SLO Management with Reinforcement Learning on Multi-tenant Serverless FaaS Platforms

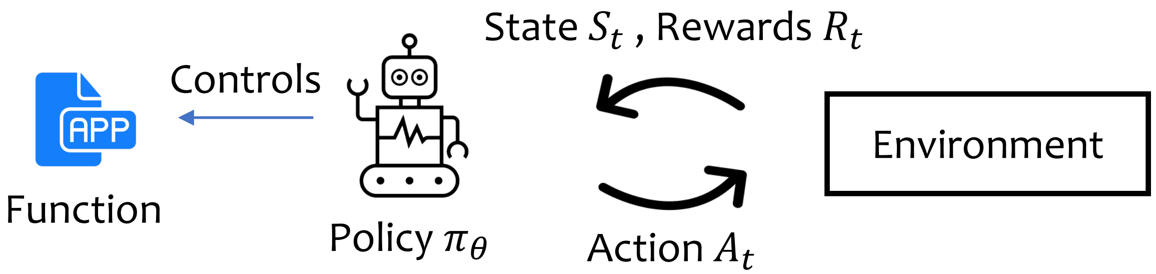
Haoran Qiu[1], Weichao Mao[1], Archit Patke[1], Chen Wang[2], Hubertus Franke[2]
Zbigniew Kalbarczyk[1], Tamer Basar[1], Ravishankar Iyer[1]

[1] UIUC [2] IBM Research

The 2nd Workshop on Machine Learning and Systems (EuroMLSys) 2022

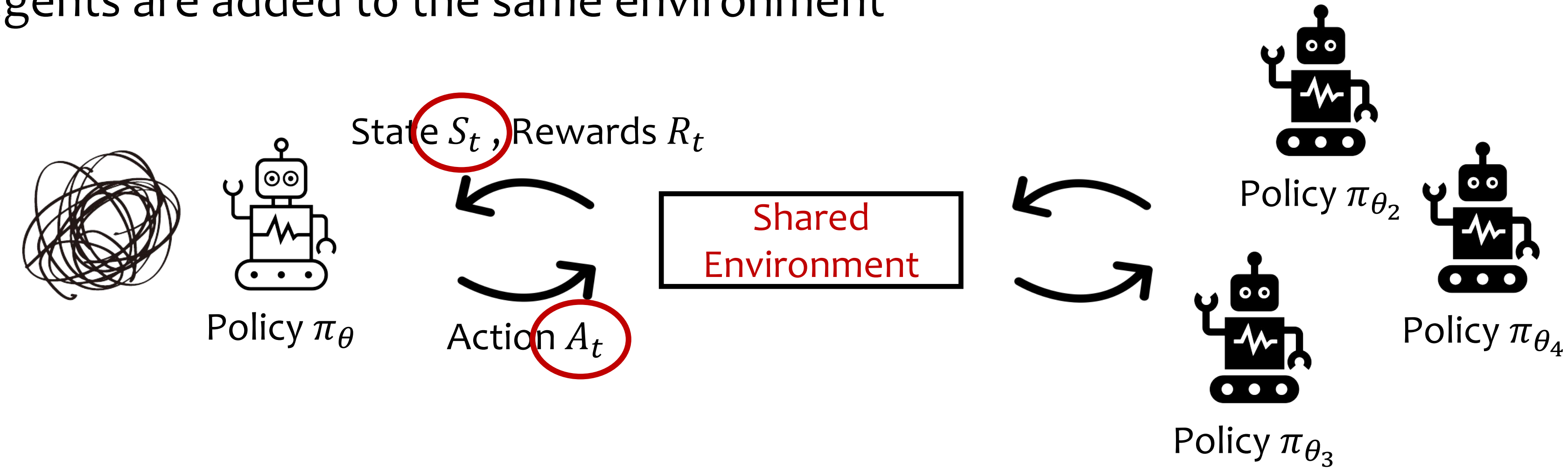
Problem Background and Motivation

- SLO (service-level objectives) management is hard in serverless FaaS platforms
  - Fine-grained resource management (concurrency, container resource allocation)
  - Heterogeneous functions (characteristics/SLO), workload dependent, frequent update, …
- Heuristics-based resource management has been proved to be inefficient and untenable
- Reinforcement learning (RL)-based resource management has been proposed in general "serverless" platforms:
  - DeepRM [HotNets '16], MIRAS [ICDCS '19], Symphony [ICML '20], FIRM [OSDI '20], ADRL [TPDS '20], AutoPilot [EuroSys '20], Q-learning-based Autoscaler [CCGrid '21], …
  - Single-agent RL in Single-tenant Environment

