Issue Report

2020. 02. 26. IS-092

AI 성능개선을 위한 클라우드 GPU 가상화 기술의 현황과 향후방향

Current state and future directions of GPU virtualization technology in the Cloud computing for improving AI performance

안성원 (swahn@spri.kr)†

SPRi 소프트웨어정책연구소

○ 이 보고서는 「과학기술정보통신부 정보통신진흥기금」을 지원받아 제작한 것으로 과학기술정보통신부의 공식의견과 다를 수 있습니다.

○ 이 보고서의 내용은 연구진의 개인 견해이며, 본 보고서와 관련한 의문사항 또는 수정・보완할 필요가 있는 경우에는 아래 연락처로 연락해 주시기 바랍니다.

- 소프트웨어정책연구소 AI정책연구실 안성원(swahn@spri.kr) 선임연구원

《 요 약 문 》

GPU(Graphic Processing Unit)를 클라우드 컴퓨팅에 활용하는 연구와 상용화가 점차 확산되고 있다. 인공지능이 주목받게 되면서, 이를 동작시키기 위한 인프라이자 데이터를 공급하는 저장소인 클라우드도 수요와 서비스가 늘고 있다. 인공지능의 성능은 데이터의 학습량과 그 학습에 필요한 컴퓨팅 파워에 따라 달라져서 고성능 클라우드 컴퓨팅이 요구된다. GPU는 다수의 코어를 가져 병렬 처리에 유리하며, CPU 대비 단위 코어당 가격이 저렴하다. 이에 따라 GPU를 클라우드 컴퓨팅 자원으로 활용하는 방법이 등장했다. 그런데 클라우드 컴퓨팅은 가상화가 전제되는 기술이기 때문에 GPU를 온전한 클라우드 자원으로 활용하기 위해서는 역시 가상화가 필요하다. 그간 GPU 가상화는 어렵고 복잡한 기술이었으나, 여러 연구를 거쳐 최근에는 하드웨어에서 가상화를 지원하는 제품이 출시되고 있다. 이 보고서에서는 GPU 가상화 기술의 등장배경과 그간의 현황에 대하여 살펴보고, 클라우드 GPU 가상화 연구에서 기술 수준을 발전시킬 수 있는 과제와 방향을 논의하고자 한다.

《 Executive Summary 》

Research and commercialization of using GPU(Graphic Processing Unit) for cloud computing is gradually spreading. AS AI rises, Cloud computing that is the infrastructure to operate AI and the storage that supplies data, is also increasing in demand and services. The performance of AI depends on the amount of data learned and the computing power required to learn it, then it require high performance Cloud computing. GPUs have a large number of cores, which are advantageous for parallel processing, and are cheaper per core than CPUs. This has led to the emergence of leveraging GPUs as Cloud computing resources. Cloud computing is a technology on which virtualization is fundamental. Therefore, virtualization is also required to utilize the GPU as a complete cloud resource. GPU virtualization has been a difficult and complex technology, but many researches have recently released products that support virtualization in hardware. This paper examines the background and the current state of GPU virtualization technology, and discusses the challenges and directions to advance the level of technology in cloud GPU virtualization research.

《 목 차 》

《 Contents 》

1. 논의배경

□ AI의 핵심 인프라로써 점점 더 높은 성능이 요구되는 클라우드 컴퓨팅

클라우드 컴퓨팅은 인공지능(Artificial Intelligence, AI)의 성능향상을 위한 학습데이터를 제공하는 저장소이자 컴퓨팅 파워를 제공하는 필수 인프라다. 클라우드 컴퓨팅은 이기종의 컴퓨팅 노드를 활용하여 컴퓨팅 인프라의 소유 및 유지비용을 줄이고, 더 높은 성능과 범용성, 에너지 효율성을 높이는 장점이 있다. 이러한 장점들은 AI의 성능을 올리는 데에 매우 적합하여, 많은 AI 연구와 서비스에 클라우드 컴퓨팅이 활용되고 있다. 최근 AI를 활용한 서비스가 우리생활 곳곳에 실현되고 있고 니즈(Needs)가 점차 다양해짐에 따라 클라우드 컴퓨팅의 보다 안정적이고 우수한 성능이 요구되고 있다.

□ 클라우드 컴퓨팅 성능향상을 위해 가성비 좋은 GPU 활용의 확대

높은 클라우드 컴퓨팅 성능을 위해서는 빠른 작업 처리가 가능한 고성능 프로세스 유닛(Process Unit, 이하 코어(Core))이 다수 필요하다. 특히 딥러닝(Deep Learning)과 같은 AI는 많은 수의 데이터를 학습할수록 성능이 좋아지는데, 이를 위해서는 다중 코어를 갖춘 병렬컴퓨팅 환경이 사실상 필수적이다. 다중 코어를 갖춘 병렬컴퓨팅 환경을 구축하기 위한 최근의 패러다임은 강력한 성능의 범용코어가 있는 멀티코어(Multi-Core) CPU[1]와 단순하지만 더 많은 코어를 가지는 GPU[2]와 같은 가속기(Accelerator)의 두 가지 아키텍처를 결합하는 방식을 추구하고 있다. 이러한 이기종 결합의 컴퓨팅 시스템은 비용과 성능의 효율성을 모두 높일 수 있는 방식으로 지난 2016년 구글의 알파고(AlphaGo)에도 활용되었다. 그래픽처리장치인 GPU를 클라우드 컴퓨팅에 활용하는 이유는 비용 효율성이 좋기 때문이며, 어려운 이기종 시스템의 결합이 있어야 한다.

이 보고서에서는, AI를 위한 필수 인프라인 클라우드 컴퓨팅에서 GPU를 활용하는 목적과 GPU를 활용하기 위해 연구해왔던 기술들을 살펴보고, 향후 기술발전을 위한 연구 주제와 방향을 논의하고자 한다.

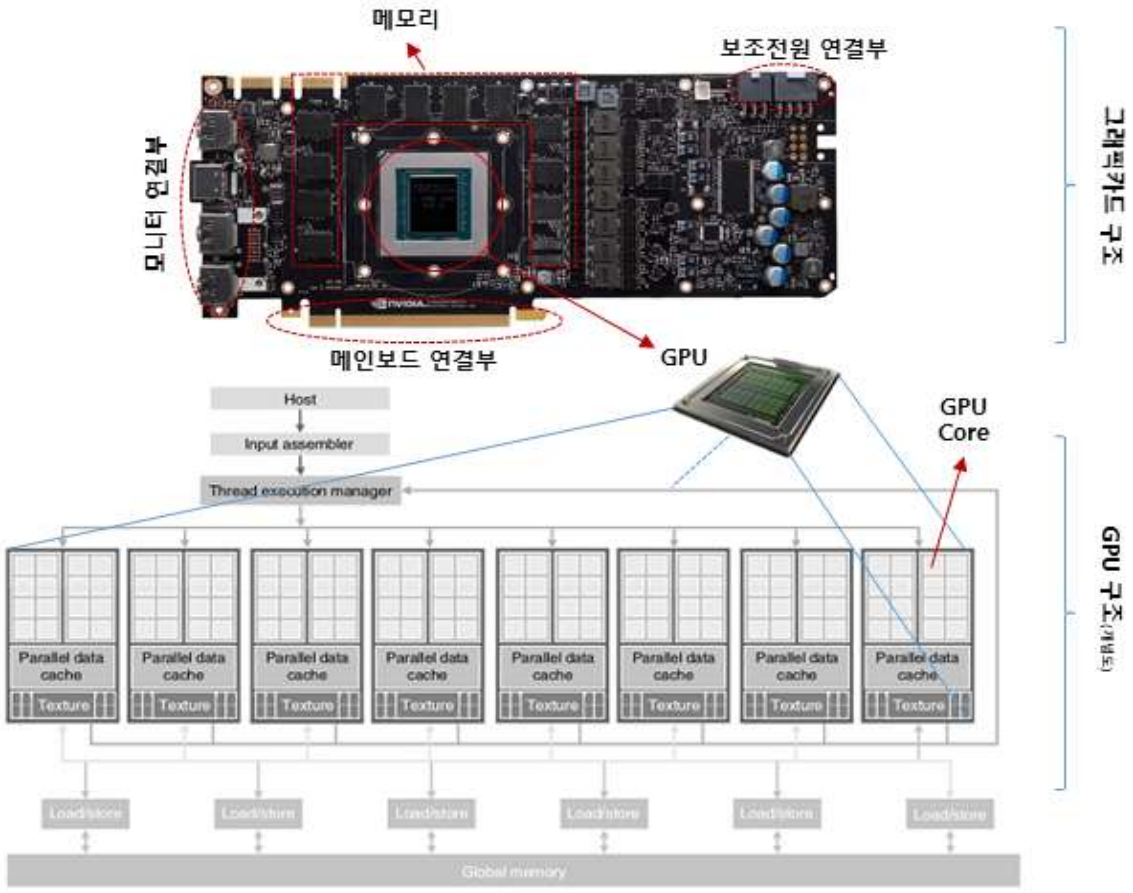
1) 두 개 이상의 독립 연산 코어를 갖는 중앙처리장치(Central Processing Unit)로 보통 다수의 코어를 하나의 칩에 집적해 놓은 형태, 코어 수는 2개 에서 72개 까지 다양함
2) Graphic Processing Unit, 그래픽처리장치

2. GPU 기술

2.1. GPU의 구조

□ GPU는 컴퓨터의 그래픽을 처리하기 위한 장치로 다수의 코어를 보유

GPU는 컴퓨터 모니터의 수천~수백만 개의 픽셀(Pixel[3])로 투영되는 그래픽의 처리를 위한 장치로 병렬연산에 특화되어 있다. 때문에 GPU는 구조적으로 다수의 코어를 보유하고 있으며, [그림 2-1]과 같다. [그림 2-1] 상단부는 PC에 일반적으로 사용하는 그래픽카드의 구조를 나타낸다. 중앙에 GPU가 탑재되어 있으며, 그래픽연산에 필요한 메모리, 컴퓨터 메인보드의 슬롯(PCIe[4])연결부, 그래픽 처리 결과를 모니터에 송신하는 연결부로 구성되어 있다.



※ 출처 : NVIDIA (nVidia GTX 1080 ti 모델의 사례)

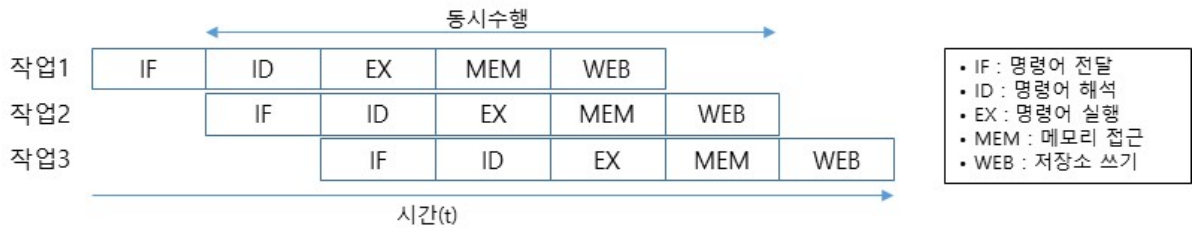
[그림 2-1] 그래픽카드와 GPU의 구조

3) 모니터에서 화면을 구성하고 있는 가장 작은 단위인 네모 모양의 작은 점, 화소
4) Peripheral Component Interconnect Express, 메인보드(Main Board, Mother Board)에 그래픽, 사운드, TV카드 등 각종 확장카드를 사용하기 위해 만들어진 규격

그래픽 카드의 중앙에 위치한 GPU는 그림 하단부와 같이 다중 코어로 구성되어 있다. GPU는 병렬 응용 프로그램을 효율적으로 실행하기 위해 기존의 멀티 코어 프로세서와는 근본적으로 설계가 다르다. 기존의 멀티 코어 프로세서는 지연시간(Latency)을 줄여서 처리 속도를 높이는 것을 목표로 설계되었다. 주로 단일 작업에 대해 빠른 속도의 연산 수행을 한다. 반면, GPU는 처리량(Throughput) 중심으로 설계가 되어 있다. 처리량을 높이기 위해 비교적 구조가 간단한 수천 개의 코어들과 높은 대역폭의 메모리를 탑재한 형태를 띠고 있다. 이러한 설계는 병렬로 동작하는 프로그램이 GPU에서 적절히 분배되어 실행되게 함으로써 응용 프로그램의 실행 처리량을 극대화 할 수 있다.

