Heterogeneous 코어 환경에서 Singularity 컨테이너의 Cgroup 정책에 따른 HPC 작업의 성능 분석

김명준 [1], 송충건 [1], 정광식 [2], 유헌창 [1]
[1] 고려대학교 대학원 컴퓨터학과,
[2] 방송통신대학교 컴퓨터과학과

{mjkim1130, security0730, yuhc}@korea.ac.kr, kchung0825@knou.ac.kr

A Performance Analysis of HPC Task According to Cgroup Policies of Singularity Container in Heterogeneous Cores Environment

Myeong-Jun Kim[1], Chung-Geon Song[1], Kwang-Sik Chung[2], Heon-Chang Yu [1]
[1]Dept. of Computer Science and Engineering, Korea University
[2]Dept. of Computer Science, Korea National Open University

요 약

최근의 인텔의 새로운 CPU 아키텍처의 도입으로 Singularity 컨테이너 내에서 cgroup 설정으로 인해 특정 작업의 성능에 영향을 초래할 수 있다. 특히 Singularity 컨테이너에서 HPC(고성능 컴퓨팅) 작업은 cgroup 정책에 의해 작업 효율이 달라질 수 있고 아직 새로운 CPU 환경에서의 HPC 작업에 대한 연구가 충분치 않다. 따라서, 본 논문은 Singularity 컨테이너 생성 시 새로운 CPU 아키텍처에 설계된 CPU 코어 유형별 cgroup 지정이 HPC 응용을 포함하여 다양한 유형의 작업 성능에 미치는 영향을 비교 분석하였고, 이를 통해 HPC 사용자에게 가이드라인을 제공하는 것이 목적이다.

1. 서론

컨테이너는 전통적인 가상화 방식에 비해 경량화되어 있고, 빠른 시작 시간과 호스트 OS 와의 밀접한 연동성을 제공한다. 이러한 이점들은 HPC 사용자들에게 복잡한 애플리케이션과 애플리케이션의 필요한 라이브러리나 환경설정 같은 종속 요소들을 신속하게 배포하고 실행하는 능력을 제공한다[1,2]. 특히 Singularity 컨테이너는 HPC 환경을 위해 특정되어 개발되었다[3]. Singularity 는 다양한 환경에서 일관된 실행을 보장하는 동시에, 기존 HPC 인프라와의 호환성에 중점을 둔 디자인이 특징이다[4].

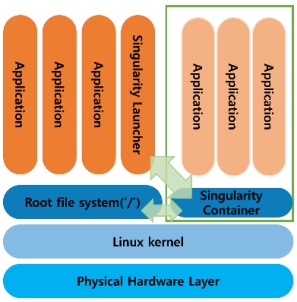
리눅스 운영 체제에서는 cgroup(control group)을 통해 특정 컨테이너가 사용하는 리소스를 제한하거나 보장한다. 그러나, 인텔의 새로운 CPU 아키텍처가 도입되면서 기존 한 가지 유형의 코어에서 두 가지 유형의 코어가 CPU 내 설계되는 이기종(Heterogeneous) 코어가 등장하였다. 이로 인해 임의로 CPU 리소스를 분배할 시, 작업 특성과 부합하지 않게 리소스가 할당될 수 있어 작업 처리 능력에 영향을 미칠 수 있다.

본 논문은 이기종 코어인 Raptor Lake 을 사용하는 시스템에서 Singularity 컨테이너 생성 시 cgroup 정책의 영향을 분석하였으며, 이를 통해 각 정책별 성능 차이와 특히 HPC 워크로드에 대한 영향을 규명하였다. 이러한 분석을 통해, 동일한 환경에서 작업하는 사용자들에게 유용한 가이드라인을 제공할 수 있을 것으로 기대된다.

2. 관련 연구

2.1 Singularity 컨테이너 런타임

Singularity 는 간단하고 이식 가능하며 재현 가능한 방식으로 HPC 환경에서 복잡한 애플리케이션을 실행하기 위해 만들어졌다[5].



(그림 1) Singularity Architecture

(그림 1)은 Singularity 아키텍처를 설명하고 있다. Singularity가 주로 사용되는 이유는 기존 Docker와 같은 컨테이너 기술과 달리, Singularity는 컨테이너 내에서 루트 권한을 갖지 않기 때문에 보안상 문제가 줄어든다. 또한 Singularity 이미지는 단일 파일로서, 다른 시스템 간에 쉽게 전송하고 복제할 수 있어 휴대성 면에서 다른 경량 런타임보다 우수하다[6].

