ISSN(Print): 2287-5905

http://dx.doi.org/10.3745/KTSDE.2013.2.2.113

Direct Pass-Through based GPU Virtualization for Biologic Applications

Dong Hoon Choi† · Heeseung Jo†† · Myungho Lee†††

ABSTRACT

The current GPU virtualization techniques incur large overheads when executing application programs mainly due to the fine-grain time-sharing scheduling of the GPU among multiple Virtual Machines (VMs). Besides, the current techniques lack of portability, because they include the APIs for the GPU computations in the VM monitor. In this paper, we propose a low overhead and high performance GPU virtualization approach on a heterogeneous HPC system based on the open-source Xen. Our proposed techniques are tailored to the bio applications. In our virtualization framework, we allow a VM to solely occupy a GPU once the VM is assigned a GPU instead of relying on the time-sharing the GPU. This improves the performance of the applications and the utilization of the GPUs. Our techniques also allow a direct pass-through to the GPU by using the IOMMU virtualization features embedded in the hardware for the high portability. Experimental studies using microbiology genome analysis applications show that our proposed techniques based on the direct pass-through significantly reduce the overheads compared with the previous Domain0 based approaches. Furthermore, our approach closely matches the performance for the applications to the bare machine or rather improves the performance.

Keywords : GPU Virtualization, Xen, IOMMU, Direct Pass-Through, Microbiologic Genome Analysis

바이오 응용을 위한 직접 통로 기반의 GPU 가상화

최 동 훈† · 조 희 승†† · 이 명 호†††

요 약

현재 개발된 GPU 가상화 기술은 모두 미세한 시분할 기법에 의한 스케줄링을 사용하기 때문에 어플리케이션 실행을 위한 오버헤드를 필요 이상으로 유발한다. 또한 이들은 가상 머신 모니터에 GPU 컴퓨팅 API를 포함하고 있어서, 가상 머신 모니터의 이식성이 약하다. 본 논문에서는 이질적 컴퓨팅 시스템에서 바이오 어플리케이션에 최적화된 GPU 가상화 기술을 제안하며, 공개 소스 Xen을 사용하여 개발하였다. 우리가 제안하는 방법은 가상 머신 간의 GPU 공유를 시분할에 의존하지 않는다. 대신에 하나의 가상 머신이 GPU를 할당 받으면 그 가상 머신이 어플리케이션을 종료할 때까지 GPU를 사용하도록 허용한다. 이렇게 하여 바이오 어플리케이션의 성능을 향상시키고 GPU의 활용률을 높인다. GPU 가상화의 이식성을 높이기 위해 하드웨어가 지원하는 IOMMU 가상화를 이용하여 GPU에 대한 직접 접근 통로를 제공한다. 미생물 유전체 분석 어플리케이션을 대상으로 성능을 분석한 결과, 본 연구에서 제안하는 직접 통로 방식에 의한 GPU 가상화는 Domain0를 통한 접근에 비해 오버헤드가 적었다. 또한 직접 접근 방식에 의한 가상 머신의 GPU 접근은, 비가상화된 머신과 거의 성능의 차이가 없었다.

키워드 : GPU 가상화, Xen, IOMMU, 직접 접근 통로, 미생물 유전체 분석

1. 서 론

현재 프로세서 산업계의 트랜드는 단순히 코어의 수가 증가하는 플랫폼에서 벗어나, 실제 시장에서 전력소모 대비 성능까지 고려하는 방향으로 진화하고 있다. 이러한 플랫폼은 고성능 컴퓨팅 능력을 많이 요구하는 바이오인포매틱스 분야 계산[1,2], 그래픽, 암호화[3,4] 등에 적용할 때 개선 효과가 클 것이라는 기대 아래, 매우 활발한 연구가 진행되고 있다. 다른 한편으로 급변하고 있는 비즈니스 환경의 변화에 대한 대응력을 확보하기 위해 컴퓨팅 시스템의 변화 수용 능력의 중요성이 부각되면서 가상화 기술이 주목을 받고 있다[5].

오늘날의 GPU는 전력소모 대비 성능 측면에서 고성능 컴퓨팅(high performance computing, HPC)의 저비용 고효율에 가장 적합한 솔루션이라고 평가되고 있으며[6], 최근에 가장 빠르게 확산되고 있는 기술 중의 하나이다. 특히 과학 기술 계산 분야의 어플리케이션에 대한 성능은 CPU와 GPU로 구

※ 본 연구는 지식경제부 및 한국산업기술평가원의 IT원천기술개발사업의 일환으로 수행하였음.
† 정 회 원 : 한국과학기술정보연구원 소프트웨어센터 책임연구원
†† 정 회 원 : 전북대학교 IT정보공학과 조교수
††† 정 회 원 : 명지대학교 컴퓨터공학과 부교수
논문접수 : 2013년 1월 8일
수 정 일 : 1차 2013년 1월 22일
심사완료 : 2013년 1월 22일
* Corresponding Author : Heeseung Jo(heeseung@jbnu.ac.kr)

