이질적 고성능 클라우드 컴퓨팅을 위한 확장형 OpenStack의 개발 및 평가

OpenStack Extension for Heterogeneous High Performance Cloud Computing

최동훈, 조희승, 박기웅[1)]
DongHoon Choi, Heeseung Jo, KiWoong Park

(34141) 대전시 유성구 대학로 245 한국과학기술정보연구원
(54896) 전북 전주시 덕진구 백제대로 567 전북대학교
(34520) 대전시 동구 대학로 62 대전대학교
choid@kisti.re.kr, heeseung@jbnu.ac.kr, woongbak@dju.kr

요 약

GPU는 저비용 고효율의 프로세서로 각광을 받고 있으며, 이질적 고성능 컴퓨팅 시스템의 주계산 자원으로 채택되고 있다. 본 논문에서는 GPU 기반의 고성능 클라우드 컴퓨팅을 위해 개발된 확장형 OpenStack을 서술한다. 확장형 OpenStack은 OpenStack에서 GPU 사용이 가능한 가상 머신의 생성 및 관리가 가능하도록, 가상 머신 간의 GPU 공유 스케줄러와 GPU 인지형 Nova 스케줄러를 제공한다. 확장형 OpenStack과 Rodinia 벤치마크를 이용한 실험에서 GPU 가상화로 인한 오버헤드가 2%내에 불과함을 보여주고 있다. 이것은 OpenStack이 이질적 고성능 클라우드 컴퓨팅에도 성공적으로 적용 가능함을 의미한다.

Abstract

GPU's are getting the spotlight in the chip processor market due to low power consumption and high efficiency, and increasingly adopted as a main computing resource in heterogeneous high performance computing systems. This paper describes OpenStack extension for heterogeneous high performance computing on the cloud. The extended features are a coarse-grained GPU scheduler and a GPU-aware Nova scheduler developed for creating and managing GPU-enabled virtual machines with OpenStack. The experiments by using Rodinia benchmark on OpenStack extension shows that the overhead due to GPU virtualization is within 2%. This means that OpenStack is successfully applicable to the heterogeneous high-performance cloud computing.

키워드: GPU 가상화, 이질적 고성능 클라우드 컴퓨팅, GPU 자원 스케줄링, GPU 인지형 Nova 스케줄러
Keyword: GPU virtualization, heterogeneous high-performance computing, GPU resource scheduler, GPU-aweare Nova scheduler

1) 교신 저자

논문투고일자 : 2016. 03. 28 심사일자 : 2016. 03. 28 게재확정일자 : 2016. 05. 10

1. 서론

클라우드 컴퓨팅은 사용자가 컴퓨팅 자원, 스토리지 자원 등을 사용한 만큼 요금을 내고 빌려서 쓰는 비즈니스 모델에 기반한 것으로, 아마존 클라우드 서비스가[1] 시작된 이래로 그 수요는 공공 시장을 중심으로 꾸준히 증가하여[2] 아마존 이외에도 IBM, Microsoft, Oracle, Google 등 클라우드 서비스 업체가 늘고 있다. 국내에서도 이에 대한 연구가[22, 23, 24] 진행되었으며, KT Cloud가 투자대비 효율의 우월성으로 인해 사용자의 관심을 끌고 있다.

클라우드 컴퓨팅은 자원 관리의 용이성과 경제성을 위해 가상화[3, 4, 5] 기술을 바탕으로 하고 있다. 가상화는 하나의 물리적 머신(physical machine, PM)에 서로 격리된 다수의 가상 머신(virtual machine, VM)을 생성하여 사용자에 특화된 VM을 사용자에게 제공할 수 있다. 이렇게 하여 사용자는 자신의 익숙한 컴퓨팅 환경에서 작업을 할 수 있는 동시에, 클라우드 서비스 제공자는 다수 사용자의 VM을 병합하여 하나의 PM에 할당하여 PM의 자원 낭비를 줄일 수 있다. VM은 파일로 저장되고 관리되기 때문에, 사용자는 자신의 VM을 필요시 재개시하여 사용할 수 있다. 클라우드 서비스 제공자는 사용자에게 VM을 제공하기 때문에, VM으로부터 해제된 PM 자원의 부가적인 관리(예를 들어, 다른 VM에 해제된 PM 자원을 할당하기 전에 해당 PM 자원에 설치되었을지도 모를 맬웨어(malware)와 같은 오염을 깨끗하게 청소하는 것) 비용을 제거할 수 있다.