예를 들어, 실행되고 있는 일부 쓰레드(Thread)[5]가 메모리 접근을 위해서 긴 대기 시간을 갖는다고 가정할 때, 이 작업이 끝날 때 까지 기다렸다가 다른 작업을 이어서 수행하는 것은 전체 작업량의 기준으로 볼 때 비효율적이다. 이러한 작업 중 대기 시간을 숨기기 위해 GPU의 하드웨어 스케줄러(Scheduler)는 실행 가능한 또 다른 쓰레드를 스케줄링(실행, 접근 등) 함으로써 병렬 처리를 가능하게 할 수 있다. 이렇게 하면 각 쓰레드의 개별 실행 시간은 길어질 수도 있지만, 전체 프로그램 입장에서 총 실행 처리량은 향상 시킬 수 있는 장점이 있다. 이를 파이프라인(Pipe-line) 기법이라고 하며, 파이프라인은 멀티코어 CPU에서도 활용되는 방식이지만, GPU는 더 많은 수의 병렬 코어를 통해 훨씬 더 방대한 양의 병렬처리를 가능하게 한다.

병렬처리(Parallel processing)

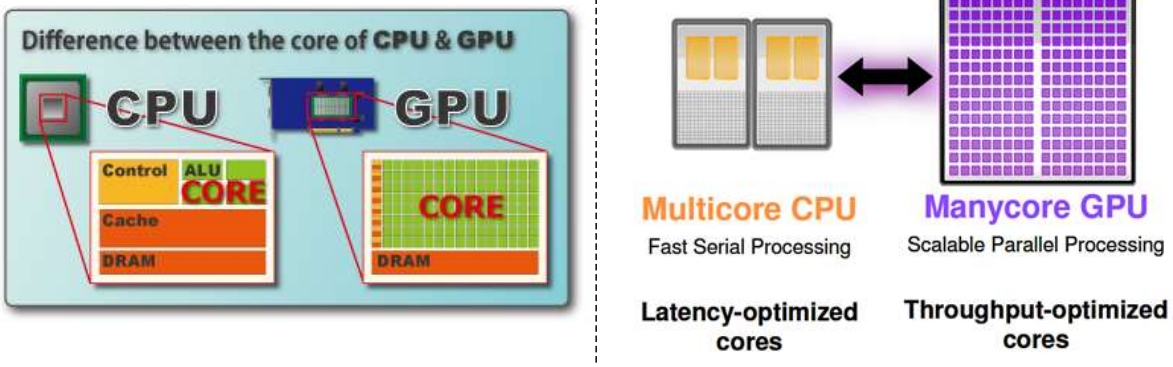


[그림 2-2] 병렬처리 기법의 예시 (파이프라이닝 기법)

○ 병렬처리는 작업을 최대한 동시에 실행하여 시스템의 성능을 올리는 방법으로 파이프라인, 멀티쓰레드, 분산컴퓨팅 외에도 여러 종류가 있음

5) 프로그램이 실행될 때의 일련의 과정을 프로세스(Process)라고 하며, 이 프로세스 내에서 실행되는 세부 흐름의 단위를 쓰레드(Thread)라고 함. 또한, 하나의 프로세스 내에서는 다수의 쓰레드가 실행될 수 있으며, 이러한 실행 방식을 멀티쓰레드(Multi-thread)라고 함.

CPU와 GPU의 비교



※출처 : NVIDIA, Microsoft

[그림 2-3] CPU와 GPU의 구조 비교

- CPU는 빠른 일련의 작업(Fast serial processing)을 목표로 하여 지연시간에 최적화된 코어를 보유한 단일 또는 멀티코어(Multi-core) 구조
- GPU는 매니코어(Many-core)의 형태로 확장 병렬 프로세싱(Scalable parallel processing)을 목표로 하여 출력값에 최적화된 코어로 구성

2.2. 왜 GPU를 컴퓨팅 자원으로 쓰는가?

□ 그래픽 처리가 아닌 다른 일반적인 목적으로 GPU를 활용하는 GPGPU

앞서 살펴본 바와 같이 GPU는 CPU보다 많은 수의 코어(Core)를 가지고 있다. GPU가 보유한 코어들은 일반적으로 CPU의 코어보다 성능은 낮지만, 이러한 코어를 수 백 ~ 수 천 개 가지고 있어서 CPU보다 단위 코어 당 가성비가 높다. 이는 같은 비용으로 더 많은 수의 계산 가용한 코어를 보유할 수 있다는 의미이다.

<표 2-1> 상용 CPU와 GPU의 단위 코어 당 가격 비교

구분	모델 명		코어 수(개)	가격($)	코어 당 가격($)
CPU	Intel Core-i9 9900k (3.6Ghz) (`18.10 출시)		8	549	68.6
GPU	nVidia Geforce RTX 2080 Ti (`18.9 출시)		4,352	1,199	0.275

※자료 : Intel, NVIDIA

특히, 높은 성능의 단일 코어가 필요한 계산 업무보다, 단일 코어대비 속도는 약간 떨어지더라도 훨씬 많은 양의 좀 더 단순한 계산 업무를 수행하는 데에 유리하다. 좀 더 직관적인 예를 들어, 4명의 대학원생과 100명의 유치원생 그룹이 있다고 가정할 때, 1개의 미적분 문제를 푸는 것은 대학원생 그룹이 유리할 수 있으나, 100개의 편지봉투에 우표를 붙이는 작업은 유치원생 그룹이 훨씬 유리하다. 앞선 예에서 대학원생 그룹은 CPU, 유치원생 그룹은 GPU에 비유할 수 있다. 이처럼, 본디 그래픽처리가 목적인 GPU를 다른 일반적인 목적으로 활용하는 것을 GPGPU[6]라고 한다. GPU를 범용적인 컴퓨팅 자원(Resource)[7]으로 활용하면 상대적으로 낮은 비용을 들여 컴퓨팅 파워를 확보할 수 있다.

□ 한계에 다다른 CPU의 성능

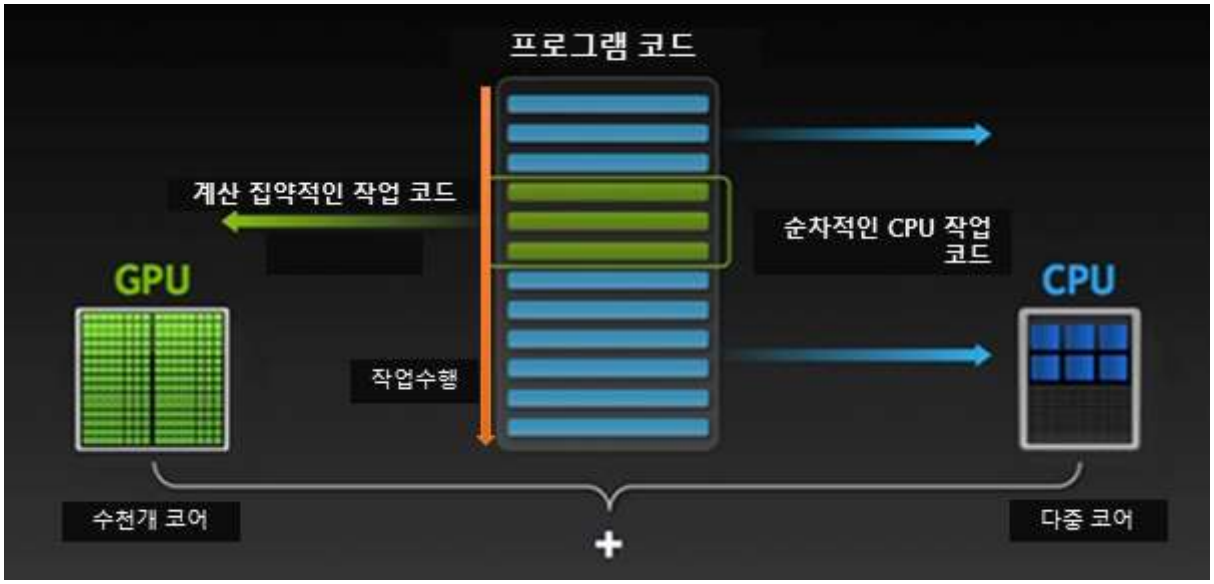
현재의 CPU는 코어 수가 점차 증가하고 있음에도 불구하고, 정교한 제어 로직과 대형 캐시 메모리를 사용하여 순차 프로그램의 지연시간을 줄이는 것을 목표로 해왔다. 이러한 설계는 각 코어에서 순차적으로 수행되는 코드의 실행 시간을 줄이기 위한 것에 최적화되어 있으므로 하나의 프로세서 패키지에 더 적은 코어가 탑재되는 대신 각 코어에 복잡성은 증가하게 된다. 더군다나, CPU의 집적도[8]를 높이기 위한 나노 공정은 현재 기술로 7nm[9]가 한계이며, 그 이하의 간격으로 집적을 할 경우 간섭이 발생하여 성능이 떨어진다.

또한, CPU의 처리속도를 나타내는 클럭(Clock) 수는 4Ghz 내외 수준을 효율성의 한계치로 보고 있다. 그 이유는 집적도 향상 및 클럭 수 향상에 따른 전력소비의 증가와 발열문제를 해결하는 것에 한계가 있기 때문이다. 이로 인해 현재의 CPU의 단일 코어 당 성능은 한계수준에 다다랐으며, 근래에 출시되는 CPU는 코어를 여러 개 두는 것을 통해 성능을 높이는 것으로 설계 방향이 잡혀있다. 물론, 여전히 복잡한 구조를 갖는 고성능 칩셋이기에 <표 2-1>에서 살펴본 바와 같이 단위 코어 당 가격도 비싸다.

6) General Purpose computing on GPU, 컴퓨터 그래픽 처리를 목적으로 하는 GPU를 전통적으로 CPU가 맡아왔던 응용프로그램의 계산에 활용하는 기술
7) CPU, 메모리, 네트워크, 서버, 스토리지, 애플리케이션 등 컴퓨터에서 가용한 요소
8) Level of integration, 실리콘 칩(반도체)에 집적할 수 있는 게이트(소자)의 수로 하나의 실리콘 칩에 논리 소자가 몇 개나 포함되어 있는지를 나타내는 정도
9) Nanometer(나노미터), 10억 분의 1미터(m)

□ 고성능 컴퓨팅 패러다임의 변화 : CPU + GPU

최근 고성능 컴퓨팅(HPC[10]) 연구자들은 프로그램의 수행 시간을 단축시키기 위한 방법으로 강력한 성능의 범용 CPU 코어가 있는 멀티 코어 프로세서와 단순하지만 많은 코어를 가지는 그래픽처리 가속기인 GPU의 두 가지 아키텍처를 결합한 컴퓨팅을 추구하고 있다. 멀티 코어 프로세서와 GPU를 결합한 이기종 시스템은 제어 집약적 구성 요소와 고도의 데이터 병렬 구성 요소를 모두 갖추고 있어서 고성능 컴퓨팅 응용 프로그램의 다양한 요구 사항을 충족시킬 수 있다. [그림 2-4]는 GPU와 CPU의 결합한 연산 방식을 나타낸다.



※ 출처 : NVIDIA (재편집)

[그림 2-4] GPU + CPU 아키텍처의 작업수행 사례

응용프로그램을 수행할 때 대량의 계산 작업이 필요한 계산 집약적인 부분을 GPU로 넘기고, 나머지의 코드는 CPU에서 처리하는 것으로 프로그램의 수행속도를 비약적으로 증가시킬 수 있다. 쉬운 이해를 위해 앞서 예로든 대학원/유치원생의 사례를 다시 응용해 보자. 프로그램 코드의 작업이 미적분 문제 3개를 풀다가, 우표붙인 편지봉투 500개를 만들고, 다시 2차방정식 6문제를 푸는 일련의 과정으로 되어 있다고 가정한다. 4명의 대학원생이 이 일련의 과정을 모두 마치는데 드는 시간은 제법 걸릴 것이다. 가장 많은 시간이 소요되는 병목구간은 편지봉투에 우표붙이기 작업으로, 1인당 125개의

10) High Performance Computing

편지봉투에 각각 우표를 붙여야 한다. 이 작업만 따로 떼어서 100명의 유치원생들에게 나눠준다면 1인당 5개씩만 만들면 되므로, 작업속도는 월등히 빨라진다. 병목 구간의 작업 시간이 단축될 뿐만 아니라, 유치원생들이 우표를 붙이는 동안 대학원생들은 나머지 방정식 문제를 풀고 있으면 된다. 전체 작업의 완료 시간이 크게 단축될 수 있다.

GPU를 사용하는 컴퓨팅 시스템이 점차 확산되고 있다는 점은 최신 Top500 목록에서 나타난다. 현재 슈퍼컴퓨터의 27% 이상[11]이 CPU와 GPU를 결합한 시스템이다. 또한, 이러한 패러다임은 가속화(19%(2017) → 27%(2019)) 되고 있다.