성된 이질적 컴퓨팅(heterogeneous computing) 시스템의 병렬 최적화를 통해 가속화되고 있다[7]. 이질적 컴퓨팅 시스템에서 CPU는 주로 운영 체제, 태스크 처리와 같이 어플리케이션의 처리에 필요한 컴퓨팅 자원의 관리 기능을 수행하고, GPU는 주로 숫자나 데이터의 대량 고속 처리를 비롯한 어플리케이션의 실질적인 처리를 수행한다[8]. NVIDIA의 Tesla GPU[9]는 CPU대비 10배 가까운 메모리 인터페이스 속도와 240개의 코어(core)에서 동시에 데이터를 처리 함으로써 최대 200배 이상까지도 계산 속도를 높일 수 있다. 이질적 컴퓨팅 시스템은 CPU뿐만 아니라 GPU의 컴퓨팅 능력을 활용하여 병렬 처리 효과를 극대화하기 위한 목적으로 개발되고 있다. 최근 NVIDIA가 1500개 이상의 코어와 2GB의 장치 메모리로 구성된 Kefler 아키텍처[10]를 발표한 이후, CPU와 GPU 간의 데이터 처리 속도는 지속적으로 격차가 벌어질 것으로 예상된다. 이러한 이유로, GPU는 고성능 컴퓨팅에 선택이 아닌 필수적 요소로 자리매김할 것이다.

바이오인포매틱스 분야에서는 데이터 분석에 필요한 컴퓨팅 자원의 수요에 탄력적으로 대응하기 위해 클라우드 컴퓨팅을 활용하고 있으며[11], 대규모 데이터의 고속 처리를 위해 이질적 컴퓨팅 시스템에 기반한 클러스터 시스템을 개발하였다[7,12]. 이러한 현상이 가속화함에 따라 고성능 클라우드를 구축하기 위한 이질적 컴퓨팅 시스템의 가상화 기술이 요구되고 있다. 가상 머신(virtual machine, VM)이 GPU를 활용하는 방법은 여러 가지가 있으나, 대규모 데이터의 고속 처리를 위한 GPU 가상화의 필수적인 요구 사항은 가상 머신 상에서 실행되는 어플리케이션의 성능 개선을 최적화해야 하고 GPU의 자원 활용을 극대화해야 한다는 것이다.

현재 개발된 GPU 가상화 기술은 gVirtuS [13], GViM [14], vCUDA [15] 등이 있다. 이들은 모두 미세한 시분할 기법에 의한 스케줄링을 사용하기 때문에 어플리케이션 실행을 위한 오버헤드를 필요 이상으로 유발한다. 또한 이들은 여러 가상 머신이 GPU를 공유할 수 있도록, 가상화의 입출력을 담당하는 소프트웨어 레이어에 GPU 컴퓨팅에 필요한 CUDA[16]나 OpenCL[17]의 API를 포함하고 있다. 이 방식은 이들 버전의 갱신에 따라 API도 지속적으로 갱신해야 하므로 VMM의 이식성이 약한 단점이 있다. VMM으로 널리 사용되고 있는 Xen[18]은 게스트 VM이 Domain0를 통해 GPU를 접근하도록 구현되어 있어 GPU 접근에 따른 오버헤드가 비교적 높다. 이 방법은 어플리케이션의 GPU 처리 시간이 매우 짧을 경우 성능에 치명적 영향을 미치지 않으나, 바이오 어플리케이션과 같이 GPU 상에서 처리 시간이 길 경우 성능에 치명적일 수 있다. 즉, 다수의 가상 머신 간에 GPU를 경쟁적으로 접근해야 하며 어플리케이션이 종료될 때까지 여러 번 다중화(multiplexing)를 위한 작업을 수행하여야 하기 때문에, Domain0를 통한 GPU 접근 오버헤드는 어플리케이션의 성능 개선에 한계를 초래한다.

본 논문에서는 이질적 클러스터 상에서 바이오 어플리케이션에 최적화된 GPU 가상화 기술을 제안한다. 제안하는 방법은 가상 머신 간의 GPU 공유를 시분할에 의하지 않는다. 대신에 하나의 가상 머신이 GPU를 할당 받으면 그 가상 머신이 어플리케이션을 종료할 때까지 GPU를 사용하도록 허용한다. 이렇게 하여 바이오 어플리케이션의 성능을 향상시키고 GPU의 활용율을 높인다. GPU 가상화의 이식성을 높이기 위해 하드웨어가 지원하는 가상화 IOMMU[19]를 이용하여 GPU에 대한 직접 접근 통로를 제공한다. 본 연구에서 제안하는 직접 통로 방식에 의한 가상 머신의 GPU 접근은 Domain0를 통한 접근에 비해 오버헤드가 적다. 또한 직접 접근 방식에 의한 가상 머신의 GPU 접근은, 비가상화된 머신과 거의 성능의 차이가 없었다. 성능 평가를 위해 미생물 유전체 분석을 수행하였으며, 실험에 사용된 어플리케이션과 데이터는 각각 CUDA로 작성된 정렬 프로그램 BarraCUDA, 미생물 유전체 데이터 집합이다. 이렇게 실현된 GPU 가상화 기술은 다수의 가상 머신이 하나 또는 다수의 GPU를 시간/공간적으로 공유 가능하게 하며, 보다 효율적인 고성능 클라우드의 구축에 기여할 것이다. 아울러 다양한 분야의 과학 기술자들이 보다 저렴한 비용으로 유연하게 고성능 컴퓨팅을 활용할 수 있는 계기를 제공할 것으로 예상된다.

