Singularity 컨테이너 런타임 환경에서 cgroups 지정에 따른 HPC 작업의 성능 분석

송충건[1], 길준민[2], 임종범[3]
[1]고려대학교 컴퓨터학과
[2]대구가톨릭대학교 컴퓨터소프트웨어학부
[3]평택대학교 ICT융합학부 스마트콘텐츠전공
security0730@korea.ac.kr, jmgil@cu.ac.kr, jblim@ptu.ac.kr

A Performance Analysis on HPC Task Using cgroups in Singularity Container Runtime Environment

ChungGeon Song[1], JoonMin Gil[2], JongBeom Lim[3]
[1]Dept. of Computer Science and Engineering, Korea University
[2]School of Computer Software Engineering, Daegu Catholic University
[3]Smart Contents Major, Division of ICT Convergence, Pyeongtaek University

요 약

컨테이너 런타임 환경에서 HPC 작업을 실행하는 수요가 증가하고 있으며, 이에 따라 컨테이너 자원 관리의 중요성이 높아지고 있다. 본 연구에서는 HPC 작업 실행에 최적화된 컨테이너 런타임 환경인 Singularity를 대상으로 cgroups 지정 여부에 따른 실행시간을 측정하는 실험을 진행하고 결과를 분석하였다. 이러한 결과는 Singularity 컨테이너 런타임 환경에서 다양한 HPC 작업을 운영할 시 성능 효율을 높일 수 있는 자원 관리 방향을 제시한다.

1. 서론

컨테이너 기술은 KVM, Xen, ESXi와 같은 하이퍼바이저 기반 가상화 기술처럼 격리된 실행 환경을 제공하는 기술이다. 그러나 컨테이너 기술은 하이퍼바이저 기반 가상화 기술과 다르게 OS 커널을 공유하고 어플리케이션 실행에 필요한 최소한의 자원만 격리하여 경량화된 실행 환경을 제공한다[1]. 이러한 컨테이너 기술은 포춘 100대 기업의 99%가 대규모 웹서비스 또는 사내 워크스테이션의 실행 환경을 관리 목적으로 활용할 만큼 가치를 인정받았다[2].

컨테이너 런타임은 이러한 컨테이너 기술을 실현하기 위해 필요한 도구와 컨테이너 운영 환경을 제공하는 플랫폼이다. 최초의 컨테이너 런타임 기술인 도커의 등장 이후에 다양한 목적을 가진 컨테이너 런타임 기술이 등장하고 있다. Singularity는 과학적 연산이 가지는 HPC 작업을 수행하는 목적으로 개발된 컨테이너 런타임 기술이다[3].

초창기 컨테이너 자원 관리 기술은 경량화된 실행 환경을 통해 대규모 서비스를 안정적으로 운영하기 위한 목적으로 발전해왔다[4]. 이에 따라 성능이 저하되는 자원 경쟁을 막기 위해서 시스템의 자원 활용률을 임계치 이하로 유지하는 로드밸런싱 기반 컨테이너 배포 알고리즘이 활용되었다[5]. 그러나 서버리스 컴퓨팅과 같은 작업 단위 배포 환경에서 시스템 활용률을 높이기 위해서 시스템 자원 활용률을 극대화하고 자원 경쟁을 최소화하는 방향으로 자원 관리를 수행하는 기술이 요구된다. 또한 컨테이너 런타임 환경에서 인공지능 연산과 과학적 연산과 같이 HPC 작업을 수행하는 수요가 증가하고 있다. 그러나 기존 연구에서는 시스템 자원 활용률이 극대화된 상황에서 발생하는 자원 경쟁과 이를 최소화하기 자원 관리 방향에 대한 연구가 부재하다.

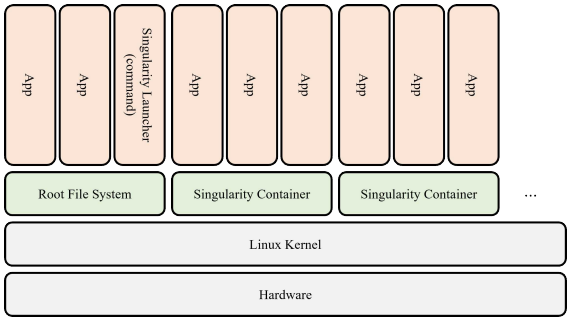
본 연구에서는 cgroups 설정이 HPC 작업의 성능에 미치는 영향과 그 원인에 대한 분석을 수행하였다. 이에 통해 Singularity 컨테이너 런타임 환경에서 HPC 작업을 운영할 시 성능 효율을 높일 수 있는 자원 관리 방향을 제시한다.