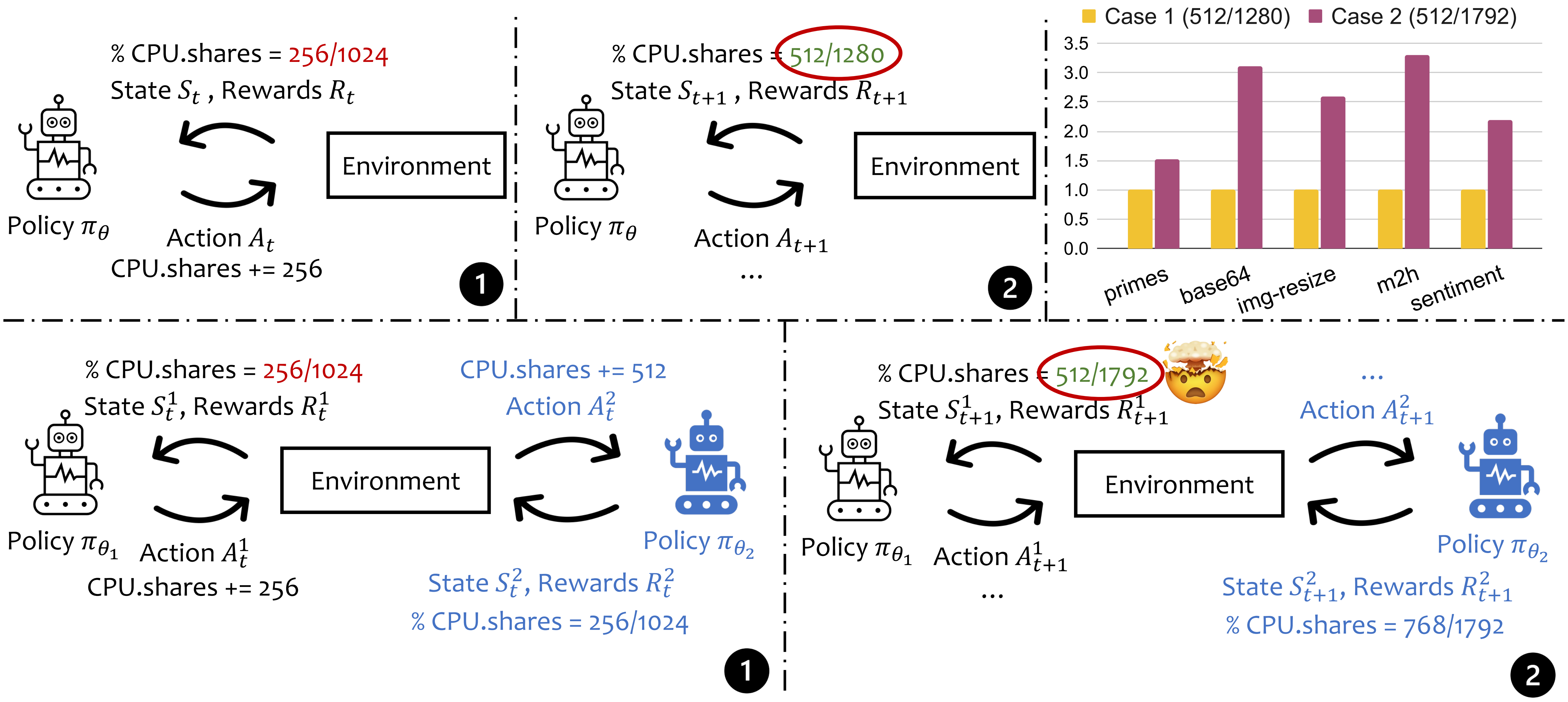


Problem Background and Motivation

- Reinforcement learning (RL)-based resource management has been proposed in general "serverless" platforms:
  - DeepRM [HotNets '16], MIRAS [ICDCS '19], Symphony [ICML '20], FIRM [OSDI '20], ADRL [TPDS '20], AutoPilot [EuroSys '20], Q-learning-based Autoscaler [CCGrid '21], …
  - Single-agent RL in Single-tenant Environment
- **Violation of the standard assumption of environment stationarity**
  - RL assumes that the underlying environment is stationary
  - Not true anymore from each RL agent's perspective when multiple self-interested RL agents are added to the same environment



Violation of Stationarity – A Motivating Example



Optimal solution: Agent 1 gets 0.4 (512/1280), Agent 2 gets 0.5 (768/1536)

Goal

- Provide system support that enables multiple RL-based controllers to coexist with each other
- Design principles

1 Performance Isolation

  - During training: converge to a collectively optimal policy
  - During execution: achieve comparable performance to single-agent RL in single-tenant cases

