



# 강화학습 기반의 클라우드 서버 오토스케일링 최적화 (Server Auto-scaling)

서버 오토스케일링 디자인 및 Reward Shaping이 에이전트의 안정성에 미치는 영향 분석

과목명: 강화학습의 기초 (2025-2학기)

팀 원: A70048 김정식

GitHub Repository: <https://github.com/sgjskim00/ServerAutoScaling>

# 프로젝트 개요(Instruction)

## 배경

기관 서버운영 책임자로서 클라우드 환경에서 트래픽 변동에 따른 서버 자원 효율적 관리는 비용 절감 및 서비스 품질 유지의 핵

## 목표

- 변동하는 트래픽 패턴을 스스로 학습 하는 강화학습(RL) 에이전트 개발
- 서비스 지연(Overload) 방지 및 운영 비용(Cost) 최소화

## 접근 방법

Gymnasium 기반의 자체 시뮬레이션 환경 구축 및 PPO(Proximal Policy Optimization) 알고리즘 적용



## 문제 정의(MDP Design)

강화학습 모델의 핵심은 환경(Environment), 상태(State), 행동(Action), 그리고 보상(Reward)을 명확히 정의

- 환경 (Environment)**
  - ServerAutoScalingEnv (Custom Gym Environment)
- 상태 (Observation)**
  - [Current Traffic, Active Server Count] – 트래픽과 현재 서버 대수를 기반으로 판단합니다.
- 행동 (Action, Discrete)**
  - 0: 유지(Hold), 1: 증설(Scale Out), 2: 감축(Scale In)
- 보상 함수 (Reward Function)**

$$R = R_{service} - (C_{server} \times N_{server}) - P_{overload}$$
  - $R_{service}$ : 서비스 생존 보상 (변수)
  - $C_{server}$ : 서버당 운영 비용 (1.0)
  - $P_{overload}$ : 과부하 패널티 (10.0)

# 알고리즘 및 환경 구현

## 알고리즘: PPO (Proximal Policy Optimization)

하이퍼파라미터에 덜 민감하며, 이산적 행동 공간에서 안정적인 성능을 보이는 것으로 알려져 있습니다.

이는 오토스케일링과 같이 명확한 행동 선택이 필요한 문제에 적합합니다.

---

## 라이브러리

- Stable Baselines<sup>3</sup>
  - Gymnasium
- 

## 트래픽 시뮬레이션

Sine Wave 기반의 주기적 패턴에 Random Noise를 추가하여 낮/밤 트래픽 변화를 모사했습니다.

이를 통해 실제 클라우드 환경의 동적인 트래픽 변화를 효과적으로 반영할 수 있습니다.

# 실험 설계: Reward Shaping의 영향력 분석

이 실험의 핵심은 보상 설계(Reward Shaping)가 강화학습 에이전트의 행동 패턴에 어떤 영향을 미치는지 분석하는 것입니다.



## 가설 설정

서비스 생존 보상( $R_{service}$ )의 크기에 따라 에이전트의 성향  
(안정성 중시 vs. 비용 절감 중시)이 달라짐



## Case A: 안정성 중시

$R_{service} = 2.0$  (보상 > 비용) → 과부하 없는 안정적 운영 예상.



## Case B: 비용 절감 중시

$R_{service} = 1.0$  (보상 = 비용) → 최소 비용으로 운영 시도 예상.



## Case C: 학습 불안정

$R_{service} = 0.5$  (보상 < 비용) → 학습 실패 또는 비효율적 운영 예상.

