스파이크 트래픽 대응을 위한 쿠버네티스 In-Place Pod Resize 를 활용한 리소스 사용 관측 간격에 따른 지연 및 부하 트레이드오프 분석

이수연, 정윤원*
숭실대학교

cocomatcha@soongsil.ac.kr, *ywchung@ssu.ac.kr (교신저자)

Delay-Overhead Trade-off Analysis based on Resource Usage Observation Intervals using Kubernetes In-Place Pod Resize for Reacting Spike Traffic

Su Yeon Lee, Yun Won Chung
Soongsil University

요 약

본 논문에서는 스파이크 트래픽 환경에서 효과적인 대응을 위해 지연과 부하의 가중합으로 정의된 비용을 정의하고, 쿠버네티스 In-Place Pod Resize 를 활용하여 리소스 사용 관측 간격에 따른 지연 및 부하 간 트레이드오프를 분석한다. 실험 결과는 지연과 부하에 설정된 가중치에 따라 비용을 최소화하는 적절한 수집 주기가 존재함을 보여준다.

Ⅰ. 서 론

가상화된 리소스를 사용하는 클라우드 컴퓨팅에서는 서비스 수요에 따라 리소스를 할당하여 서비스의 안정성을 확보하면서도 운영 비용을 절감하는 것이 중요하다. 대표적인 리소스 관리 오케스트레이션 툴인 쿠버네티스(Kubernetes)는 컨테이너화된 애플리케이션의 배포, 확장 및 자동화 기능을 효과적으로 제공하여 널리 사용되고 있다[1]. 쿠버네티스는 변화하는 부하에 효과적으로 대응하기 위해 HPA (Horizontal Pod Autoscaler) 및 VPA (Vertical Pod Autoscaler) 기능을 제공한다. HPA 는 부하 증가 시 파드(Pod)의 개수를 늘려 부하에 대응하는 수평 확장 방식으로 상태가 없는(stateless) 애플리케이션에 적합하지만, 새로운 파드가 생성되고 준비되는 데 많은 시간이 소요되는 단점이 있다[2]. 반면, VPA 는 개별 파드에 할당된 CPU 나 메모리 사양을 직접 조절하는 수직 확장 방식으로 파드의 개수를 유지하면서 성능을 높일 수 있어 HPA 와 같은 파드 생성 지연 없이 리소스를 보강할 수 있는 장점이 있다.

HPA 및 VPA 와 관련한 기존 연구에서는 HPA 효율화, 최적화에 대한 연구가 주로 진행되었으며[3],[4], 최근에는 VPA 성능 향상을 위해 강화학습이나[5] 시계열 분석 모델인 LSTM[6] 등을 적용한 연구도 진행되었다. 그러나, 이러한 연구에서는 연산 부하가 크고, 리소스 사용 수집 경로에서의 지연으로 인해 초 단위의 급격한 스파이크 트래픽에 즉각 대응하기에는 한계가 있다.

기존 쿠버네티스의 VPA 에서는 리소스 사양 변경 시 파드를 반드시 재시작해야 하는 구조적 한계가 있어 2023 년 4 월 쿠버네티스 v1.27[7]에서 In-Place Pod Resize Alpha 버전을 공개하였으며 이를 통해 파드의 삭제나 재생성 없이 실행 중인 상태에서 CPU 및 메모리의 요청(Request)과 제한(Limit) 수치를 즉시 수정할 수 있게 되었다. 이후, 지속적인 고도화를 거쳐 2025 년 12 월 v1.35 에서 정식(Stable) 출시되었으며[8] v1.35 부터는 파드의 리소스 사양을 직접 수정하거나 새롭게 추가된 In-Place Pod Resize 기능을 호출하여 즉각적인 리소스 변경을 수행할 수 있게 되었다.