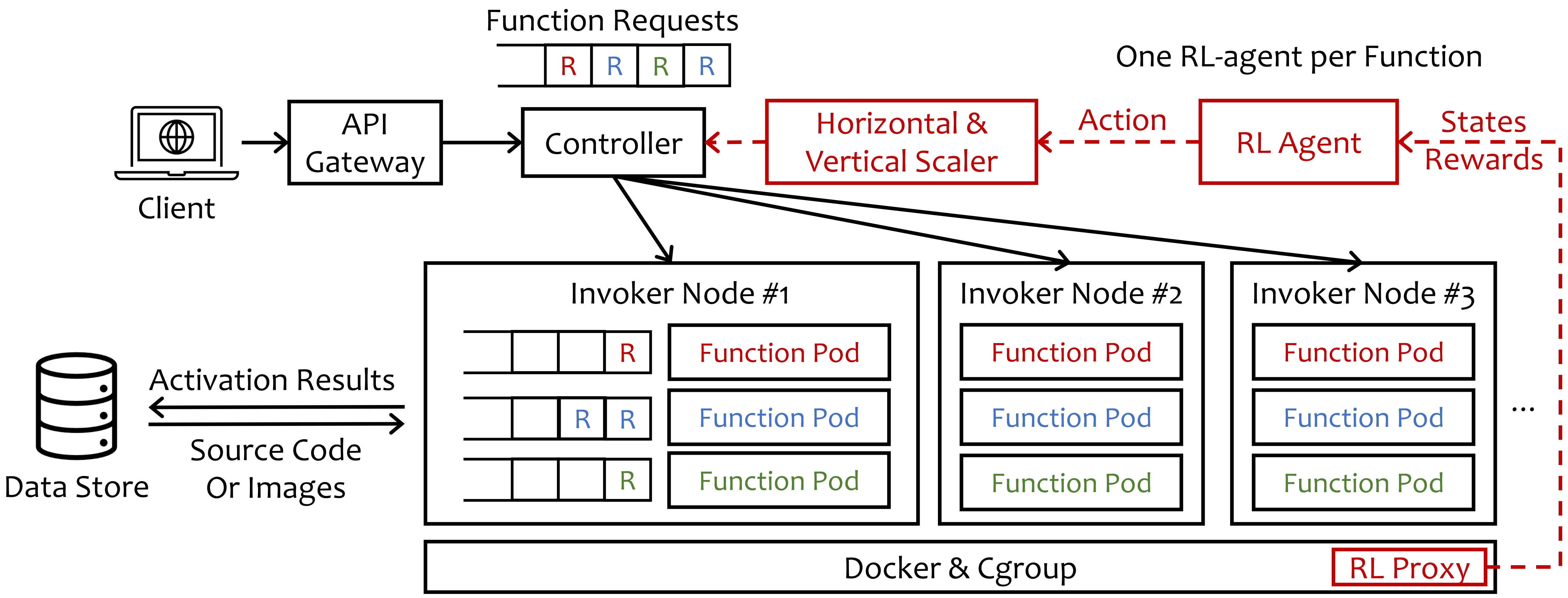
2 Scalability

  - Challenge: In a multi-tenant serverless FaaS platform, new functions from different customers can be increasingly registered

3 Fast Training/Retraining

  - Challenge: Functions from any customer can be registered, removed, or updated at any time, which changes the joint state space

Single-agent RL Pipeline in OpenWhisk



Single-agent RL Design

- **PPO:** a policy gradient method and the default RL algorithm in OpenAI
- **States:** SLO Preservation Ratio (SP_t), Resource Utilization ($RU_t(CPU, mem)$), Arrival Rate Changes (AC_t), Resource Limits ($RLT_t(CPU, mem)$), Horizontal Concurrency (NC_t)
- **Actions**
  - Vertical scaling: +/- step size of resource limits $av_t = \Delta RLT_t(CPU, mem)$
  - Horizontal scaling: +/- step size of number of function containers ($ah_t = \Delta NC_t$)
- **Reward function**

