응용 프로파일링 정보에 따른 GPU 작업 컨테이너 실행 전략

오지선 ○ 김세진 ○ 진계신◇ 김윤희 ○*

숙명여자대학교 컴퓨터과학과, 서울대학교 컴퓨터공학부◇

○{jsoh8088, wonder0702, yulan*}@sookmyung.ac.kr, charliecqc@dcslab.snu.ac.kr◇

A Execution Scenario of GPU Job Containers using Application Profiling

Jisun Oh○ Sejin Kim○ Qichen Chen◇ Yoonhee Kim*

Dept. of Computer Science, Sookmyung Women's University○

Dept. of Computer Science and Engineering, Seoul National University◇

요 약

GPU (Graphics Processing Unit)가 고성능 컴퓨팅에 필수적인 요소가 됨에 따라 클라우드 컴퓨팅 환경에서 GPU를 제공하는 데는 여전히 어려움이 있다. 현재 GPU를 제공하는 클라우드 환경을 제공하는 Kubernetes나 Yarn의 작업 스케줄러는 단일 노드에 작업을 배치하고 이전 노드가 완료 될 때까지 새 작업을 할당하지 않는 문제점이 있다. 본 논문에서는 응용 실행 패턴에 따라 적절한 GPU 노드에 작업 컨테이너를 효율적으로 할당 및 공유하는 방법을 제안한다. HPC 응용의 사전 실행 프로파일링 데이터를 기반으로 하여 적용하였다. 워크로드 작업의 GPU 노드 독점 및 공유를 결정하여 가능한 많은 자원을 짧은 수행시간으로 자원 집약적인 워크로드에 제공할 수 있음을 보였다.

1. 서 론

최근에 컴퓨터 기술이 발전함에 따라 고성능 컴퓨팅 워크로드를 많이 사용하고 있는 추세이다. 최근에는 이러한 워크로드는 개발 및 배치를 용이하게 하기 위해 각 응용을 컨테이너에 이식하여 사용한다. 이러한 우수한 확장성 및 이식성으로 인해 CaaS(Container as a Service) [1] 개념이 등장하였다. CaaS는 제공 업체가 컨테이너, 응용 프로그램 및 클러스터를 배포 및 관리하기 위한 프레임워크를 사용자에게 제공하는 컨테이너 기반 가상화를 위해 새롭게 대두되고 있다. 현재는 HPC(High Performance Computing) 응용 분야에서 뚜렷한 효과를 보이고 있다. 따라서 HPC 워크로드가 점점 중요해지고 처리하는 데이터 양도 많아짐에 따라 컨테이너 클라우드 환경에서의 제공이 필요하다.

많은 HPC 워크로드들은 계산 집약적인 특성을 가지고 있어 GPU(Graphic Process Unit 에서의 실행이 적합하다. GPU에서의 워크로드는 고성능 컴퓨팅의 높은 병렬 처리로 인하여 더 높은 성능을 달성할 수 있다. 하지만 GPU VM(Virtual Machine)는 일반적인 CPU VM보다 비용이 10배정도 더 높기 때문에 클라우드 서비스 제공 업체들은 스케줄러를 통해서 수많은 GPU의 더 효율적인 사용을 추구하고 있다. 하지만 컨테이너 클러스터 플랫폼인 Kubernetes [2]나 Yarn [3]은 CPU 기반인 스케줄러를 GPU에 적용하여 사용한다. GPU 자원 및 GPU 작업 부하를 관리하는 경우 기존 스케줄러는 워크로드의 GPU 사용량 및 GPU 메모리 사용량에 관계없이 단일 GPU에만 하나의 작업을 할당하는 문제점이 있다. 따라서 GPU의 다중 워크로드 실행 뿐만 아니라 응용 특성을 반영하여 컨테이너의 자원 공유 및 배치를 해야 한다.

본 논문의 구성은 다음과 같다. 1장 서론에 이어 2장에는 관련연구를 살펴 본다. 3장에서는 스케줄링 방식의 문제점과 컨테이너의 GPU 공유 및 응용 특성 반영 실행 방법을 보인다. 4장에서는 실험에 대해 살펴보고 마지막으로 5장에서는 결론을 맺는다.

2. 관련 연구

GPU 스케줄링 방법은 클라우드 환경에서 모든 사용자들에게 GPU를 공정하고 효과적으로 배포하기 위해 필요하다. 그러나 GPU 가상화 환경에서 GPU는 비선점적 스케줄링 방식으로 제공되고 있다. 자원 소모가 적은 작업을 장시간 실행하면 소프트웨어가 작업을 끝낼 때까지 선점 할 수 없는 문제점이 있다.