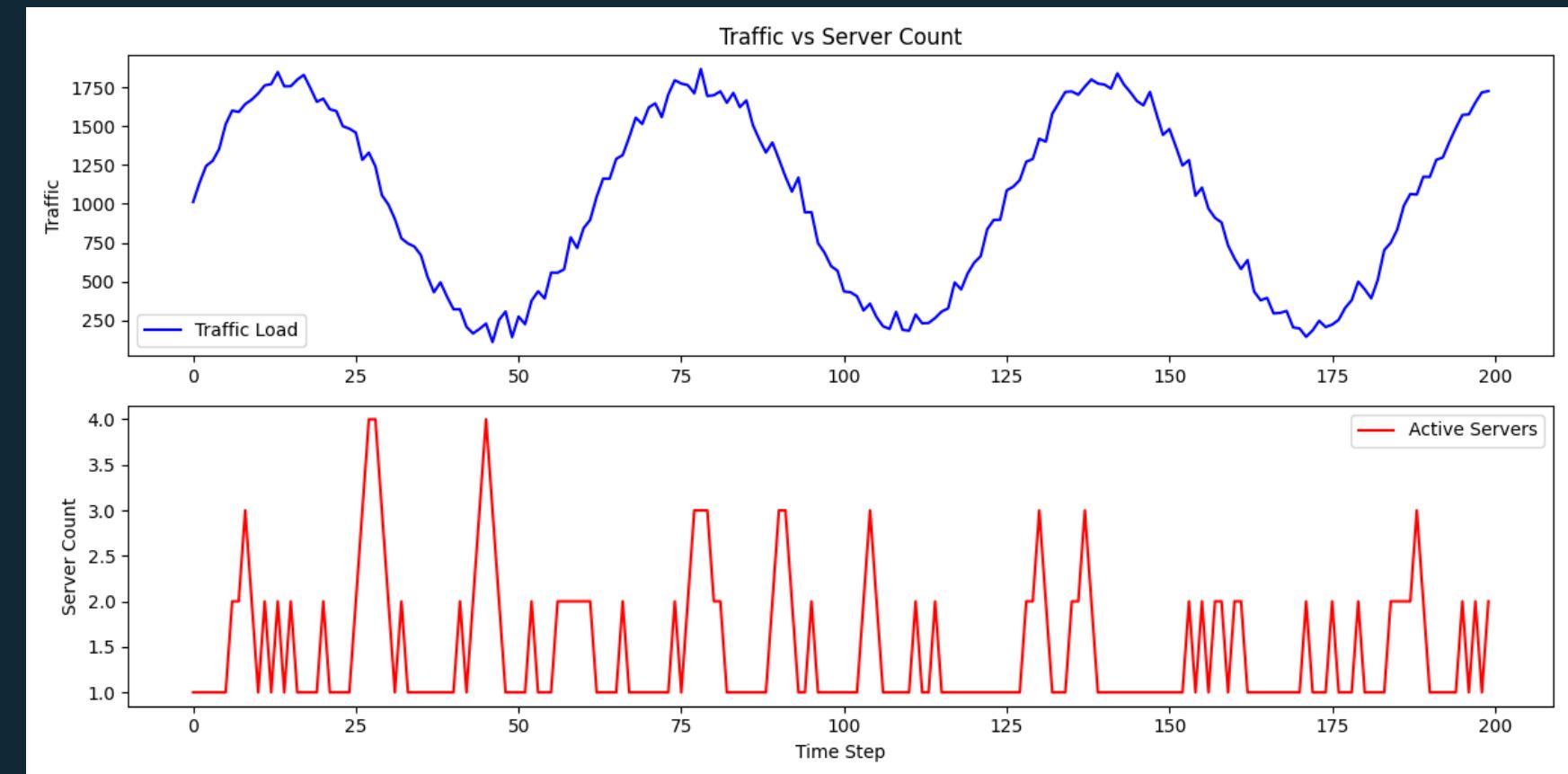
# 실험 결과 1: 안정적인 모델 (Reward 2.0)

$R_{service} = 2.0$ 으로 설정했을 때, 에이전트는 서비스 안정성을 최우선으로 학습했습니다.

## 분석:

- 파란색(트래픽) 선을 빨간색(서버) 선이 부드럽게 따라가는 양상을 보입니다.
- 트래픽 증가 시 미리 여유 서버(Buffer)를 확보하여 과부하를 효과적으로 차단합니다.
- 최대 서버 수는 4대로, 안정적인 운영을 유지하면서도 자원을 효율적으로 사용합니다.

▣ 이 모델은 서비스 품질 유지에 가장 이상적인 모습을 보여주었습니다.



