멀티 프로그램화된 컨테이너 기반의 HPC 워크로드 성능에 대한 사전 연구

유정록, 윤희준
한국과학기술정보연구원 국가슈퍼컴퓨팅본부
{junglok.yu, k2}@kisti.re.kr

A Preliminary Study on the Performance of Multi-programmed Container-based HPC Workloads

Jung-Lok Yu, Hee-Jun Yoon
National Supercomputing Center, Korea Institute of Science and Technology Information (KISTI)

요 약

최근, 응용 프로그램의 이식성, 확장성, 낮은 오버헤드 및 관리의 용이성 등을 제공하는 컨테이너 기술을 고성능 컴퓨팅 (high performance computing, HPC) 환경에 접목하려는 다양한 연구들이 진행되고 있다. 본 논문에서는 멀티 프로그램화된 환경, 즉, 컨테이너 기반의 다수개의 HPC 워크로드들이 동시에 실행되는 환경에서 멀티 프로그래밍 수준, 통신 패턴 및 비율에 따른 HPC 워크로드들의 성능 특성을 분석하고, HPC 워크로드가 실행되는 동일한 컨테이너 그룹에 속한 컨테이너들의 스케쥴링 시간 부조화가 데이터 교환 지연 시간을 증가시키고 그 결과 응용 성능을 크게 저하시킬 수 있음을 확인한다. 또한 HPC 워크로드가 수행되는 동일 그룹 컨테이너들의 CPU 점유 가능값(CPU Shares)을 동적으로 조절하는 휴리스틱을 제안, 적용함으로써, HPC 워크로드의 성능(통신소비시간 최대 약 42.5%, 워크로드 실행시간 최대 약 23.6% 감소)을 크게 향상시킬 수 있음을 확인한다.

1. 서론

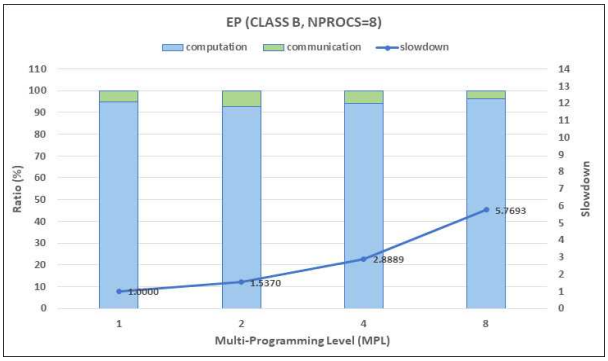
고성능 계산서버들을 고대역폭/저지연시간 네트워크로 연결한 고성능 컴퓨팅 (high performance computing, HPC) 시스템은 계산과학분야 연구자들에게 거대 계산문제를 해결할 수 있는 필수불가결한 환경이 되었다[1]. 최근에는 하드웨어 가상화 기반의 하이퍼바이저[2](예, KVM, Xen)를 사용하는 가상머신 기술과 대비해서 응용 프로그램의 이식성, 확장성, 낮은 오버헤드 및 관리의 용이성 등의 장점을 제공하는 커널 가상화 기반의 컨테이너 [3]기술을 이러한 HPC 환경에 접목하려는 다양한 연구[4-6]들이 진행되고 있다.

일반적으로 HPC 시스템의 효율적인 자원 활용을 위해, 각 계산서버는 계산자원(CPU, 메모리, I/O)을 공유하여 수행되는 수십~수백개의 컨테이너들을 구동하며, 독립적인 HPC 워크로드들은 물리적인 계산서버들에 분산, 배치되어 구동되는 다수개의 컨테이너 그룹들에 할당되어 처리된다. 본 논문에서는 이러한 멀티 프로그램화된 환경 - 즉, 컨테이너 기반의 다수개의 HPC 워크로드들이 동시에 실행되는 환경 - 에서 멀티 프로그래밍 수준, 통신 패턴 및 비율에 따른 HPC 워크로드들의 성능 특성을 분석하였다. 특히, HPC 워크로드가 실행되는 동일한 컨테이너 그룹에 속한 컨테이너들의 스케쥴링 시간 부조화(scheduling time mismatch)가 워크로드 데이터 교환/동기화 지연 시간을 증가시키고, 그 결과 응용 성능이 저하됨을 확인하였다. 아울러, 이러한 스케쥴링 시간 부조화를 완화시키기 위해 HPC 워크로드가 수행되는 동일 그룹 컨테이너들의 CPU 점유 가능 값(CPU Shares)을 동적으로 조절, 상승(boosting)시키는 휴리스틱를 제안, 적용한 결과, HPC 워크로드가 소비하는 불필요한 통신지연시간을 감소시키고 워크로드 실행 시간 역시 크게 감소됨을 확인하였다.