2. 배 경

2.1 Graph Processing Unit

그래픽 처리 장치는 부동 소수점 명령을 계산하는 그래픽 카드에 딸린 처리 장치이다. 그래픽 가속기는 그래픽 렌더링에 흔히 쓰이는 특별한 수학 명령을 포함하는 맞춤식 마이크로칩을 갖추고 있다. 이러한 마이크로칩이 효율적으로 동작하면 그래픽 가속기 또한 효율적으로 처리된다. 이들은 주로 3차원 게임이나 고사양 3차원 렌더링에 주로 쓰인다. 그래픽 처리 장치는 수많은 그래픽 기본 기하 명령을 담고 있어 호스트 CPU를 통해 화면에 표시하는 것보다 훨씬 더 빠르게 그리기를 처리할 수 있다. 또한 그래픽 처리 장치는 대량의 행렬과 벡터를 다루는 데 뛰어난 성능을 발휘하므로, 이러한 연산을 많이 사용하는 어플리케이션 프로그램은 그래픽 처리와 관련 없는 작업에도 그래픽 처리 하드웨어를 이용하기도 한다[20].

NVIDIA의 계산 전용 그래픽카드인 Tesla C1060의 아키텍쳐를 살펴보면, 30개의 멀티프로세서가 있고, 각각의 멀티프로세서는 8개의 쓰레드 프로세서가 장착되어 있다. 따라서 240개의 코어로 동작한다. 멀티프로세서는 8개의 쓰레드 프로세서, 삼각 함수 등의 계산에 사용되는 2개의 특수 함수 유닛, 정밀 연산을 처리하는 1개의 배정도(double precision) 유닛으로 구성되어 있다. 특히 각각의 멀티프로세서는 1만 6천개의 32bit 레지스터를 생성된 쓰레드가 분할하여 사용하고 각각 16KB의 공유 메모리를 사용한다[21].

2.2 가상화 기술

가상화 기술은 70년대 초반 IBM 370 메인프레임에서 고가의 하드웨어의 활용도를 높이고, 단일 하드웨어 위에 다수의 가상 머신을 제공하며 각 가상 머신마다 사용자가 원하는 운영 체계를 동작시키기 위한 목적으로, 그 연구가 시작되었

다[5]. 그러나 컴퓨터 하드웨어 시장의 급속한 발전은 하드웨어의 단가 하락을 초래하였고, 고가의 하드웨어 장비를 효율적으로 사용하기 위한 연구는 그 당시에 일단락되었다. 그러나 최근 가상화 기술은 새로운 르네상스를 맞이하고 있다. 가상화 기술의 초기 목적인 하드웨어 활용도를 높일 뿐만 아니라, 대규모의 서버 환경의 효율적인 관리와 안정성 확보에 이점이 높기 때문이다. 활용율(utilization)이 낮은 다수의 서버를 하나로 통합(consolidation)하고 서버의 이용이 증가할 경우에는 실시간 이주 기술(live migration)을 통해 이들을 여러 대의 서버에 다시 분리하는 서버 관리 기법을 적용할 수 있다. 또한, 가상 머신(virtual machine)을 이용한 운영체계간 격리(isolation)를 통해 안정성을 향상시킬 수 있다. 이와 같이 가상화 기술은 효율적인 하드웨어 자원 활용, 관리 비용 감소, 에너지 절감 등의 다양한 장점을 수반한다.

가상화 기술은 서버와 PC 환경에서 2000년도 이후에 많은 상업화를 거치고 있고 현재도 발전 중이다. 특히 VMware사[22,23]는 서버와 PC 제품군으로 각각 ESX server와 Workstation을 개발하여 상업적으로 가장 큰 성공을 거두었고 효율적인 자원 활용이나 성능 측면에서의 연구들을 활발히 진행하고 있다. Xen[18]은 공개 소스 기반의 반가상화(para-virtualization) 기술을 실현시켜 우수한 성능으로 서버 시장에서 가상화 기술을 선도하고 있다. Xen의 소스 공개는 많은 대학 및 기업들이 다양한 연구를 촉진하여 가상화 기술 발전에 크게 기여하고 있다. 또한 PC 환경에서는 Microsoft사의 Virtual PC[24], Apple사의 Parallels[25], Oracle사의 VirtualBox[26]는 PC 사용자가 가상화 기술을 손쉽게 사용할 수 있는 인터페이스를 제공하여 가상화의 일반화를 선도하고 있다.

