직접 통로 기반 GPU 가상화 환경에서 GPU 연산시간의 길이가 가상머신의 공평성에 미치는 영향 분석†

강지훈*, 유헌창*, 길준민**,‡
*고려대학교 컴퓨터학과
**대구가톨릭대학교 IT공학부
e-mail : {k2j23h, yuhc}@korea.ac.kr, jmgil@cu.ac.kr

Analysis of the Influence of GPU Task Length on the Fairness of Virtual Machines in Direct Path-through based GPU Virtualization Environment

Jihun Kang*, Heonchang Yu*, Joon-Min Gil**
*Dept. of Computer Science and Engineering, Korea University
**School of IT Engineering, Catholic University of Daegu

요 약

직접 통로(Direct Pass-through) 기반 GPU(Graphic Processing Unit) 가상화 기법은 클라우드 환경에서 가상머신에게 GPU 장치의 기능을 지원하기 위한 일반적인 방법 중 하나이다. GPU 장치는 GPGPU 기술을 통해 연산을 가속화 할 수 있기 때문에 클라우드 환경에서도 가상머신에 고성능 연산을 지원하기 위해 많이 사용되고 있다. 하지만 기존 가상머신 스케줄링 기법은 가상머신의 CPU 사용 시간을 기반으로 스케줄링 되며, GPU 자원 사용을 고려하지 않는다. 본 논문에서는 GPU와 CPU 연산을 수행하는 가상머신들이 동시에 실행되는 환경에서 성능 실험을 통해 가상머신의 GPU 연산이 다른 가상머신에게 미치는 성능 영향과 GPU 작업 길이가 다른 가상머신에게 미치는 영향을 분석한다.

1. 서론

GPU는 GPGPU[1][2] 프로그래밍 모델을 통해 대규모 병렬처리를 지원하여 컴퓨터의 연산 성능을 비약적으로 증가시킬 수 있다. 클라우드 환경에서도 가상머신의 연산 성능을 향상시키기 위해 아마존[3]이나 알리바바[4] 같이 GPU를 지원하는 클라우드 서비스가 증가되고 있는 추세이다. 가상머신에게 GPU 장치를 할당할 수 있는 방법 중 하나인 직접 통로[5] 기반 GPU 가상화 기법은 클라우드 환경에서 가상머신이 GPU 장치에 직접 접근할 수 있도록 지원하여 GPU를 이용한 고성능 연산을 가능하게 한다. GPU 장치의 연산 성능 때문에 클라우드 환경에서 가상머신에게 효율적으로 GPU를 적용하기 위한 연구[6][7]도 지속적으로 진행되고 있다.

클라우드 환경에서 GPU 장치를 사용하는 것은 가상머신의 연산 성능 증가와 클라우드 서비스 품질 향상 등 다양한 이점을 얻을 수 있다. 하지만 GPU 장치와 GPGPU 프로그래밍 모델의 특성 그리고 GPU 장치를 고려하지 않은 기존 가상머신 스케줄러의 작동 방식으로 인해 가상머신 사이의 공평성 저하를 발생시킨다.

† 이 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2014R1A1A2055463).
‡ 교신저자

클라우드 환경의 핵심인 가상화 기술은 다수의 가상머신이 단일 물리 서버의 자원을 가상머신들이 공평하게 사용할 수 있도록 독립적으로 작동하는 가상머신 스케줄러를 통해 가상머신의 작업들을 스케줄링 한다. 일반적으로 가상화 플랫폼의 가상머신 스케줄러는 가상머신의 CPU 사용 시간을 기준으로 가상머신의 작업 순서를 스케줄링하며 모든 가상머신들이 공평하게 CPU를 사용할 수 있도록 지원한다.