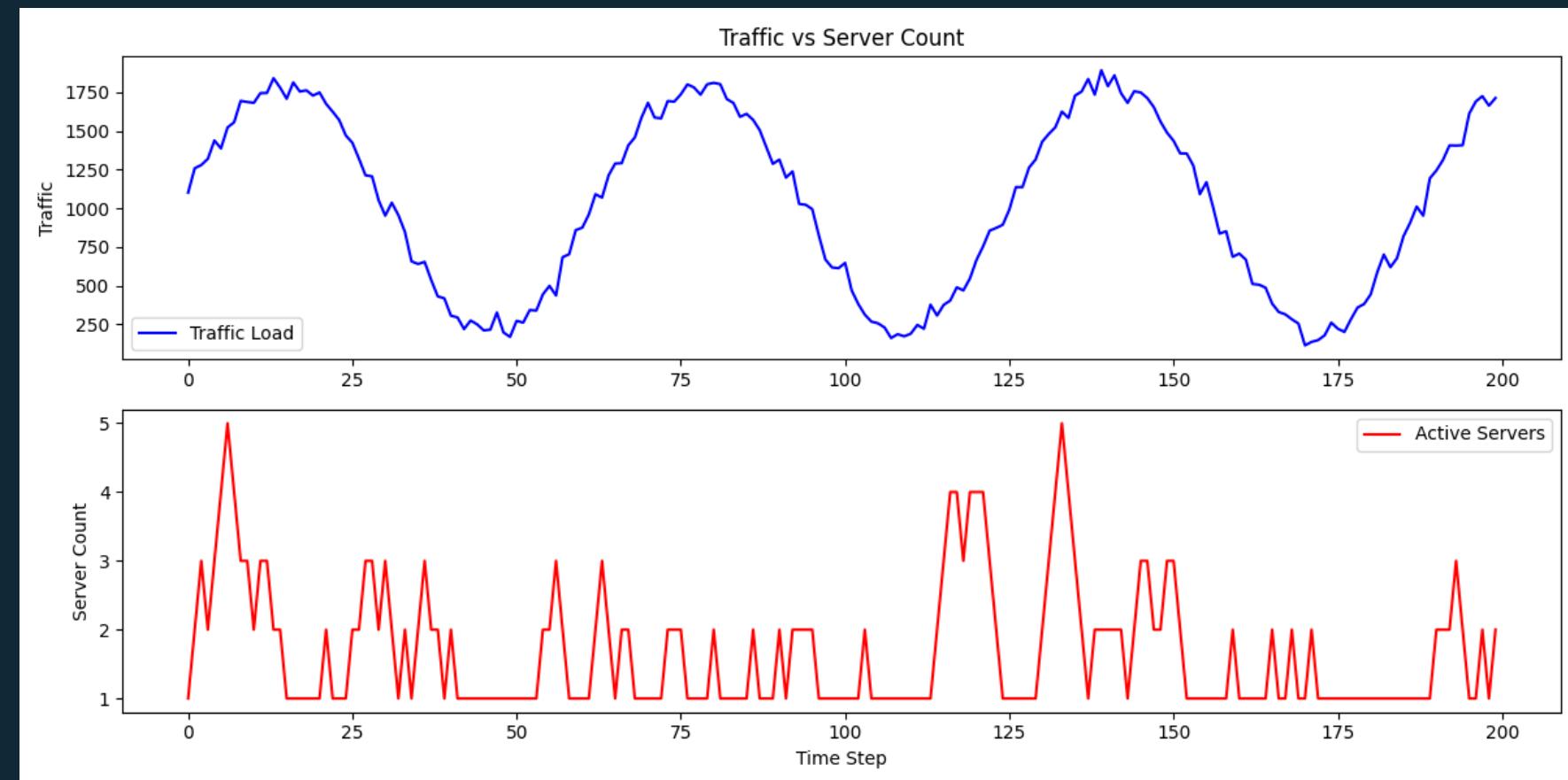
# 실험 결과 2: 비용 민감형 모델 (Reward 1.0)

$R_{service} = 1.0$ 으로 설정된 에이전트는 비용 절감에 더 민감한 반응을 보였습니다.

## 분석:

- Reward 2.0 모델에 비해 서버 증설을 **최대한 늦추려는 경향**이 나타남
- 급격한 트래픽 증가 시 서버 증설 반응이 늦어져 **순간적인 스파이크(Spike)**가 발생
- 최대 서버 수는 5대로, 타이밍을 놓쳐 급하게 증설하는 경우가 발생

▣ 안정성은 다소 떨어지지만, 비용 효율을 높이려는 시도가 돋보였습니다.