2.3 가상화 지원 하드웨어

Intel VT-x[27] : 인텔은 "Vanderpool"이라는 프로젝트를 시작으로 x86 플랫폼에서 가상화를 지원하기 위한 기술을 제안하였다. 2005년에 처음으로 Pentium 4 (Model 662와 672)에 VT-x 기술을 지원하는 CPU를 상용화하고 발표한 바 있다. VT-x 기술에는 가상화를 지원하기 위한 확장 페이지 테이블(Extended Page tables, EPT), 페이지 테이블 가상화(page-table virtualization) 등이 하드웨어 레벨에서 구현되어 있다.

AMD-V[28] : AMD는 Pacifica라는 프로젝트를 시작으로 AMD Secure Virtual Machine (SVM)을 제안한 바 있다. 2006년에 처음으로 Athlon 64, Athlon 64 X2, Athlon FX CPU에 AMD-V 기술을 상용화한 CPU를 시장에 내놓았다. 가상화를 지원하기 위하여 AMD-V는 이전의 중첩 페이지 테이블(nested page table)에서 발전된 형태의 Rapid Virtualization Indexing (RVI) 기술을 선보였으며, 이는 후에 인텔의 EPT로 발전되었다.

IOMMU 가상화 (Intel VT-d, AMD-Vi) : Input/output memory management unit (IOMMU) 가상화는 가상 머신이 주변 디바이스를 직접 접근할 수 있도록 하여주는 기술이다. IOMMU의 가상화는 주로 DMA, 인터럽트 리매핑 (interrupt remapping)에 관여하는 것으로 알려져 있으며, 인텔의 경우, snoop control, DMA pass-through, queued invalidation, interrupt remapping 기능을 지원한다. IOMMU 가상화 기술의 주된 목적은 가상 머신이 디바이스를 직접 접근하여, Domain 0 또는 VMM의 관여를 거치지 않고도, 안정적이고 빠른 접근이 가능하도록 하기 위함이다. 인텔에서는 VT-d라는 명칭으로, AMD에서는 AMD-Vi라는 명칭으로 기능을 제공하고 있다.

3. 관련 연구

가상화 기술은 크게 하드웨어의 가상화와 시스템 안정성 향상을 목표로 하고 있다. 하드웨어 가상화는 하나의 시스템에서 다수의 가상 머신을 제공하며 그 내부에 독립적인 운영체계를 비롯한 소프트웨어를 동작시키기 위한 기술로, VMM은 각각의 운영체계에 독립적인(isolated) 가상 머신을 제공해준다. 이때 프로세서, 메모리, 디스크, 네트워크 등의 하드웨어 자원을 효율적이고, 성능의 손실 없이 제공하는 것이 하드웨어 가상화의 주된 연구 방향이다. 또한 시스템 안정성을 위하여 VMM은 하드웨어를 접근하는 별개의 가상 머신을 제공한다. 이를 통해 하드웨어 또는 장치 드라이버의 오작동이 시스템 전체에 영향을 주지 않고, 자신의 가상 머신 내에서 결함 봉쇄(fault containment)되는 기능을 제공하여 시스템 전체의 안정성을 향상시킨다.

GPU를 가상화 시키기 위해서는 GPU를 PCIe 채널을 사용하는 별도의 장치로 취급하여 가상화하는 방법이 가장 적합하다. 하지만, GPU는 그래픽장치의 특성이 남아 있어서 시스템의 BIOS정보와도 연관되어 있으며, 가상화가 매우 어렵다. 기존의 연구를 살펴보면, 대표적으로 gVirtuS[13], GViM[14], vCUDA[15] (Fig. 1 참조) 등이 있으나, 모두 가상 머신 간의 GPU 공유를 궁극의 목적으로 두어 호스트 운영체제 또는 VMM가 GPU의 사용을 관리하는 구조를 갖는다. 이러한 접근 방법은 크게 아래와 같은 두 가지 문제점의 원인이 되고 있다.

- GPU 가상화의 이식성 결여: 기존의 가상화는 여러 가상 머신이 하나의 GPU를 사용할 수 있도록 하기 위해 GPU 컴퓨팅에 필요한 API를 모두 가상화 입출력 담당 소프트웨어 레이어가 포함하고 있다. 이러한 접근 방법은 가상 머신에서 실행 중인 어플리케이션의 CUDA나 OpenCL 등의 버전이 다양할 경우, VMM이 이들에 대한 GPU API를 모두 구현하여 지원해야 한다. 또한 이들의 API의 버전이 변경되거나 다른 예외 상황이 발생하는 경우 API를 모두 다시 수정해야 한다. 이에 따르는 가장 큰 문제는 GPU 가상화의 이식성의 결여다.
- 성능 저하: 기존의 가상화는 가상 머신 간에 GPU의 공유를 위해 VMM이 미세한 단위의 시분할 기법을 사용한다. 대량 데이터 분석을 미세한 시분할 기법에 의할 경우 가상 머신 간에 GPU의 경쟁으로 인해 다중화에 따르는 비용이 증가하여 오히려 본래의 성능을 매우 저하시키는 결과를 초래한다. GPU를 활용한 대량 데이터 분석은 컴퓨팅이

종료될 때까지 하나의 가상 머신이 GPU를 사용하는 것이 낫다. 더욱이 VMM의 스케줄링 정책에 의해서 지연 시간이 달라지는 문제점도 가지고 있다.

