Docker Swarm에서 컨테이너간의 메모리 자원에 대한 성능 간섭 측정

정진원, 이재학, 유헌창
고려대학교 컴퓨터학과
e-mail:{jin4812, smreodmlvl, yuhc}@korea.ac.kr

Measuring Performance Interference on Memory Resources between Containers in Docker Swarm

JinWon Jeong, JaeHak Lee, HeonChang Yu
Dept of Computer Science and Engineering, Korea University

요 약

Docker Swarm은 호스트 머신에서 여러 개의 컨테이너들을 실행할 때 발생하는 네트워크와 호스트 리소스 등을 포함한 여러 문제를 해결해 주기 위해 등장하였다. 하지만 컨테이너간의 메모리 경합으로 인한 성능 간섭 문제는 여전히 대두되고 있다. 본 논문에서는 성능 간섭 정도를 측정하기 위해 Docker Swarm을 이용하여 클러스터 환경을 구축하고 메모리 부하 작업을 수행하는 특정 스레드 개수 및 시간을 선정하여 다양한 실험을 진행하였다. 그 결과 특정 스레드 개수를 할당해 주었을 때 특정 시점에서 컨테이너간의 성능 간섭이 가장 크게 발생하였으며 그 이후의 시점부터는 성능 간섭 정도가 크게 나타나지 않는 것을 확인하였다. 이를 토대로 Docker Swarm에서 사용 중인 스케줄링 방법을 개선하여 컨테이너간의 성능 간섭을 최소화할 수 있는 향후 연구 방향을 모색할 수 있을 것으로 보인다.

1. 서론

Docker Swarm은 오픈소스 프로젝트인 Docker에서 공식적으로 만든 Docker 컨테이너를 위한 오케스트레이션 툴이다. 오케스트레이션 툴이란 여러 호스트 서버의 컨테이너들을 배포 및 관리하기 위한 툴을 의미한다. 즉, Docker Swarm을 이용하면 여러 개의 서버와 컨테이너들을 쉽게 관리할 수 있다. Docker Swarm은 manager노드와 worker노드로 구성된다. Manager노드는 Raft consensus algorithm을 이용해 클러스터에서 작동하는 다양한 서비스들이 일관된 상태를 유지할 수 있도록 관리한다. Raft consensus algorithm은 여러 서버 중 일부에 장애가 발생해도 나머지 서버가 정상적인 서비스를 제공할 수 있도록 해주는 합의 알고리즘이다. Worker노드는 manager 노드의 명령을 받아 컨테이너를 실행하는 노드를 의미한다. Docker는 호스트 OS의 커널을 공유하고 namespace를 기반으로 각각의 애플리케이션에 대한 격리된 환경을 제공한다. 사용자가 컨테이너를 실행하면, 해당 컨테이너에서 사용할 namespace를 생성하게 된다. Docker의 각 컨테이너는 자신에게 할당된 만큼의 자원을 점유할 수 있도록 격리되어야 한다. 여러 개의 컨테이너가 존재할 때 한 컨테이너가 자원을 초과해서 사용한다면 서로 간에 성능 간섭이 발생하게 되고 심각한 성능 저하 문제로 이어질 수 있게 된다[1]. 이러한 일이 발생하지 않도록 Docker는 각 컨테이너가 할당된 자원만큼만 사용하도록 조정하는데, 이때 리눅스의 cgroups를 이용해서 이 기능을 구현한다. 하지만 호스트 머신 내에서 메모리 서브시스템을 공유하는 컨테이너의 특성으로 인해 메모리 대역폭을 많이 사용하는 특정 컨테이너들이 실행되었을 때 성능이 저하되는 현상이 빈번히 발생하고 있다[2].

본 논문에서는 Docker Swarm을 이용하여 클러스

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

터 환경을 구축한 후 각 worker노드에서 메모리 부하 작업을 수행하는 컨테이너가 작동할 때 발생하는 성능 간섭 현상을 분석한다. 이를 바탕으로 구성한 실험 환경에서 어느 정도의 부하를 주었을 때 특정 애플리케이션에 대한 작업 수행시간의 증가량을 통해 성능 저하 정도를 측정한다.

본 논문의 구성은 다음과 같다. 2장에서 실험 환경과 실험을 수행하는 방법에 대해 서술하고 3장에서는 메모리 부하 작업을 수행하는 스레드의 개수에 따라 프로그램의 성능이 저하되는 정도를 측정하는 실험을 수행 및 분석하고, 마지막 4장에서는 결론 및 향후 연구에 대해 서술한다.