이러한 문제를 해결하기 위해 VGRIS [4]는 클라우드 컴퓨팅에 배포된 게임 응용 프로그램에 대한 GPU 예약 문제를 해결하였다. VGRIS는 서로 다른 성능 요구 사항을 충족시키는 스케줄링 정책을 도입하였다. SLA 요구 사항을 충족시키기 위해 각 VM에 최소 GPU를 제공한다. 그러나 소수의 VM 만 사용할 수 있는 경우

* 교신저자

GPU 사용률을 낮출 수 있다. 또한 Kato [5]는 TimeGraph의 우선 순위 기반 스케줄링 정책을 사용하여 GPU 리소스를 공정하게 배포하였다.

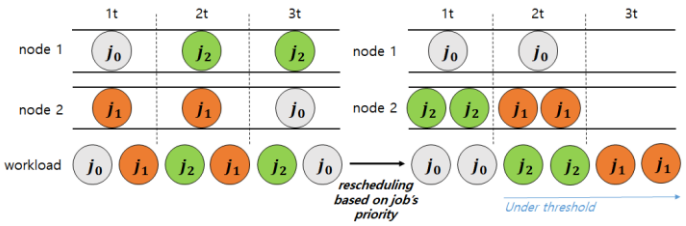
이전 관련 연구와는 달리 본 논문에서 제안하는 기법은 게임 작업 부하 대신 일반 HPC 작업 부하에 초점을 둔다. GPU 사용 및 GPU 메모리 사용에서 HPC 일반 작업 부하의 실행 특성이 더 많으므로 고정된 메트릭을 사용하는 것은 효율적으로 모든 워크로드 수행을 계획하기에 문제가 있다. 제안된 기법은 다양성을 처리하기 위해 프로파일링 단계를 도입하여 실행 메트릭을 수집하고 프로파일링된 데이터를 입력으로 사용하여 우선 순위를 계산하고 배치함으로써 자원 사용률을 높일 수 있다.

3. 컨테이너 작업 프로파일링을 통한 실행 전략

본 장에서는 컨테이너 클러스터 플랫폼인 Kubernetes 스케줄러의 문제점을 설명하고, 문제점을 해결하기 위한 자원 사용 기록에 따른 우선 순위 기반 배치 기법을 소개한다.

기본 스케줄링 정책의 문제점은 그림 1의 왼쪽 부분에서 자세히 설명한다. 2 개의 GPU 노드와 3 개의 응용 j_0, j_1, j_2를 포함하는 6 개의 작업이 있으며 각 작업의 실행 시간은 동일하다고 가정한다. 컨테이너는 각 노드에 작업이 할당되자 마자 생성된다. 그림에서 보듯이 스케줄러를 사용할 때 총 실행 시간은 3t이며 작업량이 적은 자원 일지라도 각 노드가 동시에 하나의 작업 만 할당 할 수 있기 때문에 실제 자원이 낭비된다. 적은 양의 자원이 필요한 작업이 생성되면 더 많은 자원이 낭비될 수 있다.

기본 스케줄링 전략에서의 리소스 낭비 문제를 해결하기 위해 작업 노드 할당을 위한 프로파일링 정보를 사용하는 우선 순위 기반 배치 방법을 예를 통해 설명한다. 우선 순위는 리소스에서 독점적으로 실행되거나 공유되는 작업을 결정하는 기준을 제공한다. 그림 1의 오른쪽에 표시된 것처럼 작업은 리소스 사용률에 따라 정렬된다. 많은 자원을 소비하는 작업은 단일 노드에 배타적으로 사용되는 반면, 자원 활용도가 낮은 작업은 다른 노드에 함께 배치된다. 작업이 노드를 공유하고 병렬로 실행되므로 총 실행 시간을 2t로 줄일 수 있다. 따라서 GPU 공유를 통한 작업 할당은 다중 실행을 활용하여 GPU 사용률을 향상시킬 뿐만 아니라 총 실행 시간을 줄일 수 있다.



[그림 1] 기본 및 우선 순위 기반 배치 방법을 적용한 작업 배치 예제

4. 시나리오 실험 및 결과

본 논문에서 제안하는 기법의 성능 향상을 입증하기 위해 두 가지 다른 조건과 비교하였다.