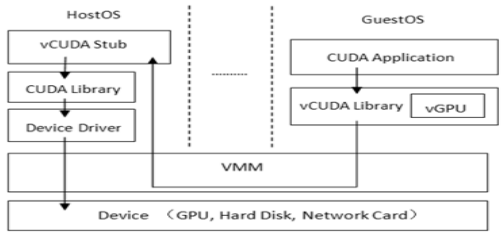


Fig. 1. Architecture of vCUDA

4. GPU 가상화 구현

이 절에서는 가상화 오버헤드를 줄이고 가상화 기술의 이식성을 높이기 위해 PCIe 채널의 직접 통로(direct pass-through)에 의한 GPU 가상화를 구현한다. 직접 통로 방식은 VMM에 의한 상호 간섭을 최소화하기 때문에 가상 머신은 물리적 머신의 성능 수준으로 GPU를 접근할 수 있다. 또한 각 가상 머신이 어플리케이션을 위한 GPU API를 설치할 수 있기 때문에 기존의 방식에 비해 GPU 컴퓨팅을 위한 API의 구현이나 수정 등에서 자유롭다. 직접 통로 방식의 가상화는 하드웨어 지원에 의한 IOMMU 가상화를 이용한다. IOMMU는 장치의 주소를 물리적 주소로 변환하고 장치의 동작 오류로부터 메모리를 보호한다.

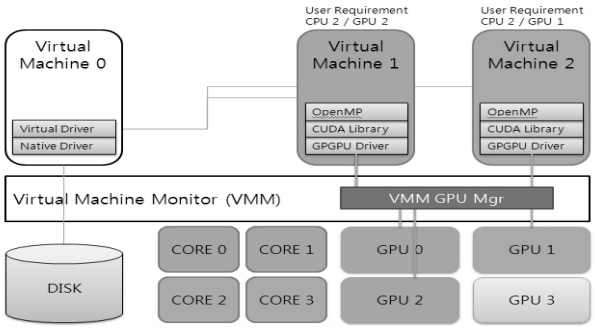


Fig. 2. Architecture of Virtualized GPU with Direct Pass-through

Fig. 2는 직접 통로에 의한 GPU 가상화 시스템의 구조를 보여주고 있다. 이 시스템에서 각 GPU는 PCIe 채널에 연결되어 있고 VMM은 PCIe와 GPU 간의 통제를 가상 머신에게 넘겨 준다. 각 가상 머신은 자신의 GPU API와 GPU 장치 드라이버를 가지고 있어서 VMM의 중재를 받지 않고서도 GPU를 통제하고 접근할 수 있다. GPU Manager는 VMM 내에서 위치하고 있으며, GPU를 가상 머신에게 할당하고 회수하는 역할을 수행한다. 가상 머신을 시작할 때, GPU Manager는 요청받은 만큼의 GPU를 가상 머신에게 할당하고 종료되면 GPU를 회수한다.

GPU 가상화를 위한 GPU 서버의 하드웨어 사양은 아래 Table 1과 같다.

Table 1. GPU Server Specification

Device	Specification
CPU	Intel(R) Xeon(R) E5620 (2.40GHz)
Chipset	Intel(R) 5520
Memory	DDR3 1333MHz (24GB)
PCI slot	PCI Express Gen2, 4EA
GPU	NVIDIA Quadro FX3800, 4EA

본 연구에서 GPU 가상화 시스템을 개발하기 위한 VMM으로 공개 소스 코드 Xen을 사용한다. 운영체제는 Linux Ubuntu 8.04 버전과 10.04를 지원하도록 GPU 가상화를 개발한다. Xen은 이론적으로 Linux의 배포 판과 무관하게 동작하나, gcc와 라이브러리 등의 버전 차이를 고려하면 배포 판에 설치 가능한 것은 각각 Xen 3.4와 4.0 버전이다. 따라서 Ubuntu 8.04와 Xen 3.4를 이용한 설치 환경과, Ubuntu 10.04와 Xen 4.0를 이용한 설치 환경으로 각각 개발을 진행하였다.