클라우드 컴퓨팅이 전파됨에 따라 클라우드 구축의 초기 비용을 줄이고 클라우드 서비스 간의 연동 문제를 해결하기 위해 이들 기능을 제공하는 소프트웨어 스택의 공동 개발을 추구하게 되었다. 이를 위해 NASA, Rackspace, AT&T, IBM, Intel, Ubuntu, RedHat, SUSE, HP 등 150개 이상의 기관을 중심으로 공개소스 클라우드 컴퓨팅 플랫폼으로 OpenStack을 개발하였으며[6], 400여 기관이 참여 하에 OpenStack 재단이[7] 설립되어 OpenStack을 산업 표준으로 널리 보급되고 있다.

GPU[8, 9]는 CPU에 비해 저전력 고성능 컴퓨팅이 가능하다. 현재 고성능 컴퓨팅 선진국에서는 GPU를 주계산용 자원으로 채택하고 CPU를 운영체제 및 어플리케이션의 실행 관리용 자원으로 사용하는 이질적 고성능 컴퓨팅(high-performance computing, HPC) 시스템[10]을 개발하고 있다. 본 논문에서는 CPU와 GPU로 구성된 이질적 HPC 시스템의 클라우드 서비스를 위해 확장 개발된 OpenStack를 소개하고자 한다. 이질적 HPC 클라우드는 GPU 가상화를 필수적으로 수반한다. 이를 통해 VM은 자신이 원하는 GPU를 사용할 수 있고, 동일한 PM에서 동작하는 다수의 VM은 GPU를 공유한다. 기존의 GPU 가상화는[11, 12, 13, 14] Dom0 또는 관리용 VM을 통해 GPU 접근을 허용하고 VM에 대한 GPU의 할당/해제를 VM의 개시/종료에 의존하고 있다. 이러한 이유로, GPU 가상화에 따른 오버헤드가 증가하고 GPU 활용이 떨어지는 단점이 있다. 이러한 문제를 해결하고 VM 간의 원활한 GPU 공유를 위해 PCIe 직접 통로(PCI express direct pass-through)에 의한 GPU 접근 허용과 hot plug-in/plug-out 방식에 의한 GPU 할당/해제 메카니즘을 제안한다. 이외에도 OpenStack 기반의 이질적 HPC 클라우드에서 GPU 컴퓨팅을 수행하려면, GPU가 사용 가능한 가상 머신(GPU-enabled virtual machine, GEVM)을 관리할 수 있어야 한다. 이를 위해 Nova Compute가 GPU를 탑재하고 있는 물리적 노드를 GEVM에 할당할 수 있도록, OpenStack의 Nova 스케줄러를 확장한 GPU 인지형(GPU-aware) Nova 스케줄러를 제안한다.

이와 같이 확장된 OpenStack은 VM 간의 GPU 자원 공유 스케줄러와 GPU 인지형 Nova 스케줄러를 제공한다. 이를 위해 KVM과 Nova Compute를 각각 확장 개발하였다. 확장된 OpenStack은 OpenCL[17]과 CUDA[18] 어플리케이션을 모두

지원한다. Rodinia[19] 벤치마크를 이용하여 측정한 결과 확장된 OpenStack의 GPU 가상화 오버헤드는 2&로, PM의 성능에 도달함을 보여주고 있다.

본 논문의 구성은 다음과 같다. 2장에서 GPU 가상화를 위한 GPU 자원 공유 스케줄러를 설명하고, 3장에서 GPU 인지형 Nova 스케줄러를 서술한다. 4장에서 성능 평가 결과에 근거하여 확장된 OpenStack의 오버헤드가 거의 없다는 것을 보여준다. 5장에서 GPU 가상화 및 GPU 확장형 OpenStack의 관련 연구를 제시하고, 6장에서 끝맺는다.