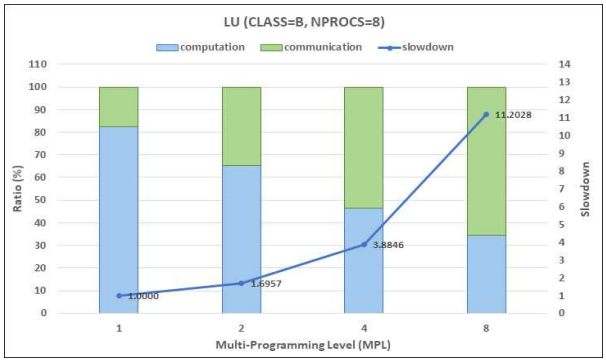
2. 컨테이너 기반 HPC 워크로드 성능 분석

멀티 프로그래밍 수준 (multi-programming level, MPL)에 따른 컨테이너 기반의 HPC 워크로드 성능 분석을 위해 Intel Core i7-6950X 10코어 CPU,

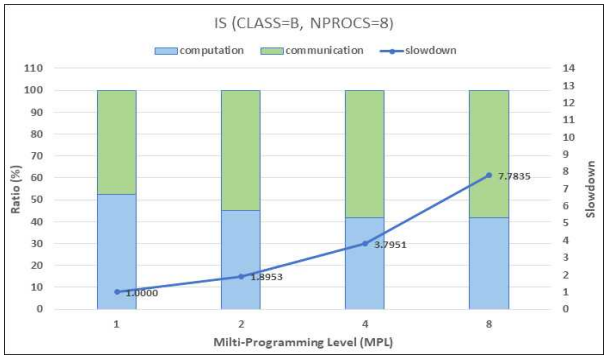
32GB 메모리를 가진 단일 계산서버를 사용하였고, OS는 리눅스 커널 버전 3.1.0, 워크로드는 NAS 병렬 벤치마크[7] 커널들 (EP, LU, IS)을 사용하였다. 컨테이너 엔진 및 오버레이 네트워크 구성을 위해 도커와 도커 스웜[4] 모드를 사용하였으며, 도커 기반의 메시지 교환 인터페이스 (message passing interface, MPI) 클러스터 구성을 위해 [6]에서 제공하는 오픈소스를 사용하였다.



(a) EP



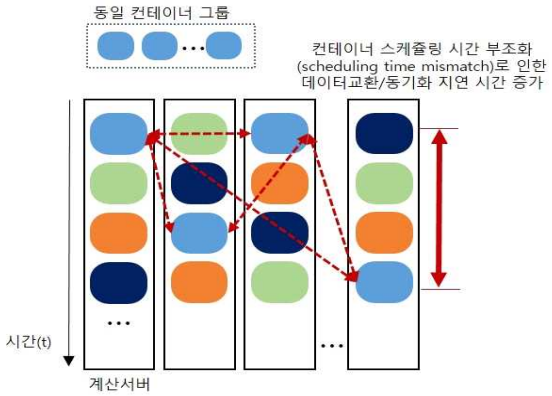
(b) LU



(c) IS

(그림 1) MPL 값 (1~8)에 따른 NAS 병렬 벤치마크 커널에서 소요되는 계산/통신 비율 및 Slowdown.