Xen은 하드웨어 가상 머신 (hardware virtual machine, HVM)과 반가상화 가상 머신 (para-virtualized virtual machine, PVM) 방식의 가상화를 지원한다. 본 연구에서 추구하는 가상화는 PVM 방식으로는 어려운 점이 존재한다. 첫째, 게스트 VM에서 GPU를 직접 통로 방식으로 가상화 레이어를 거치지 않고 직접 접근하기 위해서는 ACPI (advanced configuration and power interface)의 지원을 받아야 하나, PVM 방식은 이를 지원하지 않는다. GPU의 장치 드라이버를 수정하면 가능할 수 있으나, GPU 제조사가 자사의 장치 드라이버 소스코드를 공개하지 않기 때문에 부적절하다. 둘째, Domain0에 GPU를 설치하고, Domain0와 게스트 VM간에 통신하는 방식을 생각해 볼 수 있으나 이는 앞에 언급한 연구의 접근 방식으로, 오버헤드가 매우 높다. 따라서, HVM 방식으로 게스트 VM을 동작시키는 방법을 사용하여야 하고, HVM에서 GPU를 직접 접근 방식으로 사용하기 위해서는 IOMMU 하드웨어의 지원을 받아야만 가능하다. 본 연구에서는 Intel의 VT-d를 사용한다.

NIVIDIA사는 자사의 GPU를 쉽게 사용할 수 있도록 CUDA 개발 환경을 제공하고 있다. CUDA는 GPU의 강력한 처리 능력을 활용하여 컴퓨팅 성능을 크게 개선시켜주는 NVIDIA의 병렬 컴퓨팅 아키텍처이다[13]. GPU가 직접 통로 방식의 가상화 환경에서 사용되기 위해서는 GPU 장치에서도 이를 위한 적절한 하드웨어의 지원이 필요하다. 이를 위한 NVIDIA의 하드웨어 지원 사항은 SLI에 포함되어 있으며, 가상화 환경에서 GPU 사용 가능 여부를 결정한다. 본 연구에서 사용한 GPU Quadro FX 3800는 SLI기술을 지원하고 있으며, 대부분의 CUDA 어플리케이션을 가상 머신에서 실행하는데 전혀 문제가 발견되지 않았다.

5. 직접 통로에 의한 GPU 가상화의 성능 평가

본 연구에서 개발된 GPU 가상화 기술의 성능 분석을 위해 다음과 같은 실험을 수행하였다. 본 실험에서 사용된 환경은 Fig. 3과 같다. 실험 머신은 4개의 GPU (NVIDIA Quadro FX 3800)을 장착하고 있으며, 각 VM은 PCIe 채널을 이용하여 각각 한 개의 GPU를 사용할 수 있는 구조로 되어 있다.

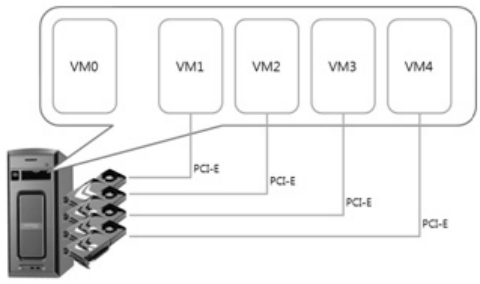


Fig. 3. Architecture of GPU Virtualization for Experimental Machine

실험을 위하여 각 VM에서 BarraCUDA[29]와 미생물 유전체 데이터를 사용하였다. BarraCUDA는 GPU를 이용하여 유전체 데이터의 리드 매핑(read mapping)을 수행한다. 실험에 사용된 데이터는 Table 2와 같다.

Table 2. Input Data(E. Coli)

	file name	size
Reference sequence	ecoli_K12_MG1665.fasta	4M
Reads	ecoli_K12_MG1665_reads_100.fasta	2.4 G

Fig. 4는 BarraCUDA의 수행시간을 보여주고 있다. x축은 Native 머신에서 동시에 동작시킨 VM의 수를 나타내며, y축은 이들 각각의 경우의 실행 시간을 나타낸다. 본 연구에서 제안된 PCIe 채널을 직접 가상 머신이 잡도록 하여 GPU를 사용하는 방식이 비가상화 머신에서 사용하는 방법과 매우 유사하다는 점을 고려하면 가상 머신에서 GPU를 사용할 때 오버헤드가 전혀 가중되지 않음을 알 수 있다. 더욱이 여러 개의 VM을 사용할 경우 비가상화 환경보다 수행시간이 오히려 약간 단축되는 것을 알 수 있고, 4개의 VM을 동시에 동작시키는 경우에도 유독 하나의 VM에서 성능이 다른 것을 확인할 수 있다. 하지만, 이는 Barra CUDA의 초기에 GPU를 탐지하는데 소요되는 시간이 약간씩 다르기 때문인 것으로 분석된다. 보다 명확한 원인 분석을 위해서는 보다 구체적이고 정확한 실험이 추가적으로 필요할 것으로 판단된다.

