컨테이너 플랫폼에서 애플리케이션 성능 영향 분석

우준, 홍태영
한국과학기술정보연구원
wjnadia@kisti.re.kr, tyhong@kisti.re.kr

Performance impact analysis of application on container platforms

Joon Woo, Taeyoung Hong
KISTI

요 약

최근 HPC에서 다양한 사용자 서비스 환경을 지원하기 위해 하드웨어 가상화보다 가벼운 OS 가상화를 제공하는 컨테이너 기술이 활용되고 있다. HPC 환경에서 호스트 운영체제와 격리된 컨테이너에서는 주요 서비스 분야 사용자 애플리케이션의 실행 시 베어메탈노드 대비 성능 오버헤드가 크지 않아야 원활한 서비스가 가능하다. 이에 따라, 본 연구를 통해 서비스 분야별로 많이 사용되는 애플리케이션을 선택하여 싱규레러티 및 도커 컨테이너와 베어메탈 노드에서 성능을 측정하고 오버헤드를 비교 분석했다

1. 서론

최근 HPC 서비스는 기존 Legacy HPC 및 HTC (High Throughput Computing) 이외에 AI, 빅데이터 분석 등으로 서비스 영역이 확장되고 있다[1]. 따라서 기존 HPC 시스템에서 지원이 어려운 특별한 사용자 서비스 환경에 대한 제공 요구가 더욱 늘어날 것으로 예상된다.

다양한 사용자 서비스 환경을 지원하기 위해 하드웨어 가상화보다 가벼운 운영체제(OS) 가상화를 제공하는 컨테이너 기술이 대안으로 등장했다. 가장 보편적인 도커(docker)와 HPC를 고려한 싱귤레러티(singularity) 및 쉬프터(shifter) 등을 포함한 여러 가지 컨테이너 플랫폼들이 존재한다[2][3]. HPC 환경에서 호스트 운영체제와 격리된 컨테이너에서는 주요 서비스 분야 사용자 애플리케이션의 실행 시 베어메탈노드 대비 성능 오버헤드가 크지 않아야 원활한 서비스가 가능하다.

본 연구에서는 서비스 분아별로 많이 사용되는 애플리케이션을 선택하여 싱규레러티 및 도커 컨테이너와 베어메탈 노드에서 성능을 측정하고 오버헤드를 비교 분석했다[4].

2. 주요 애플리케이션 시험 방안

주요 애플리케이션에서 컨테이너 플랫폼의 성능영향 분석을 위해 <표 1>과 같이 분야별로 많이 사용되는 애플리케이션 혹을 선택하여 성능 측정 및 분석 방안을 수립했다. 주요 서비스 분야별 애플리케이션을 기존 클러스터와 같이 호스트 OS만 설치된 베어메탈 노드와 도커 및 싱귤레러티 컨테이너에서 각각 실행하여 성능을 측정했다.

시험 테스트베드는 사용자 액세스와 배치 스케줄러 및 쿠버네티스의 서버 역할을 병행하는 로그인·마스터 노드와 인텔 제온 프로세서와 16GB의 메모리를 가진 4대의 계산 노드로 구성하였다. 각 노드들은 채널 당 20Gbps의 4x DDR 인피니밴드 네트워크와 1GbE 혹은 10GbE 네트워크에 연결되어 있다. 로그인 노드와 계산 노드에는 사용자 작업 디렉터리 및 컨테이너 이미지 데이터 저장소 역할을 하는 Lustre/GPFS 병렬파일시스템이 마운트 되어 있다.