2. 실험 환경 및 수행 방법

본 실험에서는 메모리 부하 작업을 수행하는 컨테이너 및 wordcount 작업을 수행하는 다수의 컨테이너를 각 worker 노드에게 동일한 개수로 배포한 뒤, 각 worker노드 내의 컨테이너들 간에 발생되는 성능 간섭 정도를 측정하기 위해 Docker Swarm을 이용한 클러스터 환경을 구축한다. 실험 환경은 <표 1>, <표 2>와 같다. Docker Swam 클러스터는 manager 노드 1개, worker노드 5개로 구성한다. 물리 머신을 manager노드로 설정하였고, 각 worker노드는 드라이버가 virtual box로 설정된 docker-machine으로 구성하였다.

실험을 위한 애플리케이션으로 직접 구성한 특정 txt파일의 단어를 세는 작업을 총 10,000번 수행하는 wordcount 프로그램을 선정하였으며, 이를 통해 작업 수행시간의 결과 값을 도출한다. 메모리 부하 작업을 수행하는 애플리케이션으로는 리눅스 과부하 테스트 프로그램인 stress tool[3]을 이용하였다.

실험을 진행하기 위해 wordcount 프로그램의 image로 빌드 된 컨테이너들이 아무런 성능 간섭 없이 동작했을 때와 메모리 부하 작업으로 인해 성능 간섭이 발생했을 때, 각 worker노드별 wordcount 컨테이너들의 작업 수행시간을 비교함으로써 성능 저하 정도를 측정한다. 각 실험마다 구분된 메모리 부하 작업을 수행하는 스레드의 개수는 1개, 5개, 10개로 구성하였으며 메모리 부하 작업의 시간(timeout)을 5초에서 시작하여 2000초까지 주었을 때 각 worker 노드 내 wordcount 컨테이너들의 작업 수행시간의 평균값을 측정하였다. 메모리 부하 작업에 따른 성능 간섭으로 인한 성능 저하 정도를 측정하기 위해 총 세 단계의 작업을 진행한다.

<표 1> 물리 머신 환경

CPU	i5-7500 CPU @ 3.40GHz x 4
RAM	32GB
HDD	1TB
OS	Ubuntu 18.04.4 LTS

<표 2> Docker Swarm 환경

구분	Manager 노드	Worker 노드
노드 수	1개	5개
RAM	27GB	1GB
HDD	924GB	20GB

가장 먼저 아무런 성능 간섭이 없었을 때 컨테이너들의 작업 수행시간을 측정한다. 5개의 worker노드에게 총 50개의 wordcount 컨테이너를 배포한다. 각 worker노드는 Docker Swarm 스케줄러에 의해 컨테이너를 10개씩 균등하게 배분받게 된다. 그 후 각 worker노드마다 배분받은 컨테이너들의 작업 수행시간을 측정한 뒤 평균값을 계산한다. 위 과정을 총 5번 수행해서 각각의 계산된 평균값을 가지고 최종적인 평균값을 계산하여 각 worker노드 내 컨테이너들의 작업 수행시간의 평균값을 나타낸다.

두 번째로 성능 간섭이 발생했을 때 컨테이너들의 작업 수행시간을 측정한다. 각 worker 노드가 10개의 wordcount 컨테이너를 배분 받은 상황에서 추가로 메모리 부하작업을 수행하는 컨테이너를 한 개씩 worker노드에게 배포한다. 그리고 각 worker노드 내에서 timeout마다 컨테이너들의 최종 작업 수행시간의 평균값을 위와 같은 방식으로 계산한다.

마지막으로 메모리 부하 작업을 수행하는 스레드 개수별로 timeout마다의 각 worker노드 내 컨테이너들의 작업 수행시간의 평균값에서 아무런 성능 간섭이 없었을 때 컨테이너들의 작업 수행시간의 평균값을 빼줌으로써 그 결과 값을 그래프로 나타낸다. 이를 통해 각 worker노드 내 컨테이너들 간의 메모리 경합으로 인한 성능 저하 정도를 측정한다.

