KVM 가상머신에서 도커를 사용하는 시스템의 호스트 메모리 부하에 따른 task 처리 성능 분석

장용현*, 이재학*, 유헌창*
*고려대학교 컴퓨터학과
wkd3475@gmail.com, smreodmlvl@gmail.com, yuhc@korea.ac.kr

Analysis of task processing performance depending on the host memory load of the system using the docker in the KVM virtual machine

Yong-Hyeon Jang*, Jaehak Lee*, Heonchang Yu*
*Dept. of Computer Science, Korea University

요 약

도커는 하드웨어 가상화를 지원하는 KVM과 달리 호스트의 커널을 공유하기 때문에 보안적인 문제가 생길 수 있다. 또한, 도커는 호스트 OS에 종속적이기 때문에 다른 OS에 종속적인 컨테이너를 실행할 수 없다는 단점도 있다. 이를 보완하기 위해 KVM을 이용해 가상머신을 실행하고 가상머신에서 도커를 이용하면 도커와 KVM의 장점을 살린 시스템을 구성할 수 있다. 실제 이 시스템의 실효성과 안정성을 평가하기 위해 실험을 진행하였고, 호스트의 메모리만 충분하다면 실효성과 안정성이 보장됨을 확인하였다.

1. 서론

도커는 cgroups와 namespace를 이용하여 호스트의 커널을 공유하고 독립적인 컨테이너를 실행한다. cgroup은 리눅스 커널에서 제공해주는 자원에 대한 제어를 가능하게 해주는 기능이고, namespace 역시 리눅스 커널에서 제공해주는 독립적인 공간을 제공하여 충돌을 방지해주는 기능이다. 이를 통해 기존의 하드웨어를 가상화하는 KVM과 같은 하이퍼바이저를 이용한 가상화 기법보다 더 적은 용량으로 빠르게 격리된 환경을 제공할 수 있다.

하지만 컨테이너가 호스트의 커널을 공유하기 때문에 작동 중인 컨테이너 중에 하나라도 공격을 당하게 된다면 호스트 커널을 통해 다른 컨테이너 까지 영향을 줄 수 있다. 하지만 하드웨어를 가상화하는 KVM은 완전히 분리된 환경을 제공하기 때문에 하나의 가상머신이 공격당해도 다른 가상머신에는 영향을 주지 않는다. 그리고 컨테이너는 호스트 OS 커널에 종속적이기 때문에 다른 OS 환경을 사용할 수 없지만 하드웨어 가상화를 통해서는 호스트 OS와는 다른 가상화된 OS를 사용할 수 있다.

위와 같은 이유로 도커는 기존 하드웨어 가상화를 완전히 대체할 수 없고 이를 보완하기 위해 KVM의 가상머신에서 도커 컨테이너를 실행시켜 컨테이너가 공격당해도 공격당한 가상머신에서만 타격을 입고 다른 가상머신에는 영향을 주지 않게 구성을 할 수 있다. 또한, 이렇게 시스템을 구성하면 컨테이너가 구동되는 게스트 OS가 호스트와 다른 OS를 기반으로 실행될 수 있기 때문에 다른 OS에 종속적인 컨테이너를 실행할 수 있다는 장점을 얻을 수 있다. 실제로 아마존과 구글에서 컨테이너 호스팅을 지원하지만 보안을 위해서 가상머신을 이용하여 공간을 격리시고 그 위에서 컨테이너를 실행하게 한다[1].

본 논문에서는 KVM 가상머신에서 도커 컨테이

I "본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구 센터지원사업의 연구결과로 수행되었음" (IITP-2018-0-01405)
II 이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기 획평가원의 지원을 받아 수행된 연구임 (No.2018-0-00480)

너를 실행시키는 환경의 실효성을 확인하기 위해 실제 서비스 환경에서의 성능 평가를 수행하고자 한다. 이를 평가하기 위해 기존 시스템 모델과 비교하여 KVM 가상머신에서 도커 컨테이너를 실행할 때 각 서버의 task 처리 속도에 어떤 영향이 생기는지 평가한다. 또한, 각각의 시스템에 대해 호스트의 메모리에 부하가 늘어남에 따라 서버의 task 처리 속도에 미치는 영향을 평가하여 안정성에 대한 평가도 진행한다.

2. 관련 연구

Xuehai Tang이 2014년에 작성한 "Performance Evaluation of Light-Weighted Virtualization for PaaS in Clouds" 논문을 살펴보면 도커와 KVM에서의 Isolation Performance를 측정하였다. 메모리 stress에 대해서는 도커는 43.3% 성능 저하, KVM은 4.2%의 성능 저하를 보여주었고, Disk stress에 대해서는 도커는 34.7%의 성능 저하, KVM은 20.9%의 성능 저하를 보여주었다. 그리고 도커와 KVM 모두 CPU stress에 대해서는 0%의 성능 저하를 보여주었다[2].