<표 1> 서비스 분야별 주요 애플리케이션 시험 방안

구분	애플리케이션	비교 플랫폼	시험 항목
Legacy HPC	• 퀀텀 에스프레소(Quantum Espresso) - 전자 구조 계산 및 소재 모델링 - MPI 병렬 연산 - 노드 간 인터커넥션 네트워크 통한 MPI 타스크 간 통신 많음	• 도커 (컨테이너) • 싱귤레러티 (컨테이너) • 베어메탈 노드 (호스트 OS)	• CPU 코어수 증가에 따른 실행경과시간 비교 • 작업 수 증가에 따른 개별 작업의 실행 경과시간 비교
AI	• 텐서플로우(TensorFlow) - 오픈소스 기계학습 라이브러리 - CPU/GPU 기반 그래프 플로우 연산 - 멀티 노드 분산처리에서 노드 간 통신발생 - (시험 애플리케이션) simple softmax 모델 사용 필기체 숫자 이미지 학습		

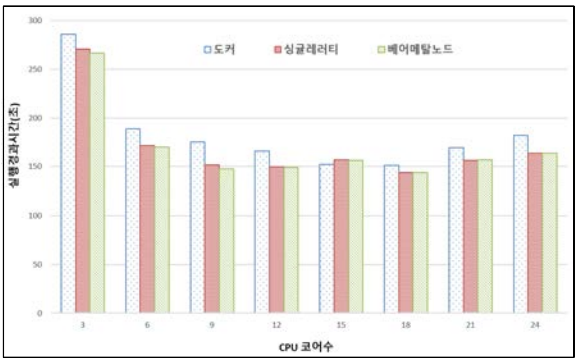
3. 주요 애플리케이션 시험 결과

Legacy HPC 분야에서는 3대의 계산 노드에서 퀀텀 에스프레소 MPI 타스크를 계산 노드에 균등하게 나누어 할당하는 방식의 병렬 연산에서 CPU 코어

수가 늘어남에 따른 실행 경과시간(초)을 도커, 싱귤레러티, 베어메탈노드에서 측정하였다.

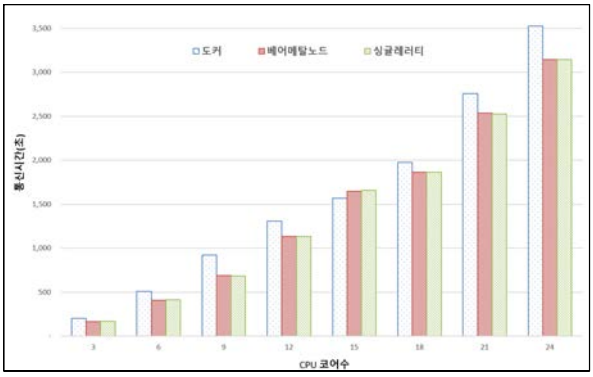
(그림 1)의 그래프에 따르면, 싱귤레러티는 베어메탈노드와 성능 차이가 없지만, 도커는 베어메탈 노드 대비 평균 약 8.9%, 최대 약 14% 성능이 낮게 나타났다.

노드 간 연결 네트워크에서 싱귤레러티는 20Gbps 대역폭을 제공하는 인피니밴드 네트워크에서 RDMA(Remote Ditect Memory Access) 프로토콜을 사용하고 있으나, 도커는 인피니밴드 네트워크에서 TCP/IP 네트워크를 지원하기 위한 IPoIB(IP over IB)와 네트워크 가상화를 지원하는 캘리코(calico) 네트워크로 구성되어 있다[5].



(그림 1) CPU 코어수 증가에 따른 실행경과시간 비교

(그림 2)는 TAU를 사용한 프로파일링 결과를 기반으로 산출된 MPI 통신 시간을 보여주고 있으며[6], 싱귤레러티는 베어메탈노드와 유사하지만, 도커는 베어메탈노드 대비 MPI 통신시간이 평균 약 15% 증가하였다. 전체 CPU 시간 중 MPI 통신 시간이 CPU 코어수 증가에 따라 약 19.2%~44.7% 정도 증가했다. 이에 대한 주요 원인은 네트워크 가상화로 인한 오버헤드가 도커의 MPI 통신 성능에 영향을 주었기 때문이다.