본 논문에서는 스파이크 트래픽 대응을 위해 v1.35 에서 정식으로 출시된 In-Place Pod Resize 기능을 활용하여 리소스 사용 관측 간격에 따른 지연 및 부하 간 트레이드오프(trade-off)를 분석하고자 한다. In-Place Pod Resize 이전에도 리소스 사용량 관측 간격은 매우 짧은 값으로 설정 가능하였으나 관측에 따른 즉각적인 리소스 조정이 불가하여 스파이크 트래픽에 실질적인 대응이 어려웠던 반면, In-Place Pod Resize 를 활용하는 경우 신속한 대응이 가능한 장점이 있다. 다만, 리소스 관측 간격이 감소하는 경우 CPU 부하가 증가하는 단점이 있어 대응 지연 및 CPU 부하를 동시에 고려한 적절한 관측 간격의 설정이 중요하다. 이를 위해 본 논문에서는 지연과 부하의 가중합인 비용을 정의하고 다양한 가중치 환경에서 지연 및 부하 간 트레이드오프를 실험을 통해 분석한다.

Ⅱ. 지연 및 부하 트레이드오프 분석

본 논문에서는 리소스 사용량 관측 간격에 따른 지연 및 부하 간 트레이드오프 분석을 위해 아래 식 (1)과 같이 지연 및 부하의 가중합으로 구성된 비용을 정의한다. 지연은 부하 발생 관측 시점부터 자원 할당 완료시까지의 시간에 해당하고, 부하는 리소스 사용량

관측을 위한 CPU 사용량에 해당한다. w_{delay} 및 $w_{overhead}$는 각각 지연 및 부하의 가중치에 해당한다.

$$Cost = w_{delay} \cdot Delay + w_{overhead} \cdot Overhead \quad (1)$$

본 논문에서는 그림 1 과 같은 단계를 통해 스파이크 트래픽을 감지하고 리소스를 조절한다. 본 논문의 실험 환경에서는 기본 메트릭 서버를 경유하여 누적 지연이 발생하는 방식 대신 컨트롤러가 전용 수집기에서 지표를 직접 확보하는 3 단계 구조를 통해 경로를 최적화하여 실시간성을 확보하였다. 이를 위해, Kubelet 에 내장되어 긴 주기로 데이터를 수집하는 기존 수집기 대신 t 초 주기로 동작하는 전용 cAdvisor 를 별도로 배포함으로써 급격한 리소스 변화에 즉각 대응할 수 있는 고속 수집 기반을 구축하였다. 또한 리소스 분석과 결정 단계가 개별 컴포넌트로 분리되어 의사결정이 지연되던 기존의 구조적 한계를 단일 컨트롤러 내 통합 처리 방식으로 개선하여 전체적인 리소스 조절 속도를 향상시킬 수 있었다.

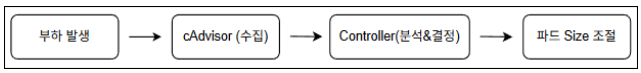

그림 1. 분석 환경

본 실험은 8 vCPU 사양의 워커 노드 2 대로 이루어진 쿠버네티스 클러스터에서 진행하였다. 200m core 가 할당된 웹 서비스 파드 10 개를 배포한 후, 급격한 CPU 과부하 상황을 부여하였다. 이러한 부하 상황에서 리소스 수집 주기(t)의 변화가 시스템에 미치는 영향을 파악하고자, 독립적으로 구축된 cAdvisor 의 housekeeping_interval 을 1 초부터 100 초까지 조정하며 실험을 진행하였다.