Top500 10위 내 슈퍼컴퓨터

○ Top500 목록 중 10위권 내의 슈퍼컴퓨터들의 절반이 GPU를 활용하는 것으로 나타남 (Summit, Sierra, Piz Daint, ABCI, Lassen 등)

<표 2-2> 전 세계 Top10 슈퍼컴퓨터

순위	이름	국가	시스템	코어 수 (개)	성능 (TFlop/s)
1	Summit	미국	IBM Power System AC922, IBM POWER9 22C 3.07GHz, ***NVIDIA Volta GV100***, Dual-rail Mellanox EDR Infiniband ,	2,414,592	148,600.0
2	Sierra	미국	IBM Power System AC922, IBM POWER9 22C 3.1GHz, ***NVIDIA Volta GV100***, Dual-rail Mellanox EDR Infiniband	1,572,480	94,640.0
3	Sunway TaihuLight	중국	Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway ,	10,649,600	93,014.6
4	Tianhe-2A	중국	TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000	4,981,760	61,444.5
5	Frontera	미국	Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR	448,448	23,516.4
6	Piz Daint	스위스	Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, ***NVIDIA Tesla P100***	387,872	21,230.0
7	Trinity	미국	Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect ,	979,072	20,158.7
8	AI Bridging Cloud Infrastructure (ABCI)	일본	PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, ***NVIDIA Tesla V100 SXM2***, Infiniband EDR	391,680	19,880.0
9	SuperMUC-NG	독일	ThinkSystem SD650, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path	305,856	19,476.6
10	Lassen	미국	IBM Power System AC922, IBM POWER9 22C 3.1GHz, Dual-rail Mellanox EDR Infiniband, ***NVIDIA Tesla V100***	288,288	18,200.0

※출처 : Top500.org
※주 : 시스템 내 GPU 활용은 볼드 이탤릭체 표기
※TFlop/s : 테라플롭스, 슈퍼 컴퓨터의 연산성능 척도로써, 1TFlop/s는 초당 1조(10^{12}) 회의 부동소수점(Flop)연산을 수행

11) nVIDIA, "Record 136 NVIDIA GPU-Accelerated Supercomputers Feature in TOP500 Ranking", 2019.11.19.

2.3. 클라우드에 GPU컴퓨팅이 필요한 이유

□ 클라우드 인프라에 대한 수요 증가 및 고성능 요구

AI 시대를 맞이하여 전통의 제조업, 서비스업 기업들도 디지털 전환을 추진하는 사례는 이미 많이 있다. 후발 주자들이 글로벌 경쟁력을 갖추기 위해 클라우드 서비스를 새롭게 제공하거나 인프라 확보를 추진하는 사례도 많다.

특히, AI 서비스의 오류를 줄이고 학습 성능을 끌어올려 완성도 있는 시스템을 확보하기 위해서는 클라우드가 필수적이다. 머신러닝(Machine Learning, 기계학습)의 방법론중 하나인 딥러닝(Deep Learning)은 수많은 데이터의 지속적인 반복학습을 통해 특징을 추출하고 판단의 근거로 삼는다. 여기서 '반복학습' 작업을 수행하는 데에는 많은 계산능력을 필요로 한다. 클라우드는 여러 개의 계산 자원들을 묶어서 활용할 수 있기 때문에 딥러닝에 요구되는 큰 계산능력을 단일 컴퓨터 보다 쉽게 제공할 수 있다. 이 때문에 클라우드 인프라에 대한 수요는 꾸준히 증가하고 있다. 또한, 쏟아지는 빅데이터를 정제하고 관리하며, 이를 기반으로 새로운 AI 서비스의 학습양분으로 활용하는 사례가 늘면서 더 빠르고 효율적인 클라우드 컴퓨팅에 대한 요구도 증가하고 있다.

보다 큰 계산능력과 효율성을 제공하기 위해서는 기존의 CPU로만 구성된 머신들을 통합하는 것만으로는 한계가 있다. 앞서 언급한 반복학습 작업도 당연히 최대 10개 내외의 코어를 갖는 CPU 보다 수천 개의 코어를 갖는 GPU가 동시 다발적으로 작업을 수행하는 것이 유리하다.

□ 가격대비 성능의 효율성 확보

클라우드의 높은 컴퓨팅 능력 요구사항을 만족시키기 위해 GPU 수준의 수 천 개 코어로 이루어진 시스템을 CPU 칩으로만 구축한다면 천문학적인 비용이 수반된다. 지난 2012년 구글은 구글브레인(Google Brain)이라는 신경망 시스템을 개발했다. 이 시스템은 고양이 이미지 1천만 개를 학습해서 고양이를 분별해 내는 인지 능력을 보여주었다. 그런데 이 시스템은 CPU 서버 1천개를 병렬로

연결한 딥러닝 클라우드 인프라를 활용하였다. 당시 인프라 구축비용은 50억 원 이상이었으며, 전력소모는 60만 와트(W)에 달하여 구축과 운영에 많은 비용이 소요되었다. 이후 GPU 제조 업체인 엔비디아(NVIDIA)에서 GPU를 활용한 서버 3대로 동일한 성능을 제공할 수 있음을 입증[12]했고, 구축비용은 3,300만원 수준에 전력 소모량도 4천 와트에 불과했다.

상대적으로 복잡한 매커니즘을 갖는 CPU들 간의 업무 분배 및 통합관리를 위한 시스템의 구현 또한 어려운 일이다. 일례로, CPU 제조업체인 인텔(Intel)에서 Xeon Phi[13]라는 72개(4세대 기준)의 CPU 코어로 이루어진 병렬처리 전문 제품을 상용화 하였으나, 병렬컴퓨팅 시장에서 GPU 제품 대비 효율성 측면의 경쟁에서 밀렸으며, 제품 생산을 중단을 발표한 것이 현실이다.[14]

앞서 언급하였지만, 빅데이터를 전처리하고 가공하는 일과 같은 작업은 코어의 클럭 수가 높은 것 보다, 코어의 숫자가 많아야 유리한 작업에 속한다. GPU 코어의 클럭 수가 현재의 CPU 코어 대비 낮더라도 코어의 숫자가 압도적으로 많고, 최근 출시되는 GPU 코어의 클럭 수(1.6Ghz)[15]는 수년전의 CPU 클럭 수 못지않은 성능을 가지고 있다. 이것은 하드웨어의 발전은 속도 대비 가격은 상대적으로 떨어지는 무어의 법칙을 따른다고 볼 수 있다. 물론 최근 들어 하드웨어 발전 속도의 증가가 무어의 법칙을 넘어섰으나, 여전히 큰 흐름에서 성능과 가격의 반비례는 적용된다.

□ 구글, MS, IBM, Apple 등 현재 AI 시장을 리딩하는 클라우드 기업들은 모두 GPU를 활용하여 컴퓨팅 파워를 확보

글로벌 클라우드 기업들이 자사의 클라우드 인프라에 GPU를 활용 사례는 쉽게 찾아볼 수 있다. 글로벌 자산가치 순위 Top 5에 드는 기업들은 공교롭게도 모두 클라우드 인프라를 보유하고 있거나 서비스 하고 있는 기업들이며, 이들은 AI 시장도 리딩하고 있다. <표 2-3>은 글로벌 기업가치 순위를 나타낸다. 2위부터 4위까지의 기업들은 자체 클라우드 인프라를 보유하고 있고, 이를

12) NVIDIA/스탠포드, “Deep Learning with COTS HPC Systems”, ICML, 2013.
13) Intel에서 제작한 병렬 연산용 마이크로 프로세서
14) TechPowerUp, “Intel is Giving up on Xeon Phi - Eight More Models Declared End-Of-Life”, 2018.7.
15) nVidia Geforce RTX 2080 Ti 모델 기준 (최대 클럭수: 1635 Mhz)

활용한 다양한 서비스를 제공한다. 1위와 5위인 애플, 페이스 북의 경우는 조금 다른데, 자체 인프라를 활용한다기보다는 타 회사의 클라우드 인프라를 활용하여 서비스를 제공하고 있다. 그러나 최근에는 클라우드 서비스를 제공하는 업체와 직접 협력하거나, 관련 업체를 인수하며 클라우드 인프라를 확보하는 방향으로 사업영역을 확장하고 있다. 애플은 자체 클라우드 인프라 구축을 위해 인재를 채용하는 등 사업을 추진하고 있으며('20.2), 페이스 북의 경우 최근 AI 클라우드를 구축을 위해 중국의 알리바바와 협력('19.9)하고 유럽의 클라우드 게임업체인 플레이기가를 900억에 인수('19.12) 하기도 했다.

<표 2-3> 전 세계 기업가치 순위 Top5

순위	기업	국가	시가총액 (십억 달러)	클라우드 인프라/서비스
1	Apple	미국	1,190	iCloud
2	Microsoft	미국	1,155	Azure
3	Google (Alphabet)	미국	905	Google Cloud Platform(GCP)
4	Amazon	미국	901	AWS
5	Facebook	미국	576	Intel, AWS, MS 클라우드 활용

※주: 2019.11.30.기준
※자료참고 : PWC, "Global Top 100 companies by market capitalisation", 2019.07.
iWeblists, "US commerce Stock Market Capitalization of the 50 Largest American Companies", 2019.11.30.
CorporateInformation, Top 100 list, 2019.

Apple의 클라우드 스토리지 이자 서비스인 iCloud는 Azure, AWS, GCP를 기반으로 서비스 하며, iCloud는 이들의 최대 고객 중 하나이다. MS의 클라우드 Azure는 N시리즈를 통해서 GPU 컴퓨팅을 제공하고 있다. Google은 알파고 당시 GPU 클러스터를 활용했으며, GPU와 유사한 병렬 구조의 텐서플로우(TensorFlow)[16]전용 TPU(Tensor Processing Unit)을 개발 하여 클라우드에 적용하고 있다. Amazon은 GPU 제조업체 엔비디아와 협력하여 GPU 가속 클라우드 서비스인 AWS EC2 P3를 보유 및 서비스 하고 있다.

이처럼 현재의 클라우드 인프라에는 GPU가 활용되고 있으며, 이에 대한 수요는 AI 및 클라우드 서비스와 함께 커지고 있다. 클라우드는 공급자 측면의 '컴퓨팅 자원의 효율적 활용'과 수요자 측면의 '필요시 컴퓨팅 자원 대여'라는 특성과 장점을 가짐과 동시에, '대표적인 병렬컴퓨팅 인프라'로써 활용되고 있다.

16) 2015년 11월 구글에서 공개, 머신러닝 프로그래밍용 공개SW기반 라이브러리

3. GPU 가상화 기술과 분류

3.1. GPU 가상화의 필요성과 클라우드 적용

□ GPU도 자원의 효율적인 활용을 위해 가상화가 필요

클라우드는 기본적으로 가상화(Virtualization)[17] 기술을 기반으로 구현되는데, GPU 컴퓨팅을 효율적으로 활용하기 위해서도 가상화 기술이 필요하다. GPU 자원을 가상화 하는 이유는 '자원의 효율적 활용'이라는 측면에서 가상화 기술의 목적과 맥락에 일치한다. 일반적인 경우, 시스템에서 GPU의 활용률은 평균 15% 수준에 불과[18]한 경우가 종종 있고, 자원 요구가 급증하는 특정시간(peak time)에는 컴퓨팅 자원이 부족한 경우가 발생한다.

GPU 자원을 가상화 하게 되면 매번 필요한 만큼만 자원을 최적화하여 활용할 수 있도록 하여 자원의 비효율적인 낭비 또는 부족현상을 최소화 할 수 있다. 또한, 유동적으로 조절하는 컴퓨팅 능력은 머신러닝 및 딥러닝의 작업 생산성을 향상시킨다. 가상화를 통해 기존의 가상머신(Virtual Machine, VM)[19]과 호환성도 보장할 수 있다.

□ GPU 가상화는 그동안 어려웠던 일

그동안 GPU의 가상화는 제품 제조업체의 하드웨어(HW) 및 소프트웨어(SW) 지원이 없어서 매우 어려운 문제였다. 가상화는 소프트웨어적으로 구현되고 동작하는 경우가 대부분이지만, 하드웨어차원의 호환이 없다면 구현이 어려운 이슈이다. 하드웨어를 생산하는 업체가 해당 하드웨어의 구조를 가장 잘 알고 있고, 하드웨어가 잘 동작하게끔 펌웨어(firmware)[20]를 탑재하거나, 시스템

17) 가상화 기술은 기존 하드웨어 장비를 마치 여러 개의 장치가 동작하는 것처럼 논리적으로 나누거나, 여러 개의 하드웨어 장비를 묶어 하나의 장치인 것처럼 동작하게 하며, 하드웨어 장치가 가진 고정된 능력을 보다 유연하게 활용하여 자원효율성을 높이는 기술.
(상세내용 - 소프트웨어정책연구소, "클라우드 가상화 기술의 변화", 2018.12. 참조)

18) TowardsDataScience, "Monitor and Improve GPU Usage for Training Deep Learning Models", 2019.3.

