가상화 응용을 위한 L4 마이크로커널 플랫폼

L4 Microkernel Platform for Virtualization Applications

강창호*, 조상영**
Chang-Ho Kang, Sang-Young Cho

Abstract - Recently, researchers are focusing on the usefulness of microkernels regarding to the virtualization of embedded systems. We ported OKL4 2.1 microkernel on MBA2440 and SMDK6410 that use ARM920T and ARM1176JFZ-S cores respectively. The ported OKL4 2.1 environment will be used to develop various virtualization applications.

Key Words : L4, 마이크로커널l, ARM 코어, 가상화

1. 장 서론

SoC 기술과 무선 통신 기술의 발달로 휴대폰, PDA, 개인용 미디어 단말기와 같은 개인용 단말 장치와 산업계의 측정, 계측 장비와 같은 이동형 계측 단말 장치가 다양한 형태로 출시되고 있다. 이러한 모바일 단말 장치는 여러 응용 프로그램을 사용할 수 있도록 복합화된 서비스를 요구하고 있다. 복합 기능의 단말기 형태의 임베디드 시스템은 서비스를 제공하는 타스크들을 관리하기 위한 운영체제를 필수적으로 요구하고 있다. 자원의 제한을 많이 받는 임베디드 시스템에서는 경량화된 운영체제가 필수적이며 정형화된 무거운 모노리틱 커널보다는 모듈 방식을 이용하여 필요로 하는 기능만을 추가할 수 있는 마이크로커널의 형태를 취하는 운영체제가 매우 효율적일 수 있다.

초기의 Mach[1]와 같은 마이크로커널은 개념적 우월성과 구조의 유연성의 장점으로 인하여 운영체제 분야에서 많은 연구가 이루어졌으나 빈번한 IPC(Inter-Process Communication)로 인한 콘텍스트 스위칭(Context Switching)의 부담으로 좋지 않은 성능을 보임에 따라 그 연구의 흐름이 많이 멈추었다. 그러나, QNX[2]와 같이 실시간 시스템에서 지속적으로 사용이 확대된 예가 있으며 L4[3]의 경우와 같이 마이크로커널의 정밀한 설계 및 구현에 따라 성능이 기존의 모노리틱 커널보다 뛰어날 수 있다는 결과들이 나오고 있다. 이에 따라, 복잡해지고 다양해지는 임베디드 시스템의 운영체제의 근간으로 마이크로커널을 사용하는 연구 및 제품이 점점 증가하는 추세이다. 특히 컴퓨터 시스템의 가상화가 뜨거운 이슈로 떠오르고 있는 최근에, 임베디드 시스템의 가상화와 관련하여 마이크로커널의 유용성이 많이 부각되고 있다[4]. 본 논문에서는 마이크로커널의 가상화 응용에 적용하기 위한 마이크로커널 플랫폼 구축에 대하여 기술한다. 플랫폼 구축을 위해 ARM920T 코어를 사용하는 MBA2440 보드와 ARM1176JZF-S 코어를 사용하는 SMDK6410 보드에 OKLab의 OKL4 2.1 마이크로커널과 부분 가상화 리눅스를 이식하였다.

본 논문의 구성은 다음과 같다. 2장에서는 마이크로커널에 대한 소개를 하고 3장에서는 OKL4 이식과 성능 실험에 대해 기술한다. 마지막으로 4장에서 결론을 맺는다.

저자 소개
* 준 회 원 : 한국외국어대학교 컴퓨터공학과 석사과정
** 정 회 원 : 한국외국어대학교 컴퓨터공학과 교수
"본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (IITA-2009-C1090-0902-0020)

2. 장 마이크로커널 소개

2.1 절 커널 소개

커널은 컴퓨터 운영체제의 핵심 부분으로 응용 프로그램이나 운영체제의 다른 부분에 서비스를 제공하기 위하여 꼭 필요한 운영체제의 일부이다. 이러한 커널은 CPU, 메모리, 입출력 장치와 같은 컴퓨터 자원을 관리하고 다른 프로그램이 실행을 하면서 이들 자원을 사용하도록 하며 프로세스 관리, 메모리 관리, 장치 관리, 시스템 호출을 제공하고 있다. 일반적으로 커널은 위와 같은 서비스를 슈퍼바이저 모드(Supervisor mode)의 단일 커널 공간 (kernel space)에서 제공한다. 커널이 운영체제의 내용 중에서 어느 정도를 포함하느냐에 따라서 여러 가지 형태로 구분될 수 있다. 전통적으로 운영체제의 대부분을 기능을 커널 공간에서 운용하는 모노리씩 커널 (Monolithic kernel)이 주류를 이루었으나 1980년대 중반에 커널이 운영체제의 최소한의 기능만을 갖는 마이크로 커널 (Micro kernel)이 등장했으며 이들의 장점을 취한 중간 단계의 하이브리드 (Hybrid kernel)이 나왔다. 모노리씩 커널은 전체가 하나의 주소 공간을 갖고 있기 때문에 풍부하고 강력한 하드웨어 접근이 가능하며 프로그램의 운영