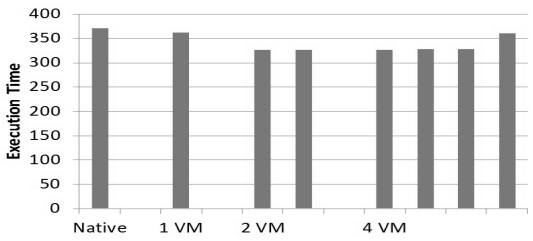


Fig. 4. BarraCUDA Execution Time

상기 BarraCUDA의 실험 결과는 어플리케이션에서 상당한 디스크 I/O의 시간을 포함하고 있다. 디스크 I/O를 제외하고 순수한 GPU의 계산 성능을 검증하기 위하여 Black Sholes [16] 알고리즘을 추가로 수행하였고, 그 결과가 Fig. 5에 나타나 있다. BarraCUDA의 경우와 비슷한 결과 양상을 보이고 있으며, 4개 VM을 동시에 동작하였을 경우 마찬가지로 하나의 VM만 다른 VM보다 느린 것을 확인할 수 있다. 이를 통해 디스크 I/O에 상관없이 성능 평가 결과가 유사한 패턴을 갖는다는 것을 알 수 있다.

향후, 시간적인 차이는 매우 작지만, 가상화된 GPU를 사용할 때 가상 머신에서의 수행시간이 왜 더 줄어드는지에 대한 분석과 4개의 가상 머신을 동시에 동작할 때 유독 하나의 가상 머신만 수행시간이 길어지는 지에 대한 추가적인 분석이 필요할 것으로 판단된다.

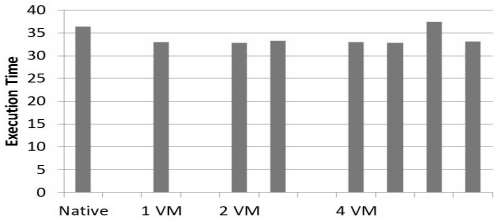


Fig. 5. BlackSholes Execution Time

6. 결 론

본 연구에서는 IOMMU와 Xen을 활용하여 바이오 데이터 분석을 지원하기 위한 GPU 가상화 시스템을 개발하였다. 특히, 대규모의 컴퓨팅이 필요한 GPU 가상화는 바이오 어플리케이션의 특성을 고려하여, 보다 나은 성능을 나타낼 수 있도록 노력하였다. 가상 머신에서 GPU를 바이오 어플리케이션의 실행에 사용하기 위해, 주로 CPU에서 가상화의 지원여부와 IOMMU의 지원여부, 메인보드 칩셋에서 IOMMU의 지원여부, GPU에서 가상화의 지원여부를 테스트하였으며, 직접 통로에 의한 GPU 가상화 시스템을 구현하였다. 본 연구에서 제안하는 직접 통로 방식에 의한 가상 머신의 GPU 접근은, 비가상화된 머신과 거의 성능의 차이가 없었으며 오히려 수행시간이 약간 단축되는 경우를 확인할 수 있었다. 성능 평가를 위해 미생물 유전체 분석을 수행하였으며, 실험에 사용된 어플리케이션과 데이터는 각각 CUDA로 작성된 정렬 프로그램 BarraCUDA, 미생물 유전체 데이터 집합이다. 데이터 입출력이 아닌 순수한 GPU 계산 성능을 분석하기 위해 CUDA 툴킷의 BlackSholes 알고리즘을 실행하였다.

현재 사용하는 직접 통로 방식의 GPU 가상화는 하나의 VM이 PCIe 채널을 이용하여 부팅한 이후, 어플리케이션의 고성능 컴퓨팅을 위해 VM의 종료시까지 지속적으로 사용하는 것을 전제하고 있다. 따라서 가상 머신 간에 GPU의 원활한 공유가 이뤄지지 않는 단점이 있다. 향후 직접 통로 방식의 GPU 가상화에서도 VM간에 GPU를 보다 원활하게 공유할 수 있는 방안으로 GPU 자원의 효율적인 스케줄링을 연구하고자 한다.

또한 PCIe 채널의 Hot-plug를 통하여 VM이 GPU를 사용하지 않을 때 GPU를 다른 VM에 할당시켜주는 방법을 모색하고자 한다. Xen에서는 공식적으로 PCIe 채널에 대하여 Hot-plug를 지원하고 있다. 그러나 테스트 장비에서 실

험해본 결과, 게스트 VM의 커널 또는 GPU 카드의 펌웨어(firmware)에서 Hot-plug를 실시간으로 인식하지 못하는 것으로 나타나고 있다. 따라서 게스트 VM에서 Hot-plug를 지원하기 위해 보다 정교한 연구가 필요하다.

참 고 문 헌

[1] P. D. Vouzis and N. V. Sahinidis, "GPU-BLAST: using graphics processors to accelerate protein sequence alignment," Bioinformatics, Vol.27, No.2, 2011, pp.182-188.

[2] C-M Liu, T. Wong, E. Wu, R. Luo, S-M Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, R. Li, and T-W Lam, "SOAP3: Ultra-fast GPU-based parallel alignment tool for short reads," Bioinformatics, 2012.

