RPC 기반 GPU 가상화 환경에서 GPU 메모리의 초과 사용 시 발생하는 가상머신 사이의 성능 불균형 문제 분석†

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

Analyzing performance imbalance between virtual machines caused by excessive use of GPU memory in RPC-based GPU virtualization environments

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

요 약

클라우드 환경에서는 가상머신의 고성능 연산을 지원하기 위해 Graphic Processing Unit(GPU)를 사용한다. 가상머신들은 공평성을 위해 독립적인 가상머신 스케줄러를 사용하기 때문에 컴퓨팅 자원의 초과 사용으로 인한 성능 저하가 발생해도 동일한 작업을 수행하는 가상머신들의 성능은 균등하게 측정된다. 하지만 GPU 연산의 경우 다중 작업을 수행할 때 하드웨어 기반 스케줄러를 사용하며 가상머신의 입출력 작업을 위한 하이퍼바이저의 First In First Out(FIFO) 기반 스케줄링 기법으로 인해 가상머신 사이의 공평성을 보장할 수 없다. 본 논문에서는 GPU 메모리를 초과 사용하는 환경에서 가상머신들의 성능을 측정하고 성능 불균형으로 인한 문제를 분석한다.

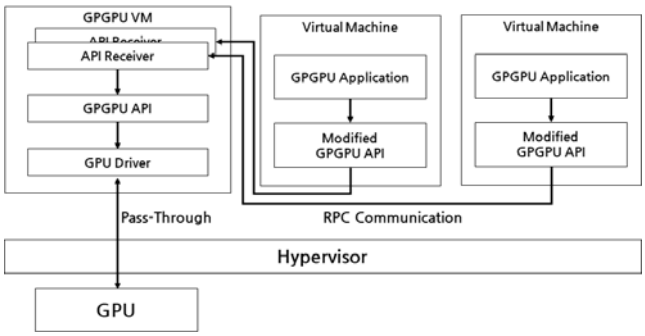
1. 서론

수천개의 연산 코어를 사용한 대규모 병렬처리를 통해 고성능 연산을 지원하는 Graphic Processing Unit(GPU)는 클라우드 영역에서도 가상머신에게 고성능 연산을 지원하기 위해 사용되어왔다. 하지만 사용자의 가상머신 사이에 공평성을 중요시하는 클라우드 환경에서 자원 공유를 고려하지 않은 GPU의 하드웨어 기반 FIFO 스케줄링 방식은 공평성을 제공하는데 제약이 따른다. 가상화 환경에서 다수의 가상머신이 GPU를 공유하는 환경에서의 GPU task의 공평한 스케줄링을 통해 공평성 저하 문제를 해결하기 위한 다양한 연구[1]도 제안되었다. 하지만 GPU task 스케줄링을 통한 기존 연구들은 GPGPU 작업에서 GPU 연산 전후에 필수적으로 수행되는 GPU 메모리의 데이터 입출력 작업에 대한 스케줄링을 고려하고 있지 않으며 GPU 메모리의 초과 사용을 고려하지 않는다.

본 논문에서는 다수의 가상머신이 단일 GPU를 공유하고 있는 RPC 기반 GPU 가상화 환경에서 다수의 가상머신에서 실행되는 General Purpose Graphic Processing unit(GPGPU) 작업의 GPU 메모리 요구량이 가용 GPU 메모리를 초과했을 때의 가상머신 성능을 측정한다. 또한, 실험에서는 다양한 규모의 GPGPU 작업을 사용해 GPU 메모리 초과 사용 정도에 따른 가상머신 사이의 성능 불균형 문제를 분석한다.

2. 기반 기술

본 논문에서는 GPU 메모리 초과 사용으로 인한 가상머신 사이의 성능 불균형 문제를 분석하기 위해 우리의 이전 연구를 통해 오픈소스 기반 가상화 플랫폼 Xen[2]에서 구성한 RPC 기반 GPU 가상화 환경을 사용한다. 본 논문에서 사용한 RPC 기반 GPU 가상화 환경의 전체 구조는 (그림 1)과 같다.



(그림 1) 본 논문의 RPC 기반 GPU 가상화 시스템

† 이 논문은 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2019R1F1A1062039).

본 논문의 RPC 기반 GPU 가상화 시스템에서는 사용자 가상머신이 RPC를 통해 요청하는 GPGPU 작업을 처리하기 위한 RPC 서버 역할을 수행하는 GPGPUvm이라고 불리는 별도의 가상머신을 사용한다. 가상머신들은 GPGPU 작업을 수행할 때 GPGPUvm과 RPC 통신을 통해 API를 전송할 수 있도록 수정된 OpenCL[3]을 사용하며GPGPUvm은 Direct Pass-through[4] 방식을 통해 GPU에 독점적으로 접근한다.