$$R_t = \alpha \cdot RU_t + \beta \cdot SP_t + penalty$$

Resource Utilization

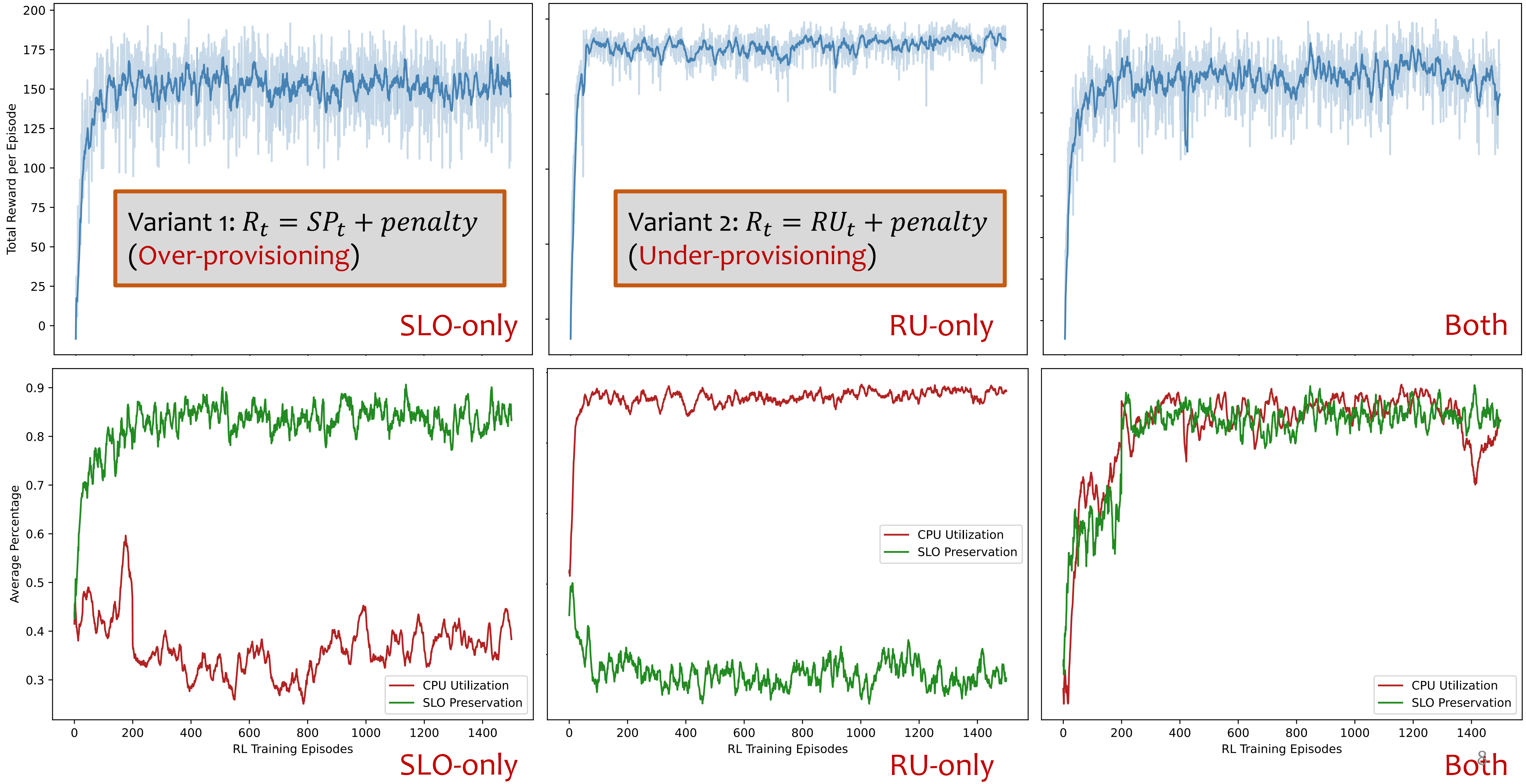
SLO Preservation

Penalize illegal or undesired actions

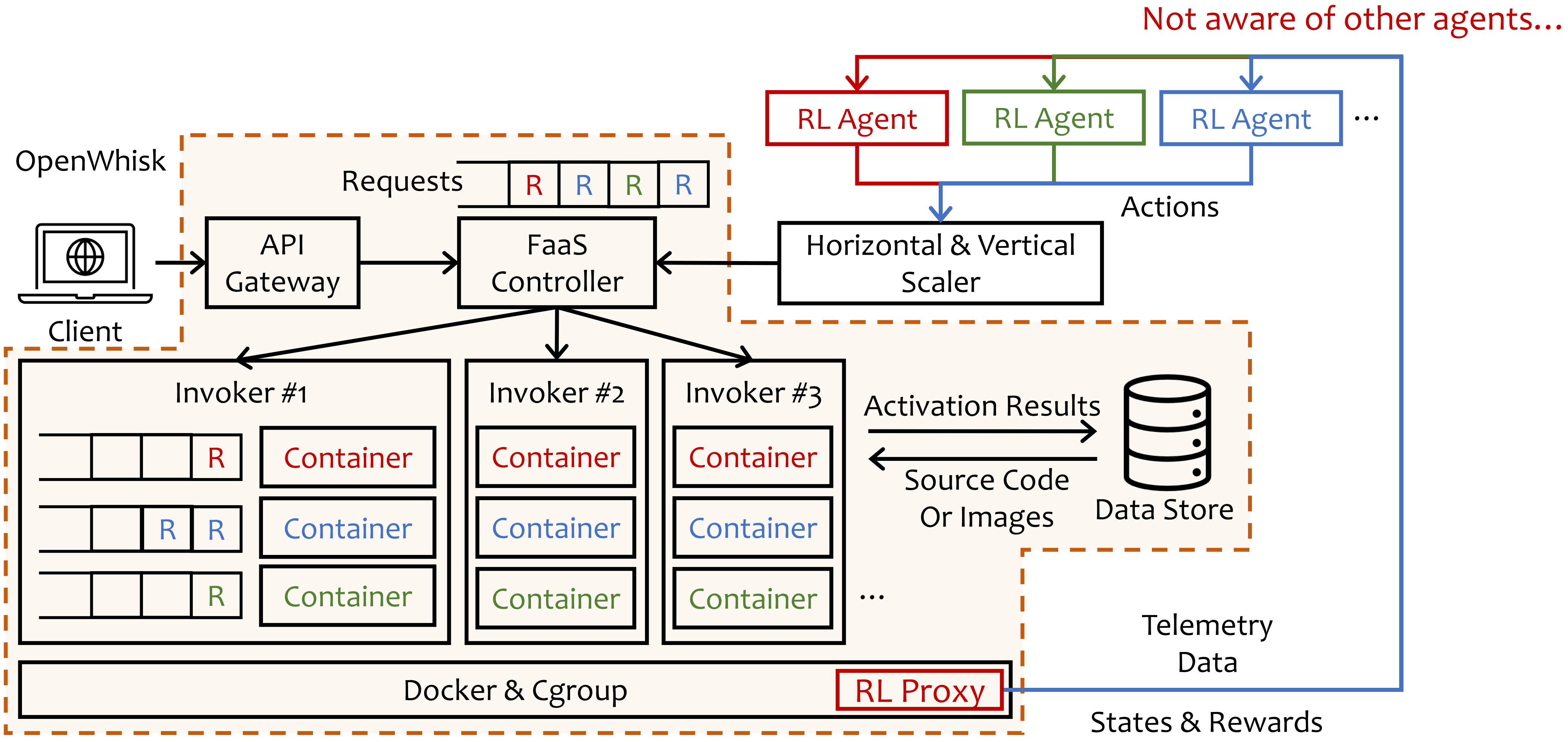
- Frequent dangling decisions
- Scale in/up/down when $NC_t = 0$

