컨테이너 기반 멀티호스트 네트워킹 성능 테스트

최윤근*, 우 준*, 리국화**
*한국과학기술정보연구원 국가슈퍼컴퓨팅연구소
**건국대학교 신기술융합과
e-mail:{ykchoi, wjnadia}@kisti.re.kr, heavenkong@gmail.com

Container based Multi-host networking performance test

Youn Keun Choi*, Joon Woo*, Guohua Li**
*National Institute of supercomputing and Networking, KISTI
**Dept of Advanced Technology Fusion, Konkuk University

요 약

컨테이너 기반 멀티 호스트 네트워킹 기술에 대하여 성능 테스트를 진행하여 컨테이너 기술이 HPC 서비스에 적용 가능한지를 검증하고 효율적인 네트워크 구성을 위한 기술 근거를 제시한다. 이를 위해 테스트베드를 구축하고 컨테이너 기술인 도커와 Singularity에 대하여 기술 비교 분석을 진행했다.

1. 서론

현재 초고성능컴퓨팅 인프라 자원은 자원 확장성, 다양한 작업 환경구성, 신속한 자원 신청 및 작업 실행 등의 요구에 효율적으로 대처하기가 용의치 않다. 이를 해결하기 위해서 해외 초고성능컴퓨팅 서비스 기관에서는 최근 각광 받고 있는 컨테이너 기술을 이용하여 이를 극복하려 노력하고 있으나 국내는 아직 활발하게 연구되고 있지 않다. 컨테이너 기술이 HPC 서비스에 적용되기 위해서는 유연한 사용자 환경 구성의 용의성과 더불어 기본적으로 HPC 시스템의 성능을 극대화할 수 있는 고성능 네트워킹 성능이 보장되어야 한다. 본 연구는 이를 판단하기 위해 컨테이너 platform으로 구성된 테스트 베드를 구축하고 HPL을 사용하여 HPC 서비스에서 가장 중요한 노드간의 네트워킹 성능을 테스트하고 이를 평가한다.

2. 테스트를 위한 시스템 구축

KREONET으로 연결된 환경에서 <그림 1>과 같이 테스트 베드를 구축한다. 총 4개의 pcn01, pcn02, pcn03, pcn04 노드로 구성되고 상세한 사양은 <표 1>과 같다. 각 노드의 OS는 CentOS 7.2.1511 버전으로 설치 되였으며 2개의 도커 컨테이너가 있다. 각 컨테이너는 도커 허브에서 가져온 이미지로 생성한 것이며 컨테이너들은 도커의 오버레이 네트워크로 연결 되어 있다. 도커의 오버레이 멀티호스트는 10G(10.1.0.0/24) 네트워크를 사용하고 퍼블릭 네트워크는 1G(134.75.117. 0/24) 네트워크를 사용한다.

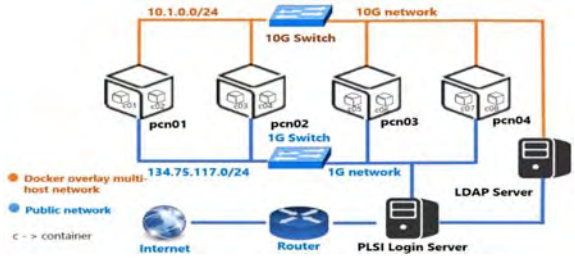
3. 테스트 베드 기반 HPL 성능 테스트

테스트 베드의 성능 테스트는 HPL를 사용한다. HPL은 대용량 메모리 시스템을 벤치마크 하는데 사용되는 벤치마크 툴로 Top 500 site에서 사용하는 공식 프로그램으로 cpu의 Flot 수를 측정하는 프로그램이다. 테스트는 pcn03, pcn04 노드에서 실행되며 컨테이너 프레임워크 2가지(도커, Singularity) 및 호스트 환경에서의 HPL 성능을 비교 테스트 한다.

pcn03, pcn04에서 도커, Singularity 및 호스트 환경에서 HPL 성능 테스트 결과는 아래 <그림 2>와 같다. 테스트 결과로부터 알 수 있다시피 bare-metal, Singularity, calico(etcd) 네트워크에서 HPL성능이 거의 비슷하게 기타 네트워크 보다 좋게 나왔다.

