Before Starting my PhD

Summer Research Internship at University of Luxemburg on "*Designing Scientific Applications on GPUs*"

Remote 07/2021 – 08/2021

- Worked with another undergraduate student to optimize a topology optimization algorithm written in PETSc, a C++ library for scientific computations
- Accelerated computation by migrating heavy operations from CPU to GPU

Master Thesis: "Transparent Control Hand-Over of Aerial Drones Between Ground Stations"

- **Theodoros Aslanidis**, Manos Koutsoubelias, Spyros Lalis. *Transparent Handover of Automated Drone Missions between Edge-based Stations*. In 20th International Conference on Embedded Wireless Systems and Networks (EWSN), 2023

During my PhD

Publications

Theodoros Aslanidis, Andreas Chouliaras, Dimitris Chatzopoulos. *Reinforcement Learning Techniques for Optimizing System Configuration on the Cloud: A Taxonomy and Open Problems*. In 1st International Workshop on Machine Learning for Autonomic System Operations in the Device-Edge-Cloud Continuum (MLSysOps@EWSN), 2023

Theodoros Aslanidis, Sokol Kosta, Spyros Lalis, Dimitris Chatzopoulos, *Cross-Domain DRL Agents for Efficient Job Placement in the Cloud-Edge Continuum*. In 5th Workshop on Machine Learning and Systems (EuroMLSys@EuroSys), 2025

Talks

Resource allocation in the cloud-edge continuum using reinforcement learning techniques. UCD DAIS Research Visit at Qualcomm, Cork, Ireland, 2023

End-to-end autonomic management of Cloud-Edge Continuum Systems using Reinforcement Learning: Design Approach and Formulation. In 1st Workshop on Enabling Technologies and Dependability in Cyber-Physical Systems (ENHANCE@HiPEAC), 2024

Posters

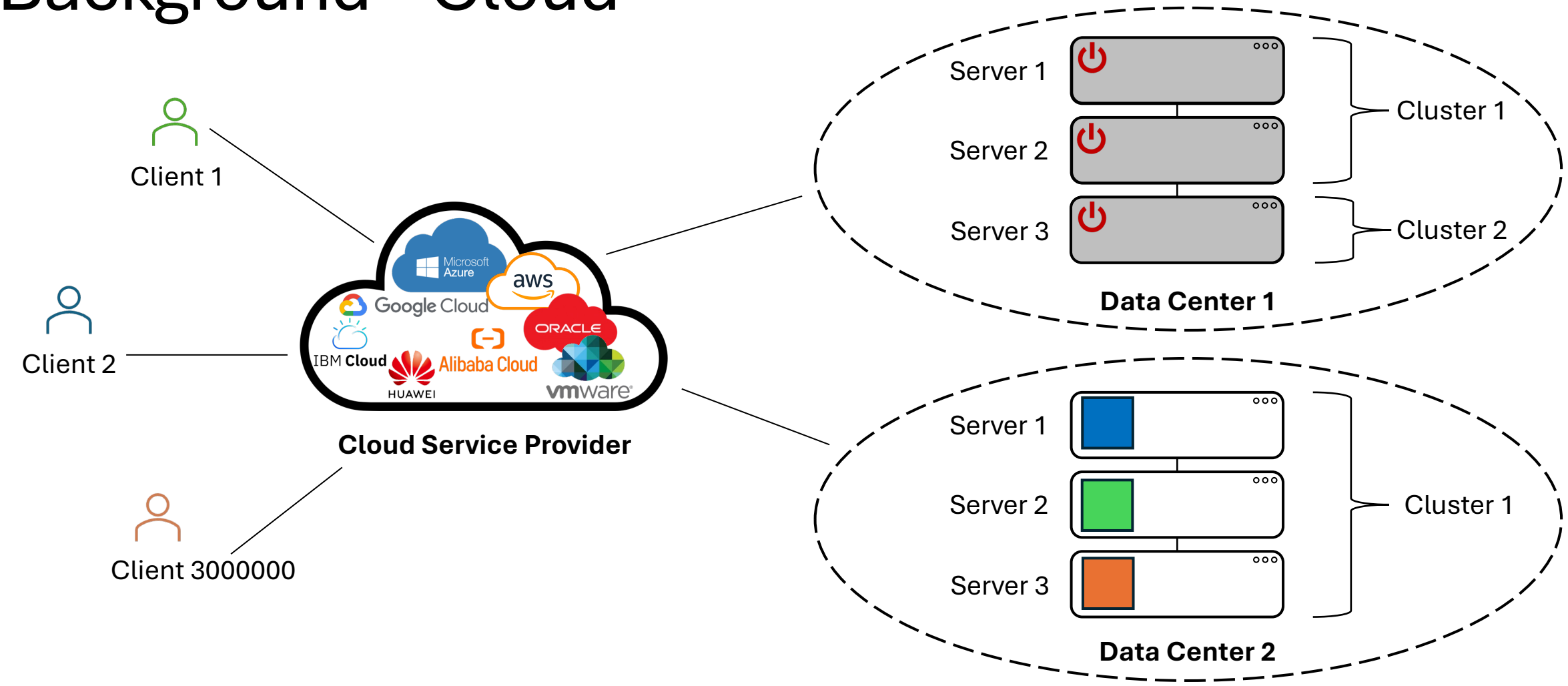
MLSysOps: Machine Learning for Autonomic System in the Heterogeneous Edge-Cloud Continuum. UCD STEM Early Career Researchers Symposium, 2023

