임베디드 시스템을 위한 하드웨어 가속기 가상화

(Accelerator Virtualization for Embedded Systems)

홍 철 호, 유 혁*

(Cheol-Ho Hong, Chuck Yoo)

Abstract : In this paper, we suggest the complete design and architecture of hardware accelerator virtualization. It is difficult to incorporate the hardware accelerator of MPSoC with virtualization environment, because hardware accelerator is often used by application directly without request to the device driver of operating system. If different kinds of operating systems on virtual machine monitor use one accelerator, this situation causes serious synchronization problem between operating systems and a accelerator. We provide the hardware accelerator virtualization models for embedded operating system using device driver for accelerator and for operating system without using it. Furthermore, we suggest the low latency communication technique between virtualized accelerator and CPU for efficient hardware accelerator virtualization.

Keywords : hardware accelerator, 하드웨어 가속기, virtualization, 가상화

I. 서 론

임베디드 환경에서 자원을 효율적으로 관리하고 보안성을 높이기 위하여 최근 가상 머신 플랫폼이 도입되고 있는 추세이다. 가상 머신 기술은 하나의 물리 머신위에서 복수개의 운영체제를 돌릴 수 있도록 가상화 레이어 또는 가상 머신 모니터(VMM)를 제공하며, 가상화 레이어 위에서 돌아가는 운영체제는 자신이 모든 하드웨어 자원을 독점하고 있다는 환상을 갖고 서비스를 제공하게 된다.

오랜 역사를 가진 가상화 기술은 높아진 시스템 성능을 최대한 활용하기 위해 최근에 다시 주목받고 있다. 현재 가상 머신에 대한 연구는 서버와 임베디드 분야에서 모두 활발히 이루어지고 있다.

서버 분야에서는 여러 대의 물리서버를 하나의 물리 서버로 통합하여 자원의 이용을 최대화하기 위해 가상화 기술이 사용되고 있다.

임베디드 분야에서는 복수개의 운영체제 탑재와 보안을 목적으로 가상화 플랫폼이 적용되고 있다. 가상화 플랫폼을 이용하면 임베디드 리눅스, Windows CE, Nucleus 등의 임베디드 운영체제가 하나의 물리 머신 위에서 돌아갈 있게 되어 각각의 운영체제에 특화된 소프트웨어를 그대로 사용할 수 있다. 한편 가상 머신 구조 환경에서는 운영체제 간에 철저한 독립(isolation)이 제공되므로 하나의 운영체제가 공격을 당하더라도 다른 운영체제는 영향을 받지 않아 보안성이 제공된다. 이를 이용한 가상화 솔루션으로 Secure Xen[1]이 있다.

위와 같은 이유로 임베디드 분야에서 가상화 기술이 필요함에도 불구하고 임베디드 시스템의 특성 때문에 가상화 기술을 적용하기 어려운 부분이 있다. 대표적인 예가 최근의 MPSoC(Multi Processor System on Chip) 구조에 통합 되고 있는 하드웨어 가속기(hardware accelerator) 부분이다.

대부분의 임베디드 운영체제에서는 하드웨어 가속기에 대한 디바이스 드라이버를 제공하지 않아 애플리케이션이 직접 물리 I/O 주소를 접근하여 가속기를 사용하고 있다. 운영체제를 거치지 않고 하드웨어 가속기를 접근하게 되면 가상 머신 레이어의 멀티플렉싱 모듈을 우회하기 때문에 복수 개의 운영체제에서 동시에 하드웨어 가속기를 사용하려고 할 때 충돌을 일으키게 된다. 예를 들어 애플리케이션 단에서 가속기를 이용하는 Nucleus와 디바이스 드라이버로 가속기를 사용하는 리눅스를 한 가상 머신 레이어 위에 올리게 되면 하드웨어 가속

* 교신저자(Corresponding Author)

홍철호, 유혁 : 고려대학교 컴퓨터전파통신 공학과

기에 대한 동기화를 제공할 수 없게 된다.

본 논문에서는 이러한 문제를 해결하기 위해 운영체제의 디바이스 드라이버를 거치는 가속기 접근 모델과 우회하는 모델을 동시에 지원하는 가상화 구조를 제안하였으며, 더 나아가 스크래치패드 메모리를 이용하여 주 프로세서와 가상화된 가속기 간에 통신 오버헤드를 줄이는 연구를 수행하여 통합적인 가속기 가상화 플랫폼을 설계하는 방법을 제시하였다.

본 논문의 구성은 다음과 같다. 2장에서 하드웨어 가속기 구조와 수행과정을, 3장에서 하드웨어 가속기 가상화 구조에 대해 기술한다. 4장에서 통신 오버헤드 감소 기법에 대해, 5장에서 구현에 대해 기술한다. 마지막으로 6장에서 결론을 맺는다.