하지만 범용 연산장치로 사용되는 GPU의 경우 가상화 환경에서 입출력 장치로 구분되기 때문에 가상머신의 GPU 사용 시간은 자원 사용 시간에 포함되지 않는다. 이러한 특징으로 인해 GPU 연산을 수행하는 가상머신은 CPU를 조금 사용한 가상머신으로 분류되어 다른 가상머신들 보다 CPU를 먼저 사용할 수 있게 되며, 이것은 공평성 저하 문제로 이어진다. 또한 GPGPU 프로그래밍 모델에서는 GPU에서 스레드들이 작업을 시작하면 스레드가 종료될 때 까지 컨텍스트 스위칭이나 작업 중지 없이 지속적으로 수행되기 때문에 작업이 시작되면 외부에서 직접 조절할 수 없게 된다.

결과적으로 GPU 연산을 수행하는 가상머신은 앞서 설명한 기존 환경의 특성으로 인해 더 많은 작업을 수행하지만 다른 가상머신들 보다 더 많은 자원 사용이 가능하고 다른 가상머신들의 영향을 받지 않기 때문에 가상머신

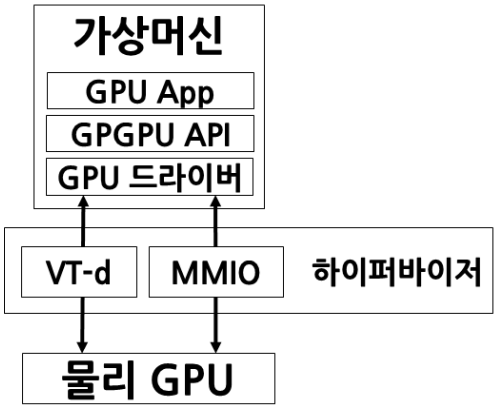
사이의 공평성에 문제를 발생시킬 수 있다.

본 논문에서는 클라우드 환경에서 GPU 연산의 길이가 가상머신에게 미치는 영향을 분석한다. 이를 위해 오픈소스 기반 가상화 플랫폼인 Xen[8]에서 여러 개의 가상머신을 사용해 GPU, CPU 집약적 연산이 동시에 수행되는 환경에서 성능 측정 실험을 통해 GPU 연산이 다른 가상머신에게 미치는 영향을 확인하고 GPU 연산의 길이가 다른 가상머신에게 미치는 영향을 분석한다.

2. 배경

2.1 직접 통로 기반 GPU 가상화

직접 통로 기반 가상화는 물리 서버 입출력 장치를 가상화 하는 기술이다. 이 기술을 통해 클라우드 환경에서 가상머신에게 GPU를 할당하여 고성능 연산 지원이 가능하게 된다. 가상화 환경에서 여러 개의 가상머신이 CPU, 메모리, 하드디스크와 같은 물리 자원을 공유하는 것과 달리 직접 통로 기반 GPU 가상화[9]는 단일 가상머신이 단일 물리 GPU를 독점하게 된다.

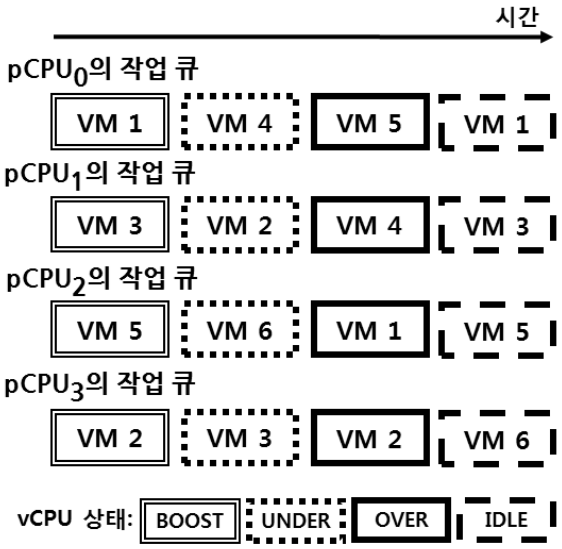


(그림 1) 직접 통로 기반 GPU 가상화 환경

그림 1은 직접 통로 기반 GPU 가상화 환경의 구조를 보여준다. 직접 통로 기반 GPU 가상화 환경은 VT-d와 같은 하드웨어 지원 기술을 사용해 가상머신이 GPU 장치에 직접 접근할 수 있도록 허용한다.

