Hanbit eBook

Realtime 23

Xen으로 배우는

가상화 기술의 이해

I/O 가상화

박은병, 김태훈, 이상철, 문대혁 지음

한빛미디어 Hanbit Media, Inc.

Xen으로 배우는

가상화 기술의 이해
I/O 가상화

Xen으로 배우는 **가상화 기술의 이해 – I/O 가상화**

초판발행 2013년 3월 29일

지은이 박은병, 김태훈, 이상철, 문대혁 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-606-7 15000 / **정가** 9,900원

책임편집 배용석 / **기획** 김창수 / **편집** 이중민, 이세진
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 박진희
마케팅 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지은이_ **박은병**

서울대학교에서 석사 학위를 받았으며, 현재 University of Toronto에서 컴퓨터 공학 박사 과정을 공부하고 있다. 석사 과정을 공부하면서 Xen을 이용해 가상화 관련 연구를 진행했다. 시스템 소프트웨어 전반에 관심이 있으며, 현재 기계학습 관련 응용 분야에 흥미를 느껴 공부 중이다.

지은이_ **김태훈**

임베디드, 커널, 가상화, 네트워크, 디바이스 드라이버를 주로 다루는 시스템 프로그래머다. (주)WIZnet 재직 시절에 개발한 W5300 네트워크 드라이버가 리눅스 커널에 포함되었다. 오픈 소스와 해커 문화를 동경하며, 특히 리누스 토발즈가 우상이다. 현재는 DINOS라는 고성능 ARM 아키텍처를 타깃으로 하는 운영체제를 개발 중이다.

지은이_ **이상철**

하드웨어 개발부터 시작해 시스템 소프트웨어 개발로 차츰 업무를 변경해왔다. 주로 임베디드 시스템 프로그램과 디바이스 드라이버를 개발했으며, 리눅스 커널 관련 업무 또한 담당했다. 현재는 알티캐스트에서 보안 관련 모듈을 개발 중이다.

지은이_ **문대혁**

한양대학교를 휴학하고 사이냅소프트에서 문서 처리 관련 프로그램을 개발 중이다. 운영체제나 하이퍼바이저와 같은 시스템 소프트웨어에 관심이 많다. 우연히 본 스터디 모집공고를 계기로 뛰어난 개발자들과 함께 Xen을 분석하는 기회를 가지게 되었다.

저자 서문

이 책은 Xen 하이퍼바이저를 통해 가상 머신 모니터를 설명한 책입니다. Iamroot 라는 스터디 그룹을 통해 1년 반 정도 같이 매주 토요일에 오프라인 모임에서 Xen 하이퍼바이저를 공부했고, 다른 저자분들의 뜻을 모아 공부한 내용을 정리하자는 의도에서 기획했습니다. 저자분들 모두 글 쓰는 데 익숙하지 않은 관계로 아는 지식을 글로 풀어내기가 쉽지 않았지만, 많은 퇴고를 거쳐 마침내 출판하기에 이르게 되었습니다. 이 책이 출판으로 이어지기까지 고생해주신 저자 분들과 한빛미디어 관계자분께 다시 한번 감사의 말씀을 드립니다.

이 책은 기본적인 가상 머신 모니터의 원리부터 Xen 하이퍼바이저의 소스 코드까지 설명해, 초보자부터 실제 Xen 하이퍼바이저를 사용하거나 개발하시는 분 모두에게 유용한 내용으로 구성했습니다. 운영체제와 컴퓨터 아키텍처에 대한 기본 지식만 있으면 읽을 수 있게 쓰였으나, 리눅스 커널에 대한 사전 지식이 있으면 이해하기 더욱 쉬울 것으로 생각합니다. CPU 가상화, 메모리 가상화, I/O 가상화 이렇게 크게 3가지 파트로 나누어서 설명했고, 이 책의 주제인 I/O 가상화는 시리즈 세 번째 책입니다. Xen 하이퍼바이저에 관심이 많다면 CPU 가상화, 메모리 가상화, I/O 가상화 순으로 읽기를 권장합니다. 전체적인 구조는 각 파트의 앞부분에 기본적인 개념을 설명하고, 뒤의 소스 코드 분석 부분에서 어떻게 실제로 구현되어 있는가를 소스 코드 라인 별로 상세히 설명했습니다. 또한 Xen 하이퍼바이저는 리눅스 커널과 매우 밀접한 관련이 있으므로, 필요한 부분에는 리눅스 커널의 소스 코드도 추가로 설명했는데, 모든 소스 코드를 분석하고 책에 담을 수 없었기에 저자가 생각하는 중요한 부분의 소스 코드를 담았습니다. 따라서 앞부분 개념을 먼저 이해하고, 실제 소스 코드를 내려받아 책의 설명과 함께 책에 빠진 부분이나 궁금한 부분의 소스 코드를 직접 찾아가면서 공부할 것을 권장합니다.