체제 서비스 요청이 한번에 처리될 수 있어 성능이 좋다. 그러나, 커널의 크기가 크며 전체가 하나의 모듈이기 때문에 커널을 확장하거나 유지하는데 드는 비용이 크다. 또한 한 곳의 오류가 전체 시스템에 영향을 줄 수 있기 때문에 신뢰성에 문제가 있는 단점이 있다.

마이크로커널은 커널 공간에서는 IPC, 가상 메모리, 스케줄링과 가장 기본적인 기능만을 수행하고 다른 운영체제의 기능은 사용자 공간에서 동작하는 서버라는 독립된 프로그램을 통해 제공한다. 각 서버는 장치 관리자, 파일 시스템, GUI 등을 담당하고 이들 서비스는 IPC를 통하여 요청되고 서비스된다. 마이크로커널은 운영체제의 구성 및 관리가 간단하며 한 서버의 에러가 자체에 국한되기 때문에 신뢰성이 높아 질 수 있다. 그러나 IPC 및 이에 따른 콘텍스트 스위칭의 부담으로 성능이 떨어지는 단점이 있다.

하이브리드 커널은 마이크로커널의 장점을 살리면서 성능을 보완하기 위하여 하드웨어와 연관이 있는 장치 관리자나 응용 IPC를 커널 모드로 포함하는 형태를 지칭한다.

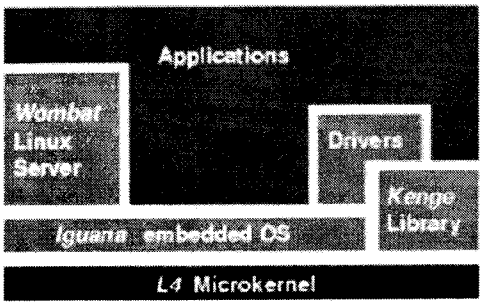
2.2 절 L4 마이크로커널

1980년대의 몇몇 운영체제 관련 프로젝트에서 탄생한 IPC 개념과 운영체제 기능의 서버화를 바탕으로 최초의 마이크로커널 형태의 Mach 3.0이 만들어 졌다.

1985년부터 1994년까지 초기의 마이크로커널은 결국 좋지 못한 성능을 보이게 되어 연구가 급속도로 위축되었다. 모노리씩 커널에 비해 성능이 좋지 않았던 이유는 빈번한 IPC와 서비스 수행을 위한 빈번한 콘텍스트 스위칭에 있었다. 그후, L4 연구진에 의하여 정밀하게 설계되고 구현될 경우에 IPC 비용이 Mach에 비하여 현저히 떨어질 수 있으며 콘텍스트 또한 매우 효율적으로 동작할 수 있음이 증명되었다. 또한 QNX 같은 경우는 다양한 실시간 임베디드 시스템에 성공적으로 사용되어 상업적으로 많은 성공을 거두었다. 최근 모바일 단말기의 기능이 복잡해 지면서 다중 운영체제의 필요성이 증가하고 다양한 모델에 맞추어 최적화된 운영체제를 제공해야 할 필요성이 증가하면서 마이크로커널이 다시 부각되고 있다. 마이크로커널 운영체제에는 Mach, ChorusOS, Minix, L4, QNX, Symbian OS 등이 있으며 하이브리드 커널 형태는 초기 WindowsNT, MacOS X 등이 있다.

L4 마이크로커]은 2세대 마이크로커널으로서, 1세대 마이크로커널을 대표하는 Mach 마이크로커널의 IPC(Inter Process Communication)를 독일의 Jochen Liedtke가 개선하여 만들어낸 마이크로커널이다. Jochen Liedtke는 마이크로커널의 기본적인 개념에는 아무런 문제가 없으며, 얼마나 잘 설계하고 구현하느냐에 따라 충분히 좋은 성능을 낼 수 있음을 증명하였고[3], 이에 대한 구현으로 L3 마이크로커널을 만들었다. L3 마이크로커널의 경험을 바탕으로 L4 마이크로커널이 만들어 졌다. 그후, 다양한 변종이 연구되었으며 National ICT Australia (NICTA)에서 임베디드 시스템을 위한 NICTA::L4-embedded로 만들었다. 그후 OKLab에서 이를 개선한 OKL4 마이크로커널을 개발하였다.