2.2 Xen의 가상머신 스케줄링 기법

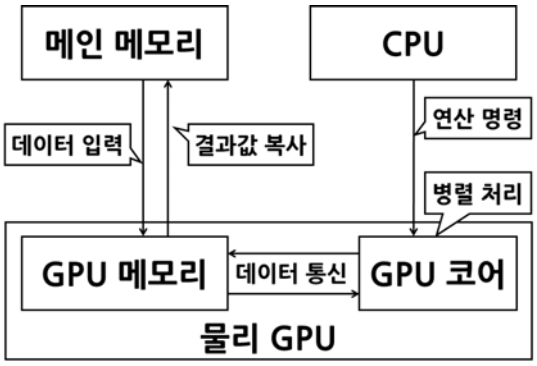
오픈소스 가상화 플랫폼 Xen은 일반적인 가상화 플랫폼과 마찬가지로 가상머신을 위한 독립적인 스케줄러를 사용한다. 가상머신 스케줄러는 단일 물리 머신의 자원을 다수의 가상머신들이 공평하고 효율적으로 공유할 수 있도록 자원 접근 순서를 정하며, 결과적으로 모든 가상머신들이 공평하게 CPU에 접근할 수 있도록 지원한다. Xen에서는 가상머신 스케줄링을 위해 기본적으로 크레딧 스케줄러[10]를 사용한다. 실행중인 모든 가상머신에게 미리 지정된 크기의 크레딧 값을 할당하고 가상머신이 CPU를 사용할 때 마다 크레딧 값을 차감시킨다. 그리고 가상머신 스케줄러는 크레딧 값이 많이 남아있는 가상머신이 CPU를 덜 사용한 것으로 분류하여 해당 가상머신이 CPU에 먼저 접근할 수 있도록 순서를 조정하여 모든 가상머신들이 공평하게 CPU를 사용할 수 있도록 지원한다.



(그림 2) Xen의 가상머신 스케줄러의 작동 방식

Xen의 가상머신 스케줄러는 그림 2와 같이 가상머신과 가상머신의 작업 상태에 따라 Boost, Under, Over 중 하나의 우선순위를 할당 하며, 자원 사용량이 아주 적은 가상머신들은 유휴 상태인 Idle 우선순위를 할당한다.

2.3 GPGPU 프로그래밍 모델



(그림 3) GPGPU 프로그래밍 모델

그림 3은 GPGPU 작업의 흐름을 보여준다. GPU는 메인 메모리에 접근할 수 없기 때문에 작업에서 사용되는 데이터들은 GPU 장치의 내장 메모리에 존재해야만 한다. 하지만 GPU 장치 특성상 GPU 메모리에 직접 접근이 불가능하기 때문에 연산 전후로 메인 메모리와 GPU 메모리 사이의 데이터 입출력이 필수로 수행된다. 또한 GPU 작업 시 CPU에게 명령어를 전달받고 실제 연산은 GPU에서 수행되며, 클라우드 환경에서 가상머신은 직접 통로 기술을 통해 GPU에 직접 접근해서 작업을 수행한다.

2.4 가상머신의 GPU 작업

기존 가상머신 스케줄링 기법은 가상머신들의 CPU 사용 시간을 기준으로 자원 접근 순서를 결정한다. 즉, 연산 작업을 공평하게 수행하도록 작업 순서를 조정하여 가상머신의 공평성을 보장해준다. 하지만 기존 환경에서 GPU 장치는 입출력장치로 분류되어 GPU 연산을 수행하는 가상머신의 실제 작업은 GPU에서 수행된다. 하지만 기존 가상화 환경에서 가상머신 스케줄러는 CPU 사용 시간을 기반으로 스케줄링하기 때문에 GPU 사용 시간은 자원 사용 시간에 포함되지 않아 자원을 덜 사용한 것으로 분류된다. 때문에 GPU 연산을 수행하는 가상머신은 다른 가상머신들 보다 CPU를 먼저 사용할 수 있으며 더 많이 사용하게 된다.