사실 책으로 출판하기에 매우 부끄럽습니다. 하지만 가상 머신 모니터를 쉽게 한글로 설명한 책이 많지 않고, 가상 머신 모니터 공부를 시작하시는 분의 수고를 조금이나마 덜어드리고자 하는 마음에 부끄러움을 무릅쓰고 출판하기로 했습니다. 너그러운 자세로 읽어주시고, 수정이나 보완 사항이 필요하다면 추후에 반영토록 하겠습니다.

집필을 마치며

저자 일동

대상 독자 및 도서 구성

초급 | 초중급 | 중급 | **중고급** | 고급

이 책은 점차 활용 분야가 늘어나는 가상화 기술의 이해를 돕기 위해 Xen을 예로 들어 가상화 기술을 설명합니다. 이 책을 통해 가상화 기술의 개념과 Xen의 동작 방식을 이해하는 데 조금이나마 도움이 되기를 바랍니다. 'Xen으로 배우는 가상화 기술의 이해' 시리즈는 가상화 기술을 다음 세 가지 파트로 나눠 설명합니다.

CPU 가상화

Xen에서 전통적으로 사용하는 반가상화 기법과 하드웨어 지원 가상화 기술을 활용하는 전가상화 기법을 다룹니다. 가상 CPU가 실제 CPU를 어떻게 나눠서 사용하는지 가상 머신 스케줄링에서 살펴봅니다.

메모리 가상화

Xen에서 동작하는 가상 머신의 메모리 접근 방식을 설명합니다. 가상화 환경에서 가상 주소를 물리 주소 변환하는 대표적인 두 가지 방식인 섀도 페이징과 CPU에서 지원하는 HAP(Hardware Assisted Paging)이 어떻게 이루어지는지 설명합니다.

I/O 가상화

가상 머신이 동작할 때 필요한 입·출력 장치에 접근하는 방법을 다룹니다. Xen에서 반가상화 입·출력 방식과 전가상화 입·출력 방식에는 많은 차이가 있습니다. 따라서 반가상화 입·출력 방식과 전가상화 입·출력 방식을 나눠서 설명합니다.

예제 테스트 환경

사용 프로그램	버전
리눅스 커널	3.6
Xen 하이퍼바이저	4.1.2

- 리눅스 커널 코드 참고 사이트

  : http://goo.gl/jmbPA

- 리눅스 커널 코드 다운로드

  : ftp://kernel.org/pub/linux/kernel/v3.x/linux-3.6.tar.gz

- Xen 코드 참고 사이트

  : http://goo.gl/PwfEA

- Xen 소스 코드 다운로드

  : http://bits.xensource.com/oss-xen/release/4.1.2/xen-4.1.2.tar.gz

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook 입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

01 가상 머신 모니터는 무엇인가? 1

1.1 왜 가상화인가? 2
1.2 하이퍼바이저 종류 4

02 I/O 가상화 6

2.1 디바이스 에뮬레이션 6
2.2 반가상화 인터페이스 7
2.3 분리 드라이버 모델 8
2.4 직접 접근 I/O 10
2.5 IOMMU 12
2.6 SR-IOV 13

03 반가상화 I/O 15

3.1 리눅스의 블록 디바이스 I/O 17
3.2 I/O Ring 21
3.3 이벤트 채널 33
3.4 XenStore & XenBus 36
3.5 그랜트 테이블 43
3.6 반가상화 I/O 정리 46

04 **전가상화 I/O** 47

4.1 QEMU란? 47
4.2 Xen에서의 QEMU 이용 49

05 **하드웨어 지원** 63

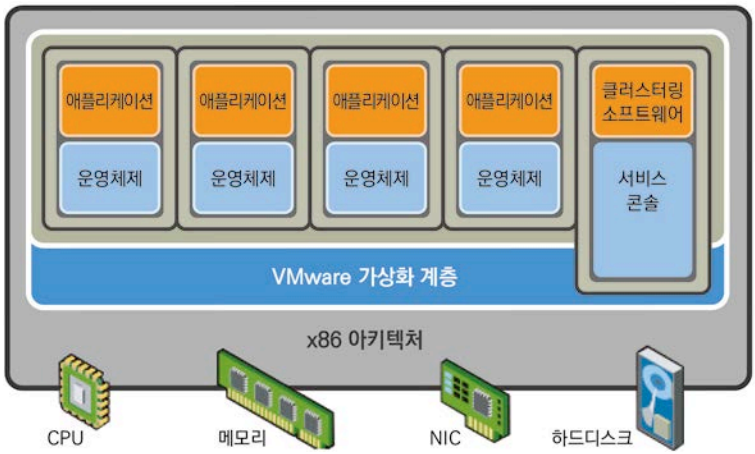
5.1 IOMMU 65
5.2 PCI passthrough I/O 71

1 | 가상 머신 모니터는 무엇인가?