# 실험 결과 3: 학습 실패 사례 (Reward 0.5)

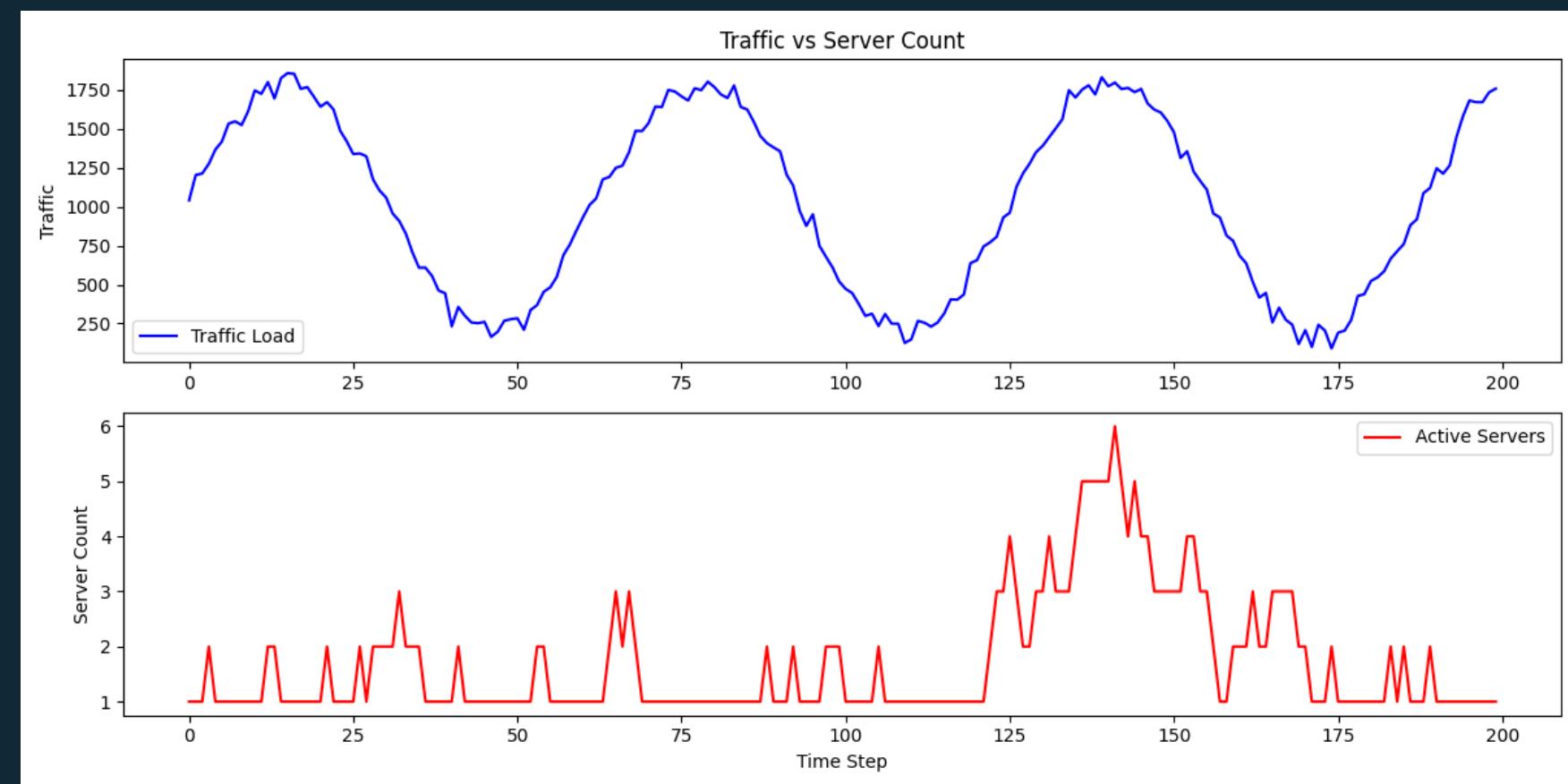
$R_{service} = 0.5$ 로 설정된 에이전트는 학습 불안정을 넘어 학습 붕괴 현상을 보였습니다.

