SIMPPO: A Scalable Online Learning Framework for Serverless Resource Management

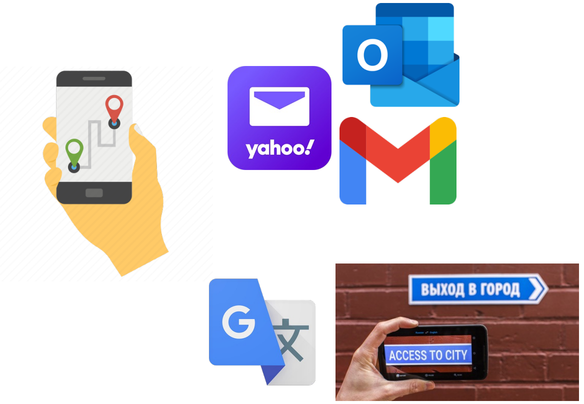
Haoran Qiu[1], Weichao Mao[1], Archit Patke[1], Chen Wang[2], Hubertus Franke[2]

Zbigniew T. Kalbarczyk[1], Tamer Basar[1], Ravishankar K. Iyer[1]

[1] UIUC [2] IBM Research

Managing SLOs in Serverless Platforms

- Meeting **Service-level Objectives (SLOs)** is critical to the success of a serverless platform, especially for user-facing applications
  - Today, performance SLOs are not supported yet in Serverless (FaaS)
- **Providing per-function performance-wise SLO agreements**
  - Customers agree for the vendor to manage the resources provided so long as SLOs are met (e.g., latency)
  - Both customers and vendor potentially benefit from meeting the SLOs



User-facing Services

FaaS Platforms

Monthly **Uptime** Percentage $\geq$ 99.95%

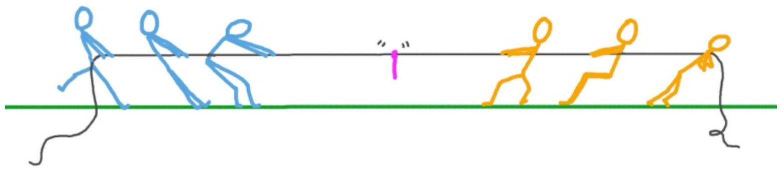
Managing SLOs in Serverless Platforms

Tension between Provider and Customers

Need for Lower latency

Desire for Higher utilization, Higher profit

Customers



FaaS Provider

Challenge: Optimizing for diverse workloads to **meet SLO constraints** while **efficiently** multiplexing shared resources

[Qiu, WoSC 2021] Is Function-as-a-Service a Good Fit for Latency-Critical Services?
In Proceedings of the 7th International Workshop on Serverless Computing (WoSC7) Co-located with ACM Middleware 2021

Managing SLOs in Serverless Platforms

Tension between Provider and Customers

Need for Lower latency

Desire for Higher utilization, Higher profit

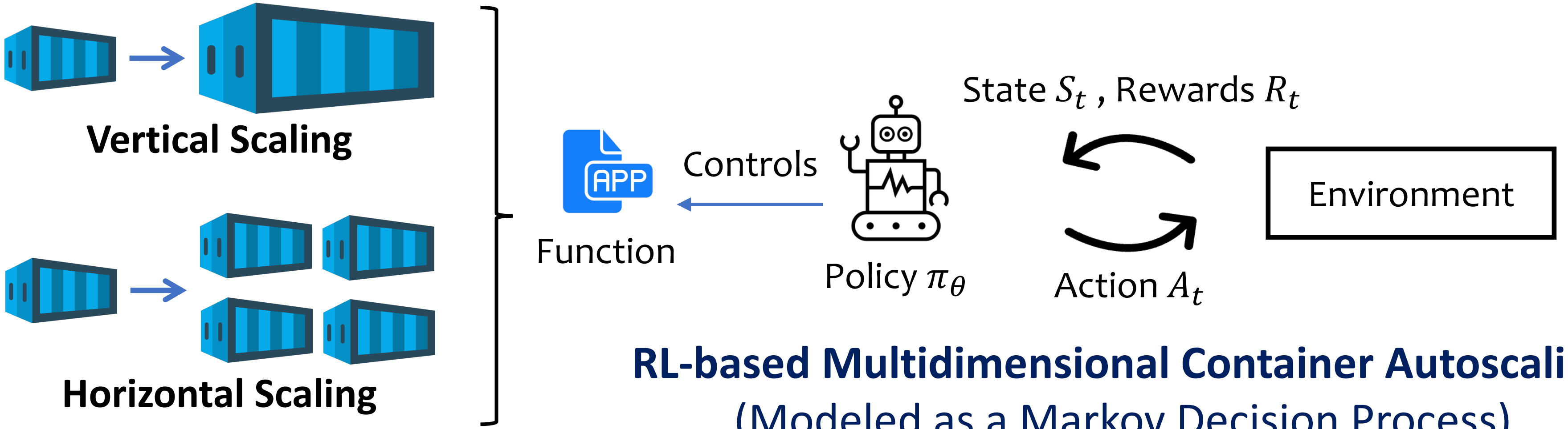
Serverless computing provides a unique opportunity!

Challenge: Optimizing for diverse workloads to **meet SLO constraints** while **efficiently** multiplexing shared resources

[Qiu, WoSC 2021] Is Function-as-a-Service a Good Fit for Latency-Critical Services?
In Proceedings of the 7th International Workshop on Serverless Computing (WoSC7) Co-located with ACM Middleware 2021

ML-managed SLO-driven Cloud Services

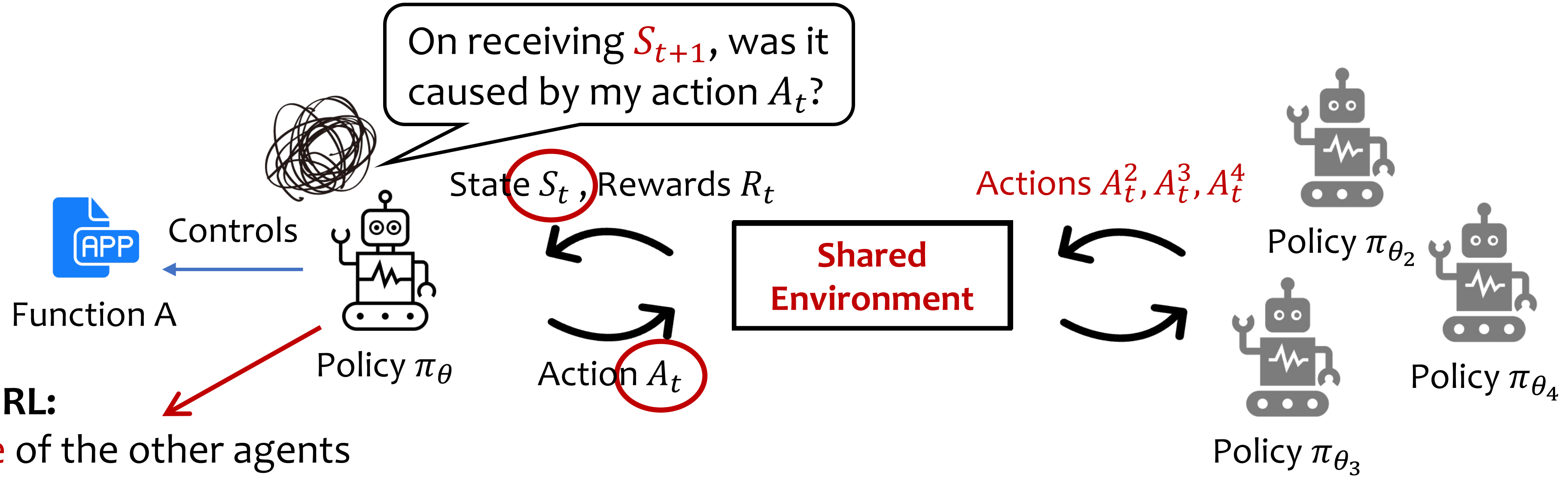
- **Why ML?:** Heuristics-based resource management are inefficient and not tenable
  - Providers dynamically manage orchestration platforms to achieve efficiency as cloud evolves
- **Contributions:**
  - **SIMPPO**: Automate the management for diverse workloads with **reinforcement learning (RL)**
  - Quantitative characterization study of existing RL approaches
  - A system that orchestrates multiple learning-based agents to achieve optimal resource allocation in the task of **multidimensional container autoscaling**
  - **Key Idea**: "Virtual Agent" and mean-field theory