[3] O. Harrison and J. Waldron, "Practical symmetric key cryptography on modern graphics hardware," USENIX Security Symposium 2008.

[4] S. Manavski, "CUDA compatible GPU as an effeicient hardware accelerator for AES cryptography," IEEE International Conference on Signal Processing and Communication 2007.

[5] E. W. Pugh, L.R. Johnson, J. H. Palmer., IBM's 360 and early 370 systems, MIT Press, ISBN 0-262-16123-0.

[6] GPU energy efficiency. http://www.nvidia.com/object/gcr-energy-efficiency.html

[7] Tianhe-1. http://www.top500.org/system/176929

[8] M. Lee, H. Jo, D. Choi, "Towards High Performance and Usability Programming Methodology for Heterogeneous HPC Architectures," International Journal of Advancements in Computing Technology, Oct., 2012.

[9] Tesla. http://www.nvidia.com/object/personal-supercomputing.html

[10] Kefler. http://www.nvidia.com/object/nvidia-kepler.html

[11] H. Lee, Y. Yang, H. Chae, S. Nam, D. Choi, P. Tangchaisin, C. Herath, S. Marru, K.P. Nephew, and S. Kim, "BioVLAB-MMIA: A Cloud Environment for microRNA and mRNA Integrated Analysis (MMIA) on Amazon EC2," IEEE Transactions on NanoBioscience, Vol.11, Issue3, Sep., 2012, pp.266-272.

[12] Tsubame. http://www.engadget.com/2011/11/23/nvidias-tesla-gpu-powers-tsubame-2-0-to-green-supercomputer-sup/

[13] G. Giunta, R. Montella, G. Agrillo, and G. Coviello, "A gpgpu transparent virtualization component for high performance computing clouds", Euro-Par 2010 - Parallel Processing, Vol.6271, chap. 37, pp.379-391, 2010.

[14] V. Gupta, A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, and P. Ranganathan, "GViM: GPU-accelerated virtual machines, Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing", pp.17-24, March 31-31, 2009.

[15] L. Shi, H. Chen, and J. Sun, "vCUDA: GPU accelerated high performance computing in virtual machines", IEEE International Symposium on Parallel & Distributed Processing (IPDPS'09), pp.1-11, 2009.

[16] CUDA toolkit. https://developer.nvidia.com/cuda-toolkit

[17] OpenCL. http://www.khronos.org/opencl/

[18] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," Symposium on Operating Systems Principles (SOSP), 2003.

[19] IOMMU. http://en.wikipedia.org/wiki/IOMMU

[20] Graphics Processing Unit. http://wikipedia.org

[21] GPU 병렬 컴퓨팅 기술을 이용한 개인용 수퍼 컴퓨터 현황과 전망. 이주석, 류현곤. 전자공학회지, Vol.36, No.5, pp.18-27.

[22] VMware Roadshow 2007 Seoul. http://vmware.event-microsite.com/kr_main.html.

[23] VMWare. http://www.vmware.com.

[24] VirtualPC. http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx.

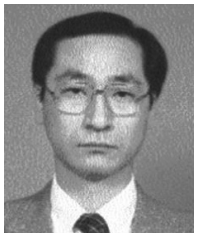
[25] Parallels. http://www.parallels.com.

[26] VirtualBox. http://www.virtualbox.org.

[27] Intel virtualization. http://www.intel.com/content/www/us/en/virtualization/intel-virtualization-transforms-it.html

[28] AMD virtualization. http://sites.amd.com/us/business/it-solutions/virtualization/Pages/virtualization.aspx

[29] BarraCUDA. http://www.many-core.group.cam.ac.uk/projects/lam.shtml



최 동 훈

e-mail : choid@kisti.re.kr
1981년 서울대학교 계산통계학과(학사)
1983년 한국과학기술원 전산학과(석사)
1986년 노스웨스턴대학교 전산학과(박사)
2005년~현 재 한국과학기술정보연구원 소프트웨어센터 책임연구원
관심분야 : 과학 데이터 관리, 데이터 집중형 컴퓨팅

조 희 승

e-mail : heeseung@jbnu.ac.kr
2000년 서강대학교 컴퓨터공학과(학사)
2010년 한국과학기술원 전산학과(박사)
2010년~현 재 전북대학교 IT정보공학과 조교수
관심분야 : 운영체제, 가상화 시스템, 클라우드 컴퓨팅, 임베디드 시스템

이 명 호

e-mail : myunghol@mju.ac.kr
1986년 서울대학교 계산통계학과 (학사)
1988년 미국 University of Southern California 컴퓨터과학과(석사)
1999년 미국 University of Southern California 컴퓨터공학과(박사)
2004년~현 재 명지대학교 컴퓨터공학과 부교수
관심분야 : 고성능 컴퓨팅, 병렬 알고리즘, 마이크로 프로세서, 컴파일러, GPU 컴퓨팅