Ⅱ. 하드웨어 가속기 구조와 수행과정

1. 하드웨어 가속기 구조

하드웨어 가속기는 애플리케이션의 내의 SIMD (Single Instruction Multiple Data)의 속성을 가지는 멀티미디어 코드 및 데이터의 실행을 도와주기 위해 MPSoC 내에 포함된다. 가속기에 의해 실행될 수 있는 멀티미디어 포맷으로는 H.264 등이 있다.

하드웨어 가속기는 몇 가지 점에서 기존의 프로세서와는 다른 구조를 갖고 있다. 우선 실행 컨텍스트라는 개념이 없다. 프로그래머가 직접 실행할 명령어와 참조할 데이터를 알려주어 제어해야만 하고 컨텍스트의 저장 또한 외부에서 이루어져야 한다. 또한 내부에 MMU가 없어 가상 메모리를 사용할 수 없어서 주 프로세서에서 동작하는 운영체제의 시스템 서비스를 바로 사용할 수 없다. 이러한 면을 고려한다면 하드웨어 가속기를 프로세서보다는 특수한 디바이스 장치 또는 보조 프로세서로 생각할 수 있다.

하드웨어 가속기 내부에는 가속기 데이터 작업을 위한 로컬 램이 있고 가속기의 로컬 램과 시스템 메모리 간의 데이터 복사에는 DMA가 이용된다.

2. 하드웨어 가속기 수행과정

운영체제의 디바이스 드라이버를 거치는 하드웨어 가속기 접근 모델은 다음과 같은 과정을 통해 하드웨어 가속기를 이용한다. 우선 애플리케이션의 유저 버퍼에서 운영체제 커널 내 버퍼로 데이터를 복사하게 된다. 다음 DMA를 위해 캐쉬를 플러쉬(flush)를 하게 되며 DMA를 이용하여 가속기의 로컬 메모리로 데이터를 복사한다. DMA에 의한 전송이 끝나면 가속기가 동작하여 멀티미디어 데이터를 처리하게 되며 결과를 다시 로컬 램에 기록하게 된다. DMA는 다시 커널 메모리로 이 데이터를 복사하게 되며 운영체제는 커널 버퍼에서 유저 버퍼로 복사하여 모든 수행을 마치게 된다.

Ⅲ. 하드웨어 가속기 가상화 구조

1. 하드웨어 가속기 가상화

하드에어 가속기를 가상화하기 위해서는 운영체제의 디바이스 드라이버를 통해 가속기를 접근하는 모델과 운영체제를 우회하는(bypass) 모델을 구분하여 가상화 플랫폼을 설계하여야 한다. 한편 본 논문에서 다루고 있는 가상화 플랫폼은 운영체제의 소스를 수정할 수 있는 반 가상화(para virtualization) 기반 방식이다[2]. 반가상화 방식을 사용하면 전체 가상화(full virtualization) 방식 보다 최적화를 통해 성능을 높일 수 있다.

2. 운영체제를 거치는 모델

운영체제의 디바이스 드라이버를 통해 가속기에 접근하려는 모델의 가상화는 이미 검증된 Xen[2] 가상 머신 모니터의 I/O 가상화 방식을 따라 설계하였다.

Xen의 경우 각각의 운영체제는 프론트 엔드(front end) 드라이버를 통해 디바이스에 대한 처리를 드라이버를 전담하는 운영체제(드라이버 도메인)에 요청하게 되며 드라이버 도메인은 이를 받아 실제 디바이스에 작업을 지시하게 된다. 드라이버 도메인은 여러 운영체제의 요청을 멀티플렉싱하는 기능도 담당하고 있다.

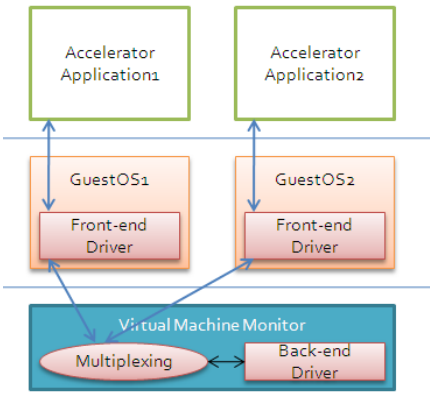

그림 1. 운영체제를 거치는 모델

그림 1은 본 논문에서 제시하고 있는 운영체제를 거치는 가상화 모델을 설명하고 있다. 운영체제에는 가속기에 대한 요청을 수행하는 프론트 엔드 드라이버를 갖고 있다. 가상 머신 모니터에는 백 엔드 드라이버가 있어 프론트 엔드 드라이버부터의 요청을 분석하여 실제 가속기에 명령을 전달할 수 있다. Xen과 달리 이 모델은 실제 디바이스 드라이버를 드라이버 도메인이 아닌 가상 머신 모니터가 갖고 있는 구조이다. 여러 게스트 운영체제에서 하드웨어 가속기에 대한 요청을 할 수 있으므로 백 엔드 드라이버에 요청을 보내기 전에 멀티플렉싱 모듈을 두어 요청을 분배할 수 있게 구성하였다.