19) HW의 가상화를 통해 컴퓨팅 환경을 에뮬레이션 하는 것으로 실제 HW와 유사하게 동작하며, OS나 응용프로그램을 설치 및 실행 시킬 수 있음. 경우에 따라 다수의 VM이 하나의 장치 상에 공존할 수 있음 (상세내용 - 소프트웨어정책연구소, "클라우드 가상화 기술의 변화", 2018.12. 참조)

20) HW에 포함된 SW로 HW제조사에서 제공하며, 해당 HW기기가 주어진 기능을 수행할 수 있도록 함

내에서 유연하게 동작할 수 있도록 드라이버(driver)[21] 소프트웨어를 배포하기 때문이다. 즉, 제품에서 지원하지 않는 가상화 기능을 강제로 구현하기 위해서는 제품의 하드웨어적인 동작 요소와 구조를 모두 알아야 하며, 드라이버나 펌웨어를 수정(대부분의 경우에는 수정이 불가능한)해야만 한다.

현재는 클라우드, AI의 수요가 증가하고 있고, 이들과 연계된 빅데이터(Bigdata), 사물인터넷(Internet of Things, IoT) 등의 확산 또한 가속화 되고 있어서 GPU 제조업체들은 클라우드 컴퓨팅에 잘 호환될 수 있도록 가상화를 지원하기 시작했다.

3.2. GPU 가상화 기술의 종류

GPU 가상화 기술은 가상화를 어디에 구현하느냐에 따라 크게 세 가지 기술로 분류할 수 있다. 이 기술들은 각각 API 리모팅, 반가상화 및 전가상화, 하드웨어 지원 가상화에 해당된다.

1 API 리모팅 : GPU 벤더가 가상화를 지원하지 않을 때 가상화를 구현하는 방법

클라우드 환경에서 GPU를 가상화하는 작업은 네트워크 카드 또는 디스크와 같은 I/O[22] 장치를 가상화하는 것보다 복잡하고 어려운 작업이다. 그 이유는 다음과 같다. 첫째, NVIDIA와 같은 GPU 공급 업체들은 상업적인 이유로 GPU 드라이버의 세부 구현 사항과 소스 코드를 공개하지 않고 있다. GPU를 가상화하는데 GPU 드라이버의 소스코드는 필수적인 정보이므로 이는 큰 제약사항이 된다. 둘째, 리버스 엔지니어링(Reverse Engineering)[23]과 같은 방법으로 GPU 디바이스 드라이버를 일일이 분석하여 수작업으로 설계를 만든다고 해도, GPU 공급업체들은 새로운 GPU 드라이버를 주기적으로 배포하고 있으므로 이를 매번 따라가기에는 시간과 노력이 많이 든다. 따라서 리버스 엔지니어링으로 만들어진 디바이스 드라이버는 시간이 흐름에 따라

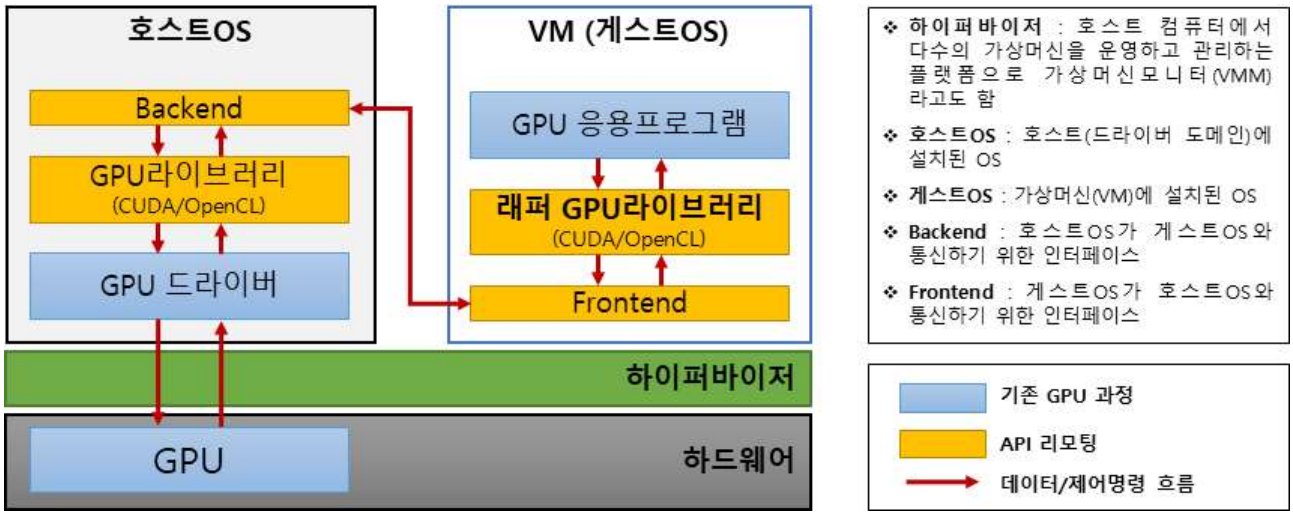
(예: 리모콘, 전기밥솥, 냉장고 등). 기본적으로 응용프로그램으로써 수정은 불가능하고, 업데이트가 가능한 경우는 있음.

21) 특정 HW를 제어하기 위해 시스템의 운영체제(OS) 내에 설치/동작하는 프로그램

22) Input/Output, 입출력 장치

23) 역공학, 구조분석을 통해 시스템의 원리를 발견하는 과정

구형의 GPU만 지원하게 된다. 마지막으로, 일부 운영체제나 가상화 솔루션 배포 업체들은 가상화를 지원하는 독점적인 GPU 드라이버를 배포하기도 하지만, 이는 모든 운영체제에서 사용할 수 없는 드라이버가 된다. 결국, GPU에 접근하기 위한 표준적인 인터페이스가 없기 때문에, GPU 디바이스 드라이버의 제작이 어렵고, GPU를 가상화하는 것이 어려워진다. API[24] 리모팅 방식은 현재까지 GPU 가상화에 가장 많이 쓰이는 보편적인 방법이었으며, 앞에서 언급한 제한 사항을 극복할 수 있는 방법으로 등장하였다. [그림 3-1]은 가상화 환경에서 API 리모팅 방식이 동작하는 원리를 도식화 한 것이다.



[그림 3-1] 가상화 환경에서 API 리모팅 기법

API 리모팅의 기본 원칙은 게스트 운영체제(Guest OS)에 네이티브(Native) 환경(호스트 OS)에서 쓰이는 GPU라이브러리(Library)[25]와 동일한 API를 가진 래퍼(wrapper) 라이브러리를 제공하는 것이다. 래퍼 라이브러리는 GPU 응용프로그램이 GPU에 대한 연산 함수를 호출할 때 그 호출이 직접 GPU 드라이버에 도달하기 전에 가로채는 역할을 맡게 된다. 이렇게 가로채어진 호출은 공유 메모리를 통해서 같은 시스템 내의 호스트 운영체제로 전달되거나 또는 네트워크를 통해 GPU를 갖고 있는 원격의 시스템에 전달된다. 호스트 운영체제나 다른 원격 시스템으로 전달된 GPU 호출은 원격으로 처리되므로 API

24) API(Application Programming Interface)는 운영체제(OS)나 프로그래밍 언어에서 제공하는 기능을 응용프로그램이 제어할 수 있도록 만든 인터페이스

25) 컴퓨터 프로그램의 조직화된 집합으로 데이터, 문서, 도움말 자료, 메시지 틀, 미리 작성된 코드, 서브루틴(함수), 클래스, 값, 자료 형 사양 등 이들을 묶어서 실행에 활용할 수 있도록 하는 기능

리모팅이라는 이름이 붙여졌다. 원격에서 GPU 호출이 처리된 후 호출의 결과 값만 다시 래퍼 라이브러리로 전달되게 된다. 즉 결론적으로 API 리모팅은 게스트 운영체제에 GPU를 직접 노출시키지 않고서도 게스트 운영체제에 GPU가 있는 것처럼 에뮬레이션 시킬 수 있는 효율적인 방법을 제공한다. 이러한 접근 방식은 라이브러리 수준에서 GPU를 가상화함으로써 소스 코드가 공개되지 않는 상용 GPU 드라이버를 우회할 수 있다.

API 리모팅 방식의 GPU 가상화

○ API 리모팅 방식을 사용하는 기법들은 GViM, vCUDA, rCUDA, GVirtuS, GVM, Pegasus, Shadowfax, VOCL, DS-CUDA 등이 있음(<표 3-1>)

<표 3-1> API 리모팅 방식의 GPU 가상화 기법

기술명	특징
GViM	▪ Xen[26] 하이퍼바이저를 기반으로, NVIDIA社의 GPU 라이브러리인 CUDA를 통해 가상화를 구현하여 가상머신상에 CUDA 응용프로그램(App) 및 API를 탑재 ▪ GPU App이 데이터를 대량으로 전송할 수 있는 공유메모리(Xenstore) 활용
vCUDA	▪ Xen 하이퍼바이저 기반으로 게스트OS에 가상 GPU(vGPU)와 CUDA 래퍼(wrapper) 라이브러리를 제공하는 형태로 구현
rCUDA	▪ 클라이언트-서버 구조를 기반으로 하여 원격(Remote)으로 호스트 서버 컴퓨터의 GPU를 게스트OS에 가상화 시켜주는 기법으로 자체 통신프로토콜 및 100G 대역폭 활용 ▪ 기존 같은 물리머신 상의 호스트의 GPU를 에뮬레이션 함으로 발생하는 오버헤드 및 성능저하를 개선하기 위해 개발
GVirtuS	▪ Xen외에도 KVM, VMware 등과 같은 다양한 하이퍼바이저를 지원하는 GPU 가상화 ▪ 게스트-호스트간의 통신 병목현상을 해결하기 위해 각 하이퍼바이저가 제공하는 고속 통신 채널을 활용할 수 있는 통신모듈(Communicator)을 탑재
GVM	▪ GPU App의 성능을 예측하기 위한 가상화 지원 모델로, 사용자 API, GPU 가상화 관리자 및 가상 공유메모리로 구성된 자체 가상화 인프라를 제공
Pegasus	▪ 기존 GViM의 업그레이드 버전으로, 가상화된 GPU(vGPU)를 스케줄링 하며 공유가 가능한 단위로 관리
Shadowfax	▪ Pegasus의 업그레이드 버전으로, 연산요구량이 많아질 때 호스트 상의 GPU 뿐만 아니라 원격의 GPU를 포함하여 다양한 가상 플랫폼을 구성할 수 있도록 함
VOCL	▪ 공개SW기반의 병렬 컴퓨팅 프레임워크인 OpenCL을 지원하는 GPU 가상화 기술로, rCUDA와 유사한 형태로 원격 GPU 가상화를 지원
DS-CUDA	▪ rCUDA와 유사하게 원격 GPU 가상화를 지원하며, 두 개의 서로 다른 호스트 GPU가 동일한 연산을 수행하여 연산의 결과를 비교 검증하는 기능을 제공

※자료: 참고문헌 목록 참조(국외 참고자료 9~17)

다만 이 방식은 기존의 가상화 환경에서 게스트 OS의 프로그램들을 수행하는 데에 있어서 성능 저하를 유발할 수 있다. 게스트 OS의 응용프로그램은

26) 여러 게스트 운영체제를 하나의 호스트 컴퓨터에서 동작하게끔 하는 대표적인 하이퍼바이저, 케임브리지 대학교에서 개발이 시작되어 2003년 첫 공개버전 발표. 초기에는 게스트 운영체제를 수정해야 하는 반가상화만 지원하였으나 현재는 전 가상화도 지원

하드웨어를 활용할 때, 반드시 호스트 OS를 거쳐서 접근 및 실행 할 수 있도록 되어 있다. 이는 자원을 공유하는 과정에서 호스트 OS가 관리자(Manager)역할을 수행하기 때문이다. 따라서 [그림 3-1]과 같은 Backend-Frontend 통신이 수반된다. 이는 가상화가 없는 상태의 GPU를 활용하는 것에 비해 단계가 추가됨을 의미하여, 곧 성능 저하로 이어질 수 있다. 이를 해결하기 위한 방법으로 하드웨어 제조사들은 가상환경에서 게스트 OS가 호스트 OS를 거치지 않고 직접 하드웨어와 통신할 수 있는 다양한 방법(Intel-VT 기술 등 - 후술)들을 제시하고 있다.

