KIPS Trans. Comp. and Comm. Sys.
Vol.11, No.4 pp.105~112
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2022.11.4.105

Transfer Learning Technique for Accelerating Learning of Reinforcement Learning-Based Horizontal Pod Autoscaling Policy

Yonghyeon Jang† · Heonchang Yu†† · SungSuk Kim†††

ABSTRACT

Recently, many studies using reinforcement learning-based autoscaling have been performed to make autoscaling policies that are adaptive to changes in the environment and meet specific purposes. However, training the reinforcement learning-based Horizontal Pod Autoscaler(HPA) policy in a real environment requires a lot of money and time. And it is not practical to retrain the reinforcement learning-based HPA policy from scratch every time in a real environment. In this paper, we implement a reinforcement learning-based HPA in Kubernetes, and propose a transfer leanring technique using a queuing model-based simulation to accelerate the training of a reinforcement learning-based HPA policy. Pre-training using simulation enabled training the policy through simulation experience without consuming time and resources in the real environment, and by using the transfer learning technique, the cost was reduced by about 42.6% compared to the case without transfer learning technique.

Keywords : Kubernetes, Reinforcement Learning, Autoscaling

강화학습 기반 수평적 파드 오토스케일링 정책의 학습 가속화를 위한 전이학습 기법

장 용 현† · 유 헌 창†† · 김 성 석†††

요 약

최근 환경의 변화에 적응적이고 특정 목적에 부합하는 오토스케일링 정책을 만들기 위해 강화학습 기반 오토스케일링을 사용하는 연구가 많이 이루어지고 있다. 하지만 실제 환경에서 강화학습 기반 수평적 파드 오토스케일러(HPA, Horizontal Pod Autoscaler)의 정책을 학습하기 위해서는 많은 비용과 시간이 요구되며, 서비스를 배포할 때마다 실제 환경에서 강화학습 기반 HPA 정책을 처음부터 다시 학습하는 것은 실용적이지 않다. 본 논문에서는 쿠버네티스에서 강화학습 기반 HPA를 구현하고, 강화학습 기반 HPA 정책에 대한 학습을 가속화하기 위해 대기행렬 모델 기반 시뮬레이션을 활용한 전이 학습 기법을 제안한다. 시뮬레이션을 활용한 사전 학습을 수행함으로써 실제 환경에서 시간과 자원을 소모하며 학습을 수행하지 않아도 시뮬레이션 경험을 통해 정책 학습이 이루어질 수 있도록 하였고, 전이 학습 기법을 사용함으로써 전이 학습 기법을 사용하지 않았을 때보다 약 42.6%의 비용을 절감할 수 있었다.

키워드 : 쿠버네티스, 강화학습, 오토스케일링

1. 서 론

쿠버네티스는 탄력성을 위해 수평적 파드 오토스케일러(HPA, Horizontal Pod Autoscaler)를 통해 수평적 파드 오토스케일링 기능을 지원한다[1]. HPA는 임곗값 기반 오토스케일링을 수행하며, 워크로드와 애플리케이션의 성능을 고려해서 목적에 맞는 적절한 임곗값을 설정해야 한다. 예를 들어, SLA이 엄격해서 자원을 더 사용해서라도 SLA 위반을 최소화하고 싶다면 임곗값을 낮게 설정해서 워크로드의 변화에 민감하게 반응하게 하여 목표를 달성할 수 있고, 반대로 자원 비용이 너무 비싼 경우에는 임곗값을 높게 설정해서 최소한의 자원만을 사용하게 할 수 있다. 이때, 목적에 맞는 적절한 임곗값을 설정하기 위해서는 많은 실험을 통한 튜닝 과정이 요구된다. 또한, 워크로드의 성격과 애플리케이션의 처리 능력이 변화함에 따라 동일한 임곗값이라도 다른 성능을 보이기 때문에 고정된 임곗값은 환경에 적응적이지 못한다는 문

※ 이 논문은 정부(과학기술정보통신부, 교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019M3E7A1113102).
† 준 회 원 : 고려대학교 컴퓨터학과 석사과정
†† 종신회원 : 고려대학교 컴퓨터학과 교수
††† 종신회원 : 서경대학교 소프트웨어학과 교수
Manuscript Received : November 5, 2021
Accepted : November 22, 2021
* Corresponding Author : SungSuk Kim(sskim03@skuniv.ac.kr)

제를 가지며, 최적의 HPA 정책을 설정하는 것을 더 어렵게 만든다.

그래서 환경에 적응적인 최적의 HPA 정책을 만들기 위해 강화학습을 사용하는 연구가 많이 진행되고 있다[2-5]. 강화학습을 사용하면 어떻게 목적에 도달할지보다는 어떤 목적에 도달할지에 집중해서 정책을 학습할 수 있다는 장점을 가지며, 온라인 학습을 통해서 환경에 적응적인 정책을 학습할 수 있다. 하지만 강화학습을 사용해서 정책을 학습하는 과정을 실제 운영 환경에서 진행하게 될 경우 실제로 발생된 워크로드를 바탕으로 학습이 수행되어야 하기 때문에 많은 시간과 자원이 요구된다. 대기행렬 기반 시뮬레이터를 사용하면 수집된 워크로드 기록과 측정된 애플리케이션의 성능을 바탕으로 파드의 CPU 이용률과 만료되는 요청 수를 시뮬레이션 할 수 있으며, 시뮬레이션을 통해 강화학습 기반 HPA 정책 학습을 수행함으로써 시간과 자원을 절약할 수 있다. 또한, 시뮬레이션을 통해 사전 학습된 정책을 사용하여 전이 학습함으로써 환경에 적응적인 정책을 학습하기 위한 온라인 강화학습 기반 HPA 정책의 학습을 가속화할 수 있다.