2. GPU 자원 공유 스케줄러

본 절에서는 VM이 직접 통로를 통해 GPU를 사용할 수 있도록, Xen과 KVM 환경에서 개발된 GPU 공유 스케줄러를 서술한다. 직접 통로 기반의 GPU 가상화는 가상머신 모니터(virtual machine monitor, VMM)가 PCIe를 점유하지 않고, VM에게 투명하게 보여주는 방법을 사용하고 있다. 직접 통로에 의한 GPU 접근은 VMM에 의한 관리용 VM 또는 Dom0를 거치지 않기 때문에, 사용자의 VM은 보다 빠르게 GPU에 접근할 수 있고 가상화에 따른 오버헤드를 줄일 수 있어서 보다 좋은 성능을 나타낸다[16].

2.1 과립형 GPU 공유 메카니즘

본 논문에서 GPU 가상화는 과학 어플리케이션을 대상으로 하기 때문에 가시화나 비교적 짧은 계산 위주의 GPU 어플리케이션과 목적이 다르다. 이들 어플리케이션은 비교적 짧은 시간의 반복적인 처리를 요구하는 반면, 과학 어플리케이션은 처리 시간이 길어서 GPU의 모든 처리 능력을 동시에 사용하는 고성능 컴퓨팅을 요구한다. 따라서 과학 어플리케이션의 고성능 컴퓨팅을 위한 GPU 가상화는, 기존의 GPU 가상화와[11, 12, 13, 14] 달리 동시에 여러 VM에 GPU를 공유시키는 방식이 아니라, 한번에 하나의 어플리케이션에 GPU 자원 하나를 전부 할당하고 계산이 종료되면 회수하는 과립형 GPU 스케줄링(coarse-grained GPU scheduling)에 의한 GPU 공유가 효율적이다. 다만, 직접 통로에만 의존하면 VM 간에 GPU의 원활한 공유를 하기가 어렵다는 단점이 있다. 예를 들어, 최초 VM의 부팅 시 GPU를 가진 PCIe 채널을 점유하면 VM이 종료될 때까지 해당 GPU를 다른 VM에서는 사용할 수 없다. 이러한 문제를 해결하고자 PCIe hot plug 기능을 이용하여, VM 부팅 이후에 동작중인 상태의 여러 VM 간에 GPU를 공유할 수 있도록 과립형 GPU 스케줄링 기술을 개발하였다. 과립형 GPU 공유 메카니즘의 절차는 먼저 각 VM은 GPU가 필요시 관리용 VM에게 GPU할당을 요청하며, 요청을 받은 관리용 VM(Dom0, VM0, Linux 호스트)은 GPU Admin을 통하여 할당 가능한 GPU가 있는지 확인하고, 가용한 GPU를 hot plug-in 방식으로 요청하여 VM에게 할당한다. PCIe를 통해 GPU를 이용하여 연산을 실행하고, 연산 종료한 후에는 이를 다시 hot plug-out 방식으로 관리용 VM에게 반납한다[16].

2.2 GPU 자원 관리

GPU Admin은 GPU Pool을 관리하면서 GPU Manager의 요청에 따라 사용 가능한 GPU를 VM에게 할당한다. GPU Manager는 VM에서 실행 중인 어플리케이션으로부터 GPU 자원 요청을 받아서 GPU Admin을 통해 사용 가능한 GPU를 할당 받아 어플리케이션에게 제공한다. 이들 간의 상호 작용은 (그림 1)과 같다.

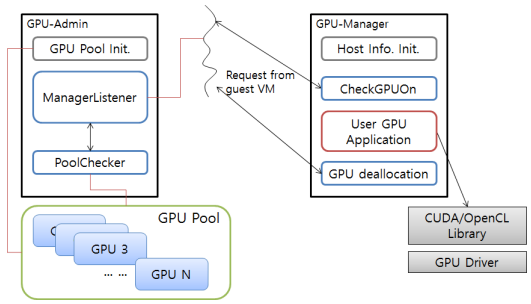
2.2.1 GPU Admin

GPU Admin은 설정파일에 저장된 정보를 이용하여, 해당 GPU들을 관리하기 위해 GPU Pool에 등록시킨다. 등록된 GPU들은 초기화되어 어느 VM에도 할당되지 않은 상태로 있게 되고, GPU Manager의 요청에 따라 VM에게 할당한다. 할당된 GPU들은 GPU Pool에서 요청한 VM의 ID와 IP 주소를 기록한다. 초기화되면, GPU Admin은 Manager

Listener와 PoolChecker을 위한 두 개의 쓰레드를 생성한다.