또한 기존 가상화 환경에서 가상머신들은 일반적인 컴퓨팅 작업을 수행하면서 CPU나 입출력 컨트롤러 등을 공유한다. 하지만 직접 통로 기반 GPU 가상화는 가상머신들 간의 자원 경쟁 없이 하나의 가상머신이 독점하기 때문에 자원 공유로 인한 성능 저하 없이 GPU 작업을 수행할 수 있다.

3. GPU 연산 길이가 가상머신에게 미치는 영향

이 절에서는 실험을 통해 GPU 연산으로 인한 공평성 저하 문제를 확인하고 GPU 연산 길이에 따라 가상머신들에게 미치는 영향을 실험을 통해 분석한다.

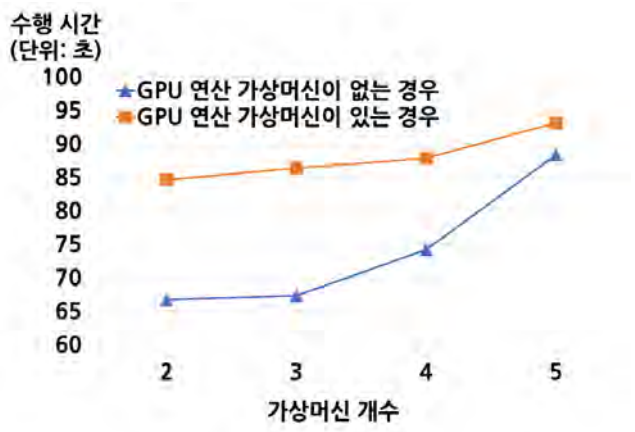
본 논문의 실험은 오픈소스 가상화 플랫폼인 Xen에서 게스트 OS로 Windows 7을 사용하는 가상머신을 생성해서 수행하며, 가상머신 하나에만 직접 통로 방식으로 GPU를 할당하여 OpenCL을 사용한 GPGPU 작업을 수행시키며, 다른 가상머신은 CPU를 사용한 연산을 수행 시킨다. 자세한 실험 환경은 <표 1>과 같다.

<표 1> 실험 환경

	Host	VM
CPU	Intel Xeon E3-1231V3	4 VCPU
Memory	32GB	4GB
HDD	1TB	100GB
GPU	-	Radeon HD 6570 (가상머신 1개에만 할당)
OS	Ubuntu 14.04	Windows 7
Hypervisor	Xen 4.4.1	-

실험은 GPU 연산을 수행하는 가상머신이 있는 경우와 없는 경우에 실행 중인 모든 가상머신의 성능을 측정하여 비교한다. 또한 GPU 연산을 분할하여 작업 길이를 다르게 하여 GPU 작업 길이가 다른 가상머신의 성능에 미치는 영향을 확인하고 실험 결과를 통해 이를 분석한다. 가상머신은 GPU 연산을 수행하는 가상머신 하나와 CPU 연산을 수행하는 가상머신을 1개~4개로 개수를 증가시켜가며 실험을 수행한다. GPU 연산을 수행하는 가상머신은 부동소수점으로 구성된 5120×5120 행렬 곱셈을 4회 연속 수행하고 CPU 연산을 수행하는 가상머신은 1600×1600 행렬 곱셈을 수행한다.

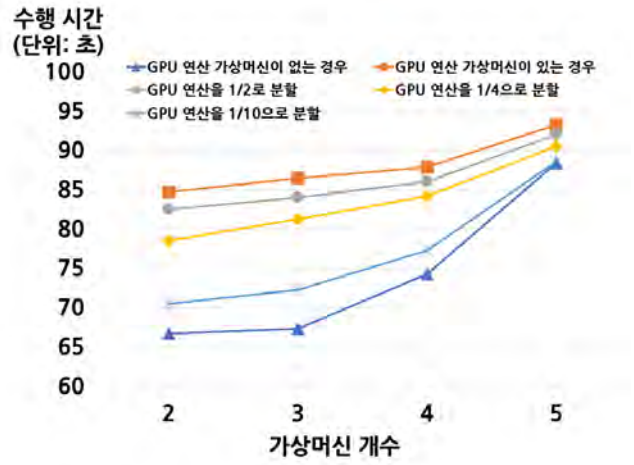
가상머신에서 수행하는 CPU 연산과 GPU 연산은 개별 수행 시 각각 약 1분, 약 73초 동안 수행되며, 성능 비교를 위한 충분한 시간과 GPU와 CPU 연산이 동시에 수행되는 시간을 최대화하기 위해 연산 크기를 비슷하게 조정하였다. GPU 연산의 경우 GPU의 하드웨어 성능과 GPU 메모리의 크기를 고려해 행렬 사이즈를 결정했다.