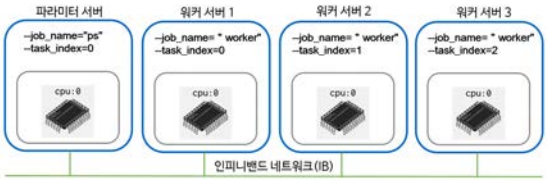


(그림 2) CPU 코어수 증가에 따른 MPI 통신 시간 비교

그리고, 3개 계산 노드에 걸쳐서 작업 당 6개의 CPU 코어를 사용하는 동시 실행 작업의 수가 증가함에 따른 실행경과시간을 측정하였다. (그림 3)과 같이 싱귤레러티는 베어메탈노드와 성능 차이가 없지만, 도커는 베어메탈노드 대비 평균 약 6.6% 정도 성능이 낮게 나타났다. 작업수 증가에 따른 각 작업 구간(1→2개, 2→3개, 3→4개)의 성능 저하율은 평균 약 22.9%이며, 1개 작업 대비 4개 작업에서는 약 1.8배(84%)의 성능 저하가 발생하였다. 이때 MPI 통신 시간은 약 2배 정도 증가하였다.

(그림 3) 작업 수 증가에 따른 실행경과시간 비교

AI 분야에서 텐서플로우를 활용하여 (그림 4)에서와 같이 CPU만 장착된 파라미터 서버 1대와 워커 서버 3대에서 시험을 진행하였다. 이와 같은 분산 텐서플로우 환경에서 파이썬으로 작성된 simple softmax 모델과 MNIST 필기체 숫자 이미지 데이터를 사용하여 필기체 숫자 이미지 학습을 실행하였다[7]. 이때, 파라미터 서버(--jobname="ps")는 공유변수의 생성 및 업데이트를 담당하며, 실질적인 모델의 연산은 워커 서버(--jobname="worker")에서 워커를 통해 실행된다. 각 워커서버에서 워커는 CPU 전체(CPU:0) 단위로만 연산 자원을 할당 받을 수 있다.



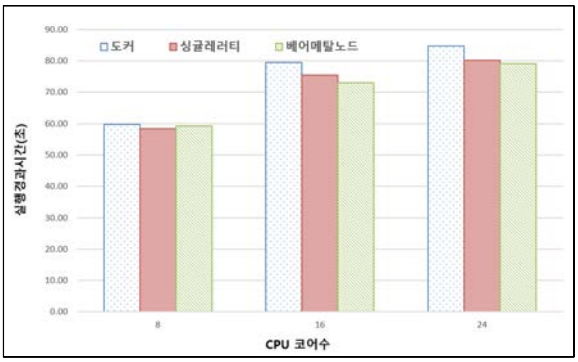
(그림 4) 분산 텐서플로우 구성도

(그림 5)는 동기식 데이터 병렬화 훈련 방식을 사용하여 각 워커서버의 워커에 8개 단위로 할당되는 CPU 코어수(워커수) 증가에 따른 학습 실행 경과시간을 비교하고 있다.

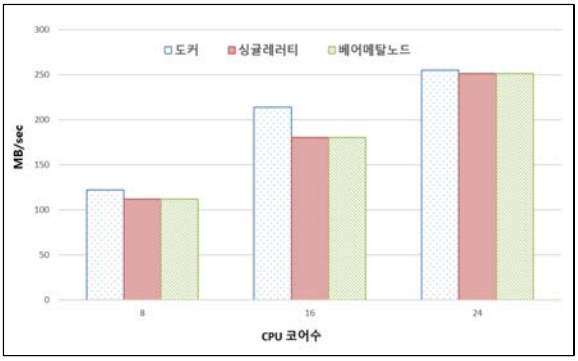
동기식 훈련에서는 CPU 코어수(워커수)가 증가함에 따라 공유변수의 동기화를 위한 오버헤드로 인해 실행경과시간이 증가하고 있다. 싱귤레러티는 베어메탈 노드 대비 성능 저하가 거의 없었으나, 도커는 평균 약 5.6% 정도의 성능 저하를 나타내었다. 특히, (그림 6)에서 CPU 코어수(워커수)가 증가함에 따라 파라미터

서버에서 네트워크 데이터 전송량 또한 증가하였다.