그림 1은 L4 마이크로커널에 대한 NICTA model을 도시하고 있다.



<그림 1. L4 마이크로커널에 대한 NICTA model>

여기서 Kenge는 라이브러리, 서비스, 어플리케이션으로 구성된 L4 기반 시스템을 만들기 위한 프레임워크이고, Iguana는 모든 시스템에서 요구하는 기본적인 OS 서비스를 제공하는 임베디드 운영체제의 핵심이 되는 부분이다. Wombat은 이구아나의 상위에서 유저모드로 동작하는 완전한 부분 가상화 Linux 커널이고, Magpie는 IDL(Interface-Definition Language) 컴파일러로서 C나 최적화된 ARM 어셈블리어로 message-passing stub 코드를 생성한다.

3. 장 OKL4 이식 및 성능

3.1 절 이식 환경 및 이식

OKL4 2.1을 이식하기 위한 환경은 다음과 같다.

<표 1. 이식에 사용된 프로그램>

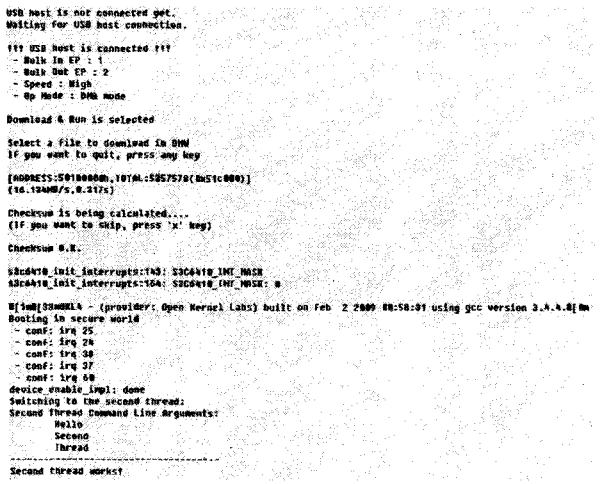
프로그램	내용
Fedora Linux 2.6.18	마이크로커널 개발 환경
Python 2.4.4	Scons 사용 도구
Scons 0.96.1	이미지 빌드
Arm-linux-3.4.4	툴 체인
OKL4 1.5.2, 2.1, 3.0	커널과 이구아나
OKLinux	부분 가상화 리눅스

대상 보드는 ARM920T 코어를 사용하는 MBA2440과 ARM1176JZF-S 코어를 사용하는 SMDK6410이었으며 각각 64MB와 128MB의 SDRAM 메모리를 가지고 있다. 이식을 위하여 기존 OKL4 2.1 폴더 내에 아키텍처에 종속된 프로그램 부분을 수정하였고 필요한 것은 전체 구조에 맞게 추가하였다. 표 2는 커널 부분에서의 이식을 위하여 수정된 파일들의 목록을 보여준다. 이외에도 Iguana 부분에서 필수 드라이버에 관련된 devicecore 폴더내에 있는 Device_core.c와 빌드를 위한 SConscript 파일이 수정되었으며 전체 빌드를 위한 Tools 폴더 안의 Machine.py 파일이 수정되었다. ARM11의 경우에는 기존에 지원하는 코어에 ARM1176이 없었고 많은 부분들이 미구현 상태였기 때문에 SWI를 사용하는 시스템 호출 부분도 새로 작성하여야 한다.

이식의 확인을 위하여 OKLab에서 제공하는 dining_philosopers, multithread 두 개의 프로그램을 실행하였다. 그림 2는 이중 multithread 예제를 실행하는 화면을 캡쳐한 것이다.