RL-based Multidimensional Container Autoscaling
(Modeled as a Markov Decision Process)

Wait! My RL solutions fail in production?

- Existing RL solutions: A single RL agent in an isolated environment or **single-agent RL**
  - DeepRM [HotNets '16], MIRAS [ICDCS '19], **FIRM [OSDI '20], Symphony [ICML '20]**, ADRL [TPDS '20], Q-learning-based Autoscaler [CCGrid '21], SOL [ASPLOS '22], …
  - **Not yet ready in production systems**
- 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 manage diverse function workloads



Single-agent RL:
- Not aware of the other agents
- Trained independently

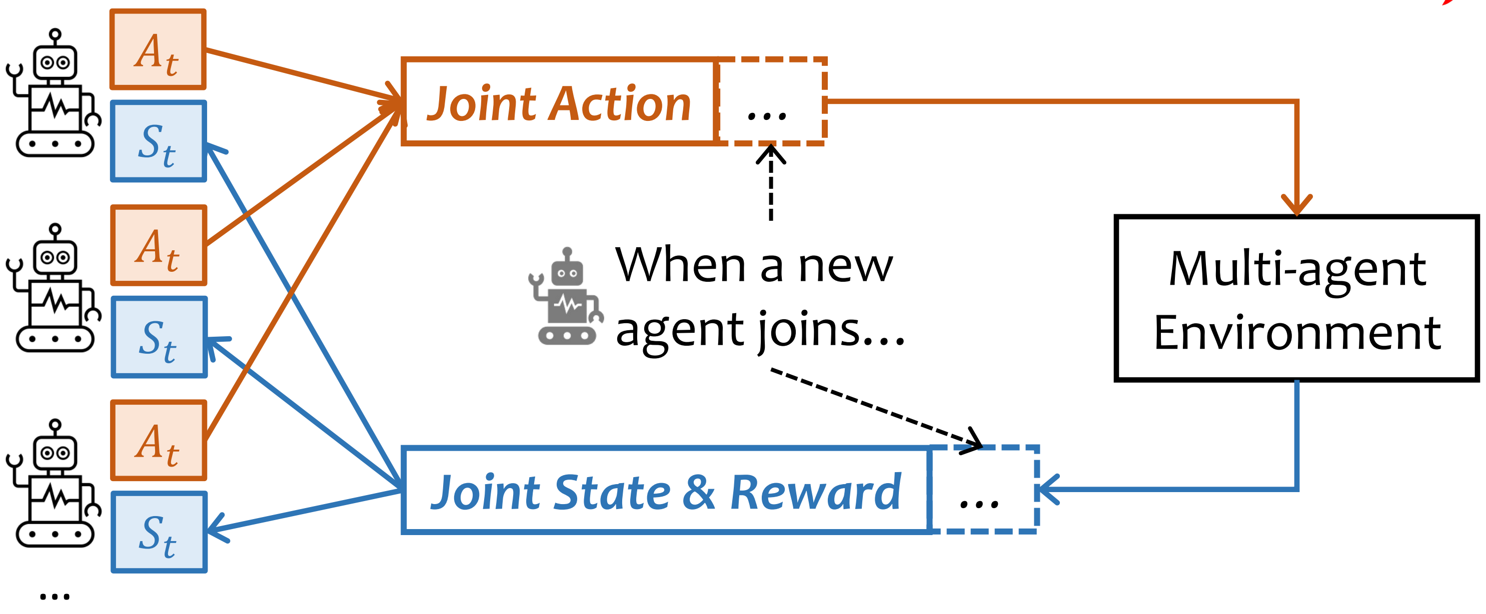
A Naïve Multi-agent RL (MARL) Solution

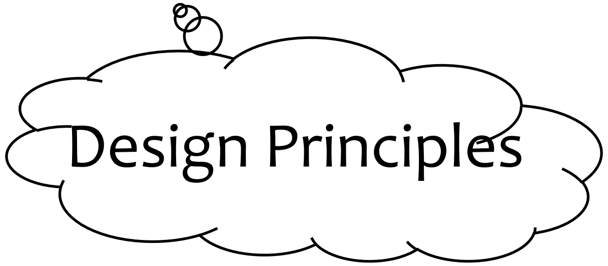
- **Joint Action Learner (JAL):** A centralized policy for all the agents is trained
  - **Input:** the concatenation of the observations of all the agents
  - **Output:** the joint actions specified to the all agents
- Eliminates the problem of non-stationarity (theoretically)
  - Allows for joint training and joint inference
- Computationally inefficient with exponential complexity
- Retraining from scratch needed for any agent group changes

Performance Isolation?

Scalability?

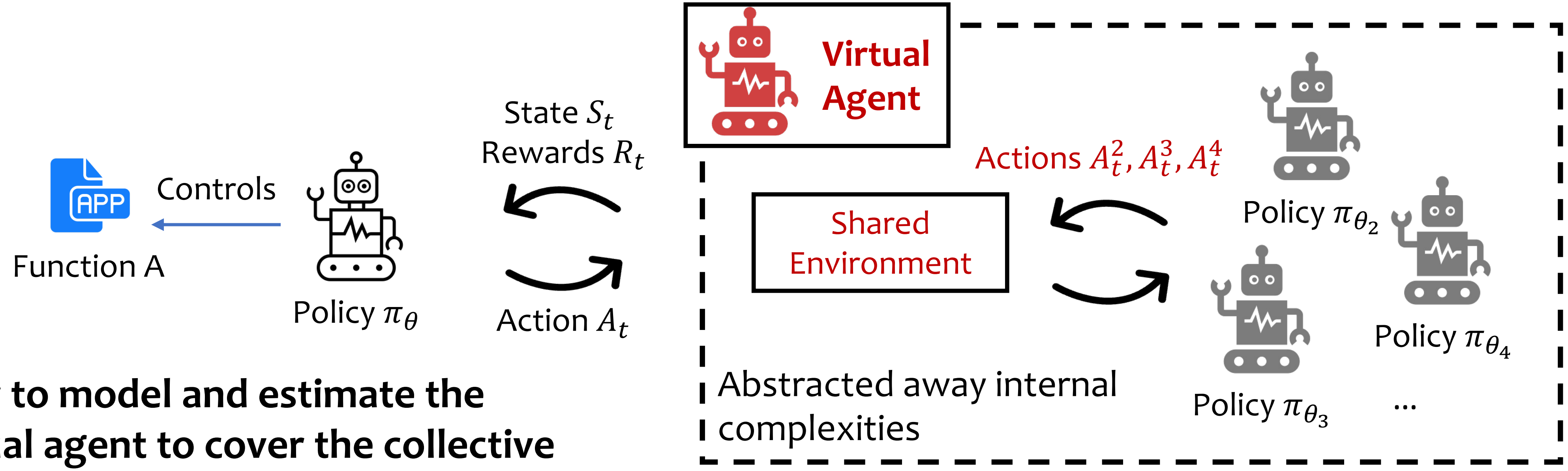
Incremental Retraining?