본 연구에서는 강화학습 기반 HPA를 구현하고, 정책 학습을 가속화하여 강화학습 기반 HPA를 실제 운영 환경에서 효율적으로 사용하기 위한 전이 학습 기법을 제안한다. 그리고 시뮬레이션을 통해 사전 학습된 정책을 전이함으로써 정책 수렴을 가속화하고 이를 통해 최대 42.6%의 총 운영비용을 절약하였다.

본 논문의 구성은 다음과 같다. 2장에서는 쿠버네티스 HPA와 강화학습에 대한 배경 지식을 소개하고, 3장에서는 강화학습 기반 파드 오토스케일링에 관련된 연구들을 소개한다. 4장에서는 전이 학습을 통한 강화학습 가속화 기법을 설명하고, 5장에서는 강화학습 기반 HPA를 실제로 구현하고, 전이 학습을 통한 총 운영비용 감소를 실험을 통해 보인다. 6장에서는 결론 및 향후 연구 과제를 제시한다.

2. 배경 지식

2.1 쿠버네티스 HPA

쿠버네티스의 HPA[1]는 임곗값을 기반으로 반응적인 오토스케일링을 수행하며, 오토스케일링 필요 여부를 결정하기 위한 scaling_tolerance와 임곗값에 해당하는 메트릭에 대한 target_value를 설정으로 가진다. target_value는 목표로 하는 메트릭의 값을 의미하며, desired # of pods = [current # of pods * (current_value / target_ value)]을 통해서 요구되는 파드 수를 결정한다. HPA는 산출된 요구되는 파드 수가 되도록 스케일링하는데, HPA를 구성할 때 설정한 최소 파드 수와 최대 파드 수의 범위를 넘지 않게 한다.

scaling_tolerance는 오토스케일링 필요 여부를 결정하는 용도로 사용되며, abs[(current_value/target_value) -1] 값이 scaling_tolerance보다 작을 때만 스케일링이 이루어지기 때문에 잦은 스케일링을 방지하는 부가적인 효과를 얻을 수 있다. 이 외에 stabilization 설정을 통해 스케일링을 수행한 이후에 특정 기간 동안 추가적인 스케일링 수행되지 않게 하는 설정 등, 사용자가 규칙을 추가할 수 있도록 한다. 이로 인해 더 정밀한 스케일링을 수행할 수 있지만, 최적의 정책을 구성하기 위해 고려해야하는 변수가 많아지기 때문에 복잡성이 높아진다. 본 논문에서는 이러한 복잡성을 해소하고, 환경에 적응적인 정책을 위해 강화학습을 기반으로 스케일링을 수행한다.

2.2 강화학습

강화학습은 의사 결정 문제를 해결하기 위한 방법으로 의사 결정과 관련된 다양한 분야에서 사용되고 있다. 강화학습은 환경에 대한 모델이 있는 model-based와 환경에 대한 모델이 없는 model-free로 나뉜다. model-based의 경우 환경의 메커니즘이 모델로써 활용되어 미래를 예측하는데 사용된다. 모델을 통해서 시뮬레이션 된 경험을 만들 수 있으며 이를 학습에 사용하여 정책을 빠르게 학습할 수 있다는 장점을 가지지만, 완벽한 모델이 요구되고 완벽한 모델을 만드는 것이 어렵다는 문제를 가진다. model-free의 경우 모델이 없어도 정책을 학습할 수 있다는 장점을 가지지만, 실제로 수행한 경험만을 사용해서 정책을 업데이트하기 때문에 model-based보다 학습이 느리다는 단점을 가진다[6].

본 논문에서는 쿠버네티스 클러스터의 상태에 대한 완벽한 모델링이 어렵기 때문에 model-free 방식을 사용한다. Q-Learning은 비활성 정책 시간차 방식을 사용하며, 활성 정책 시간차 방식을 사용하는 SARSA와 유사하다. 하지만 Q-Learning은 정책을 업데이트할 때 사용하는 정책과 행동을 수행했을 때 사용한 정책이 다른 비활성 정책을 사용한다는 점에서 SARSA와 차이가 있다[6]. 기존 연구에서 Q-Learning과 SARSA에 대한 시뮬레이션을 통한 성능 비교를 수행한 결과, 전반적으로 Q-Learning의 성능이 더 좋았기 때문에 본 논문에서는 Q-Learning을 사용한다.

Q-Learning에서 주기 t에서 Q-Table을 업데이트하는 방법은 Equation (1)과 같다.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (1)$$

하지만 오토스케일링 환경에서 현재 주기에서 측정되는 상태를 바탕으로 계산되는 보상은 이전 주기의 상태와 행동으로 인해 결정되기 때문에 기존 연구에서는 Equation (2)와 같이 주기 t에서 R_t를 구한 이후에 $Q(S_{t-1}, A_{t-1})$에 대한 업데이트를 수행했으며, 본 논문에서도 이와 동일하게 수행한다.

$$Q(S_{t-1}, A_{t-1}) \leftarrow Q(S_{t-1}, A_{t-1}) + \alpha[R_t + \gamma \max_a Q(S_t, a) - Q(S_{t-1}, A_{t-1})] \quad (2)$$

3. 관련 연구

[2]의 연구에서는 고정된 임곗값의 문제점을 해결하기 위해 강화학습을 사용했으며, Q-Threshold 알고리즘을 제안했다. Q-Threshold는 SLA를 보장하면서 자원 사용을 최소화할 수 있는 최적의 임곗값을 동적으로 제어한다. 하지만 임곗값 기반 오토스케일링을 사용한다는 한계를 넘지 못했다.