## 현상:

- 평소에는 서버를 1대(최소)로 억지로 유지
- 과부하 발생 시 패널티를 피하기 위해 **패닉(Panic) 상태**에 빠져 서버를 6대까지 무작위 증설
- 이러한 비정상적인 행동은 보상 설계의 중요성을 극명하게 보여줌

## □ Insight:

기본 보상이 운영 비용보다 낮으면( $R < Cost$ ),  
에이전트는 장기적 안정성보다 단기적 비용 회피에 집착하여 학습이 붕괴



# 종합 비교 및 결론

## 비교 요약

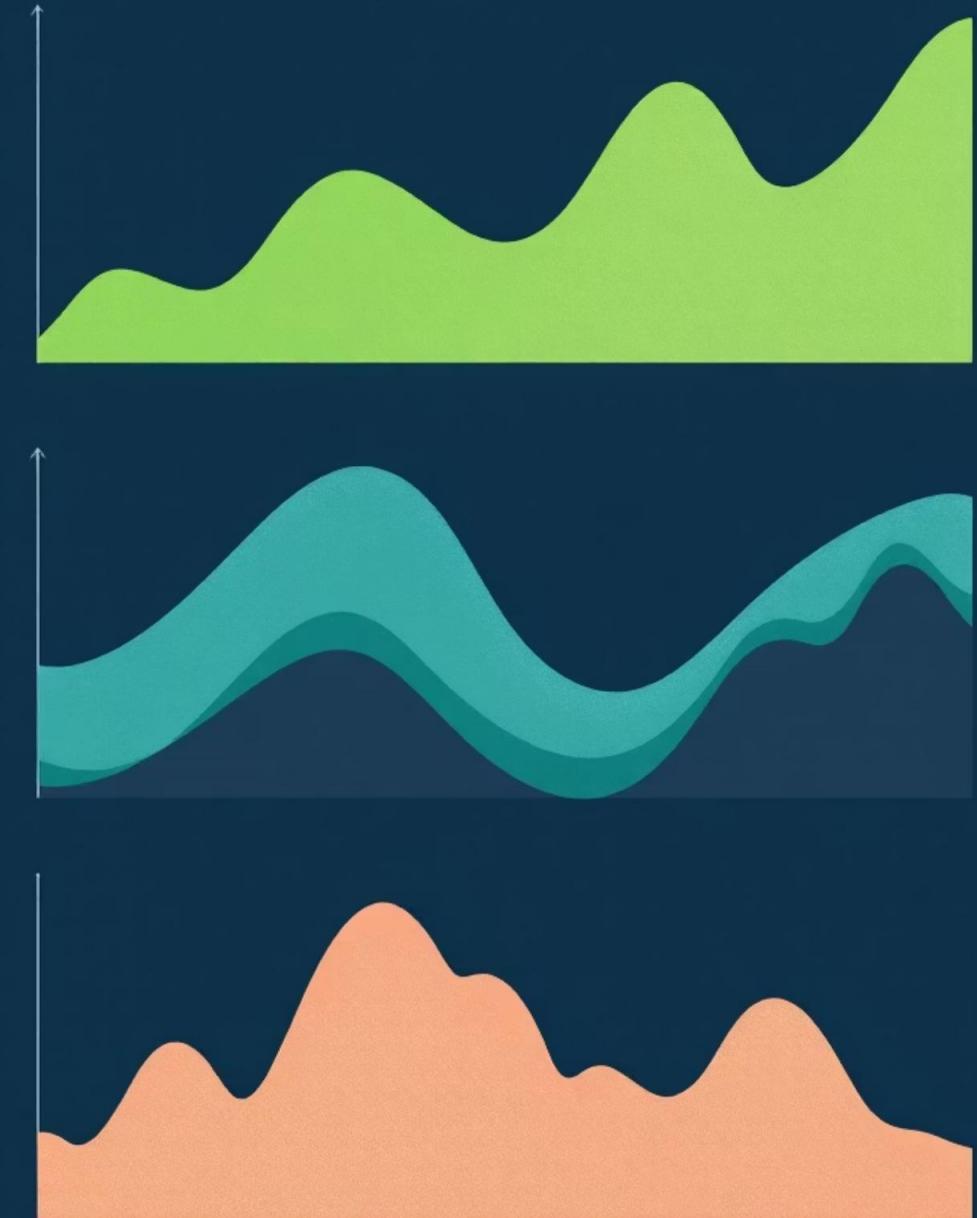
- **안정성:** Reward 2.0 > Reward 1.0 > Reward 0.5
- **학습 수렴도:** 보상이 비용보다 클 때(Reward > Cost) 안정적으로 수렴.

## 결론

- 강화학습을 통한 동적 오토스케일링 구현 성공.
- Reward Shaping(보상 설계)이 실제 성능과 정책 결정에 핵심적인 역할

## 향후 과제

- 실제 AWS/Azure 트래픽 데이터(Trace)를 활용한 검증.
- 서버 부팅 시간을 고려한 지연(Delay) 환경 모델링.





# Thank you