings of the Korean Society of Computer Information Conference , 23.1 (2015.1): 251-253.

[4] T. Lanier, “Exploring the Design of the Cortex-A15 Processor,” ARM, Tech. Rep.

[5] In-Kyu Han. "K-Hypervisor : Design and implementation of hypervisor based ARM Cortex-A15." Proceeding of the Korean Information Science Society, (2014.6): 1559-1561.

[6] P. Varanasi, G. Heiser, "Hardware-supported virtualization on ARM" APSys '11 Proceeding of the Second Asia-Pacific Workshop on Systems, pp. 11:1-11:5, 2011.

[7] FreeRTOS Project, http://www.freertos.org

[8] R. Kar. and K. Porter. "Rhealstone: A Real-Time Benchmarking Proposal". Dr. Dobb's Journal. 1989. Available: http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/DDJ/1989/8902/8902a/8902a.htm

[9]T. J. Boger, “Rhealstone benchmarking of FreeRTOS and the Xilinx Zynq extensible processing platform”, Master Thesis, Temple University, May 2013.