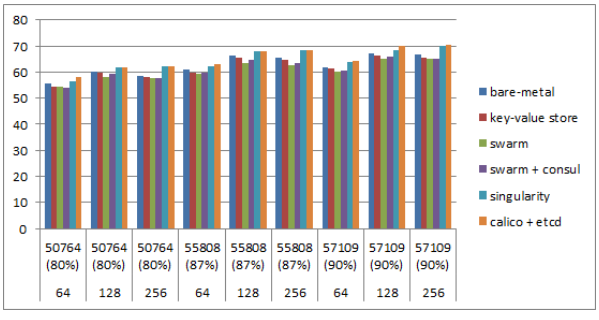
<표 1> Node Specification

노드	cpu 모델	GHz	코어	메모리 (GB)	OS
pcn01	Intel Xeon E5615	2.4	12	7.6	CentOS 7.2.1511
pcn02					
pcn03	Intel Xeon E5620		8	15	
pcn04					



<그림 1> Testbed system Architecture

성능 테스트 값 단위는 Gflops이다. Singularity에서는 실질적으로 호스트의 네트워크를 사용하기 때문에 bare-metal과 거의 비슷한 성능을 보여주지만 calico는 컨테이너 기반 멀티 호스트 네트워킹 환경으로 overlay네트워크를 형성함에도 불구하고 거의 bare-metal과 비슷한 성능을 보이고 있는데 이유는 calico는 기타 overlay 네트워크 즉 key-value store, swarm 등과 달리 기본 리눅스 overlay 드라이버를 사용하는 것이 아닌 calico자체 튜닝 드라이브를 사용하여 오버레이 네트워크를 형성하기 때문이다.



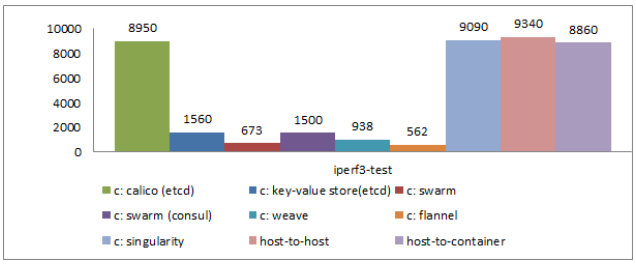
<그림 2> HPC Benchmark Tolls Test result

4. 도커를 이용한 멀티 호스트 네트워킹 성능 테스트

도커를 이용한 멀티 호스트 네트워킹 성능 테스트에서는 Iperf를 사용했다. 멀티 호스트 네트워킹 성능 테스트에서는 오버레리 네트워크, weave 네트워크, flannel 네트워크, Singularity 네트워크 등을 테스트 했다. 테스트는 호스트 사이의 컨테이너 네트워킹 성능 테스트 및 한 개 호스트안의 컨테이너 사이의 네트워킹 성능 테스트로 구성 되었으며 모두 10G 네트워크를 사용했다.

가. 호스트 사이의 컨테이너 네트워킹 성능 테스트

테스트 결과<그림 3>로부터 도커의 멀티 호스트 네트워크 환경을 구성해주는 overlay 네트워크, weave 네트워크, flannel 네트워크가 10G 네트워크에서 각각 1.56Gbits/sec, 964Mbits/sec, 562Mbits/sec의 성능임을 알 수 있음. Singularity와 호스트의 네트워크는 성능은 각각 9.09Gbits/sec, 9.34Gbits/sec의 성능을 가진다. 결과에서 볼 수 있듯이 calico 네트워크 iperf3성능은 Singularity와 호스트의 bare-metal 성능과 거의 비슷하게 나옴으로 8.95Gibts/sec에 도달했다. 이로부터 알 수 있듯이 오버레이 네트워크 구성 중 calico가 성능이 제일 높았다.



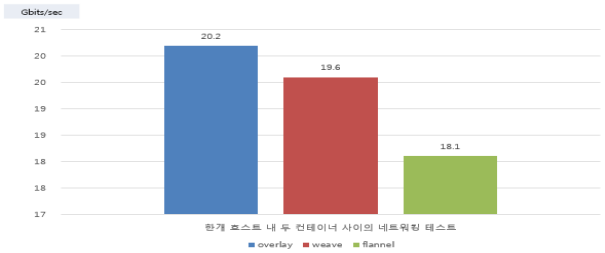
<그림 3> Iperf3 Test result

나. 한 호스트의 두 컨테이너 간 네트워킹성능 테스트

테스트 결과<그림 4>로부터 도커의 멀티 호스트 네트워크 환경을 구성해주는 overlay 네트워크, weave 네트워크, flannel 네트워크가 10G 네트워크에서 각각 20.02 Gbits /sec , 19.6Gbits/sec, 18.1Gbits/sec의 성능임을 알 수 있었다.