Rethinking the MARL Model

- **What can we do about the changing agent group?**
  - For each agent, we treat the other agents as part of the environment
- **Virtual agent = Environment + All Other Agents**
  - Many-agent problem converted to a two-agent problem
  - Agnostic to agent sequence order or the number of agents -> **Incremental Retraining**
- Neural network architecture redesigned
  - No need to reconstruct the neural network (structure remains unchanged)

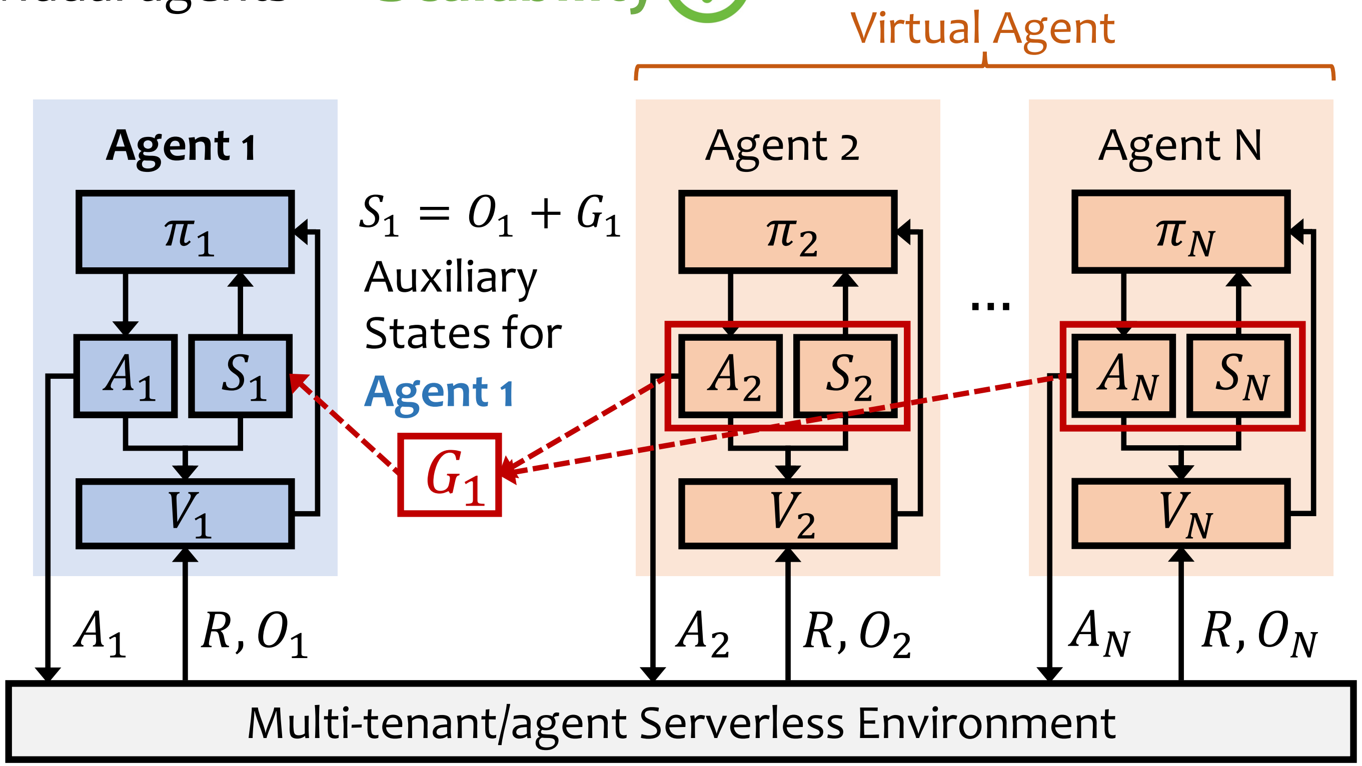


How to model and estimate the virtual agent to cover the collective dynamics?

Virtual Agent State Estimation

- Model the collective behavior of virtual agent via an **auxiliary global state distribution**
  - Aggregated actions and resource limits to represent the collective resource allocation
  - Average function performance and resource utilization to indicate how the virtual agent behaves
- Provided to each agent to learn the collective and average behavior of the virtual agent instead of all the other individual agents -> **Scalability**

A_i: Action
O_i: Observation
R: Reward
G_i: Global states
π_i: Policy (actor)
V_i: Value function (critic)



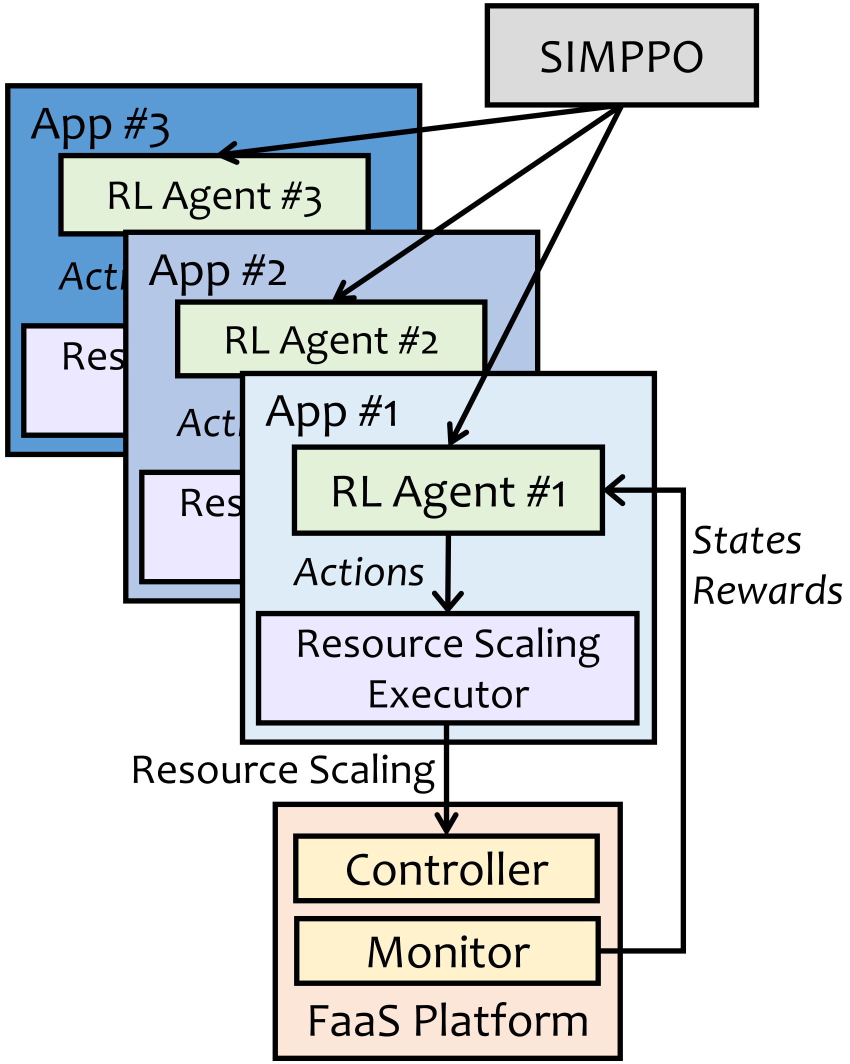
Mean-field Theory:
Modeling the collective behavior of N agents by the mean-field states provides tractable and accurate approximation of the actual N-agent scenario[1]

[Mao & Qiu, NeurIPS 2022]

[1] Weichao Mao, Haoran Qiu, et. al. A Mean-Field Game Approach to Cloud Resource Management with Function Approximation.