ManagerListener 쓰레드는 각 VM에서 동작하는 GPU Manager로부터 GPU의 할당 요청을 받아들인다. GPU Admin은 PoolChecker를 통해 GPU Pool에서 사용 가능한 GPU를 선택하여, Xen 또는 virsh의 인터페이스를 이용하여 VM에게 GPU를 hot plug-in 시켜 GPU를 할당한다. GPU Manager의 GPU의 해제 요청에 대해, GPU Admin은 Xen 또는 virsh의 인터페이스를 이용하여 요청한 VM으로부터 GPU를 hot plug-out 시켜 GPU를 해제하고 GPU Pool에 반납한다.

PoolChecker 쓰레드는 GPU Pool 내에서 이미 할당된 GPU들을 검색하여 현재 GPU가 사용 중인지 아닌지 판단한다. 주기적으로 GPU Pool을 검색하여, VM에게 할당된 GPU들을 찾고, 각 VM의 GPU Manager에게 현재 GPU가 사용 중인지 아닌지 알린다. GPU Manager는 자신의 시스템 내부에서 GPU 사용도에 따라 현재 GPU의 사용도를 알린다. PoolChecker는 이러한 정보를 기록해두고, 특정 시간 동안 사용되고 있지 않다고 판단하면, 해당 GPU를 강제로 회수한다.



(그림 1) GPU 공유 메카니즘

2.2.2 GPU Manager

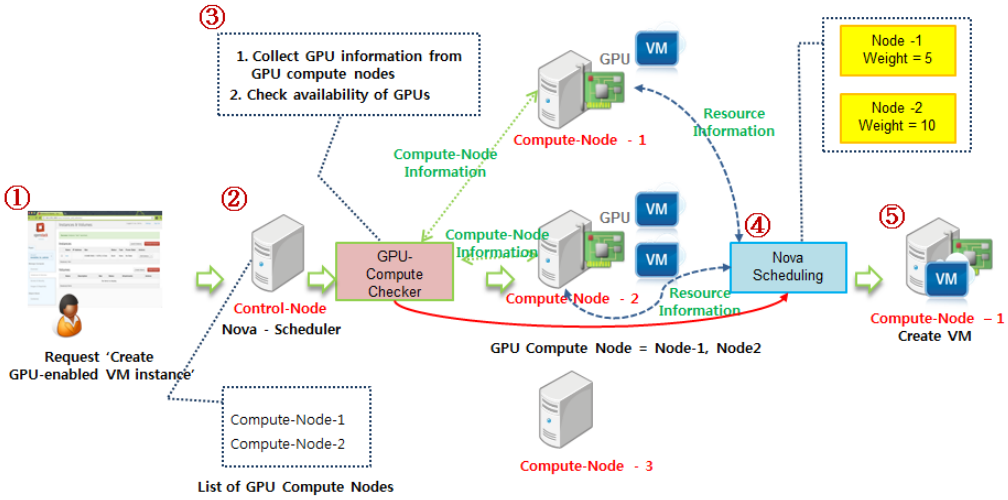
GPU-Manger는 GPU-Admin으로부터 수행해야 할 GPU 어플리케이션과 그 파라미터를 입력으로 받는다. GPU 할당 정보를 파악하고, 사용 가능한 GPU가 있으면 GPU-Admin에게 GPU 할당을 요청하고 할당이 정상적으로 완료될 때까지 대기한다. 할당이 완료되면, VM은 GPU를 사용할 수 있는 상태가 되며 GPU-Manager는 입력으로 받은 GPU 어플리케이션과 그 파라미터를 이용하여 어플리케이션의 수행을 허용한다. GPU Manager는 GPU 어플리케이션이 GPU 사용을 종료할 때까지 대기하며, 어플리케이션이 종료되면 GPU-Admin과의 연결을 종료하고 GPU 해제를 요청한다. GPU 해제가 완료되면 GPU-Manager는 자신의 역할을 정상적으로 종료한다. 이 방법은 GPU-Admin과의 연결이 종료되면 GPU-Admin은 GPU를 즉시 회수함으로써 GPU의 활용성을 높일 수 있다.