[3]의 연구에서는 학습 시간을 단축시키기 위해 근사된 모델을 사용하는 model-based 강화학습 알고리즘을 제안했으며, 도커 스웜 환경에서 수평적 파드 오토스케일링과 수직적 파드 오토스케일링을 동시에 고려해서 행동을 취한다. 그리고 제안한 model-based 알고리즘이 Q-Learning 알고리즘보다 목적을 더 잘 달성함을 보였다. 그러나 근사를 통해 만들어진 모델이 학습에 계속 사용되기 때문에 실제 환경이 아닌 근사된 모델에 편향된 정책이 학습된다는 문제점을 갖는다. 본 논문에서는 근사된 모델에 편향된 정책 학습을 방지하고, 실제 환경에 적응적인 정책을 학습하기 위해 실제 환경에서는 model-free 방식을 사용하는 Q-Learning을 그대로 사용한다.

[4]의 연구에서는 DDPG 알고리즘을 사용하는 오토스케일링 기법을 제안하였다. 그리고 각각의 마이크로서비스에 대응하는 수의 강화학습 기반 정책을 학습하기 위해 매번 새로운 정책을 생성하고 처음부터 다시 학습 하는 것이 실용적이지 않다고 주장했다. 그래서 일반적인 상황에 대해 학습된 정책을 기반으로 전이 학습을 수행하여 정책을 빠르게 학습할 수 있게 하였다. 본 논문에서는 수집된 워크로드 기록을 사용해서 시뮬레이션을 수행하며 학습된 정책을 기반으로 전이 학습을 시도하며, 조금 더 기존 워크로드에 특화된 정책을 이용해서 온라인 강화학습을 수행하려고 한다는 점에서 차이가 있다.

[5]는 강화학습 기반 오토스케일링과 관련된 연구를 조사하고 강화학습과 관련된 도전과제를 소개했다. model- based 방식은 완벽한 모델은 만드는 것이 어렵고, 모델이 변화하는 실제 환경을 반영하지 못하는 문제를 가지며, model-free 방식은 실제 경험만을 사용해서 학습을 수행하기 때문에 정책 수렴이 오래 걸리고, 학습 초기에 성능이 매우 떨어지는 문제를 가진다고 주장하였다. 본 논문에서는 model-based 방식의 모델과 관련된 문제를 해결함과 동시에 model-free 방식의 수렴 속도와 관련된 문제를 해결하기 위해 오프라인 강화학습에서는 시뮬레이션 경험을 기반으로 사전 학습을 수행하고, 온라인 강화학습에서는 시뮬레이션 경험을 사용하지 않고 실제 경험을 기반으로 학습을 수행하는 Q-Learning을 사용한다.

4. 전이 학습을 통한 강화학습의 학습 가속화

4.1 사전 학습을 위한 시뮬레이터

[7]은 환경에 대한 상태를 시뮬레이션하기 위해 대기행렬 모델로 환경을 모델링하여 실험에 사용했다. 본 논문에서는 [7]에서 제안하는 대기행렬 모델을 기반으로 하면서 시뮬레이션의 정확도를 높이기 위해 개선된 환경 모델을 사용한다. 개선된 환경 모델은 기존 연구에서 사용된 환경 모델과는 다르게 오토스케일링 주기와 시뮬레이터가 사용하는 시뮬레이션 주기를 분리하여 사용한다.

Table 1. Queuing Model-based Environment Configurations

Simulation Configurations	
- Application Profile(AP) - Workload History - Simulation Period	- Autoscaling Period - Timeout - Scaling Delay

이를 통해 기존 [7]의 모델에서는 할 수 없는 오토스케일링 주기보다 짧은 단위의 요구되는 응답시간을 설정할 수 있게 하여 0.1초 단위의 응답시간 요구 상황에 대해서도 만료되는 요청 수를 예측할 수 있게 한다. 또한, 파드가 생성되고 제거되면서 발생하는 스케일링 지연시간을 추가적으로 고려하여 실제 환경과 유사한 스케일링 결과를 시뮬레이션하도록 한다. 본 논문에서는 시뮬레이션 환경이 실제 환경과는 유사한 환경에 대해 사전 학습을 수행하고, 실제 운영 환경에서의 학습 가속화를 수행하는 용도로만 사용하기 때문에 시뮬레이션 정확도에 따른 학습 가속화 정도를 다루지 않는다.

개선된 대기행렬 기반 환경 모델은 강화학습 기반 HPA와 상호작용을 수행하며, 시뮬레이션을 통해 관측된 상태를 강화학습 기반 HPA에 제공하고, 강화학습 기반 HPA는 관측된 상태로부터 상태와 보상을 추출한다. 강화학습 기반 HPA는 이전 주기의 상태와 행동, 그리고 이번 주기에 얻은 보상을 사용해서 정책을 업데이트하고, 이번 주기에 얻은 상태는 이번 주기의 행동을 결정하는 용도로 사용한다. 시뮬레이션을 통해 사전 학습된 강화학습 기반 HPA의 정책은 이후에 실제 운영 환경에서 강화학습 기반 HPA의 초기 정책으로 설정되어 사용된다.