Xen 내부 동작을 알아보기 전에는 가상 머신 모니터 혹은 하이퍼바이저Hypervisor[01]에 대한 개념과 I/O 가상화에 대한 전반적인 내용을 알아야 한다. 1장과 2장은 I/O 가상화 전체를 그려볼 기회이므로 반드시 이해하자.

가상 머신 모니터Virtual Machine Monitor란 글자 그대로 가상 머신을 모니터링하는 소프트웨어 계층을 의미한다. 물론 단순히 모니터링만 하는 것이 아니라 하드웨어 자원 관리, 가상 머신 스케줄링 등 가상 머신을 동작시키는 데 필요한 모든 작업을 담당한다.

그림 1-1 가상 머신 모니터(출처: www.vmware.com)



01 본 책에서는 가상 머신 모니터라는 용어와 하이퍼바이저라는 용어를 섞어서 사용한다. 서로 같은 것을 부르는 말이니 혼동이 없길 바란다.

여기서 말하는 가상 머신이란 실제 머신은 아니지만 마치 물리 머신이 있는 것 같은 환경을 사용자에게 제공하는 가상화한 머신 환경을 의미한다.[02] 좀 더 쉽게 설명하면 하나의 컴퓨터에 여러 개의 운영체제를 동시에 구동할 수 있게 하는 소프트웨어라고 할 수 있다. 즉, 하나의 컴퓨터에서 가상 머신 모니터는 다수의 가상 머신을 사용자에게 제공하고, 사용자는 하나의 물리 머신 위에 다수의 가상 머신을 가질 수 있게 된다.

1.1 왜 가상화인가?

최근 들어 이곳저곳에서 가상화에 대한 논의가 뜨겁다. 실제로 가상화 기술은 이미 성숙기에 이르렀으며, 많은 기업에서 도입했거나 도입을 고려하는 상황이다. 사실 가상화 기술은 최신 기술이 아니라 1960년대 IBM의 메인프레임에서 처음 구현되었다. 하지만 당시에는 큰 이목을 끌지 못하다가 최근 10여 년 전부터 하드웨어의 발전과 함께 재조명받기 시작했다.

처음 가상화 기술이 재조명받기 시작했을 때, 기업의 가상화 기술 도입 동기는 바로 서버 통합Server Consolidation이었다. 이는 물리 서버 머신 하나가 다수의 CPU와 대량의 메모리를 갖출 수 있게 되면서 많은 자원이 유휴 상태로 남는 경우가 많아, 이렇게 유휴한 상태의 서버를 머신 하나로 통합 관리하자는 요구가 생겼다는 뜻이다. 이때 가상화 기술을 이용하면 물리 서버 하나를 가상 머신 하나로 대체하고, 이 가상 머신 여러 대를 강력한 성능을 가진 물리 머신 하나에서 동작시켜 시스템 자원의 효율성을 극대화할 수 있다.

서버 통합과 더불어 가상화 기술이 가진 또 하나의 장점은 가상 머신의 격리Isolation다. 사실 서버 통합을 하는데 가상화 기술이 반드시 필요한 것은 아니다. 예를 들어

02 JVM(Java Virtual Machine)과 같은 가상 머신을 떠올리는 독자도 있을 것이다. 이 책에서는 JVM과 같이 프로세스 레벨의 가상화가 아닌 시스템 레벨의 가상 머신에 대해 다룬다.

웹 서버, 데이터베이스 서버, DNS 서버 등 여러 서버를 하나의 머신 위에 동작시켜도 된다. 하지만 운영체제 위에서 여러 개의 서버 프로세스가 동작하게 되므로 서버 프로세스 사이에 많은 영향을 미친다는 치명적인 문제점이 있다.

또 다른 예로 자신이 홈페이지 하나를 운영하고 있고, 호스팅 업체에 웹 호스팅을 요청했다고 가정하자. 홈페이지가 다른 고객과 서버를 함께 사용해서 성능에 영향을 미친다면 당신은 해당 업체에 호스팅을 의뢰하고 싶지 않을 것이다. 가상화는 각 고객에게 독립된 가상 머신 하나를 제공하며, 가상 머신 사이에서 완벽하게 격리된 환경을 보장해 주므로 고객 각각의 요구사항을 만족시킬 수 있다.[03]

또 다른 장점은 관리의 용이성이다. 가상화되지 않은 환경에서 서버를 점검하거나 업그레이드한다면, 동작 중인 서버의 전원을 끄고 점검 및 업그레이드를 한 뒤 다시 서버 전원을 켜야 한다. 하지만 가상화된 환경에서는 단순히 가상 머신을 다른 물리 서버 머신으로 이주하고 해당 물리 서버를 점검하면 되므로, 서비스 중단없이 원하는 작업을 마무리할 수 있다.