CheckGPUOn 쓰레드는 GPU Admin으로부터 특정 요청이 오기를 기다린다. GPU Admin은 GPU가 VM에게 할당된 이후부터 주기적으로 GPU의 사용도를 확인하는 요청 메세지를 보내는데, 이에 응답하여 현재 자신의 VM에서 GPU의 사용도를 체크하여 전달하는 기능을 담당한다.

사용자 GPU 어플리케이션이 CUDA 라이브러리나 OpenCL 라이브러리를 호출하면 이때 비로소 GPU가 VM에 할당된다(hot plug-in). 반대로, GPU 어플리케이션이 종료되는 시점에서는 할당된 GPU 자원을 해제하고(hot plug-out) CheckGPUOn을 재설정하여 GPU Admin에게 알린다. GPU Admin은 이것을 GPU Pool에서 사용가능한 자원으로 재설정한다.

3. GPU 확장형 OpenStack

OpenStack에는 VM의 관리 기능(생성/삭제/시작/중지 등)을 수행하는 Nova, 객체 스토리지 Swift, 인증을 위한 Keystone, VM 이미지를 관리하는 Glance, 네트워크를 관리하는 Neutron, 블록 스토리지를 관리하는 Cinder, 웹 기반 대시보드 Horizon으로 구성되어 있다. 추가적인 서비스로는 텔레미터 서비스 Ceilometer, 오케스트레이션 서비스 Heat, 데이터베이스 서비스 Trove가 있다. OpenStack 기반의 HPC는 Heat를 통해 이루어진다. GPU 컴퓨팅을 위한 확장형 OpenStack은 기존



(그림 2) GPU 인지형 Nova 스케줄링 워크플로우

OpenStack에 GPU 자원 공유 스케줄러 및 GPU 인지형 스케줄러를 추가한 것으로, OpenStack 위에 패치 형태로 설치된다.

Nova는 GPU를 인식하지 못하기 때문에 GPU 자원을 가진 노드가 어느 것인지 알 수 있도록, 또한 이들 중에서 어느 노드에 VM 이미지를 개시할지 결정할 수 있도록, Nova를 수정 및 추가하였다. 또한 할당된 물리적 자원에서 개시된 GEVM이 제대로 동작하는지, GEVM이 어떤 작업을 실행하고 있는지 등을 볼 수 있도록 Horizon도 일부 수정하였다.

3.1 GPU 인지형 Nova 스케줄러

GPU 인지형 Nova 스케줄러는 사용자의 요청을 받아 이에 적합한 GPU 컴퓨팅 노드를 사용자에게 할당한다. 이를 위해 GPU 인지형 Nova 스케줄러는 각 컴퓨팅 노드로부터 정보를 받아서, 어느 컴퓨팅 노드에 GPU가 탑재되어 있는지, 어느 GPU 컴퓨팅 노드가 사용 가능한지 확인하고, 사용 가능한 GPU 컴퓨팅 노드 리스트를 생성하여 Nova Scheduling 모듈에게 제공한다. GPU-Compute Checker는 이러한 리스트를 생성하는 역할을 수행하는 모듈로, 새로 추가되었다. Nova Scheduling 모듈은 이 리스트에서 GPU 컴퓨팅 노드를 선정하여 사용자에게 할당한다(그림 2 참조).

3.2 GPU 사용 가능 가상 머신(GEVM)의 생성

확장형 OpenStack에서 GEVM의 생성은 다음 절차를 거친다.

1) VM을 생성할 때, GPU를 선택하고 GEVM의 생성을 요청한다.
2) GPU 인지형 Nova 스케줄러는 이 요청을 받아 GPU 노드(GPU가 장착된 컴퓨팅 노드)를 찾는 한편, Glance를 통해 GEVM 이미지를 찾는다.
3) GPU 인지형 Nova 스케줄러가 GPU 노드를 사용자에게 할당하면
4) Glance는 GPU 노드에 GEVM 이미지를 적재하고
5) GPU 인지형 Nova 스케줄러는 GEVM 이미지를 동작시켜 GEVM을 생성하고 사용자에게 제공한다.