다. 두 노드 간 컨테이너 기반 네트워크 성능 테스트

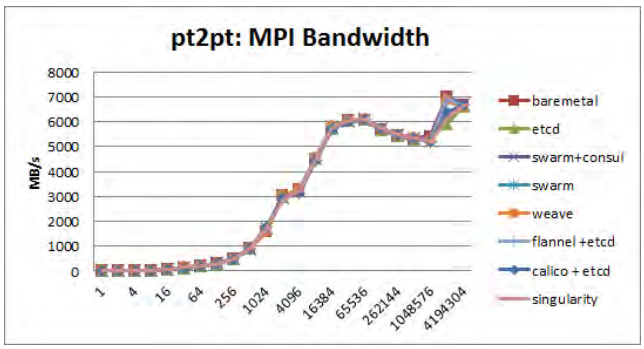
이 테스트를 위해 MPI Latency와 Bandwidth를 측정하는 네트워크 성능 벤치마크 툴인 OMB (Ohio MicroBenchmark) 를 활용하였다.



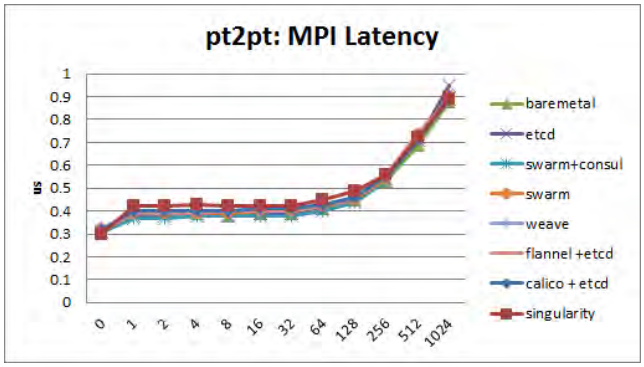
<그림 4> Container Performance Test Result on one node

OMB벤치마크 툴은 싱글 페어 Ping Pong Latency와 Ping Pong Bandwidth를 측정할 뿐만 아니라 MPI allreduce, allgather 등 MPI기능들을 실행 할 때 Latency와 Bandwidth에 대하여 측정한다. 테스트 결과는 <그림 5>와 같다. MPI Latency테스트 결과 bare-metal calico의 테스트 결과가 거의 비슷하며 기타 네트워크 성능과도 거의 차이가 나지 않고 있다.

Barrier Latency에서 베어메탈이 기타 컨테이너 기반 멀티 호스트 네트워킹과 비교 했을 경우 Latency가 낮게 나타난다. MPI Put Bandwidth와 MPI Get Bandwidth에서 베어메탈이 기타 네트워크보다 특정 메시지 사이즈에서 성능이 뛰어나게 차이가 났다. 전체 테스트 결과 두 노드에서 bare-metal, Singularity, calico에서 기타 네트워크보다 조금 더 좋은 성능을 보여준다.



<그림 5> pt2pt MPI Bandwidth



<그림 6> pt2pt MPI Latency

참고문헌

[1] Blake Cadwell, "Running Docker on Lustre", OLCF/ORNL, 6 April, 2016

[2] Docker, https://docs.docker.com/engine/understanding-docker/

[3] Singularity, http://singularity.lbl.gov/faq

[4] Shifter, http://www.nersc.gov/research-and-development/user-defined-images/

[5] Shane Canon, Doug Jacobsen, "Shifter:Container for HPC", NERSC/Lawrence Berkeley Lab, 12 May, 2016

[6] Jeff Layton, "A Container for HPC", HPC Magazine, 2016

[7] Ricardo Rocha, Mathieu Velten, Bertrand Noel, "Containers and Orchestration in the CERN Cloud", IT Technical Forum, 8 April, 2016

[8] Xinyong Xiang, Rong Zhu, "Cross-Cloud Migration of Container Persistent Data", Hwawei, 2016

[9] Burak Yenier, Wolfgang Gentzsch, "An Introduction into UberCloud Containers", 4 Nov, 2015