$$SP_t = \min\left(\frac{SLO\ Latency}{Actual\ Latency}, 1\right)$$

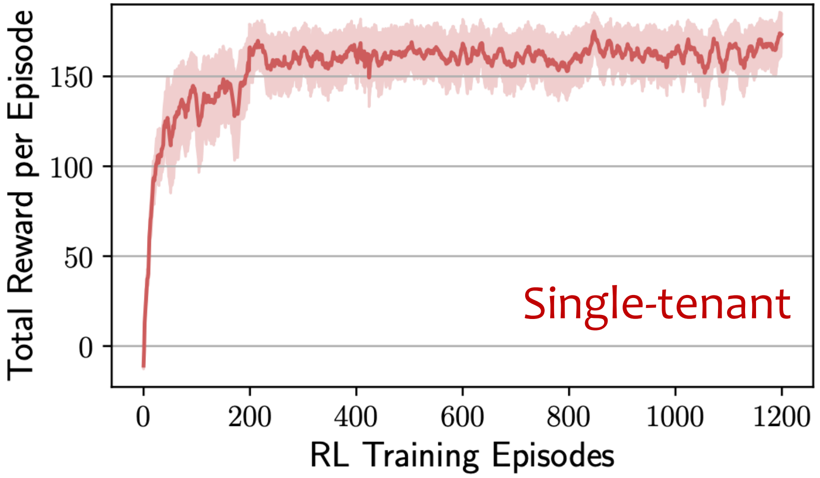
Reward Function Sensitivity Study (Single-agent RL)