이를 위해, OpenStack 위에 GPU 인지형 Nova 스케줄러가 설치되어 있어야 하며, GPU 노드에는 GPU Admin이 미리 설치되어 있어야 한다. 위와

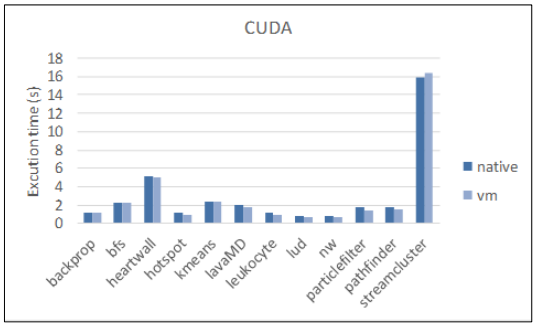
같은 과정을 통하고 나면, GPU 노드에서는 운영체제에 의해 GPU Admin이 동작하고, GEVM에서는 GPU Manager가 동작한다. 앞에 GPU 가상화에서 언급했듯이, 이들은 GPU의 원활한 공유를 위해 GPU Pool 관리에 필요한 정보를 주고받는다. GEVM은 GPU 컴퓨팅을 위한 CUDA나 OpenCL 라이브러리를 포함하고 있다.

OpenStack의 Glance는 여러 가지 VM 이미지를 등록하고 관리한다. 관리자는 사용자의 요구에 따라 필요한 VM 이미지를 만들어서 Glance에 등록한다. GEVM 이미지의 생성 및 관리도 이와 같은 메카니즘으로 동작한다. 따라서 GEVM 이미지를 생성하여 Glance에 등록하면 된다. 이를 위해 관리자는 VM을 하나 할당 받아서, GPU Manager 및 CUDA/OpenCL 라이브러리를 설치하고 종료한 후, 해당 VM 이미지의 스냅샷을 만들어 Glance에 GEVM 이미지로 등록한다

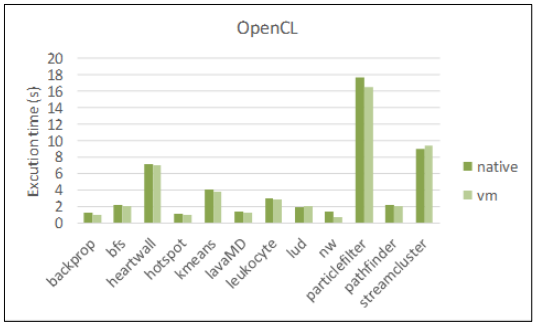
4. 성능 평가

PCIe hot plug 방식으로 GPU를 VM에게 할당하고 해제하는 GPU 자원 공유 스케줄러와 GPU 인지형 Nova 스케줄러에 의한 확장형 OpenStack이 오버헤드를 어느 정도로 내포하는지 실험으로 확인하였다. 이 실험에서 사용한 벤치마크는 Rodinia[26]로, 여러 가지 GPU 어플리케이션을 CUDA, OpenCL 버전으로 동작시킬 수 있도록 구성되어 있다.

실험 환경은 5대의 노드가 1Gb 이더넷으로 연결되어 있고, 각 노드는 CPU Xeon E5, 32GB RAM, NVIDIA Quadro FX3800 GPU로 구성되어 있다. (그림 3과 4)는 각각 CUDA와 OpenCL 코드로 구성된 Rodinia 벤치마크를 수행한 결과이다. 이들 그림에서 native와 vm은 각각 PM에서 그리고 VM에서 Rodinia 응용을 수행한 실행시간을 나타낸다. PM과 VM에서 수행한 CUDA 및 OpenCL 코드의 실행 시간은 길어야 0.2초 이내의 시간차를 나타내고 있으며, 경우에 따라서는 PM의 실행 시간이 VM의 실행 시간보다 길기도 하다. 전자는 가상화 오버헤드로 native 대비 2% 이내이다. 후자는 자원 공유로 인한 상승효과로 추측할 수 있으나 향후 상세한 모니터링을 통한 분석이 필요하다.



