Kubernetes 마이크로서비스의 리소스 효율성을 위한 예측 기반 오토스케일링 기법

정현수[1], 길준민[2*]

[1,2]제주대학교 컴퓨터공학과

[1]jhs990909@stu.jejunu.ac.kr, [2]jmgil@jejunu.ac.kr

A Prediction-Based Autoscaling Technique for Resource Efficiency in Kubernetes Microservices

Hyunsu Jeong[1], Joon-Min Gil[2*]

[1,2]Dept. of Computer Engineering, Jeju National University

요 약

현대 클라우드 네이티브 환경에서 애플리케이션의 확장성과 자원 최적화는 서비스 품질과 안정성을 좌우한다. 이를 지원하는 쿠버네티스(Kubernetes)는 HPA(Horizontal Pod Autoscaler)를 통해 부하 변화에 따라 파드 수를 자동 조정하지만, 반응형 설계 특성으로 인해 콜드 스타트(cold start)로 인한 지연과 성능 저하가 발생한다. 본 논문에서는 이러한 한계를 극복하기 위해 LSTM(Long Short-Term Memory) 기반 예측 모델을 활용하여 미래 CPU 사용량을 사전에 추정하고 이를 오토스케일링 파이프라인에 반영하는 방식을 제안한다. 예측 결과는 프로메테우스(Prometheus)와 KEDA(Kubernetes Event-Driven Autoscaler)를 통해 HPA로 전달되며, 서비스 부하가 발생하기 전에 파드를 선제적으로 확장할 수 있도록 한다. Online Boutique 마이크로서비스 애플리케이션과 Alibaba Cluster Trace 2017 데이터셋을 기반으로 수행한 실험에서 제안 기법은 기존 HPA 대비 90번째 및 95번째 백분위수에서 지연을 줄이고 요청 실패율을 감소시키며, 불필요한 자원 과잉 확장을 억제하는 효과를 보인다. 이를 통해 클라우드 네이티브 환경에서 마이크로서비스의 성능, 안정성, 자원 효율성을 동시에 확보한다.

Ⅰ. 서 론

현대 클라우드 네이티브 환경에서는 애플리케이션의 탄력성(elasticity), 확장성(scalability)과 자원 최적화를 통해 서비스 품질을 높인다. 클라우드 네이티브 환경 중 하나인 쿠버네티스(Kubernetes)는 선언적 자원 관리와 운영에 필요한 기본 기능을 제공하여 대규모 마이크로서비스를 안정적으로 운영한다. 쿠버네티스에서 파드(pod)는 컨테이너(container)를 실행하는 기본 단위로, 하나 이상의 컨테이너와 이를 실행하는 데 필요한 네트워크, 스토리지 등을 포함하는 가장 작은 컴퓨팅 단위이다. 쿠버네티스는 이러한 파드의 개수를 동적으로 조정하여 애플리케이션의 부하 변화에 대응한다. 특히, 복제본(replica)은 동일한 애플리케이션을 실행하는 다수의 파드로, 부하가 증가할 때 여러 개의 파드를 배포하여 서비스의 가용성을 유지하고 성능 저하를 방지하는 역할을 한다. 이러한 자동 확장을 담당하는 핵심이 쿠버네티스에서 HPA(Horizontal Pod Autoscaler)이며, 메트릭 서버(metric server)로부터 수집된 CPU, 메모리 사용량 등의 자원 지표를 기반으로 목표 임계치와 비교하여 파드 개수를 동적으로 조정하는 역할을 한다. 그러나 근본적으로 반응형 방식으로 동작하기 때문에, 새 파드가 생성되어 준비되는 과정에서 콜드 스타트(cold start) 문제가 필연적으로 발생하며, 이로 인해 순간적인 사용자 응답 지연이나 요청 실패율 증가가 나타난다[1]. 이러한 문제를 해결하기 위해 본 논문에서는 LSTM(Long Short-Term Memory) 기반의 예측 모델을 활용하여 미래의 CPU 사용량을 사전에 추정하고, 이를 쿠버네티스 오토스케일링 파이프라인에 반영한다. 본 논문의 제안 기법은 예측기를 통해 산출된 미래 부하 정보를 프로메테우스(Prometheus)와 KEDA(Kubernetes Event-Driven Autoscaler)를 통해 HPA로 전달하여, 부하가 발생하기 전에 파드를 선제적으로 확장할 수 있도록 한다. 이를 통해 서비스 지연을 최소화하고, 불필요한 자원 소모를 줄이며, 마이크로서비스의 안정성과 효율성을 동시에 확보한다.