그림 1은 EP, LU, IS 병렬 벤치마크 커널에 대해 MPL을 1부터 8까지 증가시켰을 때, 각 커널에서 소요되는 평균 계산시간, 평균 통신시간 비율과 MPL=1 대비 slowdown을 보여주는 그래프이다.[1] EP와 같이 통신 비율이 낮은 (약 5%) 워크로드 경우, MPL을 증가시켜도 통신시간의 비율 변화가 거의 없고, slowdown도 약 5.7을 보임을 알 수 있다. 그러나, LU, IS와 같이 통신 비율이 높은 경우, MPL 값을 증가시킴에 따라 워크로드 실행 시 소요되는 데이터교환/동기화 지연 시간이 매우 커짐을 알 수 있으며 (MPL=8일때, LU 통신비율 약 65%, IS 통신비율 약 60%), 이는 결국 slowdown을 높이며 급격한 성능 저하의 원인이 됨을 알 수 있다.



(그림 2) 컨테이너 스케쥴링 시간 부조화로 인한 통신 지연시간 증가.

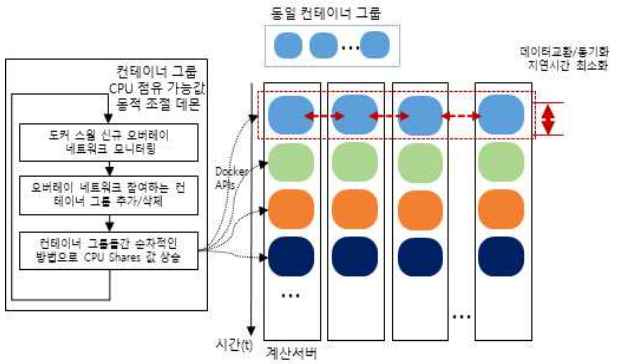
3. 제안 휴리스틱 및 사전 실험 결과

2장 결과에서 보듯이, 멀티 프로그래밍 수준이 높을 때, 컨테이너 기반 HPC 워크로드들이 소요하는 평균 통신지연/동기화 시간은 급격히 늘어나게 된다. 이에 대한 근본적인 원인은 그림 2에서 보듯이 동일한 컨테이너 그룹에 속한 컨테이너들의 스케쥴링 시간 부조화(scheduling time mismatch) 때문이다. 이러한 부조화는 각 계산서버의 부하 불균형 및 개별 계산서버에서 동작하는 리눅스 커널이 컨테이너 통신 특성을 인지하지 못하고 CFS 스케쥴링 정책[8]에 따라 컨테이너(즉, 독립적인 이름공간을 갖는 프로세스 집합)를 자율적으로 스케쥴링하기 때문이다.

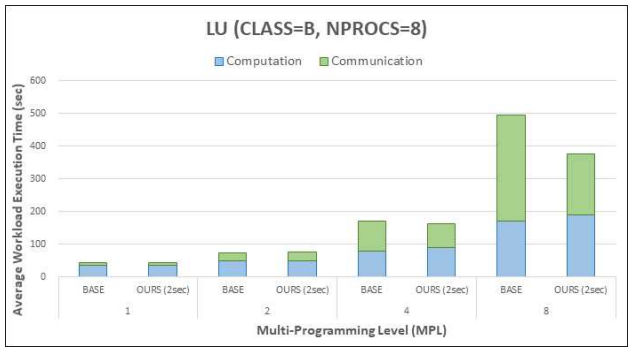
이러한 문제점을 완화시키기 위해 본 논문에서는 사전 연구로 HPC 워크로드가 할당되어 수행되는 동일 그룹 컨테이너들의 CPU 점유 가능값(CPU Shares)을 사용자 수준 (user-level)에서 순차적으로 일시 상승(boosting)시키는 휴리스틱을 설계, 구현하였다. 이 휴리스틱의 기본 아이디어는 그림 3에서 보듯이 각 계산노드의 CPU 계산자원을 대상으로 경쟁하는 다수개의 컨테이너 그룹들을 대상으로 해당

1) 단일 계산서버에서의 메모리 쓰레싱 (memory thrashing) 효과를 제거하고 모든 워크로드의 메모리 풋프린트가 물리 메모리에 적재 가능하도록 CLASS B를 사용하였고, MPL을 최대 8로 고정하였다.