본 논문의 구성은 다음과 같다. 2절의 관련 연구에서는 본 연구의 대상의 되는 기술인 Singularity 컨테이너 런타임과 cgroups 기술을 소개한다. 3절에서는 Singularity 컨테이너를 대상으로 cgroups의 적용 유무에 따라 도출되는 HPC 작업의 성능을 설명하며, 4절에서 연구의 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

A. Singularity 컨테이너 런타임

Singularity는 HPC 작업을 수행하는 복잡한 형태의 애플리케이션을 실행하기 위하여 단순하면서 포터블한 실행 환경을 제공하는 것을 목표로 하는 컨테이너 런타임 기술이다. 초기 Lawrence Berkely 연구소에서 개발되었으며, 오픈소스 프로젝트로 발전되고 있다[6]. 컨테이너 런테임 관리하기 위한 데몬이 없으며, 각 컨테이너가 프로세스와 같이 독립적으로 실행된다.



(그림 1) Singularity 구조

Singularity는 (그림 1)과 같은 구조를 가진다. Singularity 컨테이너는 하나의 프로세스 형태로 동작하고 컨테이너 사이에서 호스트의 서비스와 기능을 공유한다. 그리고 시스템 오버헤드 최소화를 위하여 최소의 namespace를 사용한다.

B. cgroups을 이용한 컨테이너 자원 관리

cgroups은 일반적인 리눅스 환경에서 프로세스를 대상으로 자원 격리 기능을 제공하는 기술이다. namespace가 컨테이너에 추상화된 환경을 제공하는 목적으로 활용되는 기술이라면, cgroups은 컨테이너의 자원 격리 기능을 제공하는 목적으로 활용된다.

```
[cpu]
0-3
[mem]
2G
```

(그림 2) cgroups 정책의 예

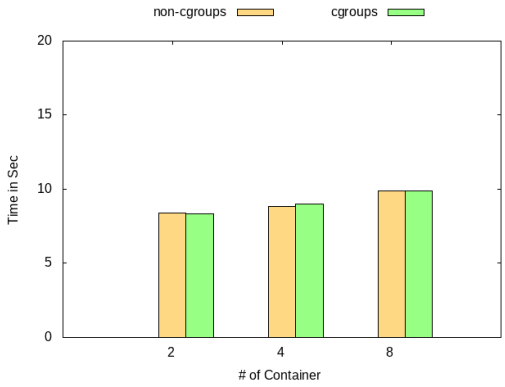
(그림 2)에서는 컨테이너 런타임 환경에서 활용하는 cgroups 정책의 예를 나타내고 있다. 컨테이너에 cgroups 정책을 적용과 자원 사용 현황에 대한 모니터링은 리눅스 커널이 수행되며, 사용자 영역과의 통신을 위해 /sys/fs/cgroup/ 이라는 특수 파일시스템을 제공한다.

3. cgroups 지정에 따른 HPC 작업의 성능 분석

이 절에서는 컨테이너 수에 따른 HPC 작업의 성능과 쓰,레드 수에 따른 HPC 작업의 성능 분석을 수행하고 결과에 대한 설명을 서술한다. 실험에서 사용한 시스템은 Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz를 가지고 있다. 메모리는 16GB이며, 스토리지 용량은 256GB SSD를 가지고 있다. 운영체제는 리눅스 4.15 커널을 가진 CentOS 7을 구축하여 진행하였다. Singularity는 3.8 버전을 활용하였다. 실험에서 HPC 작업은 sysbench를 통해 생성하였으며 1~10000까지의 소수 연산을 수행한다. 해당 작업은 5회 실행하고 실행 시간의 평균을 구하였다.

A. 컨테이너 수에 따른 HPC 작업의 성능

첫 번째 실험에서는 다양한 컨테이너 수에 따라 cgroups 지정이 성능에 어떤 영향을 미치는지 확인하는 실험을 진행하였다. 컨테이너 수는 2, 4, 8개로 구성하고 cgroups을 적용한 경우와 cgroups을 적용하지 않은 경우를 구분하여 실행 시간을 비교하였다. cgroups 정책은 모든 컨테이너가 CPU 자원을 균등하게 나누어 경쟁이 발생하지 않는 방향으로 설정하였다.