강화학습에서 환경의 역할을 수행하는 개선된 대기행렬 기반 환경 모델은 Table 1과 같은 구성을 가진다. Application Profile(AP)은 애플리케이션의 성능을 의미하며, 초당 파드 하나가 몇 개의 요청을 처리할 수 있는지를 나타낸다. 워크로드 기록은 시뮬레이션을 위해 사용되는 기존에 측정된 초당 요청 수에 대한 기록을 의미한다. 시뮬레이션 주기는 시뮬레이션 환경 모델이 사용하는 최소 시간 단위로 사용된다. 오토스케일링 주기는 강화학습 기반 HPA가 오토스케일링을 수행하는 주기이다. 타임아웃은 요구되는 응답시간을 의미하며, 본 논문에서는 실시간성이 요구되는 애플리케이션을 대상으로 하고 있기 때문에 타임아웃 이내에 처리되지 못하는 요청은 처리할 필요가 없다고 가정한다. 스케일링 지연시간은 스케일링 결정 이후에 실제로 스케일링이 적용되기까지 걸리는 지연시간을 의미한다.

4.2 강화학습 기반 HPA 설계

강화학습 기반 HPA의 설정을 위해 필요한 것으로는 최소

파드 수, 최대 파드 수, 상태 공간, 액션 공간, 보상 함수, 학습 알고리즘이 있다. 최소 파드 수와 최대 파드 수는 스케일링 범위를 정의하는 데 사용되며, 상태 공간, 액션 공간, 보상 함수는 강화학습을 위해 필요한 요소로 사용된다. 학습 알고리즘은 Q-Table을 업데이트하기 위해 사용되며, 2.2절의 Equation (2)와 같다.

1) 상태

강화학습의 상태는 파드 수와 파드의 애플리케이션 컨테이너의 평균 CPU 이용률을 사용한다. 기존 쿠버네티스 HPA에서는 평균 CPU 이용률만을 주로 사용하지만, [7]의 연구에서 파드 수가 늘어남에 따라 파드 하나의 처리 성능이 감소하는 점을 고려할 수 있도록 파드 수를 상태로 사용한다. 그리고 평균 CPU 이용률을 강화학습의 상태로 사용하기 위해 이산값으로 변환해서 사용한다. 예를 들어, [0, 10)을 0, [90, 100)을 9와 같이 변환하고, 평균 CPU 이용률이 100%인 경우 10으로 변환해서 사용한다.

2) 행동

행동은 고정된 크기의 행동 공간 $A \in \{-1, 0, 1\}$을 사용한다. -1은 파드 수를 1개 줄이는 것을, 1은 파드 수를 1개 늘리는 것을, 0은 아무 액션도 취하지 않는 것을 의미한다. 요구되는 파드 수를 행동으로 정의해서 스케일링할 경우 급격한 워크로드 변화에도 적절한 스케일링을 수행할 수 있겠지만, 가능한 파드 수의 범위가 커질수록 학습해야 하는 Q-Table의 크기가 급격하게 커지기 때문에 학습이 잘 되지 않는 문제가 발생한다. 그래서 본 논문에서는 행동 공간의 크기를 3으로 고정해서 사용한다.

3) 보상

강화학습은 미래의 보상을 최대화하는 행동을 취할 수 있는 정책을 학습하는 것에 있으며, 목적에 맞는 보상을 설정하는 것이 중요하다. 본 논문에서는 자원 사용 비용과 응답시간 초과로 인한 SLA 위반 페널티 비용의 합인 총 운영비용을 최소화하는 것을 목적으로 한다. 따라서 보상은 다음 주기 동안의 총 운영비용에 음수를 취한 값으로 설정한다. 해당 주기가 아닌 다음 주기의 총 운영비용으로 보상을 설정하는 이유는 해당 주기의 마지막에 행동을 결정하게 되고, 행동에 대한 영향은 다음 주기부터 발생하기 때문이다.

4) 행동 정책

Q-Table과 현재 상태를 기반으로 행동이 결정되며, 본 논문에서는 시뮬레이션과 실제 환경에서 사용하는 행동 정책이 다르다. 시뮬레이션에서는 ϵ확률로 임의의 행동을 취하고, $1-\epsilon$ 확률로 현재 상태에서 Q 값이 최대가 되는 행동을 선택해서 취하는 ϵ-greedy 행동 정책을 사용한다. 사전 학습된 정책을 사용하는 실제 환경에서는 사전 학습된 정책을 최대한 따르면서도 탐험적인 행동을 취할 수 있도록 노이즈를 추가한 행동 정책을 사용한다. 본 논문에서는 노이즈를 통해 실제 운영 환경에서는 90% 확률로 현재 상태에서 Q 값이 최대가 되는 행동을 취하고, 10% 확률로 Q 값이 최대가 되는 행동에 근접한 행동을 취하게 한다. 예를 들어, 10% 확률로는 결정된 행동이 -1이면, -1 또는 0 중에 하나를 임의로 선택하고, 결정된 행동이 0이면 -1, 0, 1 중에 하나를 임의로 선택한다. 이를 통해, 기존에 학습한 정책을 활용하면서도 탐험적인 학습을 수행하여 사전 학습된 정책에 비교적 덜 편향된 학습을 수행할 수 있다.

4.3 전이 학습 기법

전이 학습은 시뮬레이션으로 사전 학습된 Q-Table을 실제 쿠버네티스 환경에서의 강화학습 기반 HPA로 전이해서 진행되며, Fig. 1은 전체 과정을 묘사한다.

1) 사전 학습

사전 학습은 강화학습 기반 HPA의 초기 정책으로 사용될 Q-Table을 학습하고, 이를 통해 강화학습 기반 HPA의 학습을 가속화하는 것을 목표로 한다. 사전 학습은 실제 쿠버네티스 환경에서 트래픽을 발생시켜가며 학습을 시킬 수 있지만, 이는 너무 많은 시간과 자원을 소모하기 때문에 본 논문에서는 시뮬레이션 환경에서 사전 학습을 수행한다. 이를 통해, 5,000초의 워크로드에 대한 강화학습을 몇 초 이내에 끝낼 수 있으며, 시뮬레이션의 장점을 활용해서 여러 번 반복해서 학습을 수행하여 학습 효율을 높일 수 있다. Algorithm 1은 시뮬레이션을 통해 사전 학습을 수행하는 과정이며, 하나의 에피소드는 주어진 워크로드를 1회 시뮬레이션하는 것을 의미한다. 입력으로 사용되는 Agent_Config와 Env_Config는 시뮬레이션을 위해 설정해야 하는 파라미터들이며, 자세한 내용은 5.2절에서 다룬다.