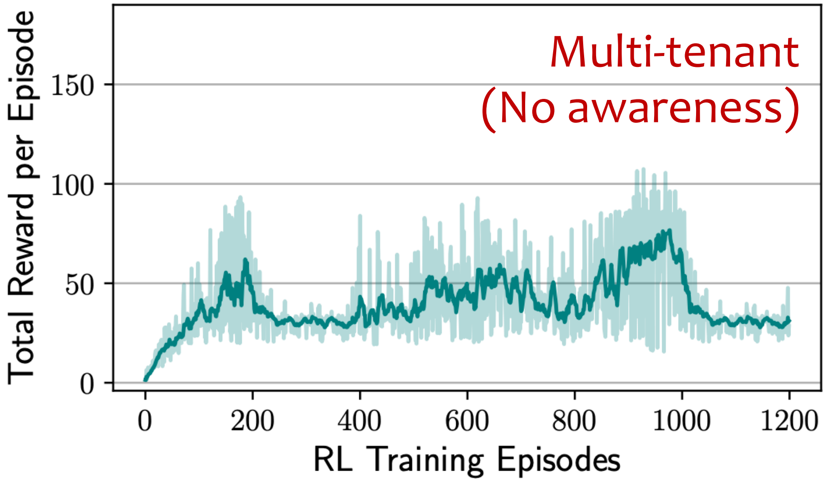


Multi-tenant RL Pipeline in OpenWhisk

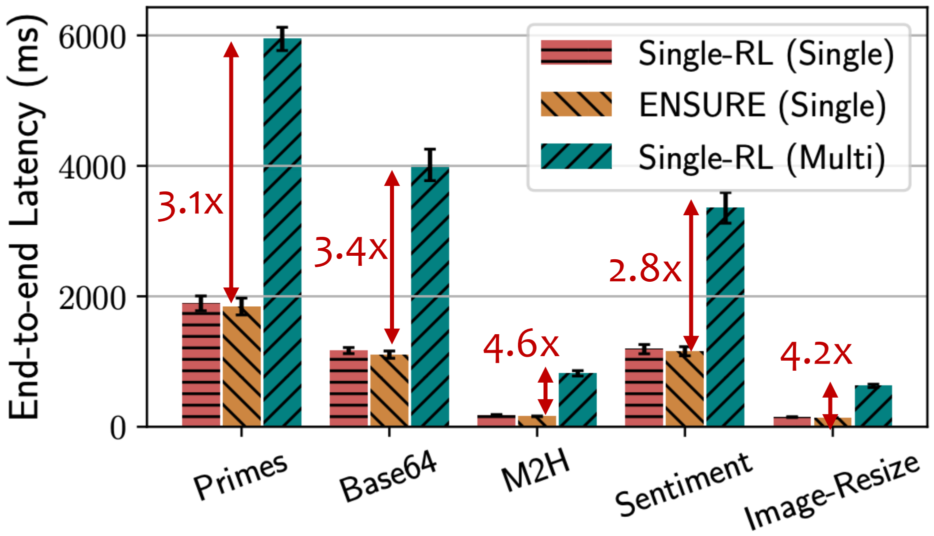


Single-agent RL Evaluation





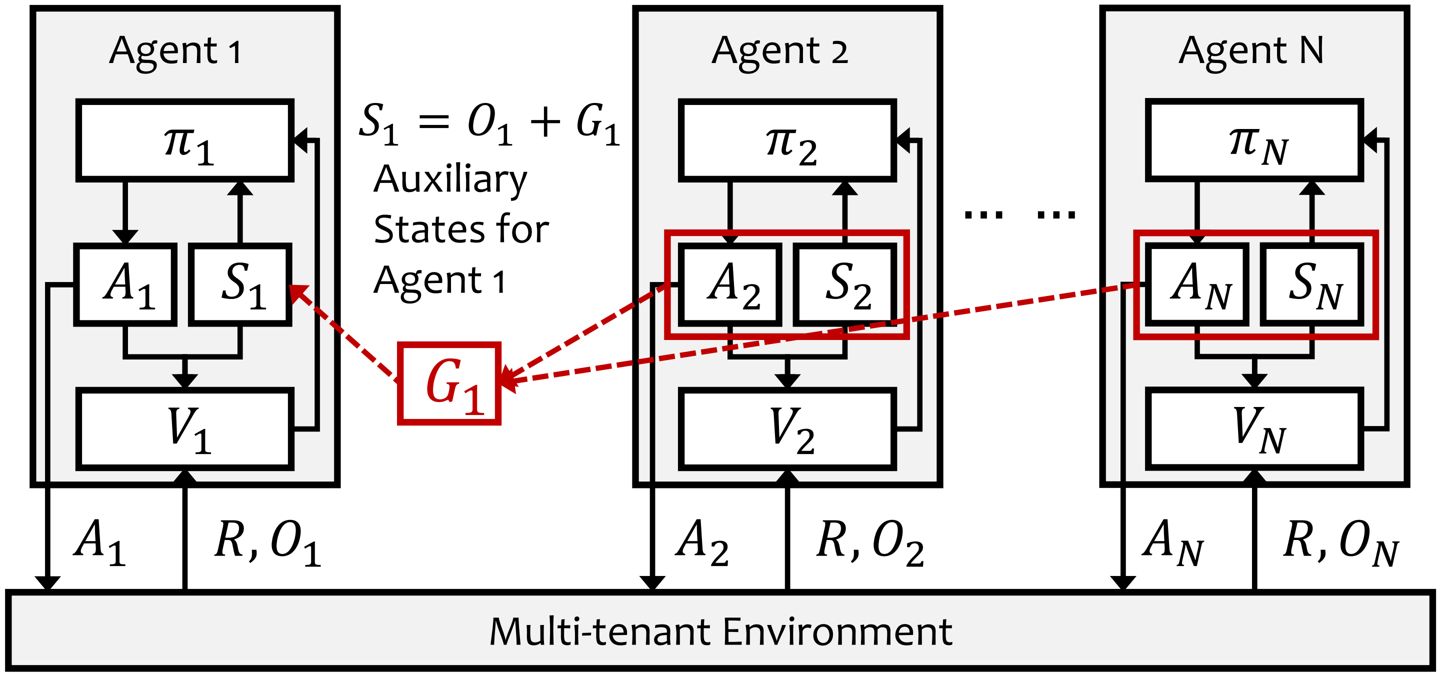
- Comparison baseline: ENSURE [ACSOS '20]
  - Heuristics-based horizontal & vertical autoscaler
- Single-agent RL achieved similar end-to-end latency with ENSURE in single-tenant cases
- Multi-tenancy results in 64-78% function performance degradation (in terms of end-to-end latency)



Multi-agent PPO (MA-PPO)