이처럼 처음에는 서버 통합과 관련한 기업 요구 때문에 재조명을 받았던 가상화 기술이지만, 최근에는 클라우드 컴퓨팅이라는 거부할 수 없는 흐름에 맞춰 가상화 기술이 탄생한 이래로 가장 큰 호황을 누리고 있다. IaaSInfrastructure as a Service라는 클라우드 서비스로 분류하는 아마존 EC2Elastic Compute Cloud는 바로 지금 설명하는 가상화 기술을 기반으로 구축되어 사용자는 웹 사이트에서 단순한 클릭 몇 번만으로 수 분 안에 서버에 생성한 가상 머신을 사용할 수 있다.

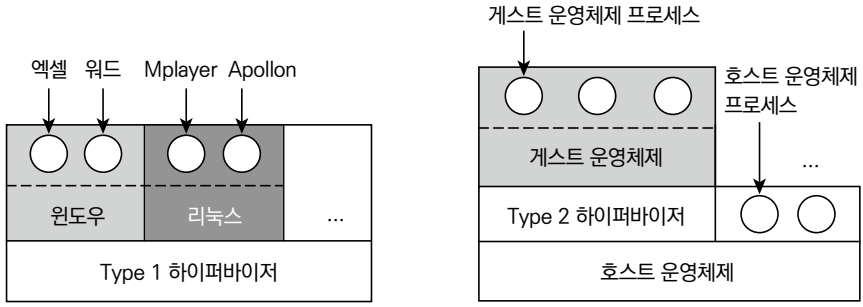
이 외에도 다양한 클라우드 서비스들이 가상화 환경 아래에서 서비스되는 추세며, 앞으로도 수많은 서비스와 애플리케이션이 등장할 것이다.

---

03 물론 운영체제 스스로 프로세스 사이의 간섭을 줄이는 방법들이 있다. 하지만 가상 머신만큼 완벽히 격리된 환경을 제공하지 못한다.

1.2 하이퍼바이저 종류

그림 1-2 Type 1과 Type 2 하이퍼바이저



가상 머신 모니터는 시스템 위치에 따라 크게 Type 1[native or bare-metal]과 Type 2[hosted]로 분류한다. Type 1 하이퍼바이저는 하드웨어 바로 위의 소프트웨어 계층으로 존재하며 그 위에 다양한 게스트 운영체제가 동작하는 방식이다. Xen이나 VMware의 서버용 하이퍼바이저 제품군 등이 여기에 해당한다. Type 1은 하드웨어 전원이 들어오면 가장 먼저 하이퍼바이저가 부팅을 시작하게 하고, 부팅을 완료하면 관리자가 가상 머신을 생성해서 여러 개의 가상 머신이 동작한다.

Type 2는 이와 조금 다르게 호스트 운영체제가 존재하고, 그 위에서 하이퍼바이저가 동작하며, 다시 그 위에 게스트 운영체제가 동작한다. 버추얼박스[VirtualBox]나 KVM, 그리고 VMware의 데스크톱을 위한 제품군인 워크스테이션[workstation] 계열이 여기에 속한다. 동작 방식도 조금 다른데 우선 호스트 운영체제를 가장 먼저 부팅해 실행하고, 그 위에서 사용자가 하이퍼바이저를 실행시킨다. 그런 다음 실행된 하이퍼바이저가 여러 개의 가상 머신을 생성하고 동작하게 한다.

어떤 종류의 하이퍼바이저가 좀 더 좋고 효율적인지는 여러 가지 이견들이 있으니 관심 있는 독자는 개별적으로 찾아보기 바란다.

이미 언급한 바와 같이 Xen 하이퍼바이저는 Type 1에 해당하며 다른 여러 하이퍼바이저와 비교해 완성도가 매우 높고 성능도 뛰어나다고 알려졌다. 또한 아마존 웹 서비스Amazon Web Service에서 Xen을 기본 하이퍼바이저로 채택해 사용하는 만큼, 안정성도 이미 검증되었다고 볼 수 있다.