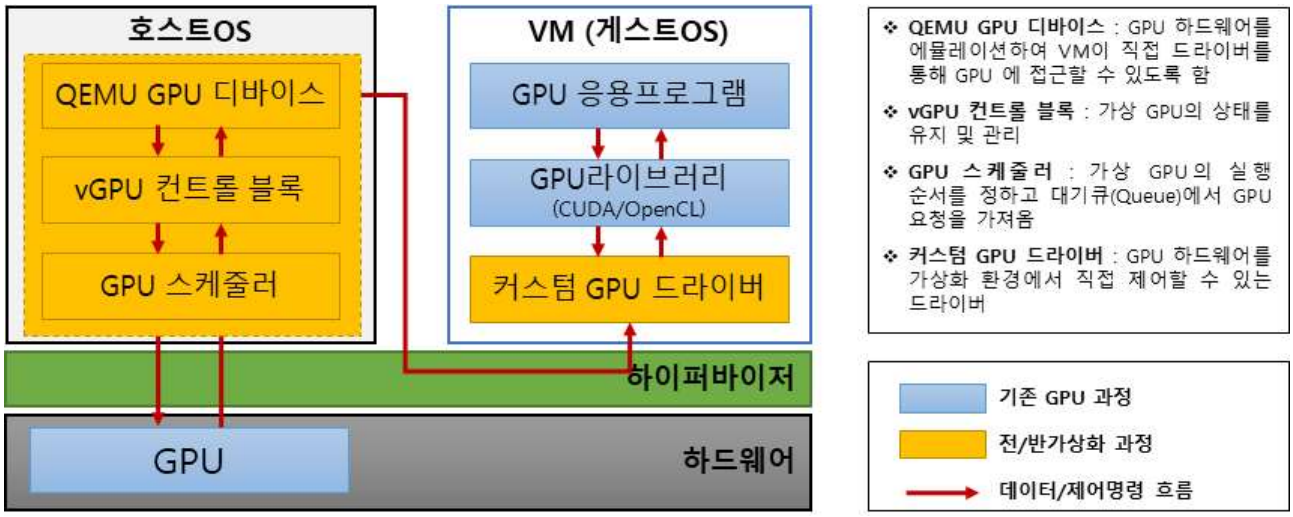
API 리모팅 방식은 새로운 기능이 GPU 공급 업체에 의해 원래 라이브러리에 추가 될 때 래퍼 라이브러리 또한 업데이트 되어야 하며, 이는 상당히 번거로운 작업이다. 이로 인해서, API 리모팅 솔루션은 최신의 GPU 라이브러리 버전을 사용하지 못하는 경우가 많다.

2 전가상화 및 반가상화 기법을 통한 커스텀 GPU 드라이버 가상화

API 리모팅 방식의 최신성 유지 측면의 한계를 개선하기 위해서 GPU 드라이버를 가상화 하는 방식이 등장했다. 가상화의 방식에는 전가상화(Full-Virtualization)와 반가상화(Para-Virtualization)가 있는데, 전가상화는 GPU 드라이버에 대한 수정 작업 없이 통째로 에뮬레이션 하는 방법이고, 반가상화는 GPU 드라이버를 수정하는 방식으로 VM의 직접 제어가 어느 정도 가능하게 하여, 성능을 좀 더 올릴 수 있는 방식이다.

최근에 CPU 및 GPU 칩셋 제조사중 하나인 AMD[27]는 특정 모델에 한해 GPU 아키텍처에 관한 공개 문서를 발표했다. 또한 일부 개발자는 리버스 엔지니어링 기술을 사용하여 NVIDIA GPU와 호스트 간의 인터페이스를 추측했다. 이러한 노력 덕분에 AMD 또는 NVIDIA GPU를 제어 할 수 있는 커스텀(Custom) GPU 드라이버가 개발되었다. 이러한 커스텀 드라이버의 출현은 반가상화 기술과 전가상화 기술에 대한 많은 연구를 촉진시켰다. [그림 3-2]는 커스텀 GPU를 활용한 가상화 방법을 나타낸다.

27) 글로벌 시장에서 CPU 부문 Intel(Intel, 1위)다음, GPU 부문 NVIDIA(nVidia, 1위)다음으로 시장을 점유



[그림 3-2] 커스텀 GPU 드라이버 가상화

이 아키텍처에서 게스트 운영체제는 수정되지 않은 또는 수정 된 커스텀 GPU 드라이버를 활용하고 있다. 우선, 커스텀 GPU 드라이버는 기존의 상용화된 드라이버가 아닌 가상화를 위한 튜닝(Tuning)을 거친 드라이버이다. 이후, 별다른 수정 없이 GPU 드라이버를 완전하게 에뮬레이션 하거나(전가상화), 하이퍼바이저를 통한 VM에 의해 제어될 수 있도록 수정(반가상화)을 하여 가상화를 구현한다. 이 방식은 VM이 직접 GPU 드라이버를 제어하는 방식이다.

게스트 운영체제의 커스텀 드라이버에서 나오는 GPU 접근 및 실행 요청은 하이퍼바이저에서 관리하는 공유 메모리를 통해 호스트 운영체제의 QEMU[28] GPU 장치로 전달된다. QEMU 장치는 장착된 GPU 하드웨어를 에뮬레이션하고 GPU가 장착된 PCIe 주소값(BAR[29])을 게스트 드라이버에 노출한다. 이렇게 하면 게스트 드라이버는 QEMU 장치를 실제 GPU로 간주하게 된다. 가상 GPU(vGPU) 제어 블록은 각 가상 GPU의 상태를 유지하고 QEMU 장치에서 발행 된 GPU 요청을 대기 큐에 보관한다. GPU 스케줄러는 다음에 수행할 가상 GPU를 선택하여 실행하고 관련 대기 큐에서 VM으로부터 온 GPU 요청을 가져온다. 게스트 운영체제의 GPU 요청은 GPU에서 직접 처리되고 결과는 역 경로를 통해 응용 프로그램에 반환된다.

이 구조의 장점은 기존 GPU 라이브러리를 재사용 할 수 있으며 GPU

28) Quick EMUlator의 약자로, 하드웨어 가상화 기능을 갖춘 오픈소스 기반의 에뮬레이터
29) Base Address Register

가상화의 구현 지점이 게스트 운영체제의 드라이버 계층으로 내려가므로 향후 라이브러리의 변경에 대비할 수 있다. 또한 게스트 운영체제의 GPU 요청을 하이퍼바이저가 모니터링하고 조정할 수 있으므로 API 리모팅과 비교하여 실시간 작업이동(Live Migration)[30]과 같은 필수 가상화 기능을 어려움 없이 지원할 수 있다. 반면, 이 구조의 단점은 커스텀 GPU 드라이버에 크게 의존한다는 것이다. 새로운 GPU 마이크로 아키텍처가 출시되었을 때 리버스 엔지니어링 또는 공개 문서가 아직 없다면 해당 커스텀 드라이버의 개발에 상당한 부담을 줄 수 있다.

전/반가상화 방식의 GPU 가상화

○ 전가상화 기법으로는 GPUvm, gVirt, gHyvi, gScale 등이 있고, 반가상화 기법으로는 VMware SVGA II, LoGV, VGRIS, AMD HSA 솔루션 등이 있음(<표 3-2>)

<표 3-2> 전/반가상화 방식의 GPU 가상화 기법

기술명		특징
전가상화	GPUvm	▪ 리눅스 기반의 공개OS인 우분투(Ubuntu)의 그래픽드라이버 nouveau를 활용하고, Xen 하이퍼바이저 기반의 GPU 가상화를 제공하며 전가상화와 반가상화를 모두 지원
	gVirt	▪ Xen 하이퍼바이저 환경에서 Intel CPU 내장형 GPU에 대한 전가상화를 지원하며, GPGPU 보다는 그래픽 가속 자체에 중점을 둠
	gHyvi	▪ gVirt의 성능을 개선하기 위한 후속 기술로, 가상머신에서 구동되는 GPU App의 빈번한 메모리 접근 및 업데이트 과정을 개선
	gScale	▪ gVirt의 확장성을 개선하기 위한 후속 기술로, 기존 최대 4개까지의 vGPU를 제공하던 것을 12~15개 수준으로 확장한 버전
반가상화	VMware SVGA II	▪ 시스템 가상화 업체 VMware社의 GPU 가상화 제공 기술로, AMD社의 GPU 및 공개문서를 기반으로 구축한 게스트OS용 GPU 드라이버 ▪ 하이퍼바이저가 가상머신과 실제 GPU 하드웨어간의 접근에 보다 적극적으로 개입하고 기본적인 가상화 기술 중 하나인 실시간 마이그레이션(Live Migration) 기능을 지원하며 API 리모팅 방식이 갖는 한계점을 개선
	LoGV	▪ NVIDIA社의 GPU 드라이버를 리버스 엔지니어링을 통해 오픈소스로 구현한 드라이버인 PathScale를 통해, KVM[31] 하이퍼바이저 기반의 GPU 가상화를 지원
	VGRIS	▪ VMware SVGA II에서 구현한 반가상화 기술을 기반으로 하며, 각 가상머신의 성능을 모니터링하고 GPU 스케줄러에 전송하는 관리 에이전트를 제공
	AMD HSA 솔루션	▪ AMD社에서 2016년에 개발한 가상화 기술로 KVM 하이퍼바이저를 기반으로 하며, GPU와 CPU를 동시에 활용하되 이들간의 통신 오버헤드를 완화시키는 기능을 제공

※자료: 참고문헌 목록 참조(국외 참고자료 18~21, 22~25)

30) 가상화 환경에서 가상머신의 실행환경을 유지시키는 기능. 호스트 시스템에 하드웨어 및 소프트웨어상의 업데이트와 같은 이벤트가 발생하더라도 가상머신의 복제, 이전 등을 통해 실행환경을 유지

31) Kernel-based Virtual Machine(커널기반가상머신), 리눅스 커널(운영체제 내에서 시스템을 통제하는 핵심 프로그램 - 각주40 참조)을 하이퍼바이저화 한 가상화 지원 플랫폼으로, QEMU라는 가상머신 실행 프로그램을 기반으로 구동되며 전가상화와 반가상화 모두 지원

3 하드웨어 지원 GPU 가상화 (하드웨어 벤더가 가상화를 지원)

하드웨어 지원 가상화는 GPU 또는 메인보드를 제작하는 벤더들이 제품상에서 가상화 기능을 제공하는 것이다. 기존 벤더들이 가상화 기능을 제공하지 않음으로 인해 API 리모팅 기법으로 우회하거나, 리버스 엔지니어링을 통해 수정된 GPU 드라이버를 제어하는 방식 보다 가상화가 쉽고 빠르다. 이 방식에서 가상머신은 GPU 하드웨어에 직접 액세스 할 수 있다.

이 접근법은 Intel, AMD, NVIDIA 등 CPU 및 메인보드 칩 제조업체 또는 GPU 공급 업체가 제공하는 Intel VT-d, AMD-Vi, NVIDIA-GRID 등과 같은 I/O 가상화를 위한 하드웨어 확장 기능을 활용한다. 하드웨어 지원 GPU 가상화 기법은 하드웨어 장치로 전송되는 데이터 또는 인터럽트(Interrupt)[32] 등과 같은 제어명령들을 처리하거나 담아두기 위한 장치 내 메모리 공간을 게스트 운영체제에 직접 매핑(Mapping)[33]해주는 방식으로 GPU를 가상화한다. 즉, 가상머신이 직접 GPU 하드웨어에 접근하여 읽기/쓰기/실행 등의 명령을 수행하도록 하는 방식이다.