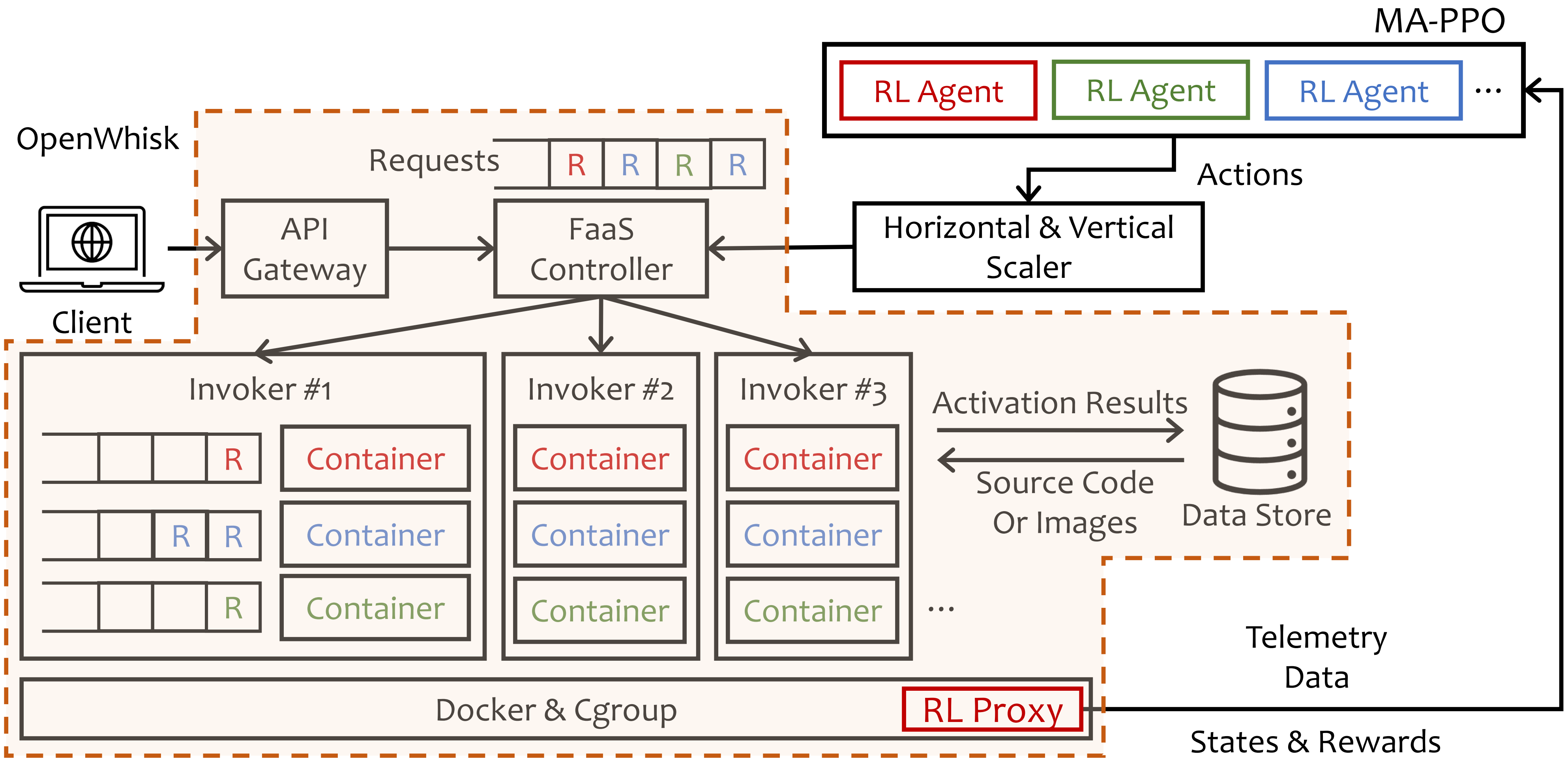
- Reward function for each agent: $R_t = \frac{\sum_i^N R(i)_t}{N}$
- Extended states for each agent: $S(i)_t = O(i)_t \cup G_t(i)$, where G_t is the global system states:
  - Aggregated horizontal actions: $AH(i)_t = \sum_{j\neq i}^N \frac{ah_t^j}{N-1}$, aggregated vertical actions: $AV(i)_t = \sum_{j\neq i}^N \frac{av_t^j}{N-1}$
  - Average SLO preservations: $ASP(i)_t = \sum_{j\neq i}^N \frac{sp_t^j}{N-1}$, average resource utilizations: $ARU(i)_t = \sum_{j\neq i}^N \frac{ru_t^j}{N-1}$

A_i: action
O_i: observation
R_i: reward
G_i: global states
π_1: policy (actor)
V_i: value function (critic)



All the other agents are treated as part of the multi-agent environment.