2 | I/O 가상화

I/O(입출력)는 CPU, 메모리와 더불어 컴퓨터가 어떤 의미 있는 작업을 하기 위한 필수 요소의 하나다. 키보드나 마우스 등으로 이루어지는 사람과의 통신, 네트워크로 이루어지는 다른 컴퓨터와의 통신, 그리고 디스크, 그래픽 카드 등과 같은 디바이스와의 통신은 컴퓨터가 어떤 작업을 할 때 반드시 필요하다.

CPU와 메모리 그리고 디바이스 사이의 실행 속도 차이 때문에 운영체제에서 I/O를 어떻게 처리하느냐는 시스템의 성능과 매우 밀접한 연관이 있다. 마찬가지로 가상화 환경에서 I/O를 어떻게 처리하느냐도 가상화 시스템의 전체 성능을 결정하는 중요한 문제다. 따라서 I/O 가상화를 이해하는 것은 전체 가상화 환경을 이해하는 데 매우 중요하다. 2장에서는 I/O 가상화의 기본 개념, Xen에서 사용하는 방법, 그리고 효율적인 I/O 가상화에 필요한 하드웨어 지원의 전반적인 내용을 소개한다.

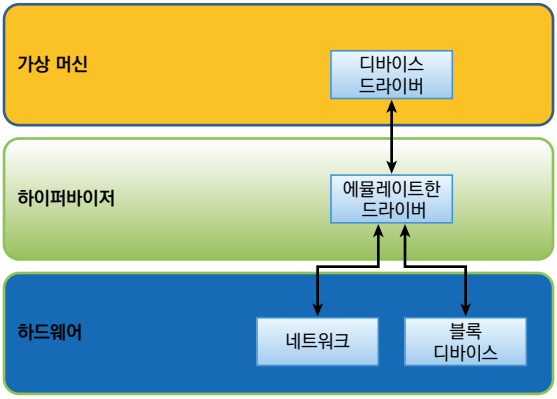
2.1 디바이스 에뮬레이션

I/O를 가상화하는 가장 직관적인 방법은 디바이스의 기능을 소프트웨어적으로 똑같이 구현하는 에뮬레이션 방식이다. 하이퍼바이저는 게스트 도메인에 실제로 디바이스가 존재하는 것처럼 설정하고, 게스트 도메인은 실제 디바이스에서 동작하는 것과 똑같이 동작하게 된다.

에뮬레이션 방식의 가장 큰 장점은 가상의 디바이스를 제공함으로써 게스트 도메인의 드라이버를 수정 없이 그대로 쓸 수 있다는 점이다. 또한 잘 알려진 디바이스를 에뮬레이션하면 게스트 운영체제 대부분이 해당 디바이스의 드라이버를 사용할 수 있으므로 다양한 게스트 운영체제를 지원할 수 있다. 그러나 디바이스 에뮬레이션 방식은 에뮬레이션 방식의 전형적인 단점을 그대로 가진다. 디바이스를 소프트

웨어로 모두 구현해야 하므로 구현이 복잡하고, 하드웨어를 소프트웨어로 동작시켜야 하므로 오버헤드가 커서 성능이 저하된다. 예를 들면 모든 저 수준의 I/O 명령(ex. x86 아키텍처에서 in/out 명령)을 하이퍼바이저가 에뮬레이션해 주어야 하고 이때 게스트와 하이퍼바이저 사이에 컨텍스트 스위치가 항상 발생한다.

그림 2-1 디바이스 에뮬레이션



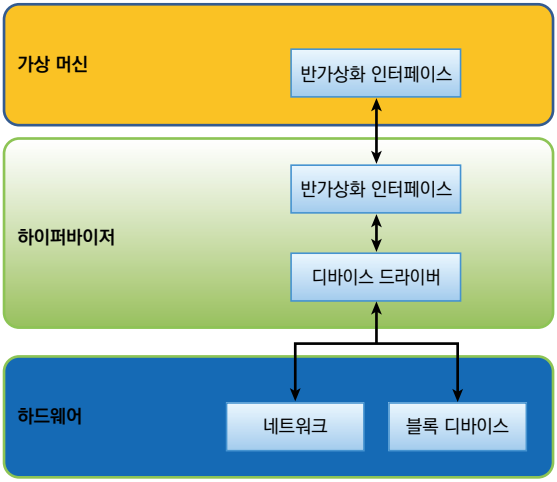
2.2 반가상화 인터페이스

게스트 운영체제를 수정할 수 있다면, 좀 더 효율적으로 I/O 가상화를 구현할 수 있다. 즉, 게스트와 하이퍼바이저 사이에 I/O 디바이스에 대한 새로운 상위 수준 인터페이스를 정의해 게스트가 직접 하이퍼바이저에 I/O 요청을 할 수 있다.