3. 운영체제를 우회하는 모델

운영체제를 우회하는 모델은 운영체제 내부에 가속기에 대한 디바이스 드라이버가 없으며 대신 가속기를 지원하기 위해 별도의 라이브러리가 사용된다. 소형 멀티미디어 디바이스를 위한 실시간 운영체제와 가속기 라이브러리의 조합이 이러한 모델의 대표적인 예이다. 애플리케이션은 가속기 라이브러리를 이용하여 프로그래밍을 하게 되며 운영체제를 거치지 않고 바로 가속기에 접근하여 데이터를 인코딩/디코딩하게 된다.

운영체제를 우회하는 모델은 하드웨어 가속기를 중앙 집중적으로 관리하기 어려우므로 가상화하기가 어렵다. 따라서 본 연구에서는 다음과 같은 방법으로 운영체제를 우회하는 모델을 가상화하려고 한다.

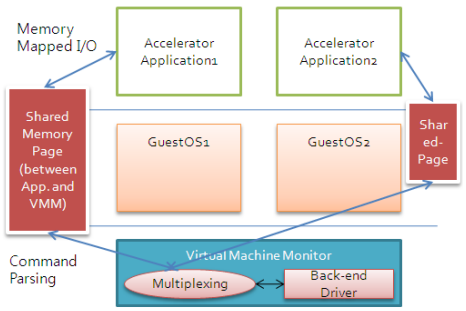

그림 2. 운영체제를 우회하는 모델

그림 2는 운영체제를 우회하는 모델의 가상화를 설명하고 있다. 운영체제를 우회하는 모델에서는 애플리케이션이 메모리 매핑 된 I/O(memory mapped I/O)를 이용하여 물리 주소에 직접 접근함으로써 가속기에 명령을 내릴 수 있다. 이때 가상화를 위해, 하드웨어 가속기와 매핑된 애플리케이션 주소 영역에 가상머신 모니터와 공유된 실제 물리 페이지를 매핑 하여 가속기에 대한 명령이 공유된 물리 페이지에 기록되게 한다. 이러한 방식을 사용하면 각 애플리케이션에서는 하드웨어 가속기를 바로 사용한다고 믿겠지만 애플리케이션의 요청이 하드웨어로 직접 전달되는 것이 아니라 가상 머신 모니터와 매핑된 공유 메모리에 기록되게 되어 가상 머신 모니터가 가속기에 대한 요청을 관리할 수 있게 된다.

가상 머신 모니터에서는 각 애플리케이션과 공유된 물리 페이지를 확인하여 새로운 기록이 남아 있으면 요청을 파싱하여 멀티플렉싱 할 수 있게 된다.

4. 가속기의 스케줄링시 고려해야 할 사항

본 논문에서 제시하고 있는 멀티플렉싱 모듈에서는 실제 물리 가속기의 할당과 스케줄링을 담당하고 있다.

각 운영체제의 가상화된 가속기 요청을 처리할 때 우선선위 변경이 일어나지 않기 위해서는 각 운영체제의 우선순위를 가속기의 스케줄링에 반영하여야 한다.

또한 레이턴시를 낮추기 위해, 멀티플렉싱 모듈은 하나의 가속기 작업이 끝난 경우 가상 머신 모니터 내의 도메인 스케줄러에 이를 알려주어, 가속기의 결과를 기다리고 있는 도메인을 바로 스케줄링할 수 있는 부스트(boost) 기능을 갖추어야 한다.

IV. 통신 오버헤드 감소

가속기를 이용하려고 할 때 유저 버퍼에서 가상 머신 모니터의 버퍼로 또는, 가상 머신 모니터의 버퍼에서 유저 버퍼로 데이터를 복사하는 시간은 큰 오버헤드로 작용한다.

따라서 본 논문에서는 최신 ARM 프로세서에 내장되어 있는 TCM(Tightly Coupled Memory) 스크래치패드 메모리를 가상 머신 모니터의 버퍼로 사용하여 통신 오버헤드를 줄이려고 한다.