Cross-Domain DRL Agents for Efficient Job Placement in the Cloud-Edge Continuum (EuroMLSys@EuroSys), 2025

Current Status

Work Title	Action Point	Notes
Reusable Deep RL Agents for VM Management Across Diverse Cloud Infrastructures	Waiting for Results (June 23, 2025)	• Submitted to ACM MobiHoc 2025 • Previously rejected from USENIX ATC 2025
Cross-Domain DRL Agents for Efficient Job Placement in the Cloud-Edge Continuum	Extend for Journal Publication	• Submit to IEEE Transactions on Cloud Computing 2025
TBD	Work on New Research Paper	• Enhance KAI-Scheduler with GVirtuS to enable GPU virtualization and seamless remote GPU access nodes in a heterogeneous cluster

Background - Cloud



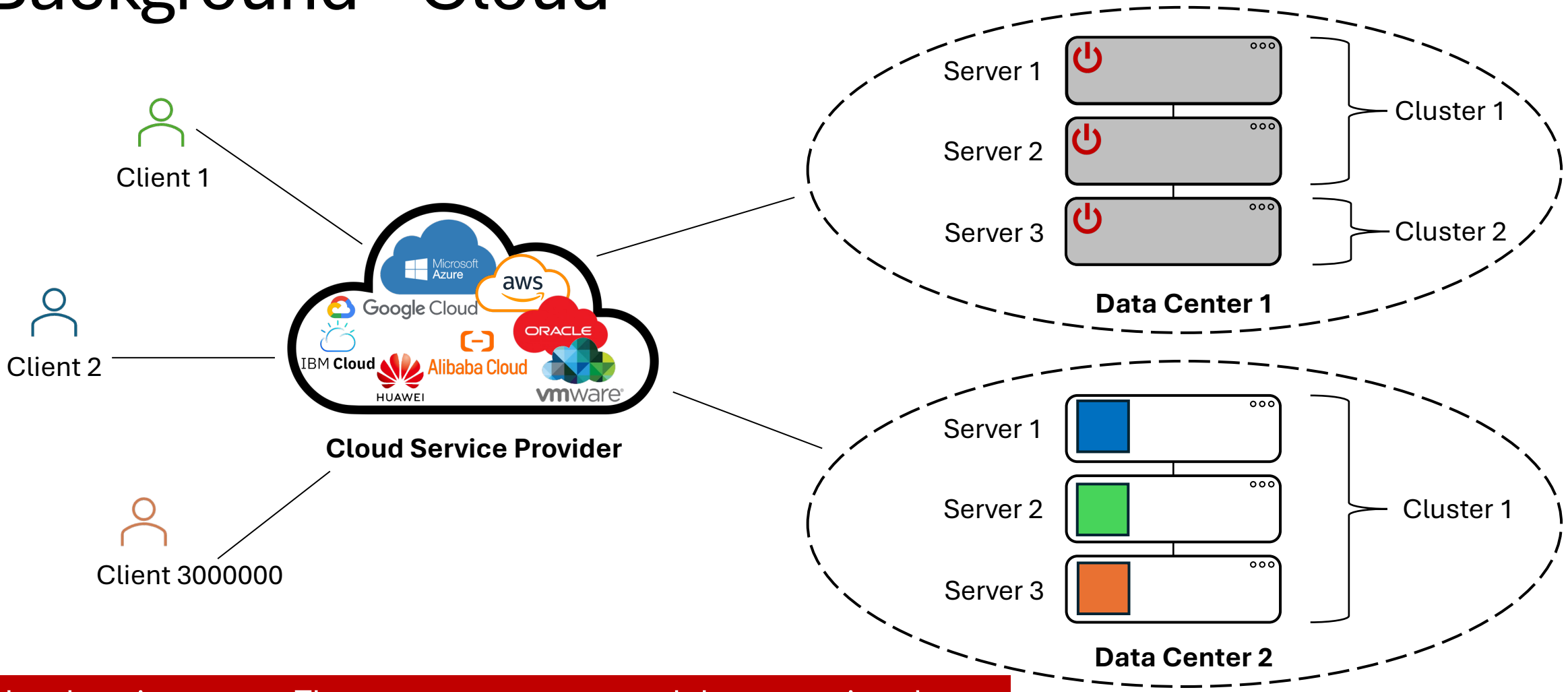
1% improvement in resource management, can save hundreds of millions of dollars. [1]

Service quality in cloud systems is strongly correlated with client satisfaction. [2]

[1] Hadary, Ori, et al. "Protean:{VM} allocation service at scale." *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*

[2] Agarwal, Rajesh, and Sanjay Dhingra. "Factors influencing cloud service quality and their relationship with customer satisfaction and loyalty." Heliyon 9.4 (2023).