예를 들어 네트워크 디바이스를 생각해 보자. 디바이스 에뮬레이션은 모든 저 수준 I/O 명령어를 있는 그대로 에뮬레이션해야 한다. 그리고 패킷 하나를 네트워크 카드를 통해 전송하려면 많은 저 수준 I/O 명령어를 요구한다. 이는 에뮬레이션을 실행할 때의 오버헤드뿐만 아니라, 게스트와 하이퍼바이저 사이에 많은 수의 컨텍스트 스위치를 일으키며 이때 발생하는 오버헤드도 상당하다. 따라서 반가상화를 통해 게스트와 하이퍼바이저 사이의 인터페이스를 정의할 때는 좀 더 상위 수준의 인

터페이스로 정의한다. 이렇게 하면 사용자의 저 수준 I/O 명령을 send_packet과 같은 한 번의 요청으로 처리할 수 있다. 게스트와 하이퍼바이저의 컨텍스트 스위치는 단 한 번으로 충분하고, 디바이스 에뮬레이션을 실행할 필요도 없으므로 더 높은 성능의 I/O 가상화를 구현할 수 있다.

그림 2-2 반가상화 I/O 인터페이스



2.3 분리 드라이버 모델

앞서 언급한 디바이스 에뮬레이션과 반가상화 방법은 I/O 명령을 어떻게 가상화할 것인가에 대한 두 가지 방법론이다. 이와 더불어 I/O 가상화를 구현할 때 논의하는 것은 바로 디바이스 드라이버에 대한 관점이다. 일반적으로 디바이스 드라이버는 운영체제를 이루는 핵심 요소이며 현존하는 많은 디바이스를 운영체제가 지원해야 하므로, 실제로 리눅스 커널에서도 디바이스 드라이버의 소스 코드가 커널의 대부분을 차지한다.

이런 상황에서 디바이스 제어를 하이퍼바이저가 하도록 설계한다면, 디바이스 드라이버의 모든 코드를 하이퍼바이저에서 유지해야 하므로 상당히 비효율적이다. 즉, 이미 운영체제가 가진 소스 코드를 또 소유해야 하는 중복성 측면의 문제와 하이퍼바이저를 유지 보수하는 과정에서 대용량의 소스 코드를 허용하기는 쉽지 않다는 문제가 발생한다. 또한 하이퍼바이저가 디바이스 드라이버를 유지하면, 디바이스 드라이버의 오동작 때문에 에러가 발생하면 시스템 전체에 영향을 미칠 것이다.

이런 모든 단점을 극복하려고 디바이스 드라이버를 하이퍼바이저와 별도로 관리하자는 분리 드라이버 모델Split Driver Model이 등장했으며, 디바이스 드라이버로 실제로 디바이스를 제어하는 별도의 특수한 도메인을 두도록 하였다.

Xen 하이퍼바이저에서는 도메인 0(DOM 0)을 디바이스 드라이버를 전담하는 도메인으로 할당했고, 도메인 0에서 리눅스를 구동시켜 리눅스 커널에 존재하는 모든 디바이스 드라이버를 사용할 수 있다.

그림 2-3 분리 드라이버 모델

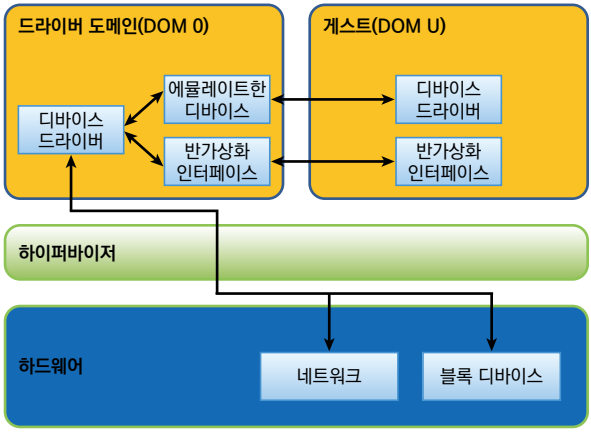
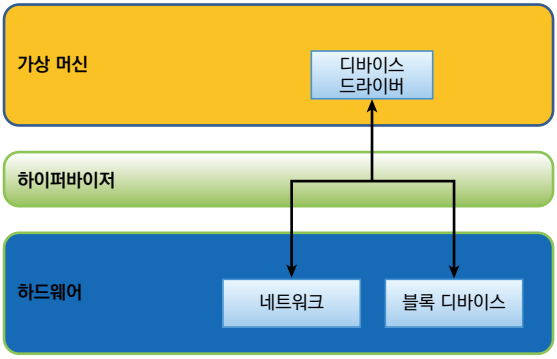


그림 2-3은 분리 드라이버 모델의 예다. 디바이스 드라이버는 하이퍼바이저가 아닌 특정 도메인에 존재하고 하이퍼바이저가 소유한 드라이버를 사용해 직접 하드웨어 디바이스에 접근한다. 게스트가 반가상화 인터페이스를 사용했다면 드라이버 도메인의 반가상화 인터페이스가 게스트로부터 I/O 요청을 받아서 디바이스 드라이버로 전달한다. 반가상화 인터페이스를 사용하지 않았다면, 드라이버 도메인에서 디바이스를 에뮬레이션해 I/O 요청을 실제 디바이스 드라이버에게 보낸다.