(그림 3) 확장형 OpenStack 오버헤드: CUDA



(그림 4) 확장형 OpenStack 오버헤드: OpenCL

5. 관련 연구

기존의 클라우드에서 GPU 컴퓨팅은 두 가지로 나눌 수 있다. GPU를 디스플레이나 가시화를 목적으로 사용하거나 고성능 컴퓨팅을 위한 주계산 자원으로 사용한다. 전자는 GPU를 시간적 및 공간적으로 분할하여 하나의 GPU를 가능한 한 많은 가상 머신에게 할당한다[11, 12, 13, 14]. NVIDIA의 vGPU를[14] 이용하여 OpenStack에서 클라우드 서비스를 지원하는 경우도 이에 해당한다. 후자는 [20, 21] 클라우드에서 다수의 가상 머신이 PCIe 직접 통로나 rCUDA를 통해 GPU 자원을 사용할 수 있게 한다. [20, 21]에서 가상 머신은 PCIe 직

접 통로 대신에 rCUDA를 통해 GPU를 사용한다. 이것은 GPU가 모든 물리적 노드에 탑재되어 있지 않은 고성능 컴퓨팅 환경에서 가상 머신 간에 GPU 자원을 원격으로 공유하기 위한 목적으로 개발되었다. 이 접근 방법은 가상 머신 간에 GPU 자원 경쟁이 심하지 않은 이질적 컴퓨팅 환경에 적합하다. 이와 달리, 본 연구는 GPU를 주계산 자원으로 간주하기 때문에 가상 머신 간에 GPU에 대한 자원 경쟁을 적절히 통제해야 할 경우를 전제하였다. 이러한 클라우드 환경에서 GEVM의 생성 및 관리, GPU 활용 극대화 등을 위해, GPU 공유 스케줄러 및 GPU 인지형 Nova 스케줄러를 확장형 OpenStack에 추가하였다.

6. 결론

본 연구에서는 클라우드 서비스에서 고성능 GPU 컴퓨팅이 가능하도록 GPU 확장형 OpenStack을 제시하였다. 확장형 OpenStack의 과립형 GPU 자원 공유 스케줄러는 직접 통로에 근거한 hot plug-in/out 방식으로 VM에게 GPU 접근 허용을 제공하고 있으며, 다수의 VM 간에 원활한 GPU 공유 메카니즘을 제공한다. 또한 확장형 OpenStack의 GPU 인지형 Nova 스케줄러는 각 노드의 GPU 유무를 확인하고 VM에게 물리적 자원을 할당하고 회수할 수 있도록 Nova Compute를 확장하였다. GEVM 이미지의 생성 및 관리는 Glance를 그대로 활용한다. GEVM 이미지를 물리적 자원에 할당하여 개시하면, 대쉬보드를 통해 실행 상태를 모니터링할 수 있도록 Horizon을 수정하였다. OpenStack의 Heat는 다수의 VM으로 구성된 가상 클러스터를 생성하고 개시할 수 있는 도구이다. Nova 스케줄러를 확장한 GPU 인지형 Nova 스케줄러가 GEVM을 지원하기 때문에, 기존의 Heat를 이용하여 다수의 GEVM으로 구성된 가상 GPU 클러스터를 생성하여 일괄처리 형태의 이질적 HPC가 가능하다.

GPU 확장형 OpenStack 위에서 Rodinia 벤치마크를 이용하여 실험하였다. 실험 결과는 GPU 확장형 OpenStack의 VM의 오버헤드가 PM에 비해 기껏해야 2% 수준에 불과하다.

OpenStack은 다수의 VM을 동시에 스케줄링할 수 있는 집단 스케줄러(gang scheduler)를 내포하지 않기 때문에 대화식의 이질적 HPC 클라우드 서비스 제공은 불가능하다. 이를 위해서는 향후 집단 스케줄러의 추가 개발이 필수적이다. 또한 GPU뿐만 아니라 인텔의 Xeon Phi에 대한 가상화 기술 개발이 필요하다.

Acknowledgment

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원(No.B0101-15-0104, 유전체 분석용 슈퍼컴퓨팅 시스템 개발)과 2016년도 정부(미래창조과학부)의 지원(No.K-16-L02-C01-S02, 국내외 핵심 과학기술정보 자원 개발 및 구축)으로 수행된 연구임.

