L4 마이크로커널을 이용한 가상화 설계

강창호*, 조상영*
*한국외국어대학교 컴퓨터공학과
e-mail:seasonyun,sycho@hufs.ac.kr

A Design of Virtualization Using L4 Micorkernel

Chang-Ho Kang*, Sang-Young Cho*
*Dept of Computer Engineering, Hankook University of Foreign Studies

요 약

최근에 다기능 복합화되는 임베디드 시스템을 위한 가상화 연구가 활발하다. 본 논문에서는 L4 마이크로커널을 기반으로 한 실시간 운영체제 MicroC/OS-II 가상화를 위한 L4 환경에서의 전체 구성, 인터럽트 핸들링, 게스트 운영체제 초기화 과정의 설계에 대해 기술한다.

키워드: 마이크로커널, 가상화, 하이퍼바이저, 실시간 운영 체제

1. 서론

일반적으로 가상화(Virtualization)는 한 대의 컴퓨터 시스템에서 여러 개의 운영체제를 함께 운용할 수 있도록 지원하는 기술을 의미한다[1]. 이 기술은 과거 메인프레임 환경에서 보편적으로 사용되었던 것으로 x86 계열의 프로세서를 쓴 서버들이 가상화 기술을 빠른 속도로 흡수하면서 가상화는 서버 시장의 메인 트렌드로 자리 잡게 됐다. 그후 인프라 자원에 해당하는 스토리지뿐만 아니라 데이터, 애플리케이션, 데스크톱, 임베디드 등 IT를 이루는 대부분의 영역에서 가상화 개념을 도입하기 시작했다.

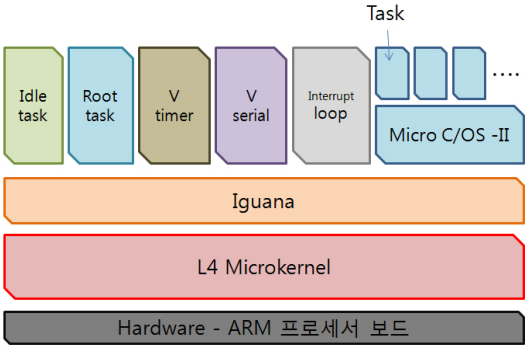
최근 임베디드 시스템에서의 가상화가 활발하게 이루어지고 있다. 가상화가 관심 받는 이유는 시스템의 물리적 자원에 대해서 다수의 사용자에게 각각 독립적인 사용 환경을 제공함으로써 하드웨어 비용절감, 높은 시스템 사용 효율성, 오류의 격리, 기존 어플리케이션의 재사용등의 이점이 있기 때문이다[2].

본 논문에서는 임베디드 시스템을 위한 가상화를 위하여 OKL4 마이크로커널[3]을 기반으로 한 실시간 운영체제 MicroC/OS-II[4] 가상화에 대해 다룬다.

2. 가상화 설계

일반적인 운영체제는 하드웨어와 직접적으로 연관되어서 그것을 관리하는 역할을 한다. 반면에 가상화된 운영체제는 사용자 모드에서 동작하게 되고 하이퍼바이저가 제공하는 인터페이스를 사용하여서 간접적으로 하드웨어에 대한 관리를 하게 된다. 따라서 가상화를 위해서는 새로운 인터페이스에 맞게 운영체제의 변경이 필요로 하게 된다. 본 논문에서는 L4 마이크로커널을 하이퍼바이저로의 역할을 수행하도록 하여 그 위에 MicroC/OS-Ⅱ를 게스트 운영체제로 이식함으로써 MicroC/OS-Ⅱ에 대한 가상화를 구현하였다. L4 마이크로커널은 관리자 모드에서 동작하게 되며 그 위에 가상화된 MicroC/OS-Ⅱ의 경우 사용자 모드로 동작하게 된다. 즉 가상화된 MicroC/OS-Ⅱ에서 하드웨어 리소스에 대한 접근은 불가능하며 이에 가상화된 게스트 운영체제에서 하드웨어 리소스 접근에 필요한 인터페이스가 필요하고 MicroC/OS-Ⅱ의 하드웨어에 의존적인 Port 제작이 필요하게 된다[4].

본 논문의 가상화 시스템 구성은 L4 마이크로커널과 Iguana 프레임워크 위에 MicroC/OS-Ⅱ를 하나의 애플리케이션으로 가상화하는 구성이다. 이를 위하여 기본적인 Idle 태스크와 Root 태스크, 타이머와 콘솔 출력을 위한 v_timer와 v_serial 쓰레드, 애플리케이션 형태의 MicroC/OS-Ⅱ 쓰레드의 인터럽트를 담당해 줄 Interrupt_loop 쓰레드로 구성된다. 그림 1은 설계된 전체 가상화 시스템 구성을 보여주고 있다. MicroC/OS-II는 하나의 쓰레드로 가상화 되어 동작하게 된다.



(그림 1) 가상화 시스템 구성

가상화 시스템의 각 구성 요소에 대한 자세한 설명은 아래와 같다.

- Iguana_Server(root task)

L4 마이크로커널은 기본적으로 root 태스크를 생성하며 root 태스크만이 쓰레드를 생성하거나 파괴할 수 있다.

Iguana와 같이 컴파일 하게 되면 L4 마이크로커널의 root 태스크는 Iguana의 Iguana_server가 되어 이것이 root 태스크의 역할을 이어 받아 수행한다.

- **idle task**

L4 마이크로커널은 기본적으로 root 태스크 이외에 idle 태스크를 생성하는데 이 태스크는 운영체제의 일반적인 idle 태스크의 역할을 수행한다.