스케줄링(Scheduling)과 가상화

○ 기본적으로 컴퓨터에서 응용프로그램이 하드웨어 장치에 접근하는 것은 엄격한 시스템 감시와 관리 하에 이루어짐

- 이는 다양한 응용 프로그램들이 동시에 한 장치를 활용하고자 할 때 발생할 수 있는 충돌을 방지하고자 하는 것임
- 자원의 효율적인 활용을 위해 꼭 필요한 기능

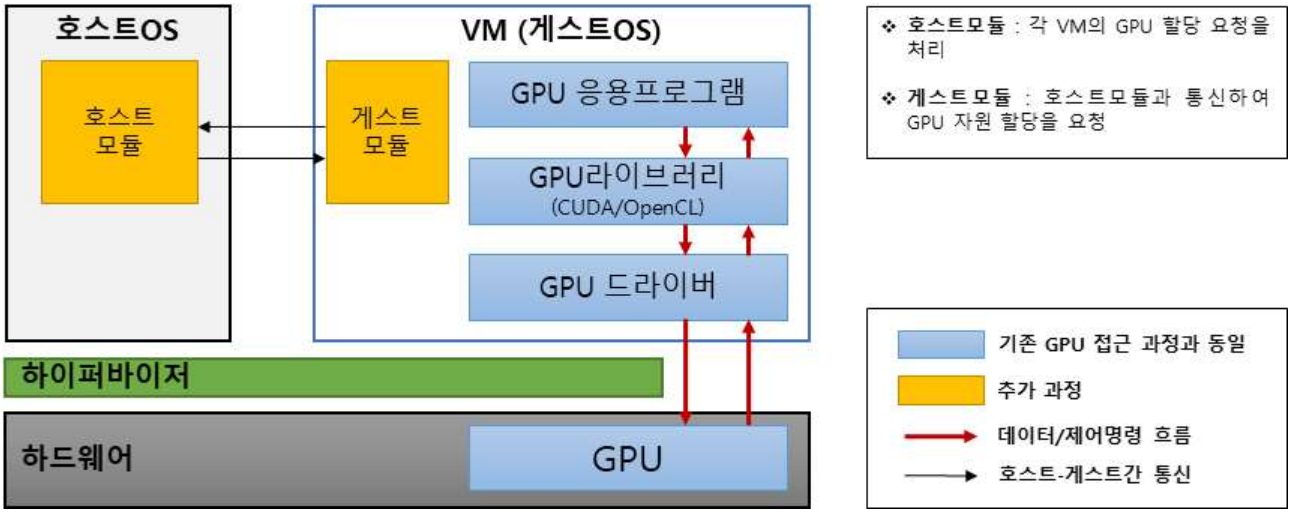
○ 때문에 가상화를 통해 공유되는 자원(GPU 등)은 반드시 한 순간에 하나의 인가된 VM에서 접근하도록 하는 방식을 취함

- 결국, 드라이버 도메인(호스트 OS)에서 이러한 관리역할을 담당
- 이러한 구조적인 한계는 결국 성능저하, 호환성 결여 등과 같은 문제를 야기

32) 실행 중인 명령을 잠시 중단하고 다른 명령을 먼저 실행하는 것
33) 입력 데이터를 원하는 목적지에 배치하는 것으로, 여기서는 할당(Allocation)의 의미

① 단일 가상머신 지원형

하드웨어 지원방식의 가상화는 하이퍼바이저의 개입 없이 GPU 장치에서 가상머신의 메모리 공간으로 데이터를 직접(Direct) 전송할 수 있으며, 이 과정에서 발생하는 인터럽트 또한 게스트 운영체제에 직접 전달된다. [그림 3-3]은 하드웨어 지원 GPU 가상화 시스템의 예를 나타내며, Intel VT-d 또는 AMD-Vi로 구현된 하드웨어 지원 GPU 가상화를 채택한 시스템과 같다.



[그림 3-3] 하드웨어 지원 GPU 가상화

이 아키텍처에서 가상머신의 GPU 접근/처리과정은 라이브러리나 드라이버를 수정하지 않고도 GPU를 활용할 수 있다. 이렇게 하면 게스트 운영체제는 DMA[34]와 인터럽트가 가상머신에 직접 매핑되기 때문에 GPU와 통신하는 동안 하이퍼바이저를 우회하게 되며, 하이퍼바이저와의 통신 오버헤드가 사라지면서 성능을 올릴 수 있다.

하드웨어 지원 GPU 가상화를 활용하는 대표적인 최초 사례는 2010년에 발표된 아마존(Amazon)의 EC2(Elastic Compute Cloud)가 있다. EC2는 Intel VT-d 기술을 사용하여 클라우드 사용자에게 GPU 가상화를 통한 인프라 환경을 제공한다. EC2는 가상머신 당 2개의 NVIDIA Tesla GPU를 제공하는 CGI(Cluster GPU Instance)라는 기술을 도입했다. CGI는 각 게스트 운영체제에 GPU를 직접

34) Direct Memory Access(직접 메모리 접근), 입출력 장치가 CPU에 의한 제어 없이 직접 자료의 이동 및 교신을 할 수 있도록 하는 방식

연결하여 대규모 병렬 처리 능력이 필요한 하이퍼포먼스 컴퓨팅(HPC) 응용 프로그램을 지원할 수 있다. 이후, 다양한 하드웨어 지원형 GPU 가상화 기법들이 연구 및 발표되었는데, <표 3-3>과 같다.

하드웨어 지원 방식의 GPU 가상화 (단일머신 지원형)

<표 3-3> 하드웨어 지원 방식의 GPU 가상화 연구들

연도	내용
2013	▪ Amazon EC2에서 32개의 클러스터로 구성된 CGI 성능 평가 연구 - 가상화 환경과 자체 클러스터를 사용하는 기본환경에서 성능 비교 - 계산 집약적인 프로그램이 EC2와 같은 설정에서 더 유리함을 증명
2013	▪ 하드웨어 지원형 GPU 가상화 시스템에서 클라우드 게임 성능 평가 - 일부 프로그램에서 GPU 가상화를 활용하게 되면 성능이 저하되는 원인을 분석한 결과 빈번한 하이퍼바이저와 가상머신간의 작업 전환 요청에 의한 것임을 입증
2014	▪ Xen과 KVM환경에서 Intel VT-d 기술 기반의 GPU 패스스루 시스템 구현 - 위 두 하이퍼바이저에서 VT-d 기술을 사용했을 때 GPU 성능이 네이티브 수준에 도달하는 것을 증명
2014	▪ GPU 하드웨어 가상화 성능 측정 연구 - Xen 하이퍼바이저 환경에서 Intel VT-d 기술을 활용하면, 가상화 환경에서 성능저하가 API 리모팅 기법(40%) 대비 성능 저하가 1.2%수준에 불과함을 실험
2014	▪ 다양한 하이퍼바이저를 기반으로 하는 Intel VT-d 기술의 실험 - VMWare ESXi, KVM, Xen 및 Linux 컨테이너 (LXC)의 성능을 실험한 결과 네이티브에 근접하며 가장 일관된 성능은 KVM, 평균 수준은 Xen, 최고 성능은 LXC 임을 밝힘
2015	▪ SR-IOV[35]를 활용한 고성능 컴퓨팅 부하 실험 - GPU 가상화 환경에 대한 부하실험 결과 네이티브 환경 대비 1.5% 수준의 오버헤드만 보임

※자료: 참고문헌 목록 참조(국외 참고자료 26~31)

하드웨어 지원 가상화 방식의 장점은 하이퍼바이저와 같은 추가 소프트웨어 계층 없이 GPU 가상화를 실현하면서 거의 네이티브 환경에 근접하는 성능을 달성 할 수 있다는 것이다. 반면, 단점으로는 GPU 작업이 호스트 운영체제 또는 하이퍼바이저를 우회하기 때문에 시스템상의 전체적인 GPU 스케줄링 정책을 적용하는 것이 불가능할 수 있다는 것이다. 아울러, API 리모팅 방식과 마찬가지로 실행 체크 포인트, 실시간 마이그레이션 및 내결함성 실행 기능을 구현하기가 어렵다. 이 부분의 구현은 전적으로 하드웨어 제공 업체의 지원 여부에 달려있다.

기존의 Intel VT-d 및 AMD-Vi는 이러한 방식으로 가상화 시스템 환경의 성능을 높일 수 있었다. 그러나, Intel VT-d 및 AMD-Vi에서는 게스트 운영체제가 부팅 될 때 지정된 가상머신에 GPU가 정적(Static)으로 할당된다. 즉,

35) Single Root - I/O Virtualization, Intel에서 제공하는 네트워크 인터페이스 카드(NIC)의 하드웨어 지원형 가상화 기술로, 가상화 환경에서도 거의 네이티브 수준(10Gb)의 초고속 네트워크 대역폭을 제공

이것은 특정 시간에 하나의 가상머신에게만 GPU 가상화를 지원한다는 의미이다. 이렇게 되면 여러 게스트 운영체제 간에 단일 GPU 공유를 지원할 수 없는 한계점이 생긴다.

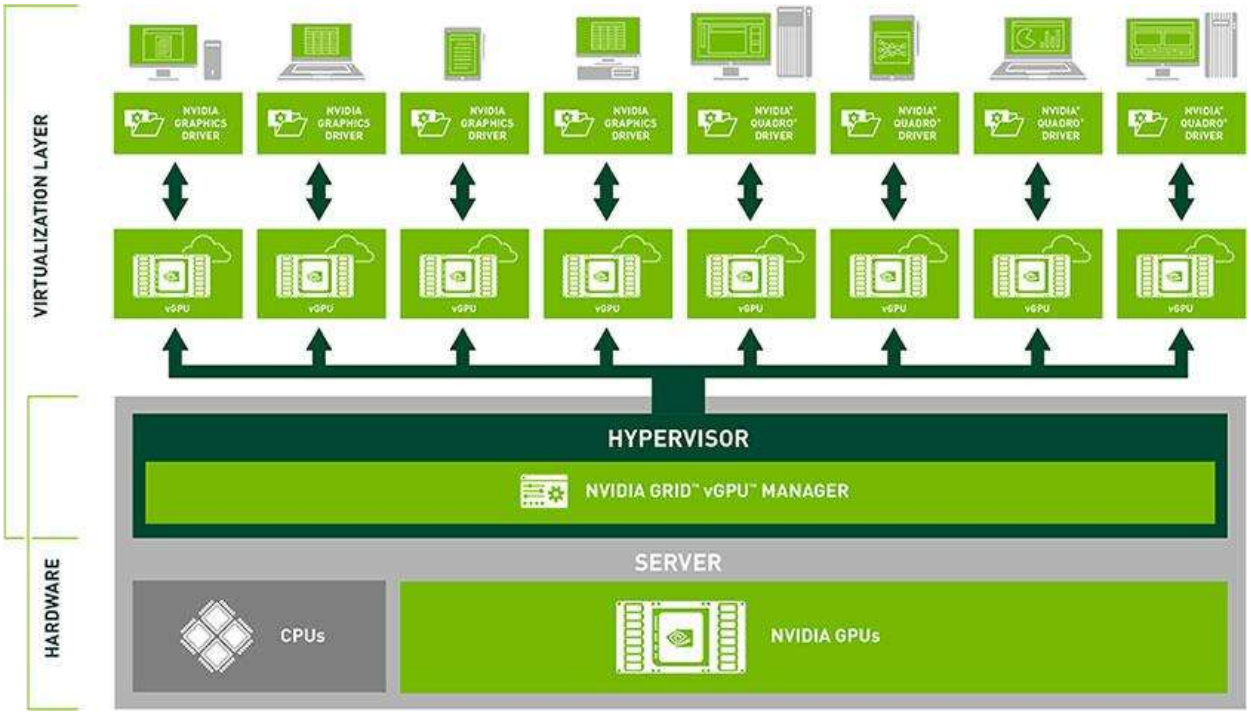
② 다중 가상머신 지원형

최근에는 가상머신간의 GPU 자원을 동적으로 활용할 수 있도록 핫 플러그(Hot Plug)[36] 기능을 탑재한 솔루션이 출시되었다. 2013년에 여러 가상머신들이 단일 GPU를 공유할 수 있는 기법 연구가 시도되었다[37]. GPU를 가상머신에 직접 연결·할당하는 기능인 패스스루(Pass through)를 확장하는 방법이다. 기존의 패스스루 방식은 오직 하나의 가상머신에만 하나의 GPU 하드웨어를 직접 할당하였기 때문에, 해당 가상머신이 종료되기 전까지 GPU를 다른 가상머신에 할당 할 수 없었다. 이 연구에서는 GPU 장치를 동적으로 설치하거나 제거 할 수 있는 PCIe 채널의 핫 플러그 기능을 활용하여 GPU를 여러 가상머신이 공유할 수 있게 구현하였다. 이 구현이 가능하기 위해서는 GPU를 구동하기 위한 래퍼 라이브러리를 각 가상머신에 제공해야하며, 이들을 모니터링 해야만 한다.

현재 시점에서 다중 가상머신에 GPU 자원을 공유하고 할당하는 최신 기술은 NVIDIA의 GRID 솔루션이 있다. [그림 3-4]는 NVIDIA GRID 아키텍처를 나타낸다. GRID 솔루션은 여러 가상머신 간의 GPU 공유를 지원하면서 좀 더 나은 확장성과 성능을 제공한다. 이 솔루션은 클라우드 환경을 타겟으로 개발된 최신의 NVIDIA GPU에서 지원 된다. [그림 3-4]의 NVIDIA GRID vGPU MANAGER는 가상 GPU의 할당과 해제, 제어를 위한 하이퍼바이저 역할을 수행한다. 이 하이퍼바이저는 각 게스트 운영체제가 GPU를 활용하기 위해 갖는 가상 메모리 영역을 GPU의 물리적 메모리에 개별적으로 매핑하고 변환 할 수 있도록, 새로운 I/O 메모리 관리 장치(IOMMU)를 가지고 있다.

36) 전원이 유지된 채 시스템의 구동 상태에서 하드웨어 장치를 활용/교체 할 수 있도록 하는 기능으로 윈도우의 Plug and Play 기능과 유사

37) Jo, Heeseung, et al. “Exploiting GPUs in virtual machine for BioCloud.“ BioMed research international 2013 (2013).