2) 전이 학습 및 온라인 학습

본 논문에서의 전이 학습은 Algorithm 1을 통해 얻은 사전 학습된 Q-Table을 실제 환경에서 사용될 강화학습 기반 HPA의 초기 정책으로 사용하는 형태로 이루어진다. 온라인 학습은 운영 중에 지속적인 학습이 이루어지는 것을 의미하며, Algorithm 2는 전이 학습을 활용한 강화학습 기반 HPA

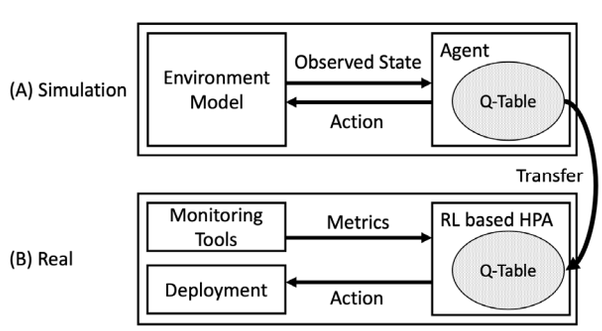


Fig. 1. Description of Transfer Learning Process using Simulation

Algorithm 1. Pre-training in Simulation

Input: Agent_Config, Env_Config
Agent ← Q_Learning_Agent(Agent_Config)
While all episodes are not finished **do**
  Env ← Environment(Env_Config)
  $t \leftarrow 1$ // Autoscaling period
  While simulation is not finished **do**
    get observed state from *Env*
    get state S_t and reward R_t from observed state
    select action A_t by using ϵ-greedy
    take action A_t to *Env*
    If $t > 1$ **then**
      update Q-Table by using $(S_{t-1}, A_{t-1}, R_t, S_t)$
      // see Formula (2)
    end If
    $t \leftarrow t+1$
  end While
  take ϵ-decay
end While
Return: *Agent*'s Q-Table

Algorithm 2. Online Learning with Transfer Learning

Input: Pre-trained Q-Table, Agent_Config
Agent ← Q_Learning_Agent(Agent_Config)
initialize *Agent*'s Q-Table as Pre-trained Q-Table
$c \leftarrow$ get # of current pods from prometheus
$u \leftarrow$ get average cpu utilization from prometheus
$S \leftarrow (c,$ discretized $u)$
$A \leftarrow 0$ // do nothing
While True **do**
  sleep as much as the autoscaling interval
  $c, c_{avg}, u, e \leftarrow$ get metrics from prometheus
  // c_{avg} : average # of pods
  // e : # of expired requests in autoscaling period
  $S' \leftarrow (c,$ discretized $u)$
  $R' \leftarrow$ calculate total cost by using c_{avg}, e
  select action A' by using Q-Table with noise
  patch deployment.spec.replicas to $c + A'$
  update Q-Table by using (S, A, R', S')
  // see Formula (2)
  $S \leftarrow S'$
  $A \leftarrow A'$
end While

의 온라인 학습에 대한 과정을 보인다. 온라인 학습에서는 사전 학습된 정책을 초기 정책으로 사용함으로써 초기부터 비용을 최소화할 수 있는 행동에 근접한 행동을 수행할 수 있으며, 이를 통해 실제 환경과 환경 변화에 적응적인 최적의 정책에 빠르게 수렴할 수 있도록 유도한다.

시뮬레이션 환경과 실제 환경이 유사하지만, 완전히 똑같지는 않기 때문에 시뮬레이션 환경에 편향된 정책은 실제 환경에서 잘못된 행동을 취하게 할 수 있다. 편향된 정책을 피하기 위해서는 탐험적인 행동을 통해 새로운 환경에 적응적인 정책으로 업데이트하는 과정이 필요하지만, 이를 위해 ϵ-greedy를 행동 정책으로 사용하기에는 잘못된 행동으로 인해 큰 손실이 발생할 가능성이 있어서 주의가 필요하다. 그래서 본 논문에서는 온라인 학습에서는 ϵ-greedy를 행동 정책으로 사용하지 않으며, 사전 학습을 통해 학습된 정책을 기반으로 행동을 취하고, 행동에 노이즈를 줌으로써 리스크를 최소화하면서도 시뮬레이션 환경에 편향된 정책의 문제를 빠르게 해소할 수 있도록 한다. 온라인 학습에서 사용되는 Agent_Config에 대한 자세한 내용은 5.2절에서 다룬다.

5. 실험 및 분석

5.1 구현

본 논문에서는 실험을 위해 kOps[8]를 사용하여 AWS에 4개의 t3.medium 인스턴스를 생성하고, 하나의 마스터 노드와 세 대의 워커 노드로 구성된 쿠버네티스 클러스터를 구축하여 사용한다. kOps는 쿠버네티스 클러스터를 구축하는 것을 도와주는 툴이며, 특히 AWS와 같은 클라우드 환경에 쿠버네티스 클러스터를 쉽게 구축할 수 있도록 도와준다.