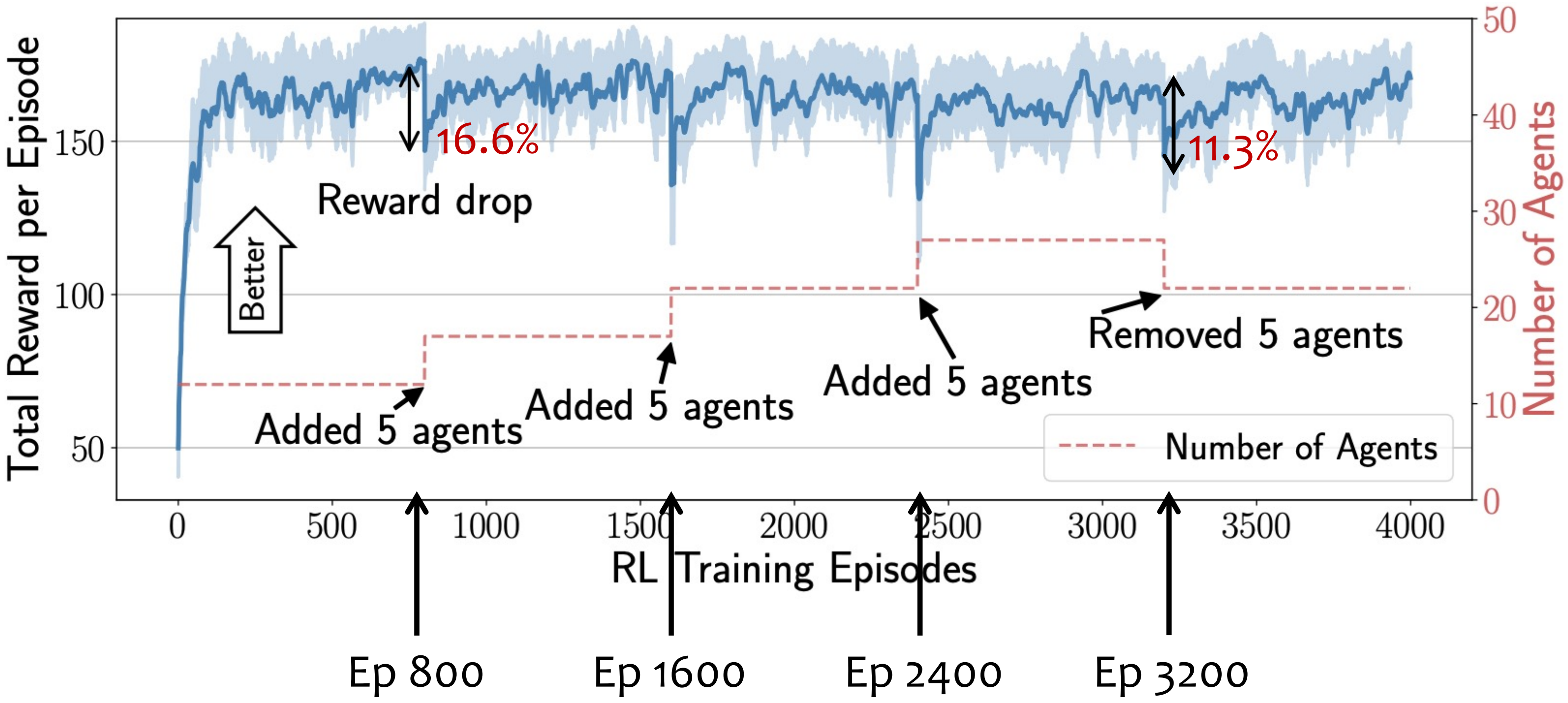
SIMPPO: Scalable and Incremental MARL

- Two building blocks of SIMPPO
  - Virtual agent
  - Auxiliary global system states
- Applied SIMPPO to multi-dimensional autoscaling of serverless platforms
  - Based on the state-of-the-art RL algorithm – PPO (Proximal Policy Optimization)
  - Serverless platform: OpenWhisk
- Evaluated SIMPPO on 12 open-source serverless benchmarks
  - Function invocation patterns from Azure Functions traces
- **RQ1: Incremental training?**
- **RQ2: Online policy-serving performance?**



SIMPPO Incremental (Re)Training

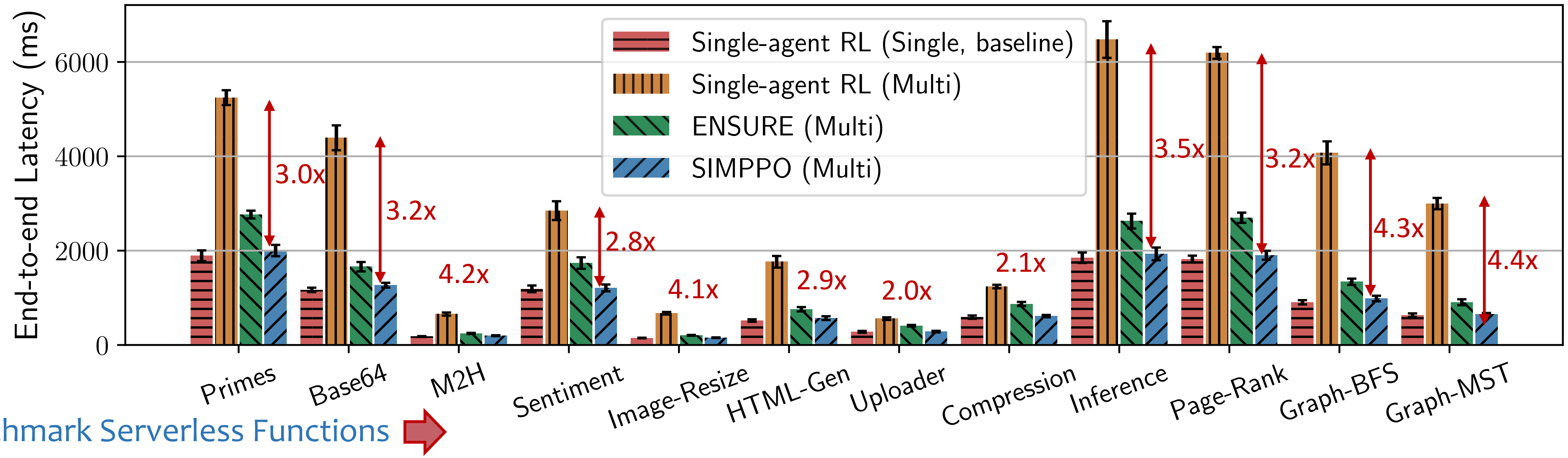
Does SIMPPO converge and support incremental training? What is the value of the auxiliary global system states?