2.4 직접 접근 I/O

가상화 환경은 디바이스 하나를 여러 게스트 도메인이 공유하므로 하이퍼바이저의 적절한 중재가 필요하다. 그러므로 게스트가 직접 디바이스에 접근하는 것은 허용할 수 없다. 그런데 특정 디바이스를 게스트 도메인 하나만 사용하도록 허용한다면, 게스트는 하이퍼바이저의 중재 없이 직접 디바이스와 통신할 수 있을 것이다. 이렇게 게스트 운영체제가 직접 디바이스에 접근해 I/O를 요청하는 것을 직접 접근 I/O(Direct Access I/O)라고 부르며, 혹은 하이퍼바이저를 통해 직접 디바이스에 접근한다고 하여 Passthrough I/O라고도 부른다.

그림 2-4 직접 접근 I/O



직접 접근 I/O는 성능이 우수하며 실제로 가상화되지 않은 환경과 동일한 성능을 보장하기도 한다. 왜냐하면 일단 디바이스를 게스트 도메인에 할당하면 하이퍼바이저의 중재 없이도 바로 I/O를 실행하기 때문이다.

하지만 직접 접근 I/O는 메모리 보호, 전가상화 지원, 디바이스 공유라는 세 가지 측면에서 치명적인 약점이 있다.

1. **메모리 보호**: 디바이스 I/O를 직접 실행한다는 것은 직접 DMA(Direct Memory Access)[01] 연산을 실행하는 것을 의미한다. 디바이스는 DMA 연산으로 직접 시스템 메모리의 데이터를 읽고 쓸 수 있다. 이때 어떤 게스트에 디바이스를 할당했는데, DMA를 통해 악의적인 게스트가 임의로 다른 게스트의 메모리 영역에 쓰기 연산을 실행하면 해당 게스트는 오작동이 일어날 것이다.
2. **전가상화 지원**: DMA 명령을 내리려면 게스트 운영체제가 머신 주소를 알아야 한다. 왜냐하면 DMA는 직접 시스템 메모리에 접근할 때 머신 주소를 입력으로 받아 DMA를 실행하기 때문이다. 그러나 전가상화 게스트 운영체제는 머신 주소를 알지 못하므로 올바른 DMA 요청을 실행할 수 없다.
3. **디바이스 공유**: 가상화를 하는 가장 큰 이유 하나는 효율적인 하드웨어 자원의 사용이다. 게스트 하나가 디바이스 하나만 사용한다면, 하드웨어 하나를 여러 게스트가 나누어 사용하는 가상화의 기본 원칙에 어긋날 수 있다.

위 세 가지 단점을 극복하고 직접 접근 I/O를 구축하려고, 각 하드웨어 벤더는 하드웨어적 해결책을 제공한다. 좀 더 안전한 직접 접근 I/O와 전가상화 게스트 지원을 위해서는 IOMMU(AMD) 혹은 VT-d(인텔)가 있고, 직접 접근 I/O를 하면서도 효율적으로 디바이스를 공유할 수 있도록 해결책을 제시한 SR-IOV도 있다.

---

01 DMA란 디바이스가 CPU의 중재 없이 직접 메모리에 접근하는 방식을 뜻한다. 일반적으로 메모리에 접근할 때는 CPU의 메모리 read/write 명령으로만 가능한데, 시스템의 효율을 극대화하려고 CPU가 다른 작업을 하는 중에도 DMA를 통해서 디바이스는 I/O를 실행할 수 있다.