참고문헌

〔1〕 Amazon EC2, http://aws.amazon.com/ec2
〔2〕 Cloud market trends, http://www.forbes.com/sites/louiscolumbus/2015/09/27/roundup-of-cloud-computing-forecasts-and-market-estimates-q3-update-2015/#4b8a0b0f6c7a
〔3〕 E. W. Pugh, L.R. Johnson, J. H. Palmer., IBM's 360 and early 370 systems, MIT Press, ISBN 0-262-16123-0
〔4〕 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. In Proceedings of the 19th ACM SOSP, 2003
〔5〕 KVM, http://www.linux-kvm.org/
〔6〕 OpenStack, https://www.openstack.org/
〔7〕 OpenStack Foundation

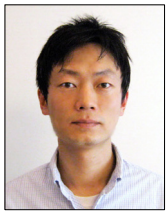
https://www.openstack.org/foundation/
〔8〕 Graphics Processing Unit, http://en.wikipedia.org/wiki/Graphics_processing_unit
〔9〕 GPU energy efficiency, http://www.nvidia.com/object/gcr-energy-efficiency.html
〔10〕 S. Mittal and J. Vetter, "A Survey of CPU-GPU Heterogeneous Computing Techniques," ACM Computing Surveys 2015
〔11〕 V. Gupta, A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, P. Ranganathan, "GViM: GPU-accelerated virtual machines, Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing", p.17-24, March 31-31, 2009
〔12〕 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
〔13〕 Giunta G., 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〕 NVIDIA vGPU, https://www.vmware.com/kr/products/vsphere/features/vGPU
〔15〕 PCI express direct pass-through, http://www.ibm.com/developerworks/library/l-pci-passthrough/l-pci-passthrough-pdf.pdf
〔16〕 H. Jo, J. Jeong, M. Lee, D.H. Choi, "Exploiting GPUs in virtual machine for BioCloud," BioMed Research International 2013
〔17〕 OpenCL, http://www.khronos.org/opencl/
〔18〕 CUDA, https://developer.nvidia.com/cuda-downloads
〔19〕 Rodinia Benchmark, https://www.cs.virginia.edu/~skadron/wiki/rodinia/index.php/Rodinia:Accelerating_Compute-Intensive_Applications_with_Accelerators
〔20〕 T. Jun, V. Dung, M. Yoo, D. Kim, H. Cho and J. Hahm, "GPGPU enabled HPC Cloud Platform based on OpenStack," Proceedings on the International Conference for High Performance Computing, Networking, Storage and Analysis, 2014
〔21〕 B. Varghese, J. Prades, C. Reano and F. Silla, "Acceleration-as-a-Service: Exploiting Virtualised GPUs for a Financial Application," Proceedings on IEEE 11th International Conference on e-Science, 2015
〔22〕 박상면, 문영성, "클라우드 컴퓨팅 환경에서 자원의 TDP와 사용률을 이용한 소비전력 예측 방안," 한국차세대컴퓨팅학회 논문지, Vol.12 No.1, 2016, pp.25-32
〔23〕 김진택, 조수지,오동하, 노준길, "클라우드 컴퓨팅 기술 스택," 한국차세대컴퓨팅학회 논문지, Vol.11 No.6, 2015, pp.79-89
〔24〕 최정열, 클라우드 데이터 센터의 에너지 효율성 평가 프레임워크, 한국차세대컴퓨팅학회 논문지, Vol.10, No.4 2014, pp.66-76

저자소개

◆ 최 동 훈

- 1981년 서울대학교 계산통계학과 학사
- 1983년 KAIST 전산학과 석사
- 1989년 Northwestern University EECS 박사
- 1989년~1992년 한국국방연구원 선임연구원
- 1992년~1999년 동덕여자대학교 부교수
- 2005년~현재 한국과학기술정보연구원 책임연구원
- 관심분야: 데이터베이스, 고성능 클라우드 컴퓨팅 등

◆ 조 희 승



- 2000년 서강대학교 컴퓨터학과 학사
- 2010년 KAIST 전산학과 석사
- 2010년 KAIST 전산학과 박사
- 2010년~현재 전북대학교 IT공학부 부교수
- 관심분야: 운영체제, 컴퓨터 아키텍처, 클라우드 컴퓨팅 등

◆ 박 기 웅

- 2005년 연세대학교 Computer Science 학사
- 2007년 KAIST Electrical Engineering 석사
- 2012년 KAIST Electrical Engineering 박사
- 2008년 Microsoft Research Asia, Wireless and Networking Grourp, Research Intern
- 2009년 Microsoft Research Redmond, Network Research Group, Research Intern Fellow
- 2012년 국가보안기술연구소 연구원
- 2012년~현재 대전대학교 정보보안학과 조교수
- 관심분야: 시스템 보안, 모바일-클라우드 컴퓨팅, 보안 프로토콜, 디지털 포렌식 등