Training curves of SIMPPO in multi-agent environment:

- Continuously reaching the convergence after incremental retraining

SIMPPO Online Performance



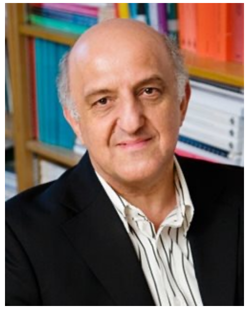
- SIMPPO provides online policy-serving performance comparable to single-agent RL in isolation (the baseline), with the performance degradation <9.2%
- In multi-tenant/agent environments:
  - SIMPPO achieves 2x-4.4x improvement compared to single-agent RL
  - SIMPPO has 21.4x less performance degradation compared to a threshold-based approach ENSURE (ACSOS 2020)

Final Words

- **SLO management in serverless platforms is critical but challenging!**
  - Serverless provides the unique opportunity of optimization of resources
  - Incorporating performance-wise SLO/SLA in the pricing model?
- **SIMPPO: Scalable and incremental multi-agent RL framework based on PPO**
  - **Key idea:** Virtual agent and auxiliary global system states
  - Able to train to convergence and achieves performance isolation
- **Limitations and Future Work**
  - Incorporating resilience management
    - Fault tolerance (e.g., agent state transition loss, agent disconnection)
  - Consideration of serverless function chains or function graphs (DAGs)
  - **Stay tuned:** Multidimensional Pod Autoscaler in Kubernetes
    - RL-based Autoscaler for general Kubernetes deployment






Thank you!

Haoran Qiu[1], Weichao Mao[1], Archit Patke[1], Chen Wang[2], Hubertus Franke[2]

Zbigniew T. Kalbarczyk[1], Tamer Basar[1], Ravishankar K. Iyer[1]

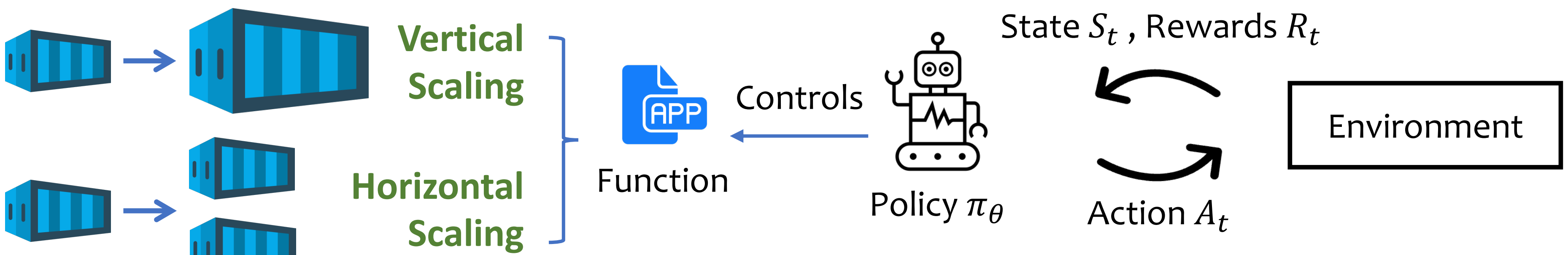
[1] UIUC [2] IBM Research

- Check out the extended version of the paper for more details: https://haoran-qiu.com/pdf/simppo-extended.pdf
- Check out our paper published at NeurIPS 2022 for the theoretical results: https://haoran-qiu.com/pdf/nips22.pdf

Backup Slides

ML-managed SLO-driven Cloud Services

- **Why ML?:** Heuristics-based resource management are inefficient and not tenable
  - Providers dynamically manage orchestration platforms to achieve efficiency as cloud evolves
- **SIMPPO**: Novel **ML-driven solutions** to automate the management of serverless platforms for diverse customer workloads with **reinforcement learning (RL)**
  - A system that orchestrates multiple learning-based agents to achieve optimality
  - Provides a **theoretical foundation** for generalization and evolution of SIMPPO (**NeurIPS 2022**)

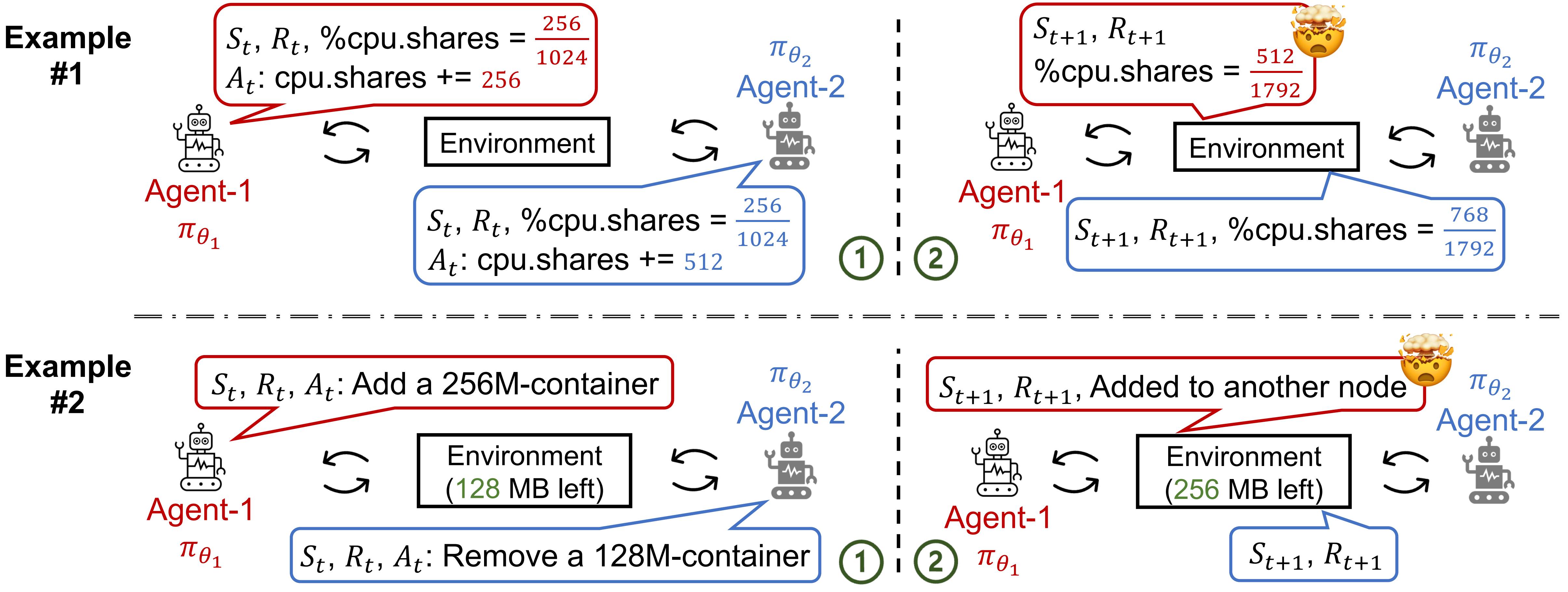


RL-based multidimensional container autoscaling (Modeled as a Markov Decision Process)

Central Idea: Minimize SLO violations by using the means of multi-dimensional workload distributions. Each dimension represents a key system attribute (e.g., resource utilization, tail latency).