Ⅱ. 제안 기법

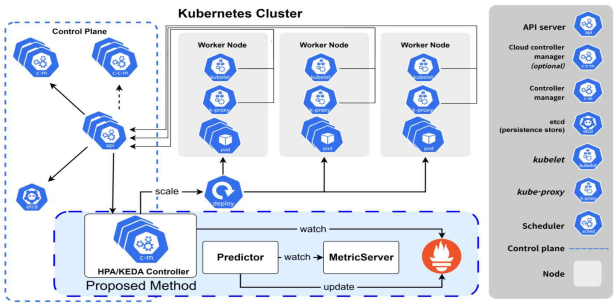

그림 1 예측 기반 오토스케일링을 위한 쿠버네티스 아키텍처

본 논문에서는 그림 1과 같이 쿠버네티스 환경에서의 예측 기반 오토스케일링 기법을 제안한다. 제안하는 기법은 메트릭 수집 단계에서 시작된다. 메트릭 서버는 각 디플로이먼트(deployment)의 CPU 사용량을 15초 주기로 수집해 쿠버네티스 API로 제공하며, 본 논문에서는 이렇게 수집된 연속된 값을 누적하여 최근 10분간의 시계열 데이터를 구성한다. 이 시계열 데이터는 LSTM 예측 모델의 입력으로 사용되며, 본 논문에서 구현한 예측기(predictor)에서 이를 기반으로 향후 15초 이후의 CPU 부하를 추정한다. 산출된 결과는 프로메테우스가 인식할 수 있는 메트릭 형식으로 변환되어 전달된다. 프로메테우스는 수집된 메트릭을 자체 시계열 데이터베이스에 저장하고 PromQL을 통해 질의할 수 있으며, 프로메테우스에서 수집한 예측된 메트릭 값들은 KEDA의 입력으로 전달된다. KEDA는 프로메테우스와 같은 외부 이벤트 소스를 활용하여 HPA를 동적으로 생성·관리할 수 있는 컴포넌트이다. 즉, KEDA는 프로메테우스에서 수집한 예측 메트릭을 기반으로 각 디플로이먼트에 대응하는 HPA를 자동으로 생성하고 해당 HPA는 실제 부하가 발생하기 전에 파드 복제본(replica) 수

표 1. k6 부하 테스트 결과: 예측 기반 오토스케일링 기법과 HPA의 비교

방식	frontend 누적 복제본 수	currencyservice 누적 복제본 수	전체 누적 복제본 수 합계	평균 응답 시간	p(90)	p(95)	요청 실패율
제안 기법	16,807	21,231	47,977	135.85ms	209.63ms	301.97ms	0.0064%
HPA	20,618	18,893	49,521	134.69ms	210.04ms	320.12ms	0.0092%

를 선제적으로 조정한다. 이 과정을 통해 마이크로서비스는 동시에 확장되며, 기존 HPA의 반응형 한계로 인해 발생하는 콜드 스타트 지연을 줄이고 불필요한 과잉 확장을 방지한다.

Ⅲ. 성능 평가

3.1 실험 환경 구성

제안한 오토스케일링 기법의 성능을 검증하기 위해서는 실제 워크로드 환경에서 부하 상황을 재현하는 과정이 필요하다. 본 논문에서는 Alibaba Cluster Trace 2017 데이터셋을 활용하여 부하 시나리오를 설계한다. 해당 데이터셋은 실제 대규모 클라우드 환경에서 수집된 워크로드를 포함하고 있다. 수집된 데이터셋은 학습과 테스트 평가 목적으로 8:2 비율로 분할하여 LSTM 기반 예측 모델의 훈련과 평가에 활용한다. 또한, 본 논문에서는 데이터셋 기반의 워크로드 패턴을 쿠버네티스 클러스터 상에서 재현하기 위해 k6[2] 부하 테스트 도구를 사용한다.

실험을 위해 사용한 마이크로서비스 애플리케이션은 Google이 공개한 Online Boutique Architecture[3]이다. 이는 전자상거래 플랫폼을 모사하도록 설계된 클라우드 네이티브 애플리케이션으로, frontend, productcatalogservice, cartservice, currencyservice, paymentservice 등 다수의 마이크로서비스로 구성된다. 이러한 구조는 서비스 간 의존성과 트래픽 전파 효과를 반영하므로, 제안 기법의 성능 검증에 적합하다.

학습 데이터셋을 기반으로 부하 패턴을 분석한 결과, 웹페이지 접속에 따른 실제 부하는 주로 frontend와 currencyservice에서 집중적으로 발생하였다. 따라서, 본 논문에서는 이 두 서비스에 대해서만 LSTM 기반 예측 모델을 적용하여 사전 확장을 수행하며, 나머지 마이크로서비스들은 기존 HPA 방식을 적용한다. 실험은 두 가지 방식으로 비교 수행한다. 첫 번째 방식은 제안한 예측 기반 오토스케일링 기법에 적용되는 frontend와 currencyservice는 LSTM 모델 예측 결과를 반영하여 선제적으로 스케일링하고 다른 서비스는 HPA에 의해 확장되는 방식이다. 두 번째 방식은 모든 서비스가 HPA에 의해 확장되는 기존 방식이다. 각 파드는 CPU 요청량(request) 100 millicore와 상한(limit) 200 millicore 조건으로 구성되었으며, 이를 기반으로 스케줄링과 자원 사용이 이루어진다. 오토스케일링의 임계 조건은 두 가지 방식 모두에서 동일하게 80%로 설정함으로써, 기존 HPA 방식과 제안하는 예측 기반 오토스케일링 방식 간 성능을 공정하게 비교할 수 있도록 한다.