이에 따라, CPU 코어수(워커)가 급격하게 증가하면 파라미터 서버와의 네트워크 트래픽도 비례하여 늘어나기 때문에 네트워크 성능이 연산 성능에 많은 영향을 줄 것으로 예상된다.



(그림 5) CPU 코어수 증가에 따른 실행경과시간 비교



(그림 6) CPU 코어수 증가에 따른 파라미터 서버 네트워크 데이터 전송량 비교

(그림 7)은 동기화된 훈련 방식과 3개 워커 서버에 걸쳐서 작업 당 3개의 워커에서 총 24개의 CPU 코어를 사용하는 작업들의 실행 경과시간을 비교하고 있다. 각 작업은 워커 서버에서 특정 개수의 CPU 코어를 고정 할당받을 수 없기 때문에, 전체 CPU 코어를 공유하여 실행된다.

(그림 7) 작업 수 증가에 따른 실행경과시간 비교

싱귤레러티는 베어메탈노드와 유사한 성능을 보였으며, 도커는 베어메탈 노드 대비 평균 약 5.3% 정도의 성능 저하를 나타냈다. 작업수 증가에 따른 각 작업 구간(1→2개, 2→3개, 3→4개)의 성능 저하율은 평균은 약 21.8% 수준이며, 1개 작업 대비 4개 작업에서는 3개 실행 플랫폼 평균 약 80.6% 정도 성능이 낮게 나타났다.

이에 따라 CPU 부하가 증가하였으며, 워커 서버들과 통신하는 파라미터 서버 간 네트워크에서 데이터 전송량도 약 2배 이상 증가하였다. 참고로, 네트워크 데이터 전송량은 dstat(실시간 부하 모니터링 프로그램)을 사용하여 파라미터 서버에서 측정하였다[8].

4. 결론

본 연구에서는 주요 서비스 분야별 애플리케이션을 기존 클러스터와 같이 호스트 OS만 설치된 베어메탈 노드와 도커 및 싱귤레러티 컨테이너에서 각각 실행하여 성능을 측정하고 오버헤드를 비교 분석했다.

싱귤레러티는 도커보다 성능 오버헤드가 작았으며, 특히, 도커는 퀀텀 에스프레소와 같이 MPI 통신이 아주 많은 워크로드에서는 네트워크 가상화 오버헤드로 인해 싱규레러티보다 성능 오버헤드가 더욱 증가하였다. 또한, 멀티 혹은 매니 코어로 구성된 계산 노드를 보다 효율적으로 이용하기 위해 다수의 컨테이너 작업을 동시에 실행하는 경우에도 도커가 싱귤레러티보다 베어메탈 노드 보다 성능이 저하되었다.

사 사

본 논문은 KISTI “국가 플래그십 초고성능컴퓨터 인프라 구축 및 서비스 사업[K-22-L02-C01-S01]”에 의해 지원된 연구임.

참고문헌

[1] KISTI, “KISTI 슈퍼컴퓨터 5호기 운영(안)”, 2017
[2] LBNL, “Singularity”, available at http://singularitylbl.gov/
[3] “Docker documentation”, available at https://docs.docker.com/
[4] 우준, “고성능컴퓨팅을 위한 이기종 컨테이너 통합 서비스 시스템”, 박사학위논문, 충남대학교, 대전, 2018
[5] Project Calico, “Learn Project Calico”, available at https://www.projectcalico.org/learn/
[6] “TAU User’s Guide”, available at https://www.cs.uoregon.edu/research/tau/tau-usersguide.pdf
[7] TensorFlow, “mnist_replica.py source code”, available at https://github.com/tensorflow/tensorflow/blob/master/tensorflow/tools/dist_test/python/mnist_replica.py
[8] Dag Wieers, “Dstat: Versatile resource statistics tool”, available at http://dag.wiee.rs/home-made/dstat/