1) Binpack : 한 노드에서 GPU 리소스가 모두 소모될 때까지 동일한 노드에 컨테이너를 배치한다.

2) Spread : 최소 컨테이너 수의 노드에 새 컨테이너를 배치한다.

3) : 본 논문에서 제안한 방법이다.

Google은 심층 학습 및 과학 컴퓨팅에 최적화 된 GPU 가속 플랫폼 인 NVIDIA GPU Cloud (NGC) [6]의 5 가지 응용 프로그램을 사용했으며 자세한 정보는 아래와 같다. 실험의 시스템 정보는 표 1에 나와 있다.

[표 1] 실험 환경

	Node Details		
	CPU(master)	***GPU(profiling)***	***GPU(monopoly/ sharing)***
Architecture	Intel® Core™ i7-5820K	Nvidia GeForce GTX 1070	Nvidia GeForce Titan Xp D5x
Core Clock	3.30GHz	1.506GHz	1.58GHz
Num of Cores	6	1920	3840
Mem. Size	32GB	8GB	12GB
Threading API	-	Nvidia CUDA 10.0	Nvidia CUDA 10.0

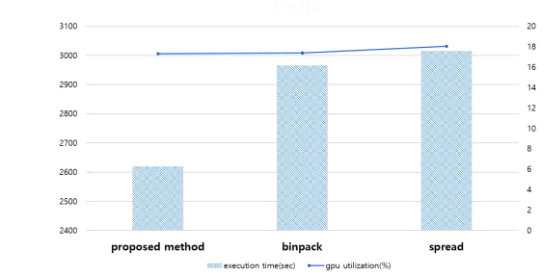
[표 2] 응용 프로파일링 정보

Application	Application Profiling Data		
	Execution Time(s)	***GPU utilization(%)***	***GPU memory usage(MB)***
LAMMPS	1195	23	8190
GROMACS	715	71	500
QMCPACK	190	46	5125
CNN	180	100	2190
BIGDFT	611	8	6367

LAMMPS [7]는 분자 동역학 시뮬레이션을 위해 설계된 응용 프로그램이다. 이 실험에서는 Leanard Jones의 3D melt 예제가 사용되었으며, binsize는 2.8로 설정되었고 LJ 시스템의 차원은 각각 8로 설정되었다. BigDFT [8]는 선형 스케일링 방법을 사용할 수 있는 웨이블릿베이스 세트를 사용하여 DFT를 대량으로 병렬화 하는 전자 구조 코드이다. 이 실험에서는 GPU 가속 FeHyb 벤치 마크를 사용하였다. GROMACS [9]는 뉴턴의 운동 방정식을 시뮬레이션하도록 고안된 분자 동역학 응용 프로그램이다. 이 실험에서 물 GMX50 베어 벤치 마크는 MPI 실행을 통해 1536K의 물 분자 데이터 세트에 대해 병렬로 실행하였다. QMCPACK [10]은 QMC(Quantum Monte Carlo) 알고리즘을 구현하는 고성능 전자 구조 코드이다. MPI 실행을 사용하여 QMC 알고리즘이 니켈 산화물의 32 개 원자

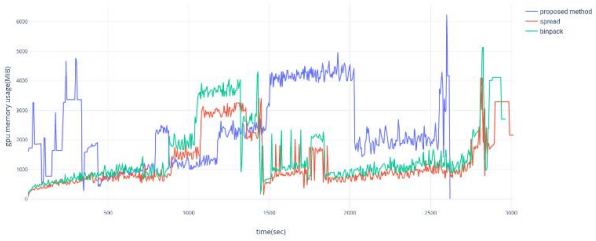
셀에 대해 병렬로 실행되었다. Convolution Neural Network (CNN) [11]는 NGC의 Tensorflow [12] 이미지를 사용하여 GPU에서 병렬로 실행되었다.. MNIST 데이터의 60,000 세트가 입력으로 사용되었으며, 28 * 28 매트릭스 형태로 된 손으로 쓴 graphemes의 예이다.

5 개의 NGC 응용 프로그램에 대한 프로파일 링 데이터를 수집하기 위해 프로파일링 노드에서 응용 프로그램을 실행하며 정보는 표 2에 나와 있다. 실험에 사용 된 워크로드는 LAMMSP, GROMACS, QMCPACK, CNN 및 BIGDFT 순서로 각각 2개, 2개, 2개, 3개, 1개 구성하여 실행하였다.