2.2 Raptor Lake

기존 인텔의 데스크탑용 CPU와 달리 12세대 Alder Lake는 p-코어(performance 코어)와 e-코어(efficient 코어)를 포함하는 하이브리드 아키텍처 기반으로 설계되었다. 하이브리드 아키텍처는 계층적 CPU 토폴로지 기반으로 설계되었고 이는 두 종류 이상의 서로 다른 프로세서나 코어를 사용하는 시스템에 적용된다, 이와 같은 아키텍처는 일반적으로 모바일 SoC 에서 효율적인 전력을 지원하기 위해 사용되었다.

인텔에서 Lakefield 로 처음으로 모바일 프로세서에 이 아키텍처를 도입하였으며, 이는 1 개의 'Sunny Cove'라 불리는 p-코어와 4 개의 'Tremont' e-코어로 구성되었다. 이러한 아키텍처 전략이 데스크톱 CPU 로 확장되면서, 12 세대인 Alder Lake가 등장하게 되었다. Alder Lake는 'Golden Cove' p-코어와 'Gracemont' e-코어로 구성되어 있고, 본 논문은 이를 계승한 13 세대 Raptor Lake 으로 연구를 진행하였다. Raptor Lake 는 'Raptor Cove' p-코어와 기존 'Gracemont' e-코어로 설계되었다. 실험에 사용된 CPU 코어의 세부 사양은 실험에 활용된 Core i9 13000k 를 기준으로 설명한다.

<표 1> Raptor Lake p-코어 e-코어 세부 사양

	p-코어	e-코어
maximum turbo frequency	최대 5.4GHz	최대 4.3GHz
base frequency	3.0 GHz	2.2 GHz
number of threads	2	1

<표 1>은 p-코어와 e-코어의 주파수 차이 및 가지고 있는 스레드 수를 보여주고 있다[7]. 주파수는 CPU 코어가 수행할 수 있는 연산 속도나 빈도를 나타낸다.

2.3 Cgroup

Cgroup 은 리눅스 운영 체제에서 프로세스 그룹을 생성하고 관리하는 기술이다. Cgroup 을 통해 컨테이너 생성 시 컨테이너가 사용할 리소스에 대한 정책을 정의하고 적용하여 시스템 리소스를 효율적으로 활용하고 프로세스 간 간섭을 제어할 수 있다.

3. 시스템 환경 및 실험 결과

3.1 실험 환경 및 방법

본 절에서는 다양한 cgroup 정책으로 생성된 Singularity 컨테이너에서 싱글 코어 및 멀티 코어 환경에서 벤치마크를 통해 특정 작업에서의 성능을 평가한다. 분석에 사용한 시스템은 앞 절에서 말한 Core i9 13000k(p-코어 8 개, e-코어 16 개), 64GB 메모리, 그리고 1TB SSD 를 스토리지를 사용한다. 운영 체제로는 Ubuntu 20.0.4.6 LTS 를 사용하며, Singularity 3.8.5 를 통해 컨테이너를 생성한다. P-코어만으로 이루어진 cgroup 은 Cp, e-코어만으로 이루어진 cgroup 은 Ce 라 정의한다 또한 HPC 에서 추가적으로 실험하기 위해 8 개의 p-코어와 8 개의 e-코어를 가진 그룹을 Cm 이라고 정의한다.

Cp, Ce 의 싱글코어 및 멀티코어 작업 처리 능력을 확인하기 위해 Geekbench 5.4.0 을 활용하여 실험을 진행하였다. 추가적으로 NPB 3.4.2 벤치마크를 활용해 실제 HPC 워크로드에서 cgroup 간 성능 차이를 확인하였다[8].