(그림 4) GPU 연산 가상머신으로 인한 성능 저하

그림 4는 GPU 연산을 수행하는 가상머신의 존재 유무에 따라 CPU연산을 수행하는 가상머신의 평균 성능(오차범위 약 2초 내외)을 측정한 결과이다. GPU 연산을 수행하는 가상머신과 CPU 연산을 수행하는 가상머신을 동시에 실행 시켰을 때 GPU 연산을 수행하는 가상머신으로 인해 다른 가상머신들의 성능이 저하된 것을 확인할 수 있다. 이것은 GPU 연산의 특징을 고려하지 않은 가상머신 스케줄러의 작동 방식으로 인해 발생하는 성능 불균형 문제이다. 앞서 설명한 것과 같이 기존 가상머신 스케줄링 기법은 GPU를 입출력 장치로 분류하지만 GPU 연산 대부분이 GPU에서 수행되는 특징으로 인해 GPU 연산을 수행하는 가상머신은 CPU 사용 시간을 덜 소모하며, CPU 사용이 요구될 때 다른 가상머신보다 더 먼저 작업을 수행할 수 있게 된다. 이러한 특징으로 인해 CPU 연산을 수행하는 가상머신은 자원 사용 기회를 빼앗기게 되는 것이다. 가상머신이 많을수록 성능 격차가 감소되는 이유는 CPU를 사용하는 경쟁자가 많아지면서 전체적으로 성능이 저하되기 때문이다.

다음 실험은 가상머신에서 수행하는 GPU 작업의 길이가 다른 가상머신들의 성능에 미치는 영향을 확인하기 위한 실험이다. CPU 연산은 앞서 설명한 것처럼 행렬 곱셈을 수행하고 GPU 연산은 행렬 곱셈 연산의 전체 처리량은 유지하고 작업 수행 시 한 번에 수행되는 스레드 그룹의 개수를 분할하여 GPU 작업 크기를 조절한다.



(그림 5) GPU 연산의 길이가 미치는 성능 영향

그림 5는 가상머신에서 처리하는 GPU 작업 길이에 따른 CPU 연산의 성능을 보여준다. 실험 결과에서 보여주듯이 GPU 연산 시간이 짧을수록 GPU 연산을 수행하는 가상머신으로 인한 성능 저하가 적어지는 것을 확인할 수 있다. 가상머신에서 수행되는 GPU 연산을 분할해서 수행하게 되면 GPU 연산을 수행하는 가상머신이 다음 GPU 작업을 수행하기 전에 다른 가상머신의 자원 접근 기회가 GPU 연산 전체를 한 번에 수행하는 경우 보다 훨씬 많아지기 때문에 GPU 연산을 수행하는 가상머신으로 인한 성능 저하를 방지하고 가상머신 사이의 공평성을 보장해 줄 수 있다. GPU 연산을 분할해서 수행하게 되면 GPU 작업 시작을 위한 초기화와 종료 시 발생하는 자원 해제 작업이 많아져 GPU 연산의 성능 저하가 발생한다. 하지만 한 번에 처리했을 때와 비교해 최대 약 7% 이내로 비교적 적게 발생하며, 가상화 환경 특성상 가상머신이 증가할 경우 자원 경쟁으로 인해 가상머신의 성능이 저하되는 것을 고려하면 다른 가상머신의 성능 저하를 방지하기 위해 GPU 연산을 수행하는 가상머신의 성능 저하는 허용 가능한 수준이라고 할 수 있다.