[그림 2]3가지 스케줄링의 GPU 활용도 및 수행시간 비교

그림 2는 3가지의 스케줄링 방법의 수행시간, GPU 활용률 결과를 비교하는 그림이다. 왼쪽 y 축은 수행시간, 오른쪽 y축은 평균 GPU 사용률을 의미한다. 제안하는 방법, binpack, spread 각각 2620s, 2965s, 3015s 소요되었다. Binpack, spread 방법은 워크로드의 작업들을 차례대로 수행해야 하기 때문에 자원을 공유하는 제안된 방법 보다 더 오랜 수행시간이 걸린다. 제안하는 방법은 나머지 두 방법에 비해 수행시간이 가장 짧으며, 수행시간을 11.3%, 11.5% 향상시킬 수 있다. 실험 수행 시 사용되는 노드의 모든 GPU 사용률을 평균 내었다. 제안하는 방법, binpack, spread 각각 17.3634%, 18.019%, 17.2958%의 평균 GPU 사용률을 나타낸다. 3개의 응용의 평균 GPU 사용률의 차이는 거의 없다. 이는 제안하는 방법이 GPU 자원 경쟁 없이 실행되어 자원을 잘 공유하였음을 보여준다.



[그림 3]3가지 스케줄링의 GPU 메모리 사용 비교

그림 3은 3가지의 스케줄링 방법의 GPU 메모리 사용량을 비교한 그림이다. Workload 수행 시, GPU 4개의 총 GPU 메모리 사용량을 총 수행시간으로 나누어 평균 내었다. 제안하는 방법, binpack, spread 각각 사용하는 GPU 평균 메모리는 2307MB, 1512MB, 1247MB이다. 제안하는 방법이 여러 응용을 한 노드 안에서 공유하면서 실행시키므로 더 많은 GPU 메모리를 사용할 수 있다. 그림 6에서 볼 수 있듯이, 총 수행시간에서 900s~1400s 구간을 제외하고 제안하는 방법이 다른 두 방법과 비교해서 GPU 메모리를 많이 사용하는 것을 알 수 있다.

전체 실험 결과를 통해 본 문에서 제안하는 스케줄링 방법이 수행시간, GPU 메모리 활용도에서 다른 두 조건보다 우수하며 GPU 사용률에서도 차이 없이 성능을 낼 수 있음을 보여준다.

5. 결 론

본 논문에서는 응용 프로파일링을 기반으로 하여 작업의 자원 사용량에 따라 GPU 자원 공유가 가능한 기법을 제안하였다. 작업의 수행시간, GPU 활용도, GPU 메모리 사용량 정보를 통해 우선순위를 계산하고, 우선순위에 따라 작업의 실행 순서를 결정한다. 결과에 따라 작업의 실행 방식인 독점 및 공유를 결정한다. 이 방식을 적용하여 binpack, spread 스케줄링 방식과 비교하였다. HPC 응용으로 워크로드를 구성하여 실험을 통해 응용 프로파일링 정보를 통한 우선순위 실행 방식이 워크로드 전체 수행시간의 감소와 높은 GPU 메모리 사용량을 보여줌으로써 효율적임을 보였다.

Acknowledgement

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2015M3C4A7065646)

참고 문헌

[1] Piraghaj, Dastjerdi, Calheiros, and Rajkumar Buyya. 2015. A framework and algorithm for energy efficient container consolidation in cloud data centers. In DSDIS, 2015 IEEE International Conference on. IEEE, 368-375.

[2] Kubernetes https://github.com/fabric8io/kansible/blob/master/vendor/k8s.io/kubernetes/docs/design/scheduler_extender.md

[3] Yarn, https://hadoop.apache.org/docs/r3.1.0/hadoop-yarn/hadoop-yarn-site/UsingGpus.html

[4] 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.

[5] Kato, Shinpei, et al. "TimeGraph: GPU scheduling for real-time multi-tasking environments." Proc. USENIX ATC. 2011.

[6] NVIDIA GPU Cloud, https://ngc.nvidia.com/

[7] LAMMPS, https://lammps.sandia.gov/

[8] BigDFT, http://bigdft.org/Wiki/index.php?title=BigDFT_website

[9] GROMACS, http://www.gromacs.org/

[10] QMCPACK, https://www.qmcpack.org/

[11] CNN, https://en.wikipedia.org/wiki/Convolutional_neural_network

[12] Tensorflow, https://www.tensorflow.org/