<표 2. 이식을 위한 수정 소스 프로그램>

폴더	파일	설명
Include	offsets.h	장치별 메모리 주소
	platform.h	플랫폼에 필요한 선언
	cache.h	cache관련 설정
	interrupt.h	인터럽트 관련 설정
	intctrl.h	인터럽트 콘트롤 설정
	timer.h	타이머 관련 설정
Kdb	console.cc	Uart 설정 및 콘트롤
src	head.spp	커널의 시작부분
	plat.cc	디바이스 주소 맵핑
	interrupt.cc	인터럽트 관련 정의
	reboot.cc	Watchdog timer 정의
	timer.cc	Timer 정의



<그림 2. multithread 실행 화면1>

3.2 절 가상화 리눅스 성능

이식된 OKL4 2.1상에 부분 가상화 리눅스인 OKLinux 2.6.23을 설치하고 OK44의 유용성을 확인하기 위하여 성능을 순수 리눅스(Linux 2.6.16)와 비교하였다. 성능 비교는 LMBench[5]를 사용하여 수행하였다. 표 2는 2, 8, 16개 프로세스의 콘텍스트 스위칭 시간에 대한 비교표이다.

<표 3. 콘텍스트 스위칭 비교>

	2P	8P	16P
Linux	2848.4	8932.9	16929.8
OKLinux	539.6	622.2	730.3

표의 값은 마이크로초 단위이며 OKLinux가 프로세스 별로 5,3, 14,4, 23,2배의 월등이 우수한 성능을 보이고 있다. 표 4는 통신 성능을 측정한 것으로 bw_pipe는 두 프로세스간 pipe를 통하여 전송되는 데이터의 대역폭이며 bw_unix는 자식 프로세스의 데이터를 pipe를 통해 읽는 성능을 측정한 것이다. 모두 MB/sec 단위를 사용한다. lat_fifo는 클라이언트/서버 프로그램 형태로 fifo를 통한 프로세스간 통신 지연을 측정하며 lat_unix는 IPC 통신 지연을 측정하며 마이크로초 단위이다.

<표 4. 통신 속도 비교>

	bw_pipe	bw_unix	lat_fifo	lat_unix
Linux	12.69	11.61	582.1	1173.7
OKLinux	13.48	14.56	596.3	530.6

대역폭은 모두 OKLinux가 우수하며 lat_unix는 2배 가까이 우수하다. lat_fifo는 기존 Linux가 근소하게 우수하다.

벤치마크 검사를 통해 마이크로커널의 단점으로 여겨지던 콘텍스트 스위칭과 IPC 통신이 최적화된 설계를 통하여 기존 Linux보다 우수할 수 있음을 확인하였다. 또한 OKLinux는 40.48MB의 메모리를 사용하며 Linux는 61.52MB를 사용하는 것으로 측정되어 메모리 사용에서도 OKLinux가 우수함을 보여주고 있다.

현재는 OKL4 2.1 상에서 uC/OS-II를 부분 가상화하는 연구를 진행하고 있다.

4. 장 결론

본 논문에서는 마이크로커널의 가상화 응용에 적용하기 위한 마이크로커널 플랫폼 구축에 대하여 기술하였다. 플랫폼 구축을 위하여 ARM920T 코어를 사용하는 MBA2440 보드에 OKLab의 L4 1.5.2, 2.1, 3.0을 이식하였다. 또한 ARM1176JZF-S 를 사용하는 SMDK6410보드에 L4 2.1을 이식하였다. MBA2440보드에서 동작하는 L4 2.1에 대하여 쓰레드 생성 및 수행 실험을 통해 쓰레드 가용 수를 확인하였다. 또한 가상화 환경의 유용성을 검증하기 위하여 기존의 리눅스와 가상화 리눅스 환경을 구축하여 성능을 분석하였다. 최종적으로 MBA2440과 SMDK6410에서 동작하는 L4 2.1을 확보하고 다양한 응용 시스템에 적용 가능한 마이크로커널 및 마이크로커널 응용 개발 환경이 구축되었다.

확보한 SMDK6410에서 동작하는 L4 2.1 마이크로커널은 ARM11 기반 시스템을 위한 마이크로커널 제품 개발 및 연구에 활용될 수 있다. 또한 L4 마이크로커널의 성능 및 신뢰성에 대한 다양한 검사 환경으로 사용할 수 있다. 추후, 독자적인 국내 마이크로커널을 개발한다면 참조 모델 환경으로 유용하게 사용될 것이다.

참 고 문 헌

[1] Mach 커널, http://en.wikipedia.org/wiki/Mach_kernel
[2] QNX 커널, http://www.qnx.co.kr
[3] Jochen Liedtke, "On u-Kernel Construction", Proc. of the 15th ACM Sym. on Operating System Principles, pp. 237-250, Dec., 1995.
[4] 유시환, 유혁, "실시간 내장형 시스템 가상화 연구 동향", 정보과학회지, 제26권, 제10호, pp.41-49, 10월, 2008.
[5] LMbench, http://lmbench.sourceforge.net