쿠버네티스 환경에서 레플리카의 자원 세분성에 따른 서비스 응답시간 성능 분석

김건우 [1], 김동균 [1], 유헌창 [1]
[1] 고려대학교 정보대학 컴퓨터학과

{kimkw0911, kdonggyun97, yuhc}@korea.ac.kr

An Analysis of Service Latency Based on Replica Resource Granularity in Kubernetes Environment

Geonwoo Kim[1], Donggyun Kim[1], Heonchang Yu[1]
[1]Dept. of Computer Science and Engineering, Korea University

요 약

최근 클라우드 컴퓨팅의 발전에 따라 서비스 품질 보장과 자원 사용의 효율성을 위해 쿠버네티스의 HPA(Horizontal Pod Autoscaling) 기술이 범용적으로 사용되고 있다. HPA 는 자원에 대해 필요한 리소스만큼 동적으로 조정하기 때문에 파드의 자원을 세분화하면 자원 효율성이 높아지지만, 반대로 느린 응답 속도를 유발한다. 본 논문은 쿠버네티스 환경에서 파드의 기본 자원량을 다양하게 조정하여 동일한 작업 요청에 따른 레이턴시와 자원 사용량을 분석한다. 실험 결과 자원의 세분성이 높아질수록 컨테이너 오버헤드의 증가로 인해 자원 사용률과 서비스 레이턴시가 증가하였다.

1. 서론

클라우드 인프라의 발전으로 많은 기업은 인프라를 소유하지 않고 클라우드 서비스 공급자로부터 임대한다. 그로 인해 기업들은 물리적인 서버 구축 및 인프라 관리 작업으로부터 독립하여, 유연하고 운영하기 쉬운 비즈니스 모델을 갖출 수 있다[1].

HPA(Horizontal Pod Autoscaling)는 쿠버네티스의 대표적인 오토스케일링 기법으로, 기본 서비스 단위인 파드를 확장하고 관리하기 위해 널리 사용된다. HPA 는 CPU 사용량의 임곗값을 초과하면 자원을 확장 및 축소하여 서비스의 가용성을 보장한다. 기존의 HPA 연구는 주로 스케일링의 시점을 조정함으로써 응답시간 또는 자원 사용량을 최적화한다. 그러나, 스케일링이 수행된 후 할당된 총 자원 할당량이 같더라도 자원의 세분성에 따라 컨테이너의 증가로 인한 오버헤드가 증가하는 문제가 발생할 수 있다. 즉, 리소스 할당 관점에서 HPA 대상이 되는 애플리케이션에 대한 리소스 양도 고려되어야 한다.

본 연구에서는 HPA 가 수행되었을 때, 총 할당 자원이 동일한 환경에서 애플리케이션의 세분화가 성능에 미치는 영향을 분석하였다. 실험 결과, 자원의 세분성이 높아질수록 서비스 응답시간이 증가하고 오버헤드로 인해 CPU 사용률이 증가한 결과가 나타났다.

2. 쿠버네티스 HPA 레플리카의 자원 세분성

HPA 는 파드의 수인 레플리카를 동적으로 조절하여 애플리케이션의 부하를 분산시킨다. 부하에 의해 결정되는 레플리카 수는 파드의 기본 자원 할당에 따라 결정된다. 예로 들어, 애플리케이션이 부하에 의해 2 코어의 CPU 를 필요로 할 때, 레플리카의 수는 파드의 기본 자원 할당량에 반비례한다.

자원 세분성은 같은 자원 내에서 레플리카의 수에 의해 결정된다. 세분성이 증가하면, 부하가 적을 때 자원을 낭비하지 않게 해주지만, 반면에 파드 하나의 애플리케이션이 실행되기 위한 기본 오버헤드를 증가시킨다. 또한 컨테이너가 대규모로 배포됨에 따라 물리 머신의 OS 프로세스가 증가한다. 따라서, 물리 머신의 동일한 리소스를 공유하는 과정에서 프로세스 간 간섭이 증가하고, 이는 호스트 OS 커널에 더 많은 부하가 발생하게 된다[2-3].

본 연구에서는 고정된 자원을 사용할 때, 컨테이너 할당 자원을 다양하게 조정하여 애플리케이션의 CPU 자원할당 비용과 응답시간을 비교 분석한다.

3. 자원 세분성에 따른 성능 분석

3.1 실험 환경

본 논문에서 실험 환경은 2 코어 3.1GHz AMD EPYC™ CPU 및 4GB 메모리로 구성된 인스턴스를 사용하였다. 실험을 위해 마스터 노드와 워커 노드 각

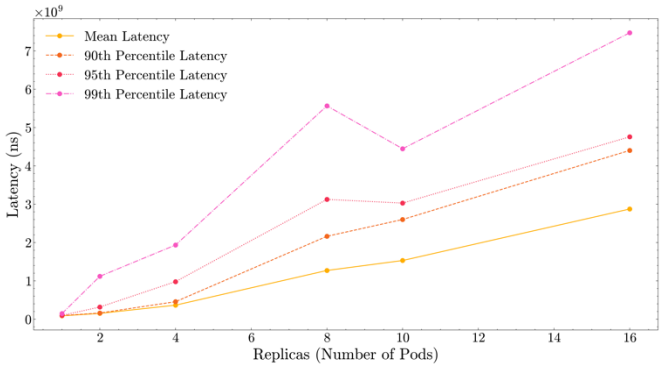
각 1 개로 구성하여 쿠버네티스 클러스터를 구축하였다. 실험 평가는 <표 1>과 같이 물리 노드의 1 코어 CPU 리소스에 대해 레플리카 수를 변화시켜 CPU 자원을 공정하게 배분하여 자원 세분성에 따른 시스템 성능을 평가하였다.