※출처 : nvidia.com

[그림 3-4] NVIDIA GRID 아키텍처 개념도

또한 각 게스트 운영체제가 제어 요청에 대해 별도의 입력 버퍼를 가질 수 있도록 했다. 이 두 가지 아키텍처 변경을 통해 GRID를 지원하는 NVIDIA GPU는 하드웨어 차원에서 GPU를 공유하는 것을 실현하였으며, 동시에 각 게스트 운영체제의 성능을 네이티브 환경처럼 유지할 수 있다. 현재 NVIDIA GRID는 클라우드 환경을 목표로 하는 NVIDIA GRID K1, K2, NVIDIA Tesla M6 및 M60 GPU 등에서 지원이 된다.

4. 향후 기술방향과 시사점

4.1. 클라우드 GPU 가상화 기술의 향후방향

클라우드 컴퓨팅을 위한 GPU 가상화 기술들은 비교적 최근에 등장한 기술들이다. 지금껏 어려웠던 GPU를 가상화하는 방법에 대해 많은 부분이 해결되기도 하였다. 그러나 아직 GPU 가상화 환경의 성능과 기능 측면에서 다양한 해결 과제가 남아 있다. 향후 더 다양해질 AI 서비스와 고객 요구를 충족시키기 위해 GPU 가상화 기술은 다음과 같은 발전방향이 고려되어야 한다.

□ 컨테이너 기반 경량 가상화의 실현이 필요

대표적인 경량화 사례인 Linux 기반 컨테이너[38]는 프로세스 수준의 가벼운 가상화를 제공하는 새로운 클라우드 기술이다. 컨테이너 기술에서는 하나의 리눅스 커널에 의해 여러 개의 컨테이너가 생성 및 관리되기 때문에 컨테이너는 GPU를 가상화하기 위해 앞서 살펴본 추가적인 래퍼 라이브러리나 프론트/백엔드 드라이버 모듈을 필요로 하지 않는다. 따라서 컨테이너를 사용하면 네이티브 환경에 가까운 성능을 얻을 수 있다.

하지만 컨테이너를 사용한 GPU 가상화에 대한 현재의 연구는 아직 초기 단계에 있다. 지금까지의 연구는 컨테이너와 다른 가상화 솔루션 간의 성능을 비교하는 것이 주를 이루고 있다. 클라우드 컴퓨팅에서 GPU가 장착된 컨테이너를 활용하려면 공정하고 효과적인 GPU 스케줄링 알고리즘이 필요하다. 대부분의 GPU 스케줄러는 GPU 호출을 가로 채기 위해 API 래퍼 라이브러리 또는 드라이버 변경이 필요하지만, 이는 컨테이너 환경에서 무시할 수 없는 오버 헤드를 일으킬 수 있다.

□ 가상화를 통한 시스템의 확장성을 보장

시스템 가상화의 주목적은 컴퓨팅 자원의 사용률을 높이고 소유 또는 운영

38) Container, ‘물체를 격리하는 공간’이라는 사전적 의미처럼, 컴퓨팅 환경에서 모듈화되고 격리된 컴퓨팅 공간 또는 컴퓨팅 환경을 의미, 프로세스를 격리하여 모듈화된 프로그램 패키지로써 수행하는 것을 의미함(상세내용 - 소프트웨어정책연구소, “클라우드 가상화 기술의 변화”, 2018.12. 참조)

비용을 줄이는 것이다. 이 목표는 데이터 센터의 각 물리적 시스템에 다수의 가상머신을 통합하여 클라우드화 함으로써 달성 할 수 있다. Xen 하이퍼바이저는 호스트 당 500 개가 넘는 가상 CPU를 생성할 수 있도록 설계되었다. 그러나 GPU 가상화의 경우에는 가상머신에게 상대적으로 적은 수의 가상 GPU를 제공한다.

앞서 살펴본 gScale 프레임워크가 확장성을 높이는 데 상당한 노력을 기울인 기술이지만, 이 프레임워크는 Linux 가상머신용으로 최대 15 개의 vGPU만 제공할 수 있다. CPU와 GPU의 리소스(=자원) 통합 성능 간에는 아직도 상당한 불균형이 존재하며 그 격차를 메우는 것은 여전히 어려운 과제이다. 따라서 방대한 vGPU를 생성할 경우 GPU 장치 메모리 크기, GPU 컨텍스트 전환(Context Switching)[39] 빈도 및 캐시(Cache)[40] 등 관련하여 성능에 영향을 미치는 모든 사항을 고려해야한다.

□ 가상화 환경에 필요한 보안책 마련

하이퍼 바이저의 중요한 기능 중 하나는 가상머신 간의 안전한 격리(Isolation)를 제공하는 것이다. 이 작업을 수행하기 위해 앞서 살펴보았던 LoGV, GPUvm, gVirt 및 gScale 등은 한 가상머신이 다른 가상머신의 GPU 주소 공간을 매핑하지 못하게 한다. 이러한 보호 메커니즘에도 불구하고 GPU 가상화 프레임워크는 DoS(denial of service)[41]와 같은 공격에 취약한 상태이며 악의적인 가상머신이 높은 빈도로 GPU 명령을 백엔드(호스트 운영체제의 Backend)에 연속적으로 보내게 되면 전체 시스템을 위험에 빠뜨릴 수 있다.

이 문제를 해결하기 gVirt는 GPU가 중단될 경우 GPU를 재설정하고 각 가상머신의 실행 상태를 검사 한 후 의심스러운 가상머신을 제거한다. 하지만 이로 인해 서비스가 일시 중지 되어 정상적인 가상머신에 피해가 갈 수 있다. GPU 재설정을 피하려면 악의적인 가상머신이 시스템을 위협하기 전에 악성

39) 문맥전환이라고도 하며, CPU(또는 GPU)가 다른 작업을 수행하기 위해 현재 수행중인 작업을 보관하고 새로운 작업으로 전환하는 과정
40) 컴퓨터에서 작업을 위한 데이터나 값을 미리 복사해 두는 임시 저장소로, 보통 CPU 칩 내부나 바로 옆에 탑재하여 하드디스크나 메인메모리의 데이터에 접근하는 시간이 오래 걸리는 경우 활용
41) 서비스 거부 공격, 악의적으로 시스템의 자원을 부족하게끔 하여 제 기능을 못하도록 만드는 공격

가상머신의 실행속도를 지연시킬 수 있는 세분화 된 접근 제어 기법이 필요하다.

□ 전력 소비 효율성 향상

가상화하지 않은 GPU 플랫폼에서 에너지 효율에 대한 연구는 많이 이루어졌지만, 가상화 환경에서의 연구는 아직 별로 없다. GPU 가상화 환경에서 에너지 효율성을 높이려는 연구는 pVOCL정도가 있는데, 앞서 살펴본 API 리모팅 기법 중 VOCL의 전력효율화 버전이다. GPU간 최대 전력 소비를 원격에서 작업량에 따라 유동적으로 제어함으로써 GPU 클러스터의 전체적인 에너지 효율을 향상시킨다. 이 방식 외에도 호스트 측에서 전력 효율에 대한 연구도 필요하다. 가상머신 간의 다양한 GPU 사용 패턴을 모니터링하고 작업 부하에 따라 GPU의 전원 상태를 동적으로 조정하는 시스템이 향후 연구개발 기술로 적합하다.

□ 자원의 공유 및 활용성 증대

최신의 GPU들은 프로세스 또는 가상머신이 단일 GPU 하드웨어에서 여러 GPU 커널(Kernel)[42]을 동시에 실행시킬 수 있도록 지원한다. 이러한 GPU 하드웨어에 대한 공간 다중화(Spatial Multiplexing, SM)[43] 기법은 각각의 여러 커널이 GPU내의 분할된 공간을 완전히 활용함으로써, GPU 활용도를 향상시킬 수 있다. 그러나 대부분의 GPU 스케줄링 방법은 여러 가상머신의 GPU 커널이 GPU에서 순차적으로 실행되는 시간 다중화(Time Multiplexing) 기법을 활용한다.

이 방법은 철저한 스케줄링 기술이 필요하며, 많은 작업을 동시에 진행해야하는 워크로드의 경우 낮은 사용률을 보일 수 있다. 또한 많은 수의 병렬 코어들을 활용하는 데에 있어서 공간 다중화가 시간 다중화보다 유리한 경우가 발생할 수도 있다. 때문에 GPU 스케줄링에서 높은 GPU 사용률과 공정성을 모두 달성하려면 두 가지 방법을 함께 사용하는 방법을 고려해야

42) 운영체제 내에서 시스템을 통제하는 핵심 영역이자 프로그램으로, 운영체제의 많은 서비스 기능들과 응용 프로그램 수행에 필요한 서비스들을 제공

43) 하드웨어내의 메모리 및 레지스터 등을 공간적으로 분리하여 활용하는 멀티프로세스 방법, 다른 방법으로는 점유시간을 분할하는 시분할(Time Multiplexing) 기법도 있으며, 컴퓨팅 및 통신분야에서 자원을 공유하는 방식으로 널리 쓰임

한다.

□ 작업의 유동적인 실시간 이동성(마이그레이션)에 대한 지원

NVIDIA GRID의 출현으로 GPU에서 진정한 하드웨어 기반 가상화가 가능해졌다. NVIDIA GRID를 사용하면 GPU 장치가 하드웨어 수준에서 여러 별도의 가상 장치로 나타난다. 그러나 이 기술은 가상화 계층(하이퍼바이저)을 우회하기 때문에 클라우드 컴퓨팅에서 가장 중요한 가상화 기능 중 하나인 가상머신의 실시간 작업이동(Live Migration)[44]에 문제를 일으킬 수 있다.

하드웨어 기반 가상화에서 실시간 마이그레이션을 사용하려면 하드웨어 상태를 캡처하여 원격 시스템에서 복원 할 수 있는 새로운 매커니즘이 필요하다. 최근 NVIDIA는 GPU 가상화 환경에서 실시간 마이그레이션을 지원할 수 있는 기술을 제안했다.[45] 가상머신에서 수행중인 작업을 다른 물리서버의 가상머신으로 옮기는 데에는 시스템의 가동 중단 시간의 최소화, 데이터 손실 방지, 접근 권한의 이동 등 고려해야할 복잡한 요소들이 모두 충족되어야 한다.

□ 통신 오버헤드의 최적화

API 리모팅 방식은 게스트 운영체제에서 GPU 호출을 가로 채서 호스트 운영체제 또는 원격 시스템으로 전달하여 GPU 가상화를 제공한다. API 리모팅 방식은 응용 프로그램이 GPU를 많이 사용하지 않는 경우 오버 헤드가 적다. 그러나 최근의 GPU 애플리케이션은 크기는 작거나(예: 100μs 미만의 수행 시간을 갖는 경우) 반복적인 GPU 커널 수행 작업을 하는 경우가 많다.

이럴 경우 매 작업마다 GPU 호출을 전달하는 것은 심각한 통신 병목 현상(Bottle neck)이 발생할 수 있다. 더구나 GPU의 공정기술과 성능이 점차 발전함에 따라 커널 단위의 실행 시간은 더욱 더 낮아질 것으로 예상되고, 이는 곧 잦은 통신 오버 헤드의 증가를 야기할 것으로 예상된다. API 리모팅 방식에서 GPU 집약적인 응용프로그램을 수용하려면 훨씬 최적화 된 통신 방법이 필요하다.

---

44) 라이브 마이그레이션, 각주 28 참조.
45) https://www.nvidia.com/ko-kr/data-center/virtualization/virtual-gpu-migration/ 참조.

□ 통합된 형태의 CPU-GPU 칩 개발

기존의 시스템은 CPU와 GPU가 분리되어 있으며, 이들 간의 통신은 시스템 버스(System Bus)[46]를 통해 이루어진다. 인공지능의 발달로 작업의 복잡성과 크기는 점차 증가하고 있기 때문에, 현재의 PCIe 시스템 버스의 대역폭 용량(16Gbps)으로는 데이터 전송 오버헤드를 해결하지 못한다. 또한, 기본적으로 분리된 데이터 공간을 활용하는 CPU와 GPU 때문에 프로그래밍의 어려움을 겪게 된다.