위 논문의 실험을 통해 각각의 실험에 사용되는 시스템의 호스트 메모리에 부하를 주면 성능 간섭이 발생할 것을 예상해볼 수 있고, 실제로 부하를 주어 각 서버의 task 수행 시간에 어떤 영향을 미치는지를 확인하고자 한다.

3. 실험 환경 구성 및 수행 방법

3.1 실험 환경 구성

KVM 가상머신에서 도커 컨테이너를 이용해 실행 중인 서버의 성능을 기존 시스템 모델들과 비교하기 위한 실험과 KVM 가상머신에서 도커 컨테이너를 이용해 실행 중인 서버의 안정성을 평가하기 위해 메모리 부하가 있을 때 서버들 사이의 성능 균형이 유지되는지를 평가하는 실험을 진행하도록 한다.

실험에 사용되는 컴퓨터 HostA와 HostB의 사양은 <표 1>과 같다. 그리고 HostA와 HostB에서 KVM으로 실행되는 하나의 가상머신의 성능은 <표 2>와 같다.

<표 1> 실험 환경

	HostA	HostB
CPU	Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz, 4 cores	Intel(R) Core(TM) i5-6400 CPU @ 3.40GHz, 4 cores
RAM	8G	32G
Disk	1TB	1TB
OS	Ubuntu 18.04 LTS	Ubuntu 18.04 LTS

<표 2> 가상머신

	HostA	HostB
vCPU	1 core	1 core
RAM	1G	2G
Disk	15GB	15GB
OS	Ubuntu server 18.04.04 LTS	Ubuntu server 18.04.04 LTS

1) 서버 및 클라이언트 구성

서버는 node.js express를 이용해서 제작하였으며, get 요청을 받으면 2,000,000번 write 작업을 수행하여 약 30MB 크기의 파일을 생성 후 저장한다. 그리고 task 수행 시간을 클라이언트에게 반환해준다. 클라이언트는 한 회차에 4개의 서버에 동시에 request를 전송하며, 수행 시간을 전부 받고 나서 task 수행 시간을 저장 한 후에 다음 회차로 넘어간다. 그리고 이런 과정을 총 50회 수행한다.

2) 메모리 부하

메모리 부하는 stress 툴[3]을 사용한다. stress 툴의 -m 옵션을 통해 부하를 주는 프로세스의 갯수를 정할 수 있고, 하나의 프로세스당 256MB 만큼의 malloc 작업을 수행하는 것이 default로 설정이 되어있다.

3.2 실험 수행 방법

본 실험은 기존 시스템 모델과 도커를 KVM 가상머신에서 실행하는 환경의 성능 비교와 도커를 KVM 가상머신에서 실행할 경우 호스트의 메모리 부하에 따라 각각의 가상머신의 도커 컨테이너의 성능이 균등하게 유지가 되는지를 평가하기 위해 총 4가지의 시스템 모델에 대하여 메모리 부하 작업을 수행한다.

메모리 부하 작업은 stress 툴의 -m 옵션으로 stress를 주는 프로세스의 갯수를 HostA는 1개씩 늘려가며, HostB는 20개씩 늘려가며 호스트의 메모리 부하가 증가함에 따라 각각의 서버에 생기는 영향을 평가한다. 4가지의 실험 유형은 3.3과 같다.