강화학습을 기반으로 오토스케일링을 수행하는 컨트롤러를 구현하기 위해 파이썬을 기반으로 쿠버네티스 커스텀 컨트롤러를 쉽게 구현할 수 있도록 도와주는 Kopf[9]를 사용한다. 강화학습 기반 HPA는 기본 HPA와는 다르게 별도의 임곗값 설정을 요구하지 않으며, 활성화될 수 있는 최소 파드 수와 최대 파드 수, 초기 Q-Table을 설정 값으로 가진 상태로 강화학습 기반 수평적 파드 오토스케일링을 수행한다. 강화학습 기반 HPA는 사전 학습을 통해 학습된 Q-Table을 초깃값으로 사용할 수 있으며, 사전 학습이 이루어지지 않았을 경우에는 0으로 초기화된 Q-Table을 초기 정책으로 사용한다.

강화학습 기반 HPA는 행동 결정과 정책 학습을 위한 메트릭 수집을 위해 프로메테우스를 사용한다. 프로메테우스는 Istio Mixer로부터 만료된 요청 수를, kube state metrics로부터 현재 파드 수와 평균 CPU 이용률을 수집하며, 강화학습 기반 HPA는 프로메테우스에 수집된 메트릭을 통해 행동 결정 및 지속적인 정책 업데이트를 수행한다.

5.2 실험 방법

실험에 사용되는 워크로드는 NASA-HTTP[10]이며 Fig. 2와 같다. 사전 학습은 Fig. 2에서 5,000초까지의 기록을 바탕으로 수행된다. 5,001초부터 20,000초까지의 기록은 실제 환경에서 온라인 학습을 위해 사용되며, 부하 테스트 툴인 Gatling[11]을 사용해서 실제 트래픽을 발생시키며 온라인 학습을 진행한다.

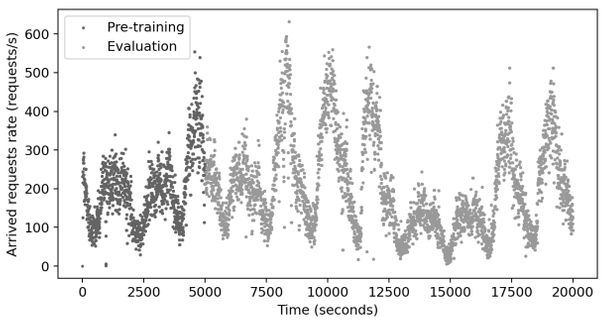


Fig. 2. NASA-HTTP Workload[10]

Table 2. Experiment Cases

Experiment	Description
(a)	No pre-training
(b)	1 pre-training in real
(c)	1 pre-training in simulation
(d)	10 pre-training in simulation
(e)	20 pre-training in simulation

실험은 Table 2의 5가지 경우와 같이 사전 학습을 수행하고, 사전 학습된 Q-Table을 쿠버네티스 클러스터에서의 강화학습 기반 HPA로 전이시킨다. 그리고 온라인 학습을 수행하며 주기적으로 만료되는 요청 수, 파드 수를 측정한다. (a)는 사전 학습의 효과를 평가하기 위해 사전 학습을 수행하지 않을 경우에 대한 실험이고, (b)는 실제 환경에서 사전 학습을 수행할 경우와 시뮬레이션 환경에서 사전 학습을 수행할 경우를 비교하기 위한 실험이다. (c), (d), (e)는 시뮬레이션을 활용한 사전 학습을 수행하는 경우이며, 사전 학습 횟수의 증가에 따른 성능 평가를 위해 수행된다.

시뮬레이션을 위한 환경에 대한 설정은 Table 3의 (a)와 같다. 시뮬레이션을 위해서는 애플리케이션 성능에 대한 측정이 필요하며, kOps로 구성한 쿠버네티스 클러스터 환경에서 애플리케이션 성능을 측정한다. 실험에는 정적 웹 페이지를 제공하는 Node.js 웹 애플리케이션을 사용하며, 애플리케이션 성능 측정을 위해서 5초마다 초당 요청 수를 5씩 증가시켜가며 파드 수에 따른 처리 가능한 최대 요청 수를 측정한다. 이때, 실험에서 최대 파드 수를 4로 설정해서 사용하기 때문에 파드 수는 1부터 4까지 증가시켜가며 측정한다. 선형 회귀를 통해 얻은 파드 수가 x일 때의 처리 가능한 최대 요청 수는 $219.1x + 72.0$으로 근사되고, 결정계수는 0.9979로 유사도가 높음을 보인다.

4.1절의 개선된 대기행렬 기반 환경 모델을 사용하기 위해서는 파드 수가 x일 때, 파드 하나가 처리할 수 있는 최대 요청 수에 대한 함수가 필요하며, 앞선 실험을 통해 측정한 $219.1x+72.0$을 파드 수 x로 나누어 $AP(x) = 219.1 + 72.0/x$로 함수를 정의해서 시뮬레이션에 사용한다. $x \times AP(x)$는 최대로 처리할 수 있는 요청 수를 의미하며, 오토스케일링 주기 동안 처리한 요청 수를 $x \times AP(x) \times autoscaling\ interval$로 나눈 값을 CPU 이용률로 근사해서 사용한다.