이러한 문제를 해결하기 위해 통합된 CPU-GPU 칩이 출현했으며 이 칩은 두 프로세서 간에 통합된 메모리 공간을 제공한다. 예를 들면 Intel의 통합 CPU-GPU 칩, AMD의 HSA 아키텍처 등이 있다. 이러한 새로운 아키텍처는 두 프로세서 간에 상당한 통신량이 필요한 큰 크기의 데이터 응용프로그램의 성능을 향상시킬 수 있다. 기존의 연구들 중 Intel 및 AMD의 통합칩에 대한 솔루션 개발이 있었지만(gVirt), 두 칩의 작업 부하를 균형있게 조정(Load Balancing)하는 것은 부족했다. 두 프로세서를 최적화 하여 사용할 수 있는 정교한 스케줄링 알고리즘이 필요하다. 또한, 최근 NVIDIA는 GPU와 CPU 사이의 통신에 높은 대역폭(80~200Gbps)을 가능하게 하는 NVLink[47]라는 기술을 출시했는데, 데이터 집약적인 애플리케이션의 고성능을 달성하기 위해서는 통합 메모리 기법과 NVLink의 결합과 같은 연구개발도 필요하다.

4.2. 요약과 시사

□ 고성능 컴퓨팅 패러다임의 변화와 GPU 가상화 기술 수요 증대

4차 산업혁명 시대의 인공지능의 확산과 여기에 맞물린 클라우드 컴퓨팅에 대한 수요가 늘어나면서, 서비스 벤더 및 연구자들은 비용 대비 높은 성능을 제공할 수 있는 컴퓨팅으로 GPU 활용을 도입했다. 이러한 CPU + GPU 조합의

46) 컴퓨터의 주요 구성요소인 주변기기들을 연결하는 데이터 및 제어명령의 통로이며, CPU의 빠른 처리 속도 대비 상대적으로 느린 주변기기들 간의 속도차를 보정해주기 위한 기능들을 탑재하고 있음
47) https://www.nvidia.com/en-us/data-center/nvlink/ 참조.

이기종 컴퓨팅은 지난 몇 년 동안 고성능 컴퓨팅(HPC) 및 클라우드 플랫폼에서 높은 성능과 자원 활용도를 상대적으로 낮은 운영비용으로 제공할 수 있는 패러다임으로 주목받아왔다. 클라우드 컴퓨팅에서 가상화는 전제가 되는(basic technology) 기술이다. 곧, 이기종의 머신들이 있는 클라우드 데이터 센터에서 GPU의 가상화는 사용자들 또는 서비스별 가상머신들 간에 GPU 장치를 효과적으로 공유하는 방법으로써 충분한 컴퓨팅 파워를 제공하기 위한 핵심 기술이 된다.

□ GPU 가상화를 위한 다양한 시도와 벤더들의 태세전환

GPU 가상화를 구현하는 것은 최근까지 어려운 일이었다. GPU를 제조·공급하는 벤더들은 GPGPU라는 개념이 등장하기 전까지, 가상화에 큰 관심이 없었다. 인공지능에 대한 관심이 늘어나고, 그 기반기술로 클라우드 컴퓨팅이 재조명 되면서, 본격적으로 GPU 벤더들은 자체적으로 클라우드 컴퓨팅을 위한 가상화 지원 기술을 개발·출시를 가속화했다. GPU를 가상화 하기 위한 방법은 크게 API 리모팅 방식, 전/반가상화를 활용한 방식, 하드웨어 지원 방식으로 나눌 수 있다. <표 4-1>은 앞서 살펴본 세 가지 방식에 대한 특징을 정리한 것이다.

<표 4-1> GPU 가상화 방식의 특징(요약)

<table>
<tr><th colspan="2">구분</th><th>특징</th><th>장점</th><th colspan="2">단점</th></tr>
<tr><td colspan="2">API 리모팅</td><td>▪ GPU 하드웨어가 지원하지 않는 GPU 가상화의 구현
- GPU 라이브러리의 복제인 래퍼를 가상머신에 제공, 원격처리</td><td>▪ 드라이버가 공개되지 않은 상용 GPU에 대해 드라이버 우회를 통한 가상화 실현</td><td colspan="2">▪ 성능저하
▪ 지속 업데이트 불가
▪ 어렵고 번거로움</td></tr>
<tr><td rowspan="2">하이퍼바이저 활용</td><td>전 가상화</td><td rowspan="2">▪ 가상화 플랫폼인 하이퍼바이저를 활용한 GPU가상화
- GPU 드라이버를 가상화</td><td>▪ GPU 드라이버의 수정 없이 에뮬레이션</td><td rowspan="2">▪ 드라이버의 존 도 높음</td><td>▪ 다소 낮은 성능</td></tr>
<tr><td>반 가상화</td><td>▪ 가상머신이 직접 드라이버 제어 및 성능향상</td><td>▪ 드 라 이 버 수정 필요</td></tr>
<tr><td colspan="2">하드웨어 지원</td><td>▪ GPU 및 CPU 벤더들의 제품에서 가상화 기술 지원</td><td>▪ 높은 성능 및 호환성</td><td colspan="2">▪ 높은 하드웨어 종속성</td></tr>
</table>

□ GPU 가상화 기술의 상용화는 이제부터 본격화

GPU를 그래픽 처리 이외의 목적으로 활용하는 GPGPU 개념 등장하게 되면서, 컴퓨팅 가성비가 상승하는 효과를 불러왔다. GPU는 AI학습을 위한 데이터 병렬처리에 최적화된 구조를 가지고 있으며, AI 및 클라우드 확산에 따른

GPGPU 수요 증가 하면서 칩제조사들은 GPU 가상화를 지원하고 나섰다. GPU 가상화 영역은 향후 기술의 성능향상 및 확장성, 최적화에 더 다양한 연구개발 필요한 분야이다. 특히, GPU의 활용은 클라우드 컴퓨팅의 인프라를 구축 및 확보하고 다양한 서비스를 가능하게 하는 기반 기술로 연계된다. AI의 안정적인 성능확보 및 활용 확산을 위해서 인프라에 대한 투자는 반드시 필요하다.

정부도 클라우드 컴퓨팅 활성화를 위한 기본계획[48]을 발표한 바 있고, 최근에는 AI 클러스터 구축[49]을 위한 사업을 추진하며 국가전략[50]을 발표하는 등 다양한 노력을 기울이고 있다. 이에 따라 GPU 컴퓨팅과 같은 AI 및 클라우드와 연계되는 인프라 원천기술의 확보를 위해, 관계부처와 산·학·연의 적극적인 협력이 있어야 한다.

48) 과학기술정보통신부, “4차 산업혁명 체감을 위한 클라우드 컴퓨팅 실행(ACT) 전략 - 제2차 클라우드 컴퓨팅 발전 기본계획”, 2018.12.28.
49) 인공지능 중심 산업융합 집적단지 조성사업(광주), 2019.8.
50) 관계부처합동, “인공지능 국가전략”, 2019.12.

[참고자료]

1. 국내

[1] 소프트웨어정책연구소, “클라우드 가상화 기술의 변화”, 2018.12.

[2] 엔비디아, https://www.nvidia.co.kr/

[3] AMD, https://www.amd.com/ko

[4] 인텔, https://www.intel.co.kr/

2. 국외

[1] http://Top500.org

[2] nVIDIA, “Record 136 NVIDIA GPU-Accelerated Supercomputers Feature in TOP500 Ranking”, 2019.11.19.

[3] Coates, Adam, et al. "Deep learning with COTS HPC systems." ICML. 2013.

[4] PWC, “Global Top 100 companies by market capitalisation”, 2019.07.

[5] iWeblists, “US commerce Stock Market Capitalization of the 50 Largest American Companies“, 2019.11.30.

[6] CorporateInformation, Top 100 list, 2019.

[7] TechPowerUp, “Intel is Giving up on Xeon Phi - Eight More Models Declared End-Of-Life”, 2018.7.

[8] TowardsDataScience, “Monitor and Improve GPU Usage for Training Deep Learning Models”, 2019.3.

[9] Gupta, Vishakha, et al. "GViM: GPU-accelerated virtual machines." Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing. ACM, 2009.

[10] Shi, Lin, et al. "vCUDA: GPU-accelerated high-performance computing in virtual machines." IEEE Transactions on Computers 61.6 (2012): 804-816.

[11] Duato, José, et al. "rCUDA: Reducing the number of GPU-based

accelerators in high performance clusters." High Performance Computing and Simulation (HPCS), 2010 International Conference on. IEEE, 2010.

[12] Montella, Raffaele, et al. "On the virtualization of CUDA based GPU remoting on ARM and X86 machines in the GVirtuS framework." International Journal of Parallel Programming 45.5 (2017): 1142-1163.

[13] Li, Teng, et al. "GPU resource sharing and virtualization on high performance computing systems." Parallel Processing (ICPP), 2011 International Conference on. IEEE, 2011.

[14] Gupta, Vishakha, et al. "Pegasus: Coordinated scheduling for virtualized accelerator-based systems." 2011 USENIX Annual Technical Conference (USENIX ATC'11). 2011.

[15] Merritt, Alexander M., et al. "Shadowfax: scaling in heterogeneous cluster systems via GPGPU assemblies." Proceedings of the 5th international workshop on Virtualization technologies in distributed computing. ACM, 2011.

[16] Xiao, Shucai, et al. "VOCL: An optimized environment for transparent virtualization of graphics processing units." Innovative Parallel Computing (InPar), 2012. IEEE, 2012.

[17] Oikawa, Minoru, et al. "DS-CUDA: a middleware to use many GPUs in the cloud environment." High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:. IEEE, 2012.

[18] Suzuki, Yusuke, et al. "GPUvm: Why not virtualizing GPUs at the hypervisor?." USENIX Annual Technical Conference. 2014.

[19] Tian, Kun, Yaozu Dong, and David Cowperthwaite. "A Full GPU Virtualization Solution with Mediated Pass-Through." USENIX Annual Technical Conference. 2014.

[20] Dong, Yaozu, et al. "Boosting GPU Virtualization Performance with

Hybrid Shadow Page Tables." USENIX Annual Technical Conference. 2015.

[21] Xue, Mochi, et al. "gScale: Scaling up GPU Virtualization with Dynamic Sharing of Graphics Memory Space." USENIX Annual Technical Conference. 2016.

[22] Dowty, Micah, and Jeremy Sugerman. "GPU virtualization on VMware's hosted I/O architecture." ACM SIGOPS Operating Systems Review 43.3 (2009): 73-82.

[23] Gottschlag, Mathias, et al. "LoGV: Low-Overhead GPGPU Virtualization." HPCC/EUC. 2013.

[24] Qi, Zhengwei, et al. "VGRIS: Virtualized GPU resource isolation and scheduling in cloud gaming." ACM Transactions on Architecture and Code Optimization (TACO) 11.2 (2014): 17.

[25] Huang, Yu-Ju, et al. "Building a kvm-based hypervisor for a heterogeneous system architecture compliant system." ACM SIGPLAN Notices. Vol. 51. No. 7. ACM, 2016.

[26] Cloud, Compute. "Amazon Elastic Compute Cloud." (2009).

[27] Expósito, Roberto R., et al. "General-purpose computation on GPUs for high performance cloud computing." Concurrency and Computation: Practice and Experience 25.12 (2013): 1628-1642.

[28] Yang, Chao-Tung, et al. "Implementation of GPU virtualization using PCI pass-through mechanism." The Journal of Supercomputing 68.1 (2014): 183-213.

[29] Shea, Ryan, and Jiangchuan Liu. "On GPU pass-through performance for cloud gaming: Experiments and analysis." Proceedings of Annual Workshop on Network and Systems Support for Games. IEEE Press, 2013.

[30] Younge, Andrew J., and Geoffrey C. Fox. "Advanced virtualization techniques for high performance cloud cyberinfrastructure." 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and

Grid Computing (CCGrid). IEEE, 2014.

[31] Younge, Andrew J., et al. "Supporting high performance molecular dynamics in virtualized clusters using iommu, sr-iov, and gpudirect." ACM SIGPLAN Notices. Vol. 50. No. 7. ACM, 2015.

[32] Walters, John Paul, et al. "GPU passthrough performance: A comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL applications." Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on. IEEE, 2014.

[33] Jo, Heeseung, et al. "Exploiting GPUs in virtual machine for BioCloud." BioMed research international 2013 (2013).

[34] Vu, Lan, Hari Sivaraman, and Rishi Bidarkar. "GPU virtualization for high performance general purpose computing on the ESX hypervisor." Proceedings of the High Performance Computing Symposium. Society for Computer Simulation International, 2014.

[35] Herrera, Alex. "NVIDIA GRID: Graphics accelerated VDI with the visual performance of a workstation." Nvidia Corp (2014).

[36] Hong, Hua-Jun, et al. "GPU consolidation for cloud games: Are we there yet?." Proceedings of the 13th Annual Workshop on Network and Systems Support for Games. IEEE Press, 2014.

[자문]

- 중앙대학교 전자전기공학부 홍철호 교수

주 의

1. 이 보고서는 소프트웨어정책연구소에서 수행한 연구보고서입니다.
2. 이 보고서의 내용을 발표할 때에는 반드시 소프트웨어정책연구소에서 수행한 연구결과임을 밝혀야 합니다.