3.3 실험 유형

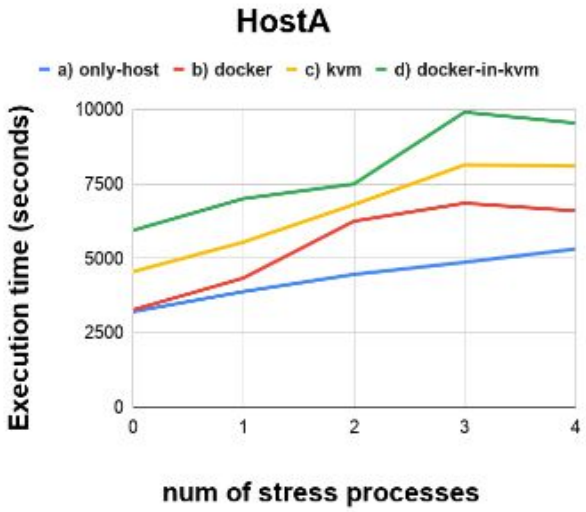
a) 호스트 컴퓨터에서 작동 중인 4개의 api 서버에 대한 성능 평가

b) 호스트 컴퓨터에서 도커 컨테이너로 격리되어 작동 중인 4개의 api 서버에 대한 성능 평가

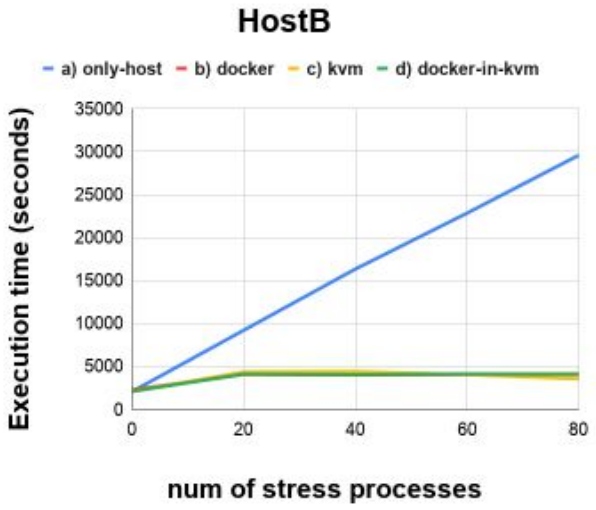
c) 4개의 가상머신에서 격리되어 작동 중인 4개의 api 서버에 대한 성능 평가
d) 4개의 가상머신에서 격리되어 도커 컨테이너로 작동 중인 4개의 api 서버에 대한 성능 평가

4. 실험 평가

4.1 기존 시스템 모델과 성능 비교



(그림 1) HostA, 시스템별 서버 평균 task 수행 시간



(그림 2) HostB, 시스템별 서버 평균 task 수행 시간

(그림 1) HostA에서 실험한 시스템들의 성능을 보면 평균적으로 호스트에서만 서버를 운영한 것보다 도커를 사용한 시스템에서는 1.26배, KVM를 사용하는 시스템에서는 1.53배, KVM에서 도커를 사용하는 시스템에서는 1.84배만큼 처리 속도가 느려짐을 확인할 수 있다. 그러나 (그림 2) HostB에서는 호스트에서만 서버를 운영한 시스템만 호스트의 메모리 부하가 증가함에 따라 수행시간이 비례해서 증가하고, 나머지 시스템에서는 호스트의 메모리부하가 증가해도 메모리 부하가 없을 때보다 1.9배 이상으로 처리 속도가 느려지지 않는 것을 확인할 수 있다.

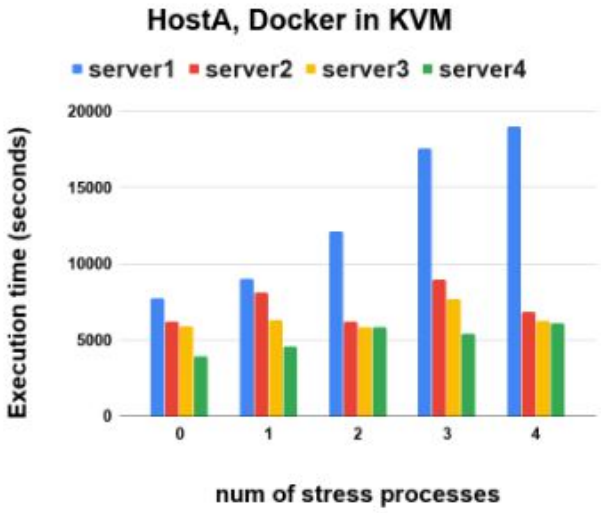
HostA와 HostB에서 실험한 결과의 차이가 생기는 이유는 HostA가 가상머신 4개를 운영하기에는 메모리 용량이 부족하기 때문이다. 호스트의 메모리 부족으로 메모리 스왑이 많이 발생하게 되고 이로 인해 도커보다 메모리를 더 많이 사용하는 가상머신이 더 많은 성능 저하를 일으켰다고 볼 수 있다.

그리고 HostA에서 메모리 스왑이 많이 발생한 원인으로는 Memory Ballooning이 발생했다고 볼 수 있다. Memory Ballooning은 호스트의 메모리가 부족하고 게스트 가상머신의 메모리는 여유가 있을 때 발생한다. 이런 상황이 발생하면, Balloon Controller가 게스트 OS 내부의 메모리를 inflate시켜 메모리 스왑을 일으키고 메모리의 내용을 disk로 스왑 아웃하여 비워낸다. 이를 통해 호스트 OS의 메모리 공간을 확보하게 된다[4]. 그리고 Memory Ballooning으로 인한 가상머신의 성능 저하를 막기 위해 Memory Ballooning을 비활성화하는 시도도 존재한다[5]. 실제 본 실험 과정에서 HostA는 가상머신을 4개를 실행시키면 스왑 공간이 사용됨을 확인하였고, 이를 통해 HostA에서는 Memory Ballooning이 발생해서 성능 저하가 일어났다고 볼 수 있다.