그림 2 는 본 실험을 통해 측정한 수집 주기의 변화에 따른 비용을 나타낸다. 다양한 가중치 조합이 비용에 미치는 영향을 분석하기 위해 w_{delay} 의 값을 1 로 고정한 환경에서 $w_{overhead}$ 의 값을 다양하게 변화시키면서 비용을 측정하였다. 실험 결과, $w_{overhead}$ 가 10 이하인 경우 지연이 비용에서 더 큰 비중을 차지하여 수집 주기가 증가할수록 비용도 증가하는 것을 알 수 있다. 반면, $w_{overhead}$ 가 30 이상인 경우 비용에서 부하의 비중도 증가하여 수집 주기가 짧은 구간에서 수집 주기가 감소하는 경우 부하 증가의 영향으로 인해 비용이 증가하고, 수집 주기가 긴 구간에서는 수집 주기가 증가하는 경우 지연 증가의 영향으로 인해 비용이 증가하며, 최소의 비용을 가지는 적절한 수집 주기가 존재하는 것을 알 수 있다. 지연에 대한 가중치는 동적 트래픽에 대응하기 위한 서비스 운영 정책에 따라 결정될 수 있는 값이며, 부하에 대한 가중치는 고려하는 수집 주기 환경에서 서비스 제공을 위해 사용되는 자원에 따라 결정될 수 있는 값으로, 고려하는 서비스 환경에서 적절한 지연 및 부하의 가중치 값이 결정되는 경우 비용을 최소화하는 수집 주기 또한 적절히 결정될 수 있음을 알 수 있다.

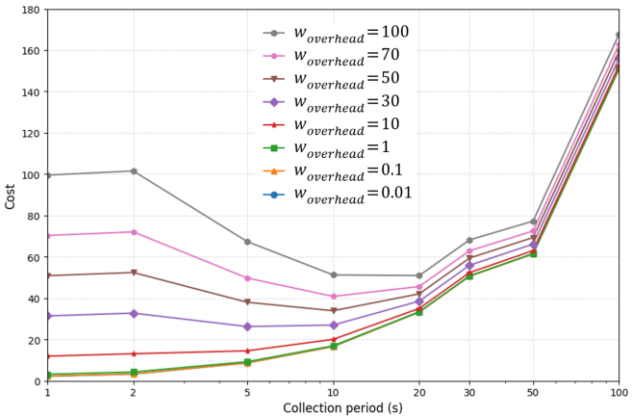

그림 2. 수집 주기의 변화에 따른 비용

Ⅲ. 결론 및 추후 연구

본 논문에서는 쿠버네티스 In-Place Pod Resize 기능을 활용하여 리소스 사용량 관측 주기에 따른 지연과 부하 간의 트레이드오프를 지연과 부하의 가중합으로 정의된 비용을 통해 분석하였다. 분석 결과 지연과 부하의 가중치에 따라 최소 비용의 관점에서 적절한 수집 주기가 존재함을 확인할 수 있었다. 추후 연구로는 스파이크 트래픽 이외의 다양한 입력 트래픽 유형에 따른 적절한 수집 주기에 대한 연구를 진행할 계획이다.

참 고 문 헌

[1] https://kubernetes.io/docs/concepts/overview/

[2] L. Larsson, et al., “Experimental Evaluation of Kubernetes Autoscaling Mechanisms”, 2020 IEEE International Conference on Cloud Engineering (IC2E), April, 2020.

[3] B. Serracanta, et al., “On the Stability of the Kubernetes Horizontal Autoscaler Control Loop”, IEEE Access, vol. 13, pp. 7160-7166, January 2025.

[4] D. Kim, et al., “LARE-HPA: Co-optimizing Latency and Resource Efficiency for Horizontal Pod Autoscaling in Kubernetes”, ICSOC’24, December, 2024.

[5] S. H. Baek, “Reinforcement Learning-based Vertical Pod Autoscaling (VPA) Technique”, MS Thesis, Korea University. August 2024.

[6] S. C. Kim, “Bi-LSTM-based Adaptive Vertical Pod Autoscaling Technique for Efficient Cloud Computing Resource Utilization”, MS Thesis, Dongguk University. 2022.

[7]https://kubernetes.io/blog/2023/05/12/in-place-pod-resize-alpha/

[8]https://kubernetes.io/blog/2025/12/19/kubernetes-v1-35-in-place-pod-resize-ga/