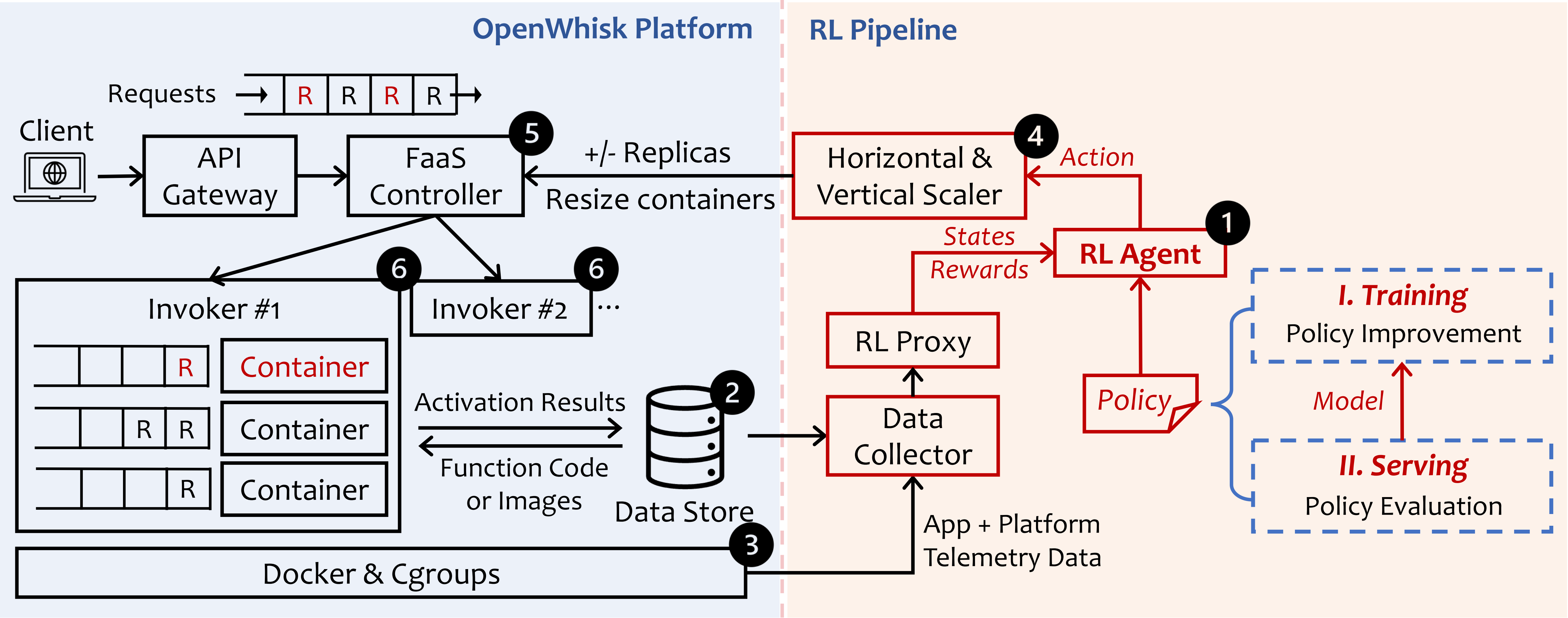
Motivating Examples

: RL agent (per Function) : RL transitions : Denote that current state is undesired π_θ: RL Policy S_t: State at time t A_t: Action at time t R_t: Reward at time t



Deeper Dive into the Popular Single-agent RL Design

[FIRM, 2020 OSDI]



R: Denotes the requests to the function managed by the RL agent.

Check out our paper for more Details!

Single-agent RL Design

- **RL Algorithms: PPO (Proximal Policy Optimization)**
  - A policy gradient method and the default RL algorithm in OpenAI
- **State Space:** 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)
- **Action Space:**
  - Vertical scaling: +/- a STEP_SIZE of the resource limits
    - $av_t = \Delta RLT_t(CPU, mem)$
  - Horizontal scaling: +/- a STEP_SIZE of the number of function containers
    - $ah_t = \Delta NC_t$
- **Reward Function:**

$$R_t = \underbrace{\alpha \cdot RU_t}_{\text{Resource Util}} + \underbrace{\beta \cdot SP_t}_{\text{SLO Preservation}} + \underbrace{penalty}_{\text{Penalize illegal or undesired actions}}$$

Resource Util

SLO Preservation

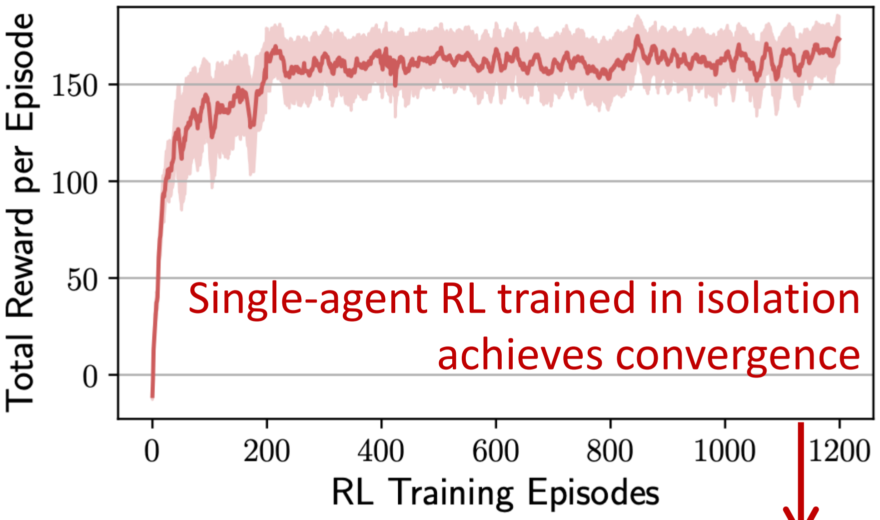
$$SP_t = \min\left(\frac{SLO\ Latency}{Actual\ Latency}, 1\right)$$

Penalize illegal or undesired actions

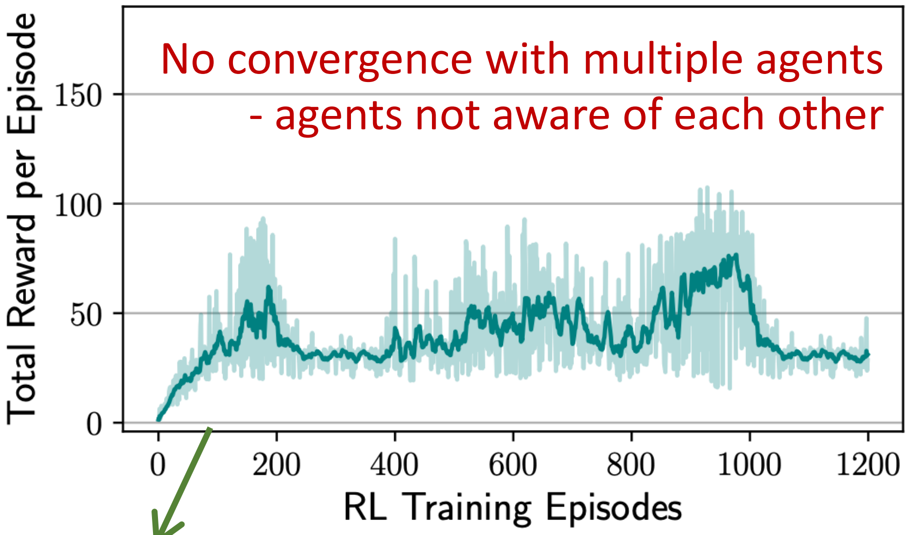
- Frequent dangling decisions
- Scale in/up/down when $NC_t = 0$