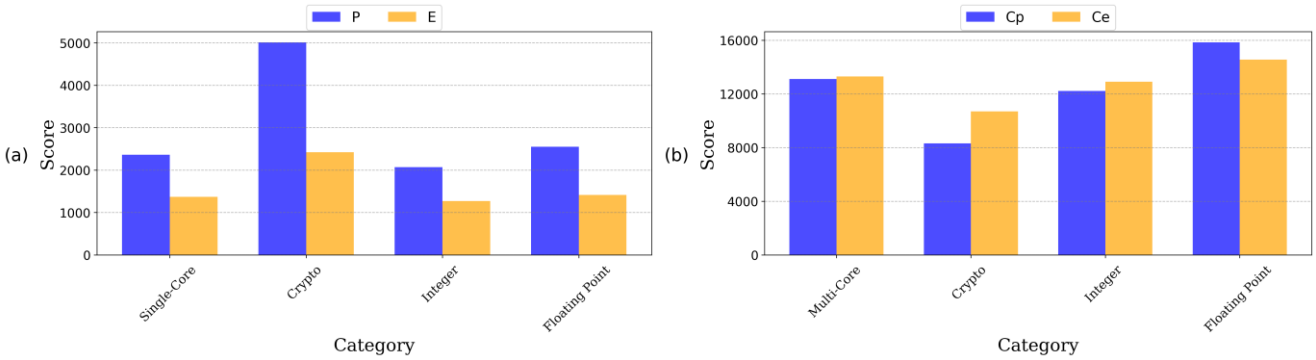
3.2 싱글코어 및 멀티코어 환경에서의 cgroup 별 성능 차이

첫 번째 실험에서는 Geekbench 가 제공하는 워크로드들을 수행하고, 각 작업의 성능 점수(초당 처리량)을 측정하였다. Geekbench 5.4.0 는 총 3 가지 연산 범주의 워크로드들을 제공한다. 암호화 워크로드 1 가지, 정수 워크로드 9 가지, 부동 소수점 워크로드 11 가지를 통해 cgroup 들의 성능을 확인하였다[9]. <표 2>는 각 범주별 워크로드 종류를 보여주고 있다.

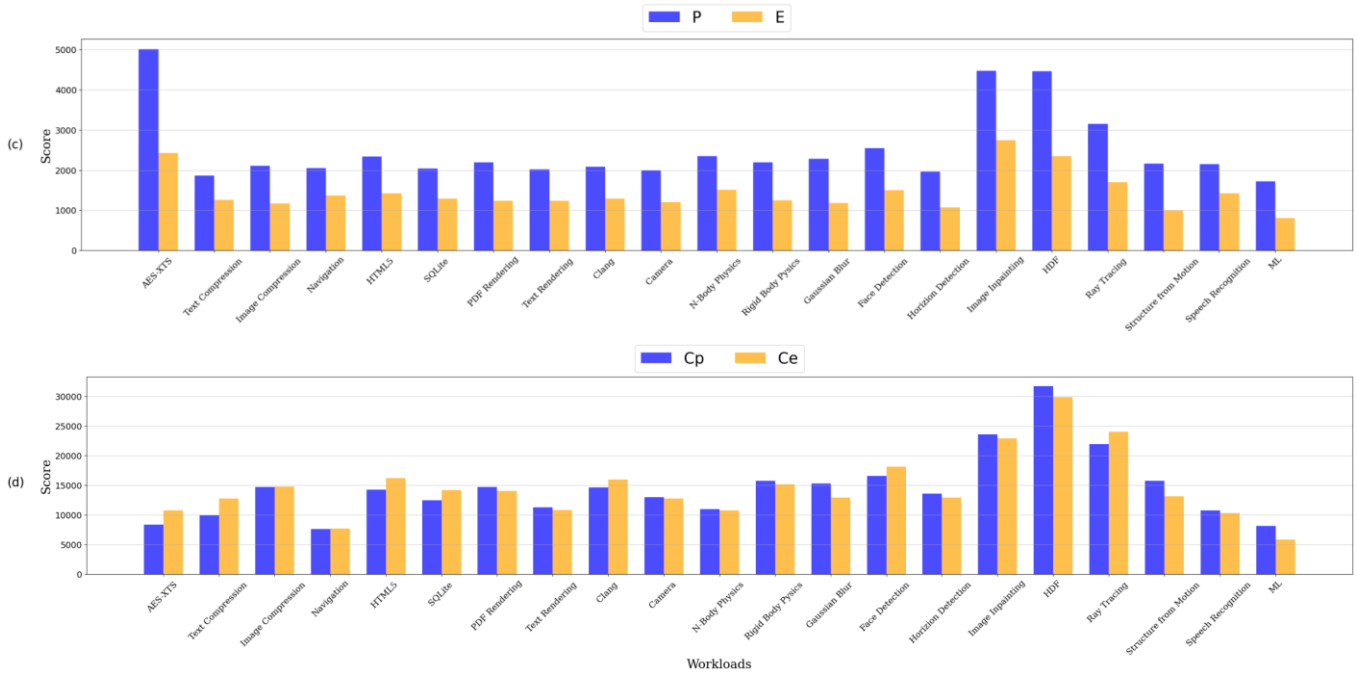
성능 점수는 Core i3-8100 의 점수인 1,000 을 기준으로 보정되어서 제공된다. 각 워크로드별로 개별 점수를 산출한 후, 기하 평균을 계산해 각 범주의 점수로 환산되고 이 점수들은 <표 2>에서 보여주는 가중치를 통해 가중 산술 평균 계산을 통해 종합 점수로 환산해서 제공된다. 실험은 총 5번 진행되고 각 cgroup 간 종합 점수 및 범주별 점수, 그리고 개별 워크로드별 점수를 구하였다.