Table 3. Environment Configurations

Configuration	(a) Simulation	(b) Real
Application Profile(AP)	$AP(x) = 219.1 + 72/x$	-
Autoscaling Period	30s	
Workload History	1s ~ 5,000s	5,001s ~ 20,000s
Timeout	0.1s	
Simulation Period	0.1s	
Scaling Delay	5s	

Table 4. Agent Configurations

Configuration	(a) Simulation	(b) Real
Minimum # of Pods	1	
Maximum # of Pods	4	
Resource Cost	5.8×10^{-7} USD/s	
SLA Violation Cost	5.8×10^{-8} USD/s	
Initial Epsilon(ϵ)	1.0	-
Epsilon(ϵ) Decay	0.9	-
Noise	-	0.1
Learning Rate	0.5	
Discount Factor	0.3	

실제 환경에 대한 설정은 Table 3의 (b)와 같으며, 시뮬레이션을 위해 필요했던 설정을 제외하고는 Table 3의 (a)와 똑같다. 타임아웃은 Istio의 타임아웃 설정을 통해 지정하며, 스케일링 지연시간은 파드가 생성될 때의 초기 지연시간 설정을 통해서 지정한다.

시뮬레이션에서의 강화학습 기반 HPA의 설정은 Table 4의 (a)와 같고, 실제 환경에서의 강화학습 기반 HPA의 설정은 Table 4의 (b)와 같다. 시뮬레이션에서는 ϵ-greedy를 통해 최대한 탐험적인 행동을 취하기 위해 ϵ의 초깃값을 1.0으로 설정한다. 실제 환경에서의 강화학습 기반 HPA는 ϵ-greedy 행동 정책을 취하지 않고 경험을 통해 학습된 정책을 활용하는 방향으로 행동을 취하며, 시뮬레이션 환경에 대한 편향을 완화시키면서 리스크를 최소화하기 위해 4.3절에서 언급한 노이즈를 사용한다.

AWS의 t3.medium은 1 vCPU 당 0.0208 USD/hour의 비용이 청구된다. 실험에 사용되는 Node.js 웹 애플리케이션 컨테이너의 CPU limit을 0.1 vCPU로 설정했기 때문에, 하나의 컨테이너에 대한 자원 비용을 약 5.8×10^{-7} USD/s로 근사하여 실험에 사용한다. 응답시간이 초과된 요청에 대한 페널티 비용은 자원 사용 비용의 0.1배로 설정해서 강화학습을 수행한다.

5.3 실험 결과 및 분석

Fig. 3은 각각의 케이스에 대해 누적된 비용이며, Fig. 4의 만료된 요청 수와 Fig. 5의 파드 수가 비용을 산출하는데 사용 된다.

Fig. 3의 (d)는 사전 학습을 10번 수행한 이후 전이 학습

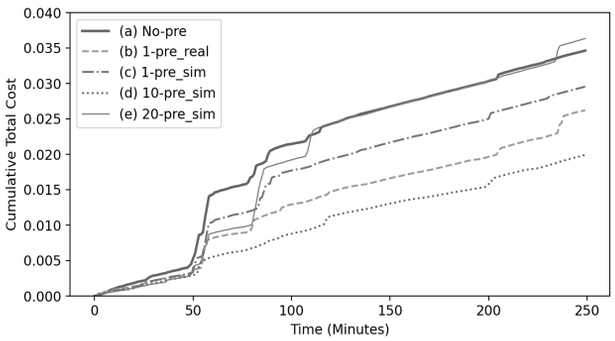


Fig. 3. Cost Comparison

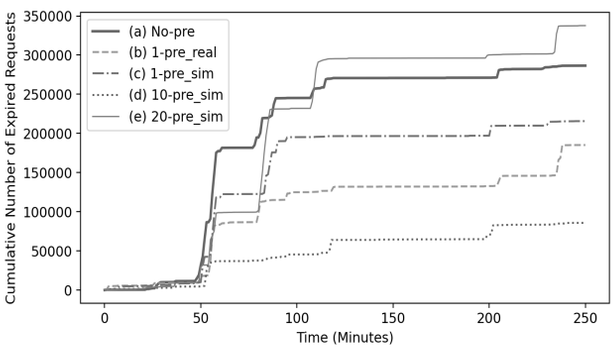


Fig. 4. Number of Expired Requests Comparison

을 통해 온라인 학습을 수행한 경우이며, 사전 학습을 수행하지 않은 Fig. 3의 (a)보다 약 42.6%의 총 비용을 절약했다. 또한, Fig. 4와 Fig. 5를 통해 (d)가 (a)보다 더 적은 수의 파드를 사용하면서도 더 적은 수의 요청이 만료되었음을 확인할 수 있다.

Fig. 3의 (b)는 실제 환경에서 5,000초 분량의 워크로드를 바탕으로 1번의 사전 학습을 수행한 경우이고, Fig. 3의 (c)는 시뮬레이션 환경에서 약 1초 동안 5,000초 분량의 워크로드를 시뮬레이션하며 1번의 사전 학습을 수행한 경우이다. 실제 환경에서 학습을 수행한 경우가 시뮬레이션을 통해 사전 학습을 수행한 경우보다 약 11.2%의 총 비용을 절약했으나, (b)의 경우 실제로 5,000초 동안 워크로드 기록을 바탕으로 트래픽을 발생시키며 학습을 수행했기 때문에 사전 학습을 위한 시간과 자원이 추가적으로 필요하다는 단점을 가진다.

Fig. 3의 (c), (d), (e)는 각각 시뮬레이션 환경에서 사전 학습을 1회, 10회, 20회를 수행한 이후에 전이 학습을 통해 온라인 학습을 수행한 경우이며, 사전 학습을 10회 수행한 경우가 가장 적은 비용을 사용하였다. (c), (d)를 통해 주어진 워크로드를 사용한 시뮬레이션 경험의 학습 효율을 높이기 위해 반복적인 사전 학습을 수행하는 것이 효과가 있음을 보였으며, (d), (e)를 통해 너무 많은 학습은 시뮬레이션 환경에 지나치게 편향된 정책으로 수렴하게 되어 이후 온라인 학습을 수행할 때 오히려 학습을 방해하여 더 많은 비용을 사용하게 함을 확인할 수 있다.