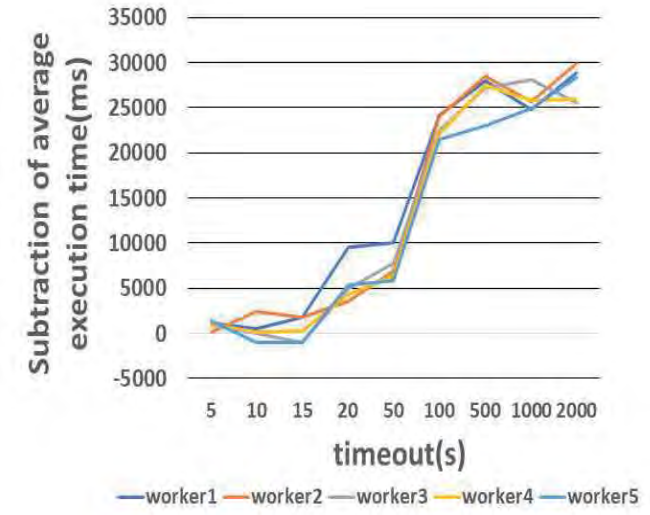
3. 실험 결과 및 분석

[그림 1]은 메모리 부하 작업을 수행하는 스레드 개

수가 1개일 때 timeout에 따른 각 worker노드마다의 작업 수행시간의 평균값 차이를 나타낸 그래프이다. Timeout이 5초 지점일 때 worker3과 worker4 그리고 worker5에서 작업 수행시간의 평균값 차이가 음수인 것을 볼 수 있다. 이는 오히려 메모리 부하 작업이 수행되었을 때의 경우가 평균적으로 작업 수행시간이 더 빨랐다는 것을 의미한다. 10초 지점일 때도 worker1과 worker3 그리고 worker4에서 같은 상황을 보여준다. 이는 이에 해당하는 각 worker노드마다 5번의 반복 작업 중 몇몇의 경우에 컨테이너를 10개씩 배분 받았다가 Docker swarm 스케줄러에 의해 컨테이너가 재배치되어 최종적으로 작업을 수행한 컨테이너의 개수가 다른 worker노드에 비해 훨씬 더 적었던 것이다. 따라서 컨테이너들 간에 메모리 경합이 비교적 덜 일어났고 평균적으로 작업 수행을 더 빨리해낸 상황이 발생한 것으로 보인다. 15초 지점 이후부터는 각 작업 수행시간의 평균값이 점점 차이가 나면서 500초와 1000초 지점에서 각 worker 노드마다 그래프가 최고점에 달성하는 것을 보였다. 그 이후의 시간부터는 그래프에 큰 변화가 없는 것을 볼 수 있으며, timeout을 더 늘려도 작업 수행시간의 평균값 차이는 더 이상 크게 나타나지 않았다.

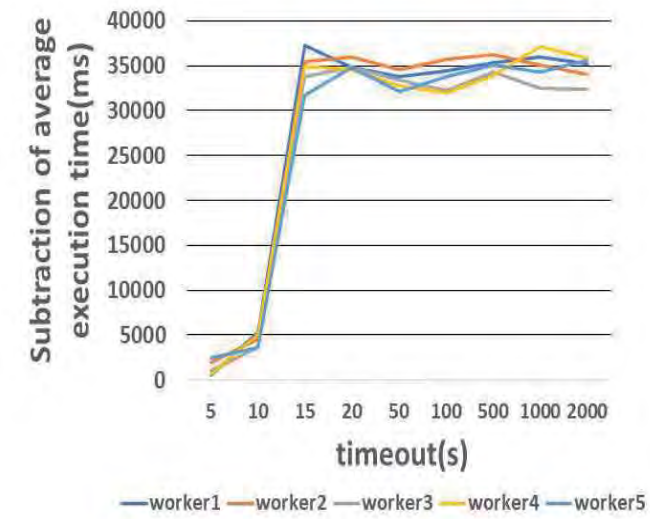
[그림 1] 스레드가 1개일 때 수행시간의 평균값 차이

[그림 2]는 메모리 부하 작업을 수행하는 스레드 개수가 5개일 때 timeout에 따른 각 worker노드마다의 작업 수행시간의 평균값 차이를 나타낸 그래프이다. Timeout이 10초 지점일 때 worker5, 15초 지점일 때 worker3과 worker5에서 작업 수행시간의 평균값 차이가 음수인 것을 볼 수 있다. 이는 [그림 1] 설명에서 서술한 내용과 같은 이유로 보인다. 그 이후의 시간부터는 각 worker노드마다 그래프가 증가하는 양상을 보이며 100초 지점일 때 크게 증가하는 모습을 볼 수 있다. 500초 지점 이후부터는 그래프에 큰 변화가 없는 모습을 보였다.