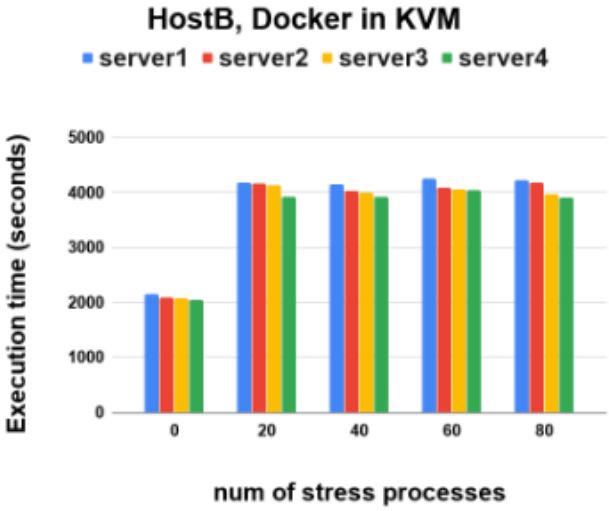
하지만 HostB처럼 호스트 OS의 물리적 메모리가 충분한 경우에는 메모리 스왑이 발생하지 않아 게스트 OS의 성능 저하가 거의 일어나지 않았음을 확인할 수 있다. 이 실험을 통해 호스트의 메모리가 충분하다면 KVM 가상머신을 사용하여도 서버의 성능은 도커만을 사용할 때와 비슷한 성능을 보여준다는 것을 확인할 수 있어 이러한 시스템의 실효성을 확인할 수 있다. 그리고 도커나 KVM과 같은 가상화를 이용하면 호스트에서 바로 서버를 운영하는 것 보다 안정적으로 성능을 유지할 수 있음을 확인할 수 있다.

4.2 메모리 부하에 따른 각 서버 사이의 간섭 현상 측정

(그림 3) HostA에서의 실험 결과를 보면, 가상머신 4대를 실행시키는 것 자체만으로도 호스트의 메모리에 부하가 심해 각 서버에서 성능 차이가 생기는 것을 확인할 수 있다. 그리고 메모리의 부하가 증가함에 따라 성능의 불균형이 더 심해지는 것을 확인할 수 있다. 이를 통해, KVM에서 실행되는 게스트 OS는 호스트 메모리에 심한



(그림 3) HostA, docker in kvm의 서버별 task 수행시간



(그림 4) HostA, docker in kvm의 서버별 task 수행시간

부하가 생기면 성능 불균형이 발생함을 확인할 수 있다,

(그림 4) HostB에서는 HostA에서와는 다르게 서버들 사이의 성능이 비교적 비슷하다는 것을 확인할 수 있다. 이런 결과나 나온 이유로 호스트의 메모리가 충분하여 Memory Ballooning과 같은 성능에 영향을 줄 수 있는 원인이 발생하지 않았기 때문이라고 유추한다. 그리고 이를 통해 호스트의 메모리가 충분하다면, 안정성을 보장함을 확인할 수 있었다.

5. 결 론

KVM을 이용해 가상머신을 실행하고 가상머신에서 도커를 이용하면 도커와 KVM의 장점을 살린 시스템을 구성할 수 있다. 그리고 실험을 통해 KVM 가상머신에서 도커를 사용하는 시스템을 사용해도 호스트의 메모리만 충분하다면 실행되는 서버의 성능은 도커만 사용하는 시스템과 KVM만 사용하는 시스템과 동일한 성능을 보이고, 동시에 운영되는 서버 사이의 성능 균형도 보장함을 확인하였다. 그러나 호스트의 메모리가 충분하지 않으면, 성능 저하와 서버간의 성능 불균형이 발생함을 확인하였다. 그래서 도커와 KVM을 같이 사용하여 보안적인 문제를 해결하는 환경을 구축하는 것은 메모리가 적은 호스트에서는 권장되지 않으며, 메모리가 충분한 호스트에서 도커를 KVM에서 사용하는 시스템의 경우에는 운영하는 서버의 task 처리 성능 저하가 없고 동시에 운영하는 서버들 사이에 성능 균형을 보이기 때문에 실효성과 안정성이 보장됨을 확인하였다.

참고문헌

[1] https://xenproject.org/2015/08/11/will-docker-replace-virtual-machines/

[2] Xuehai Tang, Zhang Zhang, Min Wang, Yifang Wang, Qingqing Feng, and Jizhong Han, “Performance Evaluation of Light-Weighted Virtualization for PaaS in Clouds”, ICA3PP Algorithms and Architectures for Parallel Processing, pp. 415-428, 2014

[3] https://linux.die.net/man/1/stress

[4] Chin-Hung Li , “Evaluating the Effectiveness of Memory Overcommit Techniques on KVM-based Hosting Platform”, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol. 6, No.10, pp. 1218-1222, 2012

[5] Nadav Amit, Dan Tsafrir, Assaf Schuster, “VSwapper: a memory swapper for virtualized environments”, ACM, pp. 349-365, March 1-4, 2014