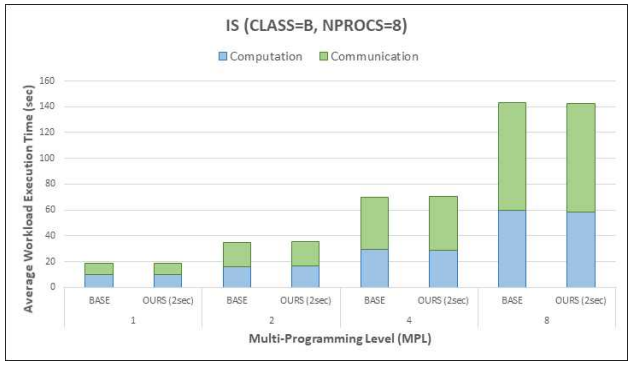
그룹이 점유할 수 있는 CPU 점유 가능값을 특정 기간 (예, 2초) 동안 주기적으로 높임으로써 전체 계산서버들에서 HPC 워크로드를 수행하는 동일 그룹 컨테이너들이 거의 같은 시간에 스케쥴링 되도록 유도하자는 것이다.



(그림 3) 컨테이너 그룹 CPU 점유 가능값 상승 휴리스틱.



(a) LU (Nearest Neighbor)



(b) IS (All-to-All)

(그림 4) LU, IS 벤치마크 커널에 대한 평균 워크로드 실행 시간 비교.

그림 4는 LU, IS 벤치마크 커널에 대해 제안한 휴리스틱을 적용했을 때, 평균 워크로드 실행 시간을 비교한 그래프이다. 컨테이너 그룹들에 대해 순차적인 방법으로 CPU Shares 값을 조정 (period : 2초)함으로써, 컨테이너 기반 HPC 워크로드에서 소요되는 데이터 교환/동기화 지연시간을 큰 폭으로 감소시킬 수 있고, 이는 결국 전체 평균 실행 시간의 감소(최대 약 23.6%)를 가져옴을 확인하였다. 이러한 경향은 멀티 프로그래밍 수준이 커질수록, Nearest Neighbor 통신 패턴을 사용하는 워크로드일수록 더욱 두드러졌다.

4. 결론 및 향후계획

본 논문에서는 멀티 프로그램화된 컨테이너 기반의 HPC 워크로드를 처리할 때, 컨테이너들간의 스케쥴링 시간 부조화가 시스템 전체 성능을 크게 저하시킬 수 있음을 확인하고, 이를 완화시킬 수 있는 휴리스틱을 제안, 그 유효성을 실험을 통해 사전 검증하였다. 향후에는 단일 계산서버가 아닌 다중 계산서버로 구성된 큰 규모 시스템에서의 성능 분석과 컨테이너 기반의 최적화된 HPC 시스템을 위한 더욱 정교한 자원 관리 기법을 연구할 계획이다.

Acknowledgement

본 연구는 한국과학기술정보연구원 주요사업(대용량 데이터센터 구축 및 운영, K-20-L02-C04) 및 수탁사업(기초연구실험데이터글로벌허브구축, NRF-2010-0018156)의 지원을 받아 수행되었다.

참고문헌

[1] Buyya, Rajkumar. "High performance cluster computing: Architectures and systems." Prentice Hall, Upper SaddleRiver, NJ, USA 1.999, 1999.

[2] Soriga, Stefan Gabriel, and Mihai Barbulescu. "A comparison of the performance and scalability of Xen and KVM hypervisors." 2013 RoEduNet International Conference 12th Edition: Networking in Education and Research. IEEE, 2013.

[3] Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." Linux journal 2014.239, 2014.

[4] Gerhardt, Lisa, et al. "Shifter: Containers for hpc." Journal of Physics: Conference Series. Vol. 898. No. 8. 2017.

[5] Le, Emily, and David Paz. "Performance analysis of applications using singularity container on sdsc comet." Proceedings of the Practice and Experience in Advanced Research Computing 2017

on Sustainability, Success and Impact. 2017. pp 1-4.
[6] Nguyen, Nikyle, and Doina Bein. "Distributed mpi cluster with docker swarm mode." 2017 ieee 7th annual computing and communication workshop and conference (ccwc). IEEE, 2017. pp 1-7.
[7] Bailey, David H., et al. "The NAS parallel benchmarks." The International Journal of Supercomputing Applications 5.3, 1991. pp 63-73.
[8] Wong, C. S., et al. "Fairness and interactive performance of O (1) and CFS Linux kernel schedulers." 2008 International Symposium on Information Technology. Vol. 4. IEEE, 2008.