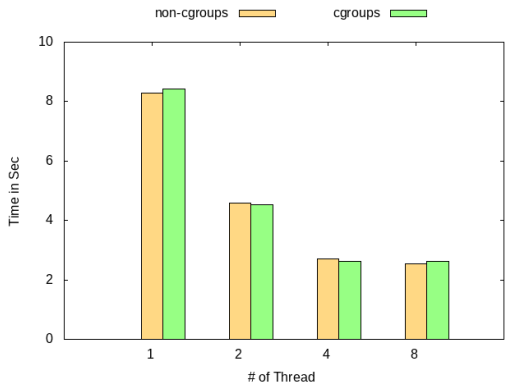
(그림 3) 컨테이너 수에 따른 성능

(그림 3)에서는 첫 번째 실험의 결과를 그래프로 나타내고 있다. 컨테이너 수를 다양하게 구성하여 실행하여도 cgroups 적용이 실행 시간에 영향을 주지 않고 non-cgroups 상황과 동일한 실행 시간을 확인할 수 있다. 이러한 결과의 원인을 분석하기 위

해 두 번째 실험에서 진행하였다.

B. 쓰레드 수에 따른 HPC 작업의 성능

두 번째 실험에서는 다양한 쓰레드 수에 따른 HPC 작업의 실행 시간을 확인하는 실험을 진행하였다. 실험 1과 동일한 작업 크기를 가지는 컨테이너 2개를 구성하고 쓰레드 수를 1, 2, 4, 8로 다양하게 설명한 후 작업을 실행하고 실행시간을 측정하였다. 컨테이너는 2개 실행하였으며, CPU 자원에 대한 cgroups 정책은 각각 “0-3”, “4-7”로 지정하였다.



(그림 4) 쓰레드 수에 따른 성능

(그림 4)에서는 두 번째 실험의 결과를 나타내고 있다. 쓰레드 수가 늘어날수록 시스템 성능이 비례하여 늘어나는 것을 확인할 수 있으며, cgroups을 지정한 환경에서 쓰레드 수가 cgroups 정책의 코어의 수와 동일한 경우 성능이 최대로 개선되는 결과를 얻었다. 이를 통해 첫 번째 실험 결과에 대한 주요 원인으로 cgroups을 변경하여도 벤치마크의 작업이 고정된 쓰레드 수로 실행되기 때문에 병렬화를 효율적으로 수행하지 못한다는 사실을 확인하였다.

4. 결론

본 연구에서는 Singularity 컨테이너 런타임 환경에서 HPC 작업 수행 시 성능 개선을 얻을 수 있는 자원 관리 방향을 제시하였다. 구체적으로 HPC 작업은 호스트 머신의 CPU 정보를 동적으로 읽어 쓰레드 수를 조절하는 프레임워크에서 실행해야 cgroups을 통한 성능 향상을 얻을 수 있다. 이러한 연구 결과는 다양한 응용 분야에서 활용되어 HPC 작업 연산 효율을 얻을 수 있을 것으로 기대된다.

이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단 기초연구사업의 지원을 받아 수행된 연구임(NRF-2021R1F1A1063307)

참고문헌

[1] Soltesz, Stephen, et al. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors." Proceedings of the 2Nd ACM SIGOPS/EuroSys european conference on computer systems 2007. 2007.

[2] Application Container Market by Service (Container Monitoring, Security, Data Management, Networking, Orchestration), Platform (Docker, Kubernetes), Application Area, Deployment Mode, Organization Size, Vertical, and Region - Global Forecast to 2023, REPORT CODE TC 6274, 2018.

[3] Kurtzer, Gregory M., Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific containers for mobility of compute." PloS one 12.5, 2017, e0177459.

[4] Rodriguez, Maria A., and Rajkumar Buyya. "Container based cluster orchestration systems: A taxonomy and future directions." Software: Practice and Experience 49.5, 2019, 698-719.

[5] Luksa, Marko. Kubernetes in action. Simon and Schuster, 2017.

[6] Singularity Github, https://github.com/apptainer/singularity.