3.2 성능 결과

제안 기법의 성능을 측정하기 위해 본 논문에서는 누적 복제본 수, 응답 시간(평균, 90번째 백분위수, 95번째 백분위수), 요청 실패율을 주요 지표로 설정하였다. 이때 응답 지연 시간은 k6에서 측정한 request duration으로, 이는 클라이언트 요청이 전송된 시점부터 응답이 완전히 수신될 때까지 걸린 총 소요 시간을 의미한다. 표 1은 제안한 LSTM 기반 예측 오토스케일링 방식과 기존 HPA 방식의 성능 비교 결과를 보여준다. 또한, frontend 누적 복제본 수와 currencyservice 누적 복제본 수는 k6 부하 테스트 시나리오 실행 시간 동안 15초 간격으로 측정된 복제본 수를 합산한 값이며, 시간에 따른 파드 사용량을 누적으로 반영하는 지표이다. 실험 결과, 제안한 예측 기반 오토스케일링 기법은 frontend에서 불필요하게 과잉 확장되는 현상을 줄이고 currencyservice와의 확장을 보다 균형 있게 수행한다. 이로 인해 테스트 시나리오 구간에서의 전체 누적 복제본 수 합계는 기존 HPA 대비 약 3.12% 감소한다. 또한 응답 지연 측면에서도 90번째 및 95번째 백분위수에서 예측 기반 오토스케일링 방식 기법이 개선 효과를 보이며, 요청 실패율 역시 0.0064%로 HPA의 0.0092%보다 낮게 나타나 서비스 안정성이 향상되었음을 확인할 수 있다.

Ⅳ. 결론

본 논문에서는 쿠버네티스 환경에서 HPA의 반응형 한계와 콜드 스타트 문제를 해결하기 위해, LSTM 기반 예측 모델을 활용한 오토스케일링 기법을 제안한다. 제안 기법은 CPU 사용량을 사전에 추정하고 이를 프로메테우스와 KEDA를 통해 HPA로 전달함으로써, 실제 부하 발생 이전에 파드를 선제적으로 확장할 수 있다. Online Boutique 마이크로서비스 애플리케이션을 활용한 실험 결과는 제안 기법의 효과를 뚜렷하게 보여준다. Online Boutique은 사용자가 웹 브라우저를 통해 frontend 서비스에 접속하면 frontend가 내부적으로 여러 마이크로서비스에 요청을 전달하는 구조를 가진다. 특히 상품 가격 표시 과정에서 frontend는 gRPC를 통해 currencyservice에 환율 변환 정보를 요청하고 그 결과를 받아 최종 응답을 구성한다. 이러한 구조적 특성으로 인해 외부 요청이 증가하면 부하가 우선 frontend에 집중되며, currencyservice는 frontend의 호출을 통해서만 부하가 발생하므로 확장이 상대적으로 늦게 이루어진다. 기존 HPA 방식에서는 frontend가 먼저 급격히 확장되고 currencyservice는 뒤늦게 확장되면서 병목이 발생한다. 반면, 본 논문의 제안 기법은 frontend와 currencyservice를 동시에 예측·확장함으로써 이러한 구조적 한계를 완화하고 특정 서비스에만 과도하게 부하가 몰리는 현상을 줄이며 서비스 간 확장을 균형 있게 수행한다.

ACKNOWLEDGMENT

본 과제(결과물)는 2025년도 교육부 및 제주도의 재원으로 제주RISE센터의 지원을 받아 수행된 지역혁신중심 대학지원체계(RISE)의 결과입니다(2025-RISE-17-001).

참 고 문 헌

[1] A. P. Jegannathan, R. Saha and S. K. Addya, “A Time Series Forecasting Approach to Minimize Cold Start Time in Cloud-Serverless Platform,” 2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Sofia, Bulgaria, 2022.

[2] B. Serracanta, A. Lukács, A. Rodriguez-Natal, A. Cabellos and G. Rétvári, “On the Stability of the Kubernetes Horizontal Autoscaler Control Loop,” in IEEE Access, vol. 13, pp. 7160-7166, 2025.

[3] H. Ahmad, C. Treude, M. Wagner and C. Szabo, “Smart HPA: A Resource-Efficient Horizontal Pod Auto-Scaler for Microservice Architectures,” 2024 IEEE 21st International Conference on Software Architecture (ICSA), Hyderabad, India, 2024