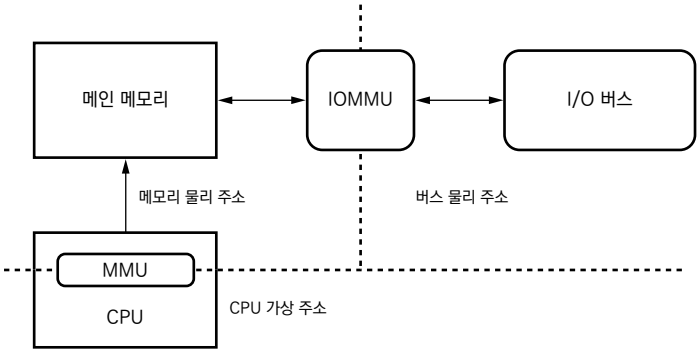
2.5 IOMMU

IOMMU[I/O Memory Management Unit]는 CPU 안의 MMU[Memory Management Unit]와 비슷하다. 즉, CPU에서 MMU가 가상 주소를 물리 주소로 변환하는 것과 비슷하게 주소 변환 기능을 I/O DMA에 적용한다. DMA 리매핑[Remapping]이라고도 불리는 이 주소 변환 기능을 이용해 전가상화 게스트 운영체제가 DMA 명령으로 물리 주소를 전달하면, 물리 주소를 머신 주소로 변환해 DMA 연산을 실행하게 된다.

또한 MMU가 페이지 레벨의 보호 기능을 제공해 승인되지 않은 메모리 영역에 함부로 쓰기를 할 수 없는 것과 같이 DMA 연산 시 승인되지 않은 메모리 영역에 쓰기를 금지한다. 이 보호 기능으로 악의적인 게스트가 다른 게스트나 하이퍼바이저 메모리를 침해하는 일을 방지할 수 있다.

그림 2-5는 시스템상에서 IOMMU의 위치를 보여준다. 그림에서 보듯이 CPU는 MMU를 통해 메모리에 접근하고, I/O 디바이스는 IOMMU를 통해 메모리에 접근할 수 있다.

그림 2-5 IOMMU(출처: http://www.linuxjournal.com/article/7104)



2.6 SR-IOV

IOMMU를 이용한 직접 접근 I/O는 게스트 도메인에서 디바이스에 직접 접근해 성능을 높이지만, 게스트 도메인 하나에만 디바이스를 할당해야 하는 단점을 극복하지 못한다.

SR-IOV(Single Root I/O Virtualization)는 이런 단점을 극복하기 위한 하드웨어 기능으로, 직접 접근 I/O에서 디바이스 공유 기능을 지원하며, 디바이스 하나를 복수 개의 가상 디바이스로 나눠 게스트 도메인에 할당할 수 있게 한다.

그림 2-6 SR-IOV를 통한 디바이스 공유

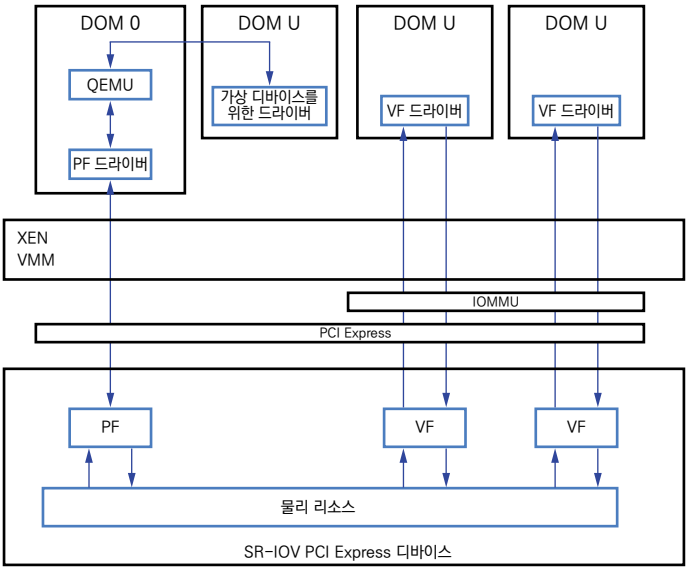


그림 2-6은 SR-IOV와 IOMMU를 통한 직접 접근 I/O의 디바이스 공유 모델을 보여준다. SR-IOV는 디바이스 하나를 PF(Physical Function)와 VF(Virtual Function)라는 두 가지 타입의 기능으로 분리한다. 따라서 디바이스 하나는 여러 개의 VF를 각 가상 머

신에 할당하게 된다. 그리고 하이퍼바이저는 PF를 제어/관리하며, 여러 개의 VF를 게스트 도메인 각각에 할당하도록 요청한다. 쉽게 말하면 디바이스 하나를 가상화해 여러 개의 인스턴스를 제공하는 것이다.

SR-IOV를 활용한 이더넷 디바이스라면 디바이스에서 스위치 역할을 할 수 있는 기능도 추가되어 있으므로 외부로부터 패킷을 수신했을 때 패킷을 적절히 각 VF에 분배하고 게스트 도메인에 전달할 수 있다. SR-IOV는 이처럼 직접 접근 I/O에서 물리 디바이스 하나를 여러 게스트 도메인에서 공유할 수 있도록 지원한다.