- **v_timer**

L4 마이크로커널에서 유저 쓰레드의 장치에 대한 접근은 장치 쓰레드에 의해서 이루어지며 타이머 장치에 대한 접근을 위한 쓰레드가 v_timer이다. v_timer의 경우 다른 유저 쓰레드의 request_time 요구에 의해서 동작하고 실제 하드웨어 타이머와 IPC(Inter-process communication)를 통해서 타이머를 동작 시키고 타이머로부터 인터럽트를 받는다.

- **v_serial**

v_serial은 하드웨어 상의 UART 시리얼 장치를 사용하기 위하여 만들어진 쓰레드이며 콘솔상의 입출력은 v_serial과 통신을 통해서 이루어진다.

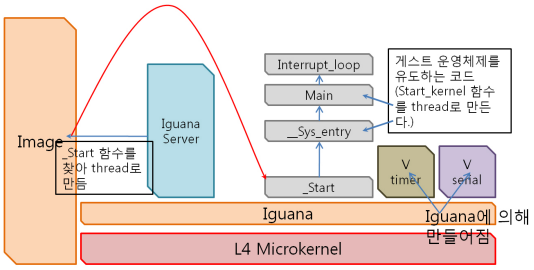
- **Interrupt_loop 쓰레드**

유저 쓰레드의 경우 인터럽트를 직접 처리할 수가 없다. 따라서 인터럽트에 관련된 처리를 위한 쓰레드이다.

- **MicroC/OS-II 쓰레드**

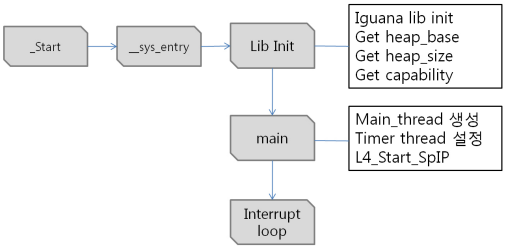
게스트 운영체제인 MicroC/OS-II의 수행을 위한 쓰레드이다.

하이퍼바이저로 사용하기 위한 L4 마이크로커널은 하이퍼바이저의 구성요소를 갖추어야 한다. 하이퍼바이저의 구성요소는 인터럽트 제어 루틴, 시스템 초기화 코드, 커널 간 통신 API이다. 인터럽트 제어 루틴의 경우 Interrupt_loop 쓰레드를 통해 구현하고 시스템 초기화 코드의 경우 _Start함수에서 호출된 main 함수에 구현한다. 그리고 커널 간 통신 API의 경우 L4 마이크로커널에서 제공해 주고 있는 IPC API를 이용한다. 그림 2는 L4 마이크로커널을 하이퍼바이저로 사용하기 위한 설계 그림이다.



(그림 2) 하이퍼바이저 설계

그림 2의 동작은 다음과 같다. 처음에 Iguana_Server가 root 태스크의 역할을 대신 수행하게 되면서 전체 Image에서 _Start함수를 찾는다. _Start함수를 찾으면 해당 함수를 쓰레드로 만들어 Iguana위에서 수행되도록 한다. _Start 함수에서는 Iguana의 API를 사용하기 위한 초기화를 수행하고 main 함수를 호출한다. main 함수는 게스트 운영체제를 유도하기 위한 함수로써 게스트 운영체제에 대한 쓰레드를 생성하고 해당 쓰레드의 시작을 게스트 운영체제의 시작인 Start_kernel 함수로 만들어준다. 게스트 운영체제의 쓰레드를 만들어 주고 나서 Interrupt_loop 함수를 호출하게 되어 Interrupt_loop 쓰레드로 동작하게 된다.



(그림 3) 시스템 초기화 과정

그림 3은 시스템 초기화 코드의 자세한 동작을 보여준다. Lib_Init 함수를 통해서 메모리 관련된 부분들을 초기화하고 해당 메모리에 대한 권한을 얻어온다. main 함수에서는 실제 게스트 운영체제가 돌아갈 main 쓰레드를 생성하고 time tick에서 사용할 v_timer를 설정한 뒤, L4 API인 L4_Start_SpIp를 이용하여 MicroC/OS-II를 main 쓰레드로 실행시킨다. L4_Start_SpIp는 L4 쓰레드를 시작 시키는데 있어서 스택 포인터와 프로그램 카운터 값을 가지고 해상 프로그램 카운터에서 혹은 해당 함수가 쓰레드의 시작과 함께 실행 될 수 있도록 해준다.

3. 결론

최근의 모바일 기기들은 다야한 형태의 복합화된 서비스가 요구되고 있으며 가상화의 장점들로 인하여 임베디드 시스템을 위한 가상화 연구가 활발하다. 본 논문에서는 임베디드 시스템용 마이크로커널인 OKL4를 하이퍼바이저로 사용하여 실시간 운영체제인 MicroC/OS-II를 가상화할 수 있는 시스템 구조를 제안하고 이의 구현을 위한 전체 구조 및 시스템 초기화 설계에 대해 기술하였다.

참고문헌

[1] 안창용, 유혁, “가상화 방법의 비교”, 한국컴퓨터종합학술대회 논문집, 제35권 제1호, pp. 446-450, 2008.

[2] G. Heiser, "Virtualization for Embedded Systems", Open Kernel Labs Inc., Technical White Paper, 2007.

[3] OKL4 Microkernel, http://www.ok-labs.com/.

[4] J. J. Labrosse, “"MicroC/OS-II 실시간 커널 제2판", 성원호 역, 에이콘 출판사, 2003.

"본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음" (NIPA-2009-C1090-0902-0020)