GPGPU 작업은 앞서 설명한 것과 같이 GPU 연산 전후로 데이터 입출력 작업을 수행한다. GPU 코어가 연산을 수행하기 위해서는 연산에 사용되는 데이터가 GPU 메모리에 존재해야 하지만 GPU 메모리에 데이터를 직접 작성하거나 직접 읽을 수 없으므로 메인 메모리와 GPU 메모리 사이의 데이터 통신을 통해 연산에 사용될 데이터를 메인 메모리에서 GPU 메모리로 복사하고 연산 결과값은 GPU 메모리에서 메인 메모리로 복사하는 입출력 작업을 필수적으로 수행한다.

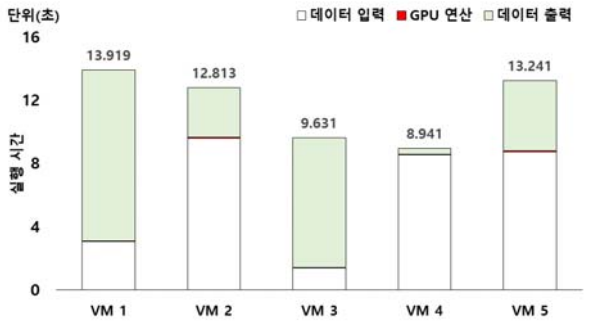
3. 실험

이번 장에서는 실험을 통해 GPU 메모리의 초과 사용 상황에서 다수의 가상머신이 GPGPU 작업을 동시에 실행할 때 GPU 메모리 경쟁으로 인해 발생하는 성능 불균형 문제를 분석한다. 실험 환경은 <표 1>과 같다.

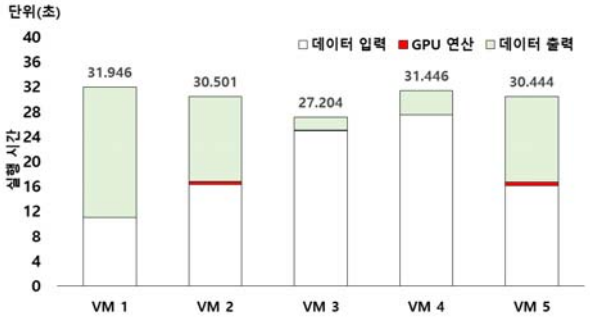
<표 1> 실험 환경

	Host Machine	GPGPUvm	VM
CPU (vCPU)	Intel Xeon E3-1231V3	4 vCPU	4 vCPU
Memory	32 GB	5 GB	4 GB
GPU	-	Radeon HD 6750 (1GB GPU Memory)	-
OS	Ubuntu 14.04	Windows 7	Windows 7
Hypervisor	Xen 4.4.1	-	-

본 논문은 실험을 위해 5개의 가상머신을 사용하며 RPC 통신을 통해 전달된 가상머신들의 GPGPU 작업을 대신 처리하기 위해 RPC 서버 역할을 수행하는 1개의 GPGPUvm을 사용한다. 가상머신들의 성능 측정을 위해 가상머신에서 OpenCL을 사용해 구현된 행렬 곱셈을 실행한다.



(그림 2) 7168x7168 행렬 곱셈 성능



(그림 3) 8192x8192 행렬 곱셈 성능

(그림 2)와 (그림 3)은 5개의 가상머신을 사용해 각각 7168x7168 그리고 8192x8192 크기의 행렬 곱셈을 동시에 실행했을 때의 성능을 보여준다. 7168x7168 행렬 곱셈과 8192x8192 행렬 곱셈은 5개의 가상머신을 사용해 동시에 실행시키기 위해서는 각각 2940 MB, 3840 MB의 GPU 메모리가 필요하므로 GPGPU 작업을 수행하게 되면 GPU 메모리 부족 상황이 발생한다. 이로 인해 GPU 메모리의 데이터 입출력 경쟁으로 인해 각각의 가상머신들은 성능 차이가 발생하게 되며, 특히, GPU 연산시간의 경우 각각의 가상머신이 큰 차이를 보이지 않지만, 데이터 입출력 작업의 경우 큰 성능 차이를 확인할 수 있다

4. 결론 및 향후연구

본 논문에서는 실험을 통해 GPU의 기존 스케줄링 방식은 GPU 메모리를 초과 사용한 경우에 공평성을 보장하는데 효율적이지 않다는 것을 확인하였다. 실험 결과에서 보여주듯이 각 가상머신에서 실행하는 GPGPU 작업의 GPU 연산시간은 균일했지만, GPU 메모리의 입출력 시간은 가상머신 사이에서 성능 차이가 존재하는 것을 확인했다. 추후 연구에서는 각 가상머신에서 실행되는 GPGPU 작업의 균등한 입출력 작업 스케줄링 기술과 각 가상머신이 GPU 메모리를 균일하게 사용할 수 있도록 지원하는 GPU 메모리 격리 기법을 연구하는 것이다.

참고문헌

[1] Zhou, Husheng, Guangmo Tong, and Cong Liu. "GPES: A preemptive execution system for GPGPU computing." 21st IEEE Real-Time and Embedded Technology and Applications Symposium. IEEE, 2015.

[2] 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.

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

[4] VGA Passthrough. https://wiki.xen.org/wiki/Xen_VGA_Passthrough.