<표 2> Workload 종류 및 가중치

범주	워크로드	가중치
암호	AEX-XTS	5%
정수	Text Compression, Navigation, HTML5, SQLite, PDF Rendering, Text Rendering, Clang, Camera	65%
부동 소수점	N-Body Physics, Rigid Body Physics, Gaussian Blur, Face Detection, Horizon Detection, Image Inpainting, HDR, Ray Tracing, Structure from Motion, Speech Recognition, Machine Learning	30%



(그림 2) 종합 점수 및 범주별 점수 (a) 싱글코어 (b) 멀티코어



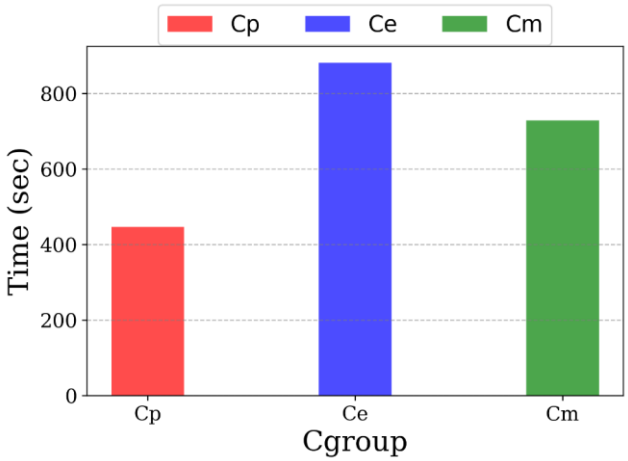
(그림 3) 워크로드 개별 점수 (c) 싱글코어 (d) 멀티코어

(그림 2)는 싱글코어 및 멀티코어에서의 종합 점수와 범주별 평균 점수를 보여주고 있다. 그림 2(a)을 통해 싱글코어 환경에서는 p-코어가 모든 범주에서 e-코어보다 우월한 성능을 보이지만, 그림 2(b)인 멀티코어 환경에서는 종합 점수, 암호화, 정수 범주에서 Ce 가 Cp 보다 평균적으로 더 우수 것을 확인할 수 있었다. 또한 정수 범주에 대해 모든 결과에서 Ce 가 Cp 보다 더 높은 점수를 기록하였다. 이는 2 개의 스레드를 가진 p-코어에서 컨텍스트 전환 및 스레드 관리 오버헤드가 증가해 멀티코어 환경에서 정수 워크로드에 Cp 를 지정하는 것은 e-코어의 집단인 Ce 에 비해 비효과적인 것을 확인하였다.

(그림 3)은 싱글코어 및 멀티코어 환경에서 워크로드 개별 적인 점수를 나타낸다. 그림 3(d)를 통해 Ce 가 AEX-XTS, Text Compression, Image Compression, Navigation, HTML5, SQLite, Clang, Face Detection, Ray Tracing 워크로드에 대해 Ce 가 평균적으로 Cp 보다 높은 점수를 기록했으며, HTML5 및 SQLite 워크로드에 대해서는 모든 시도에서 앞서는 결과를 보여주었다. 그러나 Face Detection 및 Ray Tracing 워크로드를 제외한 나머지 부동 소수점 범주의 워크로드들에 대해서는 평균적으로 Cp 가 Ce 보다 우수한 성능을 기록하였다. 따라서 Ce 에게는 I/O 작업이 주로 이루어지면서 간단한 계산 워크로드를 할당하는 것이 적합하다는 것을 확인할 수 있었다. 이와 같은 결과가 특히 HPC 환경에서 어떻게 적용되는지 알아보기 위해 실제 HPC 워크로드를 활용해 두번째 실험을 진행하였다.