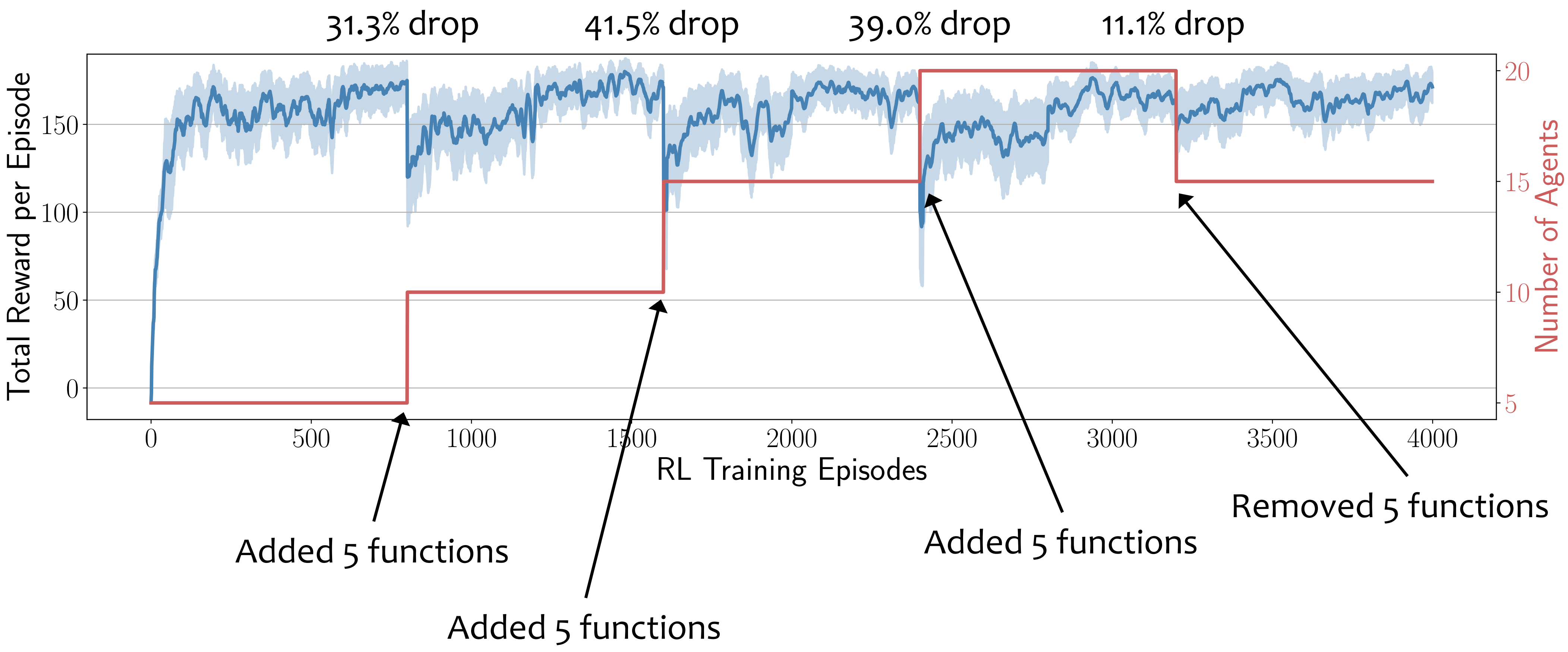
Multi-tenant RL Pipeline in OpenWhisk (with MA-PPO)



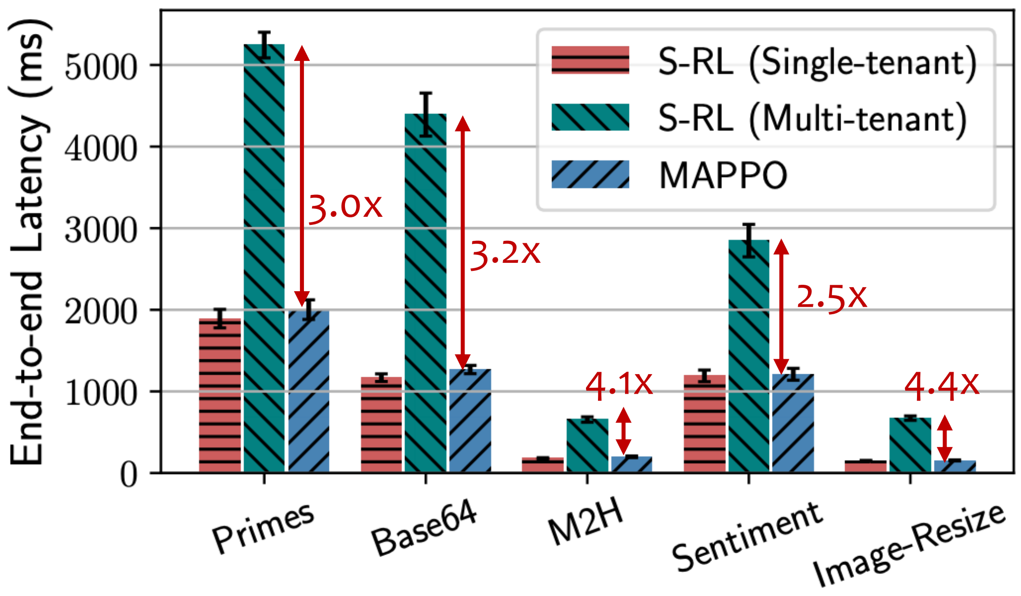
Individual local states + System states

Same rewards $R_t = \sum R(i)_t / N$

MA-PPO Training



MA-PPO Online Performance



- MA-PPO can provide online performance comparable to single-agent RL in single-tenant cases, with the performance degradation ranging from 1.8% (for sentiment-analysis, 1190.2 ms to 1211.5 ms) to 9.9% (for markdown2html, 178.4 ms to 198.1 ms).
- Compared to the single-RL trained in multitenant environments, the MA-PPO achieves 2.5× (for sentiment-analysis, 1211.5 ms to 3047.8 ms) to 4.4× (for image-resize, 154.2 ms to 672.4 ms) improvement.

Conclusion

- Single-agent RL
  - Suffers performance degradation and is unable to train (converge) in multi-tenant cases
- MA-PPO: Customized multi-agent PPO
  - Collect rewards from all agents; Agnostic to agent order and size of the agent group
    - Observations from all other agents (aggregated/averaged values)
  - Able to train (and converge) and achieves comparable performance in multi-tenant cases compared with single RL in single-tenant cases
- Future work
  - Fast retraining: Network parameter sharing, transfer learning
  - Fault tolerance

Thank you!