<표 1> 레플리카 수에 따른 파드당 CPU 할당 자원

Replicas	파드 1 개의 CPU 자원(core)
1	1
2	0.5
4	0.25
8	0.125
10	0.1
16	0.063

애플리케이션은 Apache 웹 서버를 기반으로, CPU 집약적 워크로드인 삼각 함수 계산을 수행한다. 부하 생성기를 사용하여 애플리케이션에 총 300 초에 걸쳐 분당 300 개의 HTTP 요청을 보낸다.

3.2 동일 작업 요청 수에서 레플리카 수에 따른 레이턴시 및 CPU 사용량 분석

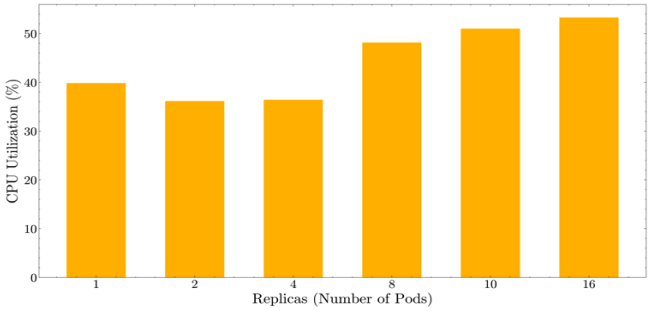


[그림 1] 자원 세분성에 따른 서비스 레이턴시

[그림 1]은 <표 1>에 따른 서비스 응답 지연시간을 보여준다. 실험 결과, 제한된 CPU 자원을 더 많은 레플리카들이 사용할 수록 레이턴시가 증가하는 추세를 확인할 수 있다. 구체적으로, 레플리카의 수가 1 과 2 일 때를 비교해보면, 95, 99 번째 레이턴시는 각각 220.58%, 660.56% 증가했다. 그러나, 레플리카의 수가 8 과 10 일 때는 비슷하거나 감소하는 추세가 보인다. 그럼에도 불구하고, 평균 레이턴시를 보면 레플리카의 수가 1 개일 때 대비 172.03%, 416.39%, 1446.61%, 1742.16%, 3270.99% 증가하여 전반적으로 서비스 품질이 저하되는 것을 확인할 수 있다.

이 현상은 프로세스 간 간섭과 함께, 워크로드의 특성도 반영된다. 컨테이너의 오버헤드가 FPU FFT 연산과 같은 CPU 집약적인 애플리케이션의 경우 레이턴시 오버헤드가 높은 경향을 보이기 때문이다[4].

[그림 2]는 부하 요청을 받는 5 분 동안의 레플리카의 평균 CPU 사용률을 나타낸다. 자원이 더 세분화될수록 부하를 처리하기 위한 CPU 자원이 더 필요하다는 것을 나타낸다. 자원에 여유가 있는 환경에서는 CPU 의 효율적인 사용을 이끌어내지만, 제한된 자원 환경에서는 오버헤드 증가와 자원 경합으로 응답 속도 감소를 유발할 수 있다.



[그림 2] 자원 세분성에 따른 CPU 사용량

4. 결론

본 연구는 자원이 제한된 쿠버네티스 환경에서 세분화에 따른 서비스 응답시간 성능 영향을 분석하였다. 클러스터 자원의 세분성이 높아질수록 응답시간이 증가하였고 더 많은 자원을 사용했다. 이러한 결과는 많은 파드가 배포됨에 따라 오버헤드가 증가하고, 독립적인 여러 파드의 자원 경합이 발생하여 자원 요구량이 늘어난 것으로 보인다.

자원 세분성은 서비스의 목적에 따라 자원 할당 전략을 결정하는 중요한 요소이다. 응답시간이 중요한 서비스는 자원 세분성을 줄여 빠른 처리 속도를 유지해야 하며, 자원 효율성이 중요한 서비스는 자원 세분화를 통해 자원 낭비를 최소화할 수 있다. 이를 통해 클라우드 환경에서 서비스 운영 목적에 따른 자원 세분화 전략을 형성할 수 있다.

Acknowledgment

본 연구는 2024 년 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학사업 지원을 받아 수행되었음(2023-0-00044)

참고문헌

[1] Saedi, Amin, and Iahad, Noorminshah A. "An Integrated Theoretical Framework for Cloud Computing Adoption by Small and Medium-Sized Enterprises." PACIS 2013 Proceedings, Jeju Island, Korea, 2013, Paper 48.

[2] GARG, Surya Kant; LAKSHMI, J. Workload performance and interference on containers. In: *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017. p. 1-6.

[3] XU, Yunjing, et al. Bobtail: Avoiding long tails in the cloud. In: *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. 2013. p. 329-341.

[4] LI, Zheng, et al. Performance overhead comparison between hypervisor and container based virtualization. In: 2017 IEEE 31st International Conference on advanced information networking and applications (AINA). IEEE, 2017. p. 955-962.