3.3 HPC benchmark 를 통한 cgroup 별 성능 분석

두 번째 실험은 NASA 에서 제공하는 HPC Parallel Benchmark 에서 CG(Conjugate Gradient)작업을 수행시켜 Cp, Ce, Cm 간 성능 차이를 확인하였다. CG 알고리즘은 대칭 행렬과 선형 방정식 시스템의 해를 찾기 위해 사용되는 방법이다. 실험 방식은 Class D 작업을 cgroup 별 5 번 수행하여 각각 평균을 구하였다.



(그림 4) cgroup 별 HPC 별 작업 수행 시간

(그림 4)은 Cp, Ce, Cm 에서 CG 작업에서 총 소요된 시간(초)의 평균값을 보여주고 있다. Cp 에서는 평균 446.47 초의 계산 시간이 소요되었고 Ce 에서는 두 배 가까운 880.47 초가 소요되었다. Cm 은 평균 728.47 초로 Ce 보다 빠르지만 Cp 보다 시간이 더 소요되는 결과를 보여주었다. 이러한 결과가 나타난 이유로는 동일한 워크로드가 각 코어에 할당되었을 때 e-코어의 계산 능력이 p-코어보다 떨어져 전체적인 작업 시간에 영향을 미쳤음을 알 수 있었다. 따라서 HPC 와 같이 고성능 계산 능력이 필요한 작업에 대해서는 p-코어로만 이루어진 cgroup 만 지정하는 것이 리소스 할당면에서도 바람직하다는 것을 알 수 있었다.

4. 결론

본 논문은 이기종 코어인 Raptor Lack 시스템에서 Singularity 컨테이너 생성 시 다른 cgroup 정책을 적용 받았을 때 다양한 분야, 특히 HPC 워크로드에 대한 성능 분석을 진행하였다. 실험 결과, 코어 개별적으로 성능이 확실히 차이나는 것을 확인하였고, 코어 유형별 cgroup 으로 나누어졌을 때 각 그룹에 효율적인 작업 특성을 확인할 수 있었다. 특히 HPC 와 같은 고사양의 계산 작업을 수행할 경우, 특정 코어로만 이루어진 cgroup 을 지정해주는 것이 리소스 할당면에서 그리고 작업 처리 능력면에서 바람직하다. 이러한 결과를 토대로 동일한 작업 환경을 가진 HPC 사용자들에게 유용한 정보를 제공할 수 있을 것으로 기대한다. 향후 Kubernetes 와 같은 컨테이너 오케스트레이션을 활용해 다양한 HPC 환경과 작업 유형에 맞게 Singularity 컨테이너의 성능을 최적화하기 위한 전략을 연구할 예정이다.

Acknowledgement

본 연구는 산림청(한국임업진흥원) 산림과학기술 연구개발사업 ´ 2022427C10-2224-0801 ´ 의 지원에 의하여 이루어진 것입니다.

이 논문은 2022 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (NO.2022-0-00983, 사용자 별 오토모티브 헬스케어 서비스를 지원하는 엣지 클라우드 기반 차량 공유 플랫폼 개발)

참고문헌

[1] Sochat, Vanessa, and Alec Scott. "Collaborative container modules with singularity registry hpc." Journal of Open Source Software 6.63 (2021): 3311.

[2] S. Abraham, A. K. Paul, R. I. S. Khan and A. R. Butt, "On the Use of Containers in High Performance Computing Environments," 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), Beijing, China, 2020, pp. 284-293.

[3] Keller Tesser, R., Borin, E. Containers in HPC: a survey. J Supercomput 79, 5759–5827 (2023).

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

[5] O. Rudyy, M. Garcia-Gasulla, F. Mantovani, A. Santiago, R. Sirvent and M. Vázquez, "Containers in HPC: A Scalability and Portability Study in Production Biological Simulations," 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil, 2019, pp. 567-577.

[6] Singularity User Guide, https://docs.sylabs.io/guides/latest/user-guide/introduction.html#why-use-singularityce

[7] Intel® Core™ i9-13900K Processor, https://www.intel.com/content/www/us/en/products/sku/230496/intel-core-i913900k-processor-36m-cache-up-to-5-80-ghz/specifications.html

[8] NPB, https://www.nas.nasa.gov/software/npb.html

[9] Geekbench5 CPU Workload, https://www.nas.nasa.gov/software/npb.html