Single-agent RL Evaluation on 12 FaaS Benchmarks

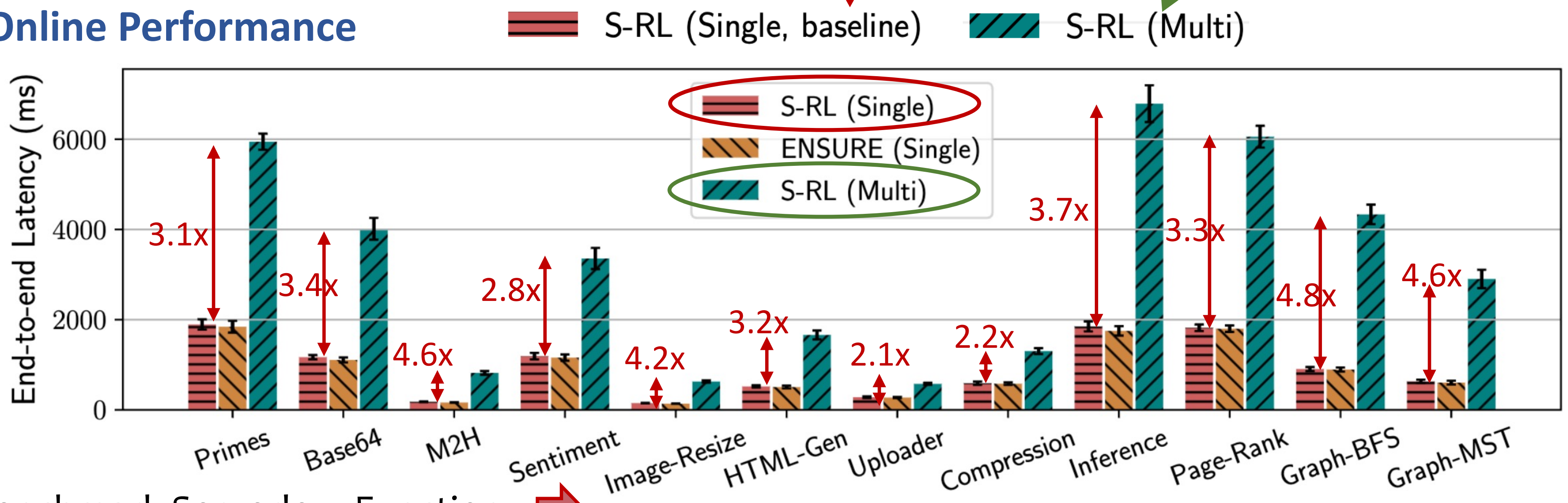
Training in Isolation



Training with Multiple Agents



Online Performance



Benchmark Serverless Functions

Independently trained RL agents result in 2.1x to 4.8x (up to 80%) function execution latency degradation

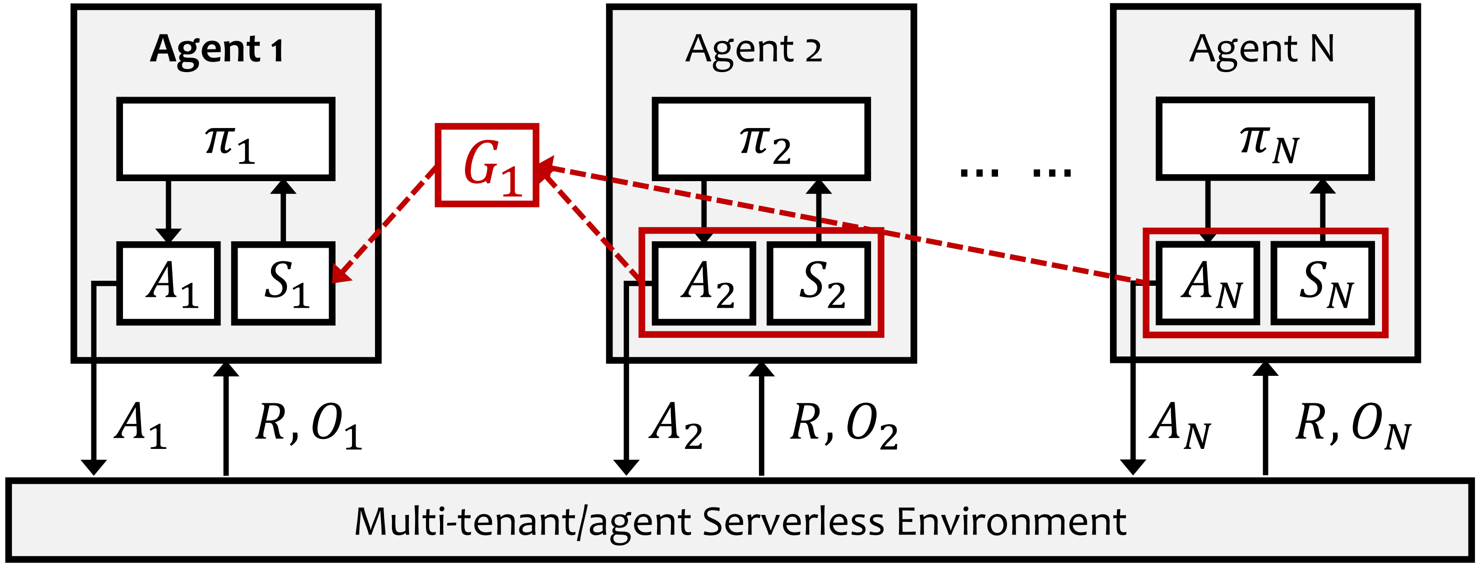
Moving to Multi-agent RL (MARL)

- Our approach: Providing system support that enables multiple RL-based controllers to coexist
  - **Performance Isolation**
    - During training: Converges to a collectively optimal policy
    - During execution: Achieves comparable performance to single-agent RL in isolation
  - **Scalability**
    - Challenge: In a multi-tenant serverless FaaS platform, new functions can be increasingly registered
  - **Incremental training/retraining (adaptability)**
    - Challenge: Functions can be registered/removed/updated at any time, changing the joint state space