TCM은 고속으로 동작하는 레벨1 메모리 시스템이며 로컬 램으로 사용하거나 스마트 캐쉬로 사용할 수 있다. TCM을 로컬 램으로 사용할 경우 자체 DMA를 통하여 시스템 램과 TCM 사이, TCM과 가속기 사이에 데이터를 교환할 수 있다.

레벨1 메모리인 TCM을 가상 머신 모니터의 버퍼로 사용한다면 CPU-가상화된 가속기간 통신 방식은 다음과 같이 바뀌게 된다.

1. 캐쉬 flush
2. 유저 버퍼에서 TCM 자체 DMA를 사용해서 TCM 버퍼로 데이터 복사
3. TCM자체 DMA를 이용하여 가속기의 로컬 메모리로 데이터 복사
4. 하드웨어 가속기 수행
5. 가속기 수행 결과를 TCM 자체 DMA를 이용하여 TCM 버퍼로 복사
6. TCM 버퍼에서 TCM자체 DMA를 사용해서 유저 버퍼로 결과 데이터 복사

이러한 과정을 거치게 되면 주 메모리 대신 낮은 레이턴시를 가지는 TCM을 이용하므로 성능향상을 이룰 수 있을 뿐만 아니라 유저-가상화 버퍼 데이터 복사에 TCM의 DMA를 이용하므로 CPU가 복사에 직접 관여하지 않아 시스템 버스를 이용하는 빈도를 줄일 수 있다. 또한 TCM의 내용은 L1 캐쉬에 기록되지 않으므로 가속기에 사용되는 데이터에 의해 캐쉬가 오염되는 것을 방지할 수 있다.

V. 구 현

본 논문에서 제시하고 있는 하드웨어 가속기 가상화 플랫폼은 현재 구현이 진행되고 있는 중이다. 대상 플랫폼은 삼성의 S3C6400 프로세서가 탑재되어 있는 한백전자의 EMPOS-III모델이며 S3C6400 프로세서[3] 내에는 미디어 가속기와 TCM이 포함되어 있어 논문의 하드웨어 요구사항을 만족시키고 있다.

가상화 플랫폼은 자체 개발한 MobiVMM[4]을 사용하고 있으며 가상 머신 모니터 위에 리눅스 두 개를 올려 실험 환경을 구축하였다.

VI. 결 론

본 논문에서는 효율적이고 통합적인 가속기 가상화 플랫폼을 제시하고 있다. 이를 위해 운영체제의 디바이스 드라이버를 거치는 가속기 접근 모델과 우회하는 모델을 동시에 지원하는 가상화 구조를 제안하였다. 또한 주 프로세서와 가상화된 가속기 간에 통신 오버헤드를 줄일 수 있는 방법을 제시 하였다.

참 고 문 헌

[1] Sang-bum Suh, “Secure Architecture and Implementation of Xen on ARM for Mobile Devices”, Presented at Xen Summit Spring 2007, IBM TJ Watson.

[2] Boris Dragovic, Keir Fraser, Steve Hand, Tim Harris, Alex Ho, Ian Pratt, Andrew Warfield, Paul Barham, and Rolf Neugebauer, “Xen and the Art of Virtualization”, Proceedings of the ACM Symposium on Operating Systems Principles 2003

[3] http://www.samsung.com/global/business/semiconductor/productInfo.do?fmly_id=229&partnum=S3C6400

[4] See-hwan Yoo, Yunxin Liu, Cheol-Ho Hong, Chuck Yoo, Yongguang Zhang, "MobiVMM: A Virtual Machine Monitor for Mobile Phones," International Workshop on Mobile Computing Virtualization(MobiVirt 2008) in colocated with USENIX/ACM MobiSys, Breckenridge, CO., June, 2008.

저 자 소 개

홍 철 호(Cheol-Ho Hong)
2001년 2월 : 고려대학교 컴퓨터학과 학사
2003년 2월 : 고려대학교 컴퓨터학과 석사
2007년~현재 : 고려대학교 컴퓨터학과 박사과정

관심분야 : 가상 머신 플랫폼, 멀티 코어 운영체제
Email : chhong@os.korea.ac.kr

유 혁(Chuck Yoo)
1982년 2월 : 서울대학교 전자공학과 학사
1983년 2월 : 서울대학교 전자공학과 석사
1986년 8월 : Master of Computer Science in University of Michigan
1990년 8월 : Ph.D of Computer Science in University of Michigan
1990년~1995년 : Sun Microsystems Lab. Researcher
1995년~현재 : 고려대학교 컴퓨터학과 교수

관심분야 : 시스템 가상화, 멀티 코어 플랫폼, 커널 네트워킹, 센서 네트워킹, 멀티미디어 스트리밍
Email : hxy@os.korea.ac.kr