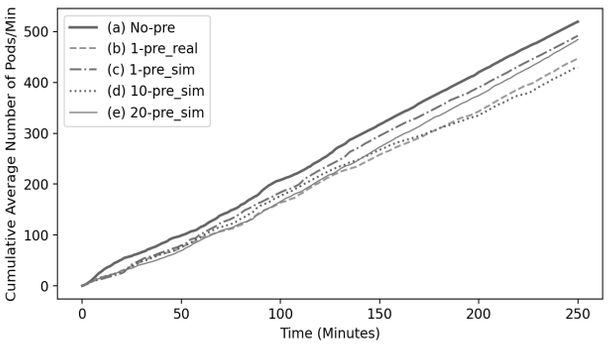


Fig. 5. Average Number of Pods Comparison

6. 결 론

본 논문에서는 시뮬레이션 환경에서 사전 학습된 정책을 이용하여 실제 환경에서 온라인 학습을 수행하는 강화학습 기반 HPA의 학습을 가속화하는 전이 학습 기법을 제안하였다. 시뮬레이션에서 5,000초 분량의 워크로드 기록으로 10번의 사전 학습을 수행하는데 걸리는 시간은 약 10초이며, 실제 환경에서는 막대한 시간과 비용을 투자하여 이루어져야 하는 사전 학습을 시뮬레이션을 통해 짧은 시간 내에 가능하게 했다. 또한, 10번의 사전 학습을 수행한 이후에 전이 학습을 수행하기만 해도 사전 학습을 수행하지 않았을 때보다 약 42.6%의 비용을 절감시킬 수 있었다. 이를 통해, 실제 환경과는 다른 시뮬레이션 환경에서 학습된 정책이어도 학습을 가속화하는 효과를 보이고, 학습 초기에도 적절한 행동 정책을 가짐을 확인할 수 있다.

마이크로서비스 아키텍처가 많이 채택되면서 작은 규모의 서비스가 많이 배포되고 있다. 마이크로서비스 아키텍처의 장점 중에는 각각의 마이크로서비스에 대해서 요구에 맞게 스케일링을 수행할 수 있다는 점이 있으나 각각의 서비스에 대해 매번 오토스케일링 정책 최적화를 수행하는 것은 비효율적이다. 그러나 시뮬레이션 환경을 활용한 강화학습 기반 HPA 전이 학습 기법은 오토스케일링 정책 최적화 과정을 가속화 및 자동화하는데 활용될 수 있을 것이며, 범용적인 상황에서 사용할 수 있는 전이 학습 기법에 대한 추가적인 연구를 진행할 계획이다.

References

[1] Horizontal Pod Autoscaler - Kubernetes [Internet], https://kubernetes.io/ko/docs/tasks/run-application/horizontal-pod-autoscale/

[2] S. Horovitz and Y. Arian, "Efficient cloud auto-scaling with SLA objective using Q-learning," In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp.85-92, 2018.

[3] F. Rossi, M. Nardelli, and V. Cardellini, "Horizontal and vertical scaling of container-based applications using reinforcement learning," In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pp.329-338, Jul. 2019.

[4] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, "FIRM: An intelligent fine-grained resource management framework for SLO-Oriented microservices," In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp.805-825, 2020.

[5] Y. Garí, D. A. Monge, E. Pacini, C. Mateos, and C. G. Garino, "Reinforcement learning-based application autoscaling in the cloud: A survey," *Engineering Applications of Artificial Intelligence*, Vol.102, 2021.

[6] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2nd ed., Cambridge: MIT press, 2018.

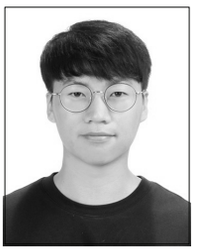
[7] L. Toka, G. Dobreff, B. Fodor, and B. Sonkoly, "Machine learning-based scaling management for kubernetes edge clusters," *IEEE Transactions on Network and Service Management*, Vol.18, No.1, pp.958-972, 2021.

[8] kOps: Kubernetes Operations [Internet], https://kops.sigs.k8s.io/

[9] Kopf: Kubernetes Operators Framework [Internet], https://kopf.readthedocs.io/en/stable/

[10] NASA-HTTP Logs, [Internet], ftp://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[11] Gatling [Internet], https://gatling.io/



장 용 현

https://orcid.org/0000-0002-9445-7727
e-mail : wkd3475@gmail.com
2020년 고려대학교 컴퓨터학과(학사)
2020년 ~ 현 재 고려대학교 컴퓨터학과 석사과정
관심분야 : 클라우드컴퓨팅, 분산시스템, 강화학습

유 헌 창

https://orcid.org/0000-0003-2216-595X
e-mail : yuhc@korea.ac.kr
1989년 고려대학교 컴퓨터학과(학사)
1991년 고려대학교 컴퓨터학과(석사)
1994년 고려대학교 컴퓨터학과(박사)
1998년 ~ 현 재 고려대학교 컴퓨터학과 교수
관심분야 : 클라우드컴퓨팅, 가상화, 분산시스템

김 성 석

https://orcid.org/0000-0002-7596-0757
e-mail : sskim03@skuniv.ac.kr
1997년 고려대학교 컴퓨터학과(학사)
1999년 고려대학교 컴퓨터학과(석사)
2003년 고려대학교 컴퓨터학과(박사)
2001년~2003년 한국산업기술대학교 컴퓨터학과 겸임교수
2014년 University of Texas at El Paso 교환교수
2003년 ~ 현 재 서경대학교 소프트웨어학과 교수
관심분야 : 데이터베이스, 인공지능 및 빅데이터