4. 결론 및 향후연구

본 논문에서는 실험을 통해 GPU 연산을 수행하는 가상머신으로 인한 공평성 저하 문제를 확인하고 GPU 연산의 길이가 다른 가상머신의 성능에 미치는 영향을 분석하였다. 앞서 수행한 실험을 통해 GPU 연산을 수행하는 가상머신으로 인해 다른 가상머신의 성능이 저하되어 가상머신 사이의 공평성이 저하되는 것을 확인할 수 있었다. 또한 GPU 연산의 길이가 짧으면 GPU 연산이 끝나고 다음 작업을 수행하기 전에 다른 가상머신들의 자원 접근 기회가 증가하여 결과적으로 GPU 연산으로 인한 가상머신의 성능 저하 격차를 감소시켜 가상머신 사이의 성능 공평성이 보장되는 것을 확인할 수 있었다.

앞서 설명한 기존 가상화 환경의 특징과 실험 결과를 통해 알 수 있듯이 가상머신의 GPU 사용은 자원 사용 시간에 포함되지 않기 때문에 GPU 연산을 수행하는 가상머신은 CPU 연산을 수행하는 가상머신들 보다 동일한 시간 동안 더 많은 작업을 수행함에도 불구하고 CPU 연산을 수행하는 가상머신보다 자원을 덜 사용한 가상머신으로 분류된다. 때문에 GPU 연산을 수행하는 가상머신은 다른 가상머신들 보다 CPU를 먼저 사용할 수 있게 되며, 이는 가상머신의 자원 사용에 대한 공평성 불균형 현상을 발생시키고 다른 가상머신들은 성능 저하가 발생한다.

하지만 본 논문의 실험 결과에서 보여주는 것 같이 가상머신의 GPU 연산의 길이를 조절하여 다른 가상머신의 자원 접근 기회를 보장해줌으로써 실행중인 가상머신들의 공평성을 기존 환경보다 향상시킬 수 있었다. 가상머신에서 수행하는 GPU 연산을 분할하고 나눠서 처리하여 작업 처리 중간에 다른 가상머신들이 자원을 사용할 수 있는 기회를 만들어주어 GPU 연산의 길이가 짧게 분할될수록 공평성이 향상되는 것을 볼 수 있었다.

본 논문에서는 가상머신에서 수행하는 GPU 연산의 길이를 임의로 조절하였지만 추후 연구에서 본 논문에서 수행한 실험 결과를 바탕으로 가상머신 사이의 공평성을 보장할 수 있도록 GPU 연산의 작업 길이를 동적으로 조절할 수 있는 GPU 작업 관리 기법과 가상머신 스케줄러와의 연동에 대한 연구를 진행할 계획이다.

참고문헌

[1] OpenCL: Open Computing Language. https://www.khronos.org/opencl/.

[2] CUDA: Compute Unified Device Architecture. http://www.nvidia.com/object/cuda_home_new.html.

[3] Amazon GPU Instance. https://aws.amazon.com/ec2/instance-types/?nc1=f_ls.

[4] Aliyun. https://hpc.aliyun.com/product/gpu_bare_metal

[5] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger,G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B.Vembu, and J. Wiegert. “Intel virtualization technology for directed I/O”. Intel Technology Journal, 10, August, 2006.

[6] Suzuki, Y., Kato, S., Yamada, H., & Kono, K. “Gpuvm: Gpu virtualization at the hypervisor”. IEEE Transactions on Computers, 65(9), 2016, 2752-2766.

[7] Tian, Kun, Yaozu Dong, and David Cowperthwaite. “A Full GPU Virtualization Solution with Mediated Pass-Through”. USENIX Annual Technical Conference. 2014; 121-132

[8] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. “Xen and the art of virtualization”. In Proceedings of the nineteenth acm symposium on operating systems principles, SOSP ’03. ACM:New York, NY, USA, 2003; 164 - 177.

[9] VGA Passthrough. https://wiki.xen.org/wiki/Xen_VGA_Passthrough. [Accessed date: 20 February 2017].

[10] Credit Scheduler. https://wiki.xen.org/wiki/Credit_Scheduler.