SIMPPO Online Inference

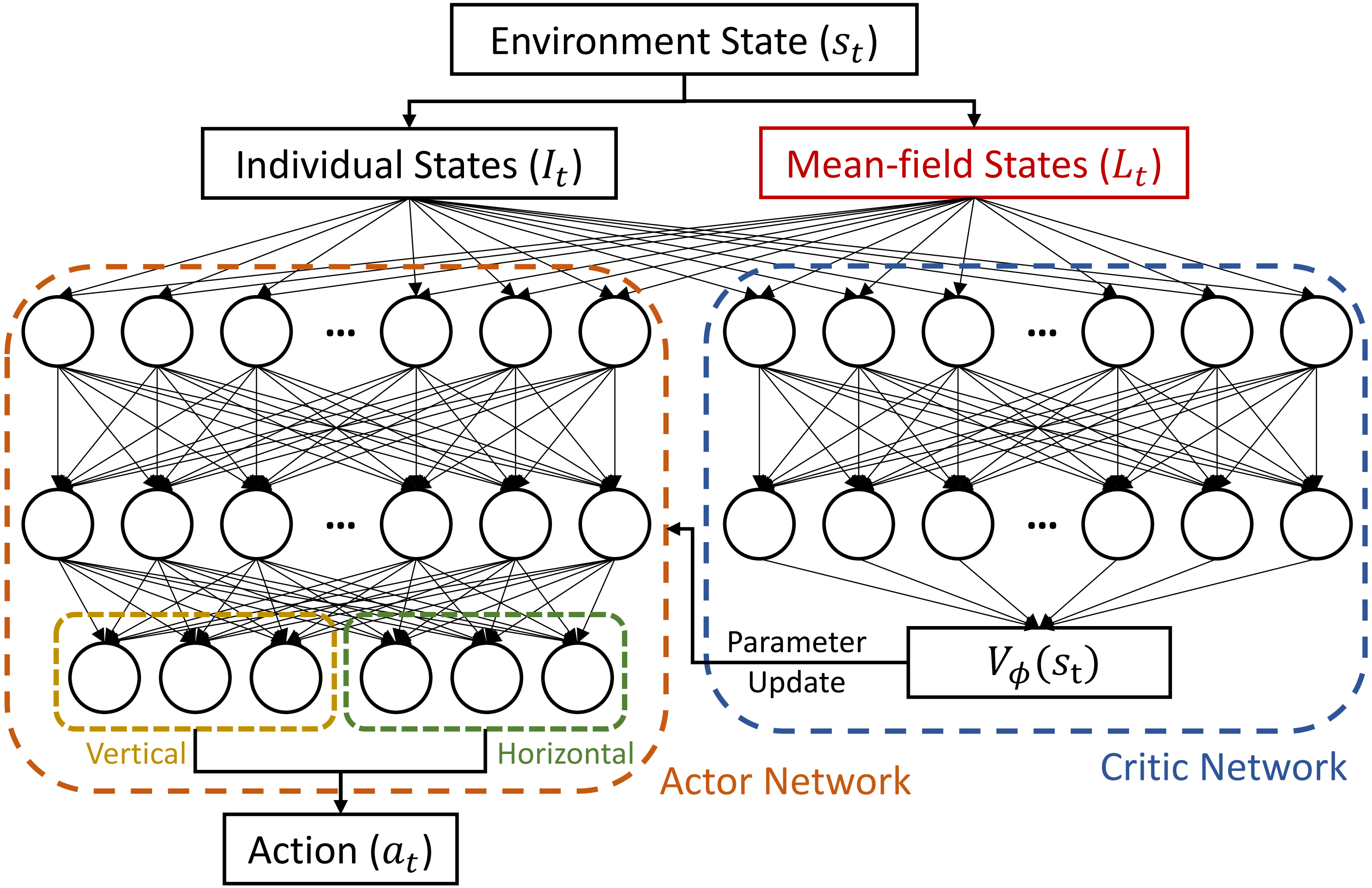
- Each RL agent is plug-and-play onto different servers
- Scalable state & action space: agent trained w/ X agents applicable to scenarios w/ Y agents
  - Zero values for empty RL agent slots
  - Other RL agents treated as part of the environment → agent-order-agnostic
    - Observations from all other agents (aggregated/averaged values)

A_i: Action
O_i: Observation
R: Reward
G_i: Global states
π_i: Policy (actor)
V_i: Value function (critic)



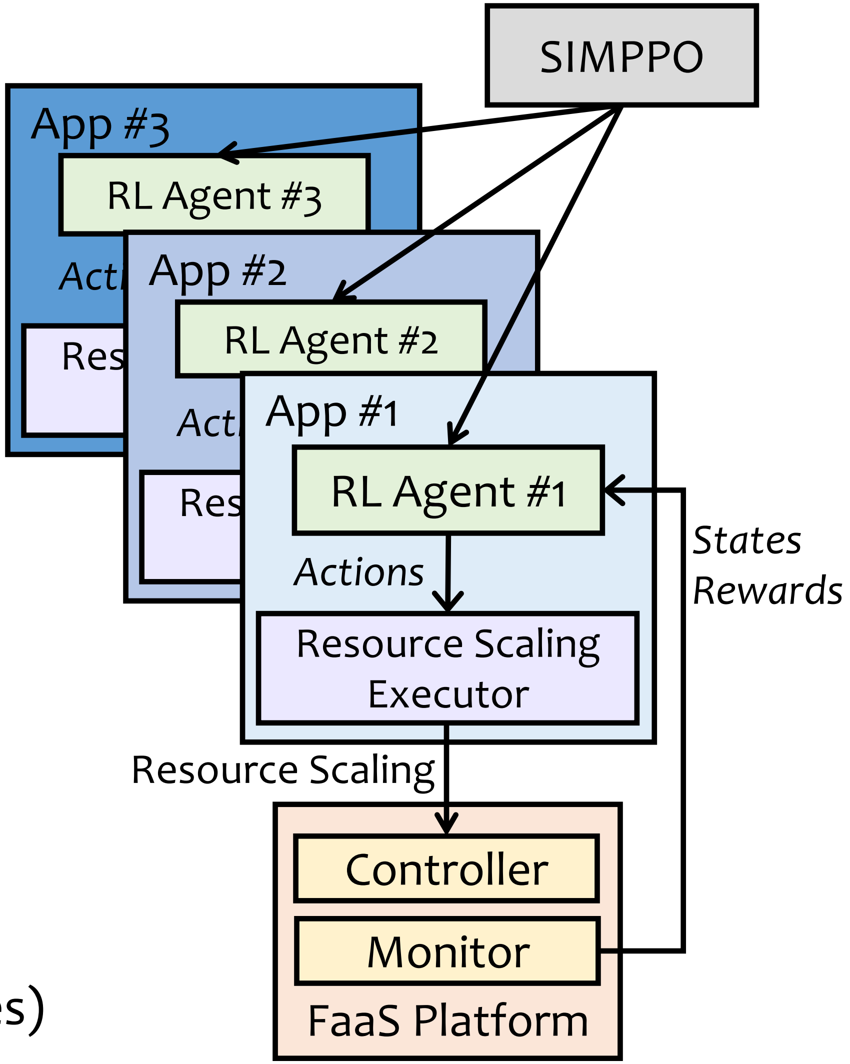
SIMPPO Neural Network Model architecture

- Model architecture



Other Use Cases

- SIMPPO as a general framework to support a variety of RL agents that employ online learning algorithms
- Assumptions / Pre-conditions:
  - Conflicts between agents in a shared environment
  - Visibility of the states of the other agents
  - Agent coming from the same distribution (same task + reward)
- Use cases:
  - **RL-based serverless resource management**
    - This work
  - **RL-based network flow congestion control (Aurora, ICML 2019)**
    - Shared network bandwidth
  - **RL-based video adaptation (streaming) (ABRL, ICML 2019)**
    - Shared network and video content server
  - **RL-based job scheduler (Decima, SIGCOMM 2019)**
    - Shared cluster resources and low-level task scheduler (queues)



SIMPPO Incremental and Scalable Training

RQ: Is SIMPPO scalable with respect to converged rewards, online policy-serving performance, and retraining time?

- As the # of functions increases from 5 to 110, the reward drop % first increases and then decreases after the # of functions is >20.
- The retraining time has similar trends, as retraining is done until the per-episode reward converges to a stable value.
- As the number of functions increases to 110, the reward drop % and the retraining cost decrease to 3.0% and 47.2 training episodes.