[그림 2] 스레드가 5개일 때 수행시간의 평균값 차이

[그림 3]은 메모리 부하 작업을 수행하는 스레드 개수가 10개일 때 timeout에 따른 각 worker노드마다의 작업 수행시간의 평균값 차이를 나타낸 그래프이다. [그림 1]과 [그림 2]를 보면 timeout이 500초와 2000초 지점 사이에서 큰 수치에 도달했고 [그림 3]은 15초 지점 일 때 모든 worker노드의 그래프가 급격하게 증가하는 것을 볼 수 있다. 이는 세 가지 실험 중에서 컨테이너들 간에 성능 간섭으로 인한 성능 저하가 가장 크게 일어난 모습을 보여준다. 그 이후부터는 그래프에 큰 변화가 없는 모습을 보였다.



[그림 3] 스레드가 10개일 때 수행시간의 평균값 차이

본 실험에서 각 worker노드마다 도출되는 컨테이너들의 작업 수행시간의 평균값을 더 정밀하게 나타내기 위해 총 5번의 반복 계산 작업을 실시하여 최종적인 평균값을 나타내었다. 이 과정에서 스레드 개수별로 메모리 부하 작업을 수행하는 컨테이너가 포함되었을 때 timeout에 따른 각각의 worker노드에서 나타난 컨테이너들의 작업 수행시간의 평균값들 중 최댓값과 포함되지 않았을 때 나타난 각각의 worker 노드 내 컨테이너들의 작업 수행시간의 평균값을 비교해본 결과 작업 수행시간의 증가율은 <표 3>과 같다. <표 3>을 보면 스레드 개수가 10개일 때 각 worker노드마다의 성능 저하율이 가장 크게 발생한 것으로 나타났다. 즉, 메모리 부하 작업을 수행하는 스레드 개수가 증가할수록 작업 수행시간이 더 크게 증가해 프로그램의 성능이 저하되는 현상을 확인할 수 있다.

<표 3> 수행시간 최대 증가율

스레드 개수	worker 1	worker 2	worker 3	worker 4	worker 5
1개	14.02%	11.7%	13.22%	11.92%	10.55%
5개	29.98%	30.93%	28.75%	28.17%	29.06%
10개	38.69%	37.45%	35.61%	38.11%	36.53%

4. 결론 및 향후 연구

본 논문에서는 Docker Swarm을 통해 구성한 클러스터 환경에서 메모리 부하 작업을 수행하는 컨테이너를 실행시켰을 때와 그렇지 않았을 때의 작업 수행시간을 측정 및 비교함으로써 나타난 성능 저하 현상을 관찰 및 분석하였다. 스레드 개수를 본 실험에서 사용한 개수보다 더 많이 할당하거나 timeout을 더 늘린다 하더라도 성능 저하 정도가 크게 증가하지 않았다. 이는 각각의 worker노드가 할당받은 메모리 용량에서의 한계점으로 보였으며 worker노드의 메모리 용량을 늘린다면 더 많은 스레드를 할당해 더 큰 성능 저하를 일으키는 모습을 확인할 수 있을 것으로 보인다.

본 실험에서 할당한 스레드 개수가 1개일 때와 5개일 때, 메모리 부하 작업이 수행되었을 때의 경우가 그렇지 않았을 때의 경우보다 평균적으로 컨테이너들의 작업 수행시간이 더 빨랐던 현상이 관찰되었다. 이는 Docker Swarm 환경에서 worker노드의 자원 여유에 따라 실행되는 컨테이너의 수가 달라질 수 있기 때문에 나타난 현상이다. 이를 바탕으로 각 worker 노드마다 다른 수의 컨테이너가 실행되고 있는 상황이라도 메모리 경합을 줄일 수 있는 적절한 스케줄링 방법을 연구해 볼 수 있다.

Docker Swarm에서 기본적으로 사용하고 있는 Raft consensus algorithm은 여러 서버 중 일부에 장애가 발생해도 제 기능을 유지하도록 해주지만 메모리 경합에 대한 고려는 하지 않는다. 그러므로 worker 노드 내 메모리 경합으로 인한 여러 컨테이너들의 성능 저하를 예방하지 못한다. 따라서 향후 연구에서는 이러한 성능 간섭으로 인해 발생하는 성능 저하를 해결하고자 Docker Swarm을 이용한 클러스터 환경에서 메모리 경합을 고려한 스케줄링 기법에 대하여 연구할 계획이다.

참고문헌

[1] Prateek Sharma et al., “Containers and Virtual Machines at Scale: A Comparative Study” 17th International Middleware Conference, 2016

[2] Junhee Park et al., “Performance Interference of Memory Thrashing in Virtualized Cloud Environments: A Study of Consolidated n-Tier Applications” IEEE 9th International Conference on Cloud Computing (CLOUD), 2016

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