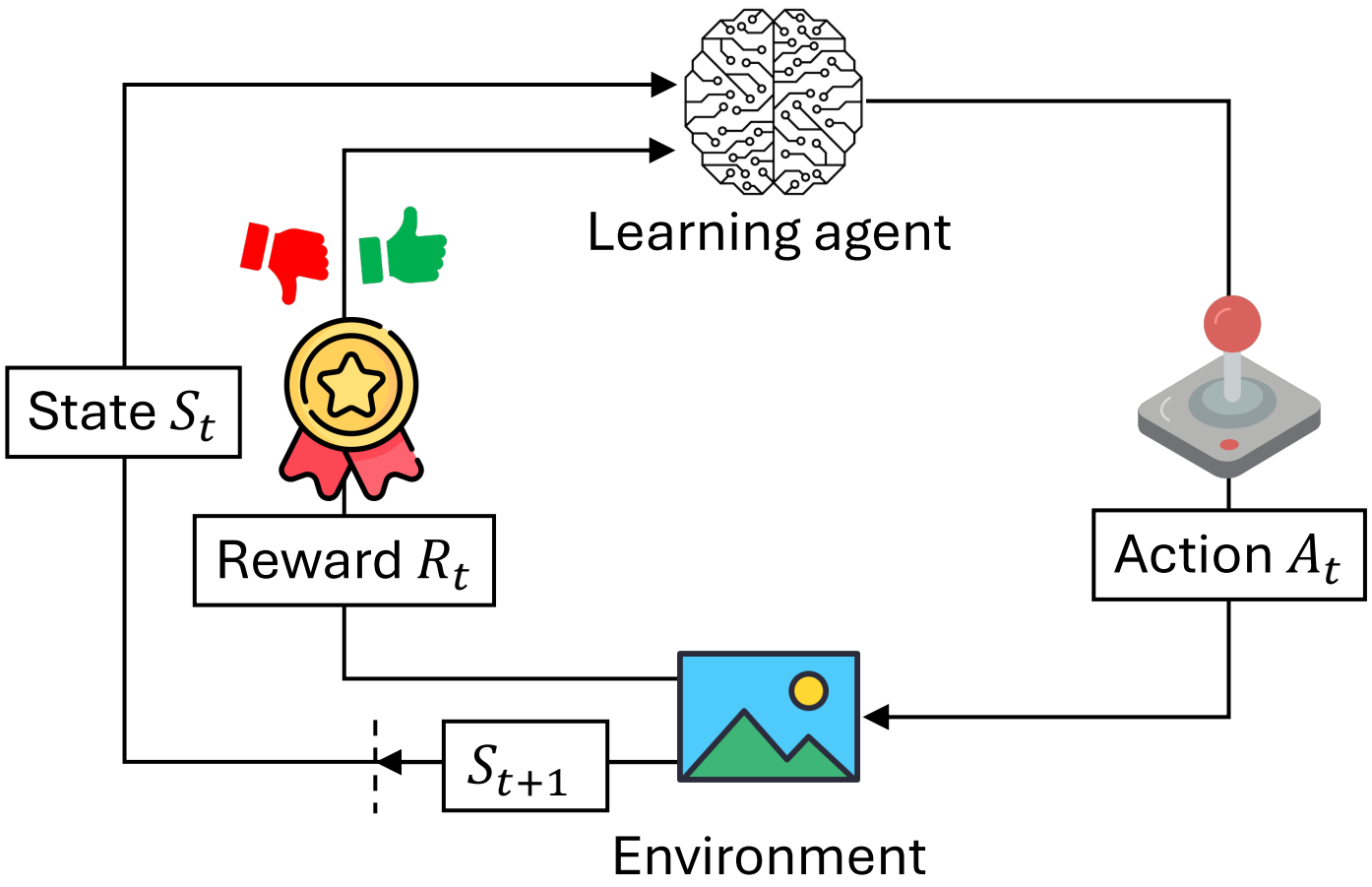
Background - Cloud



Cloud environment: The request patterns and the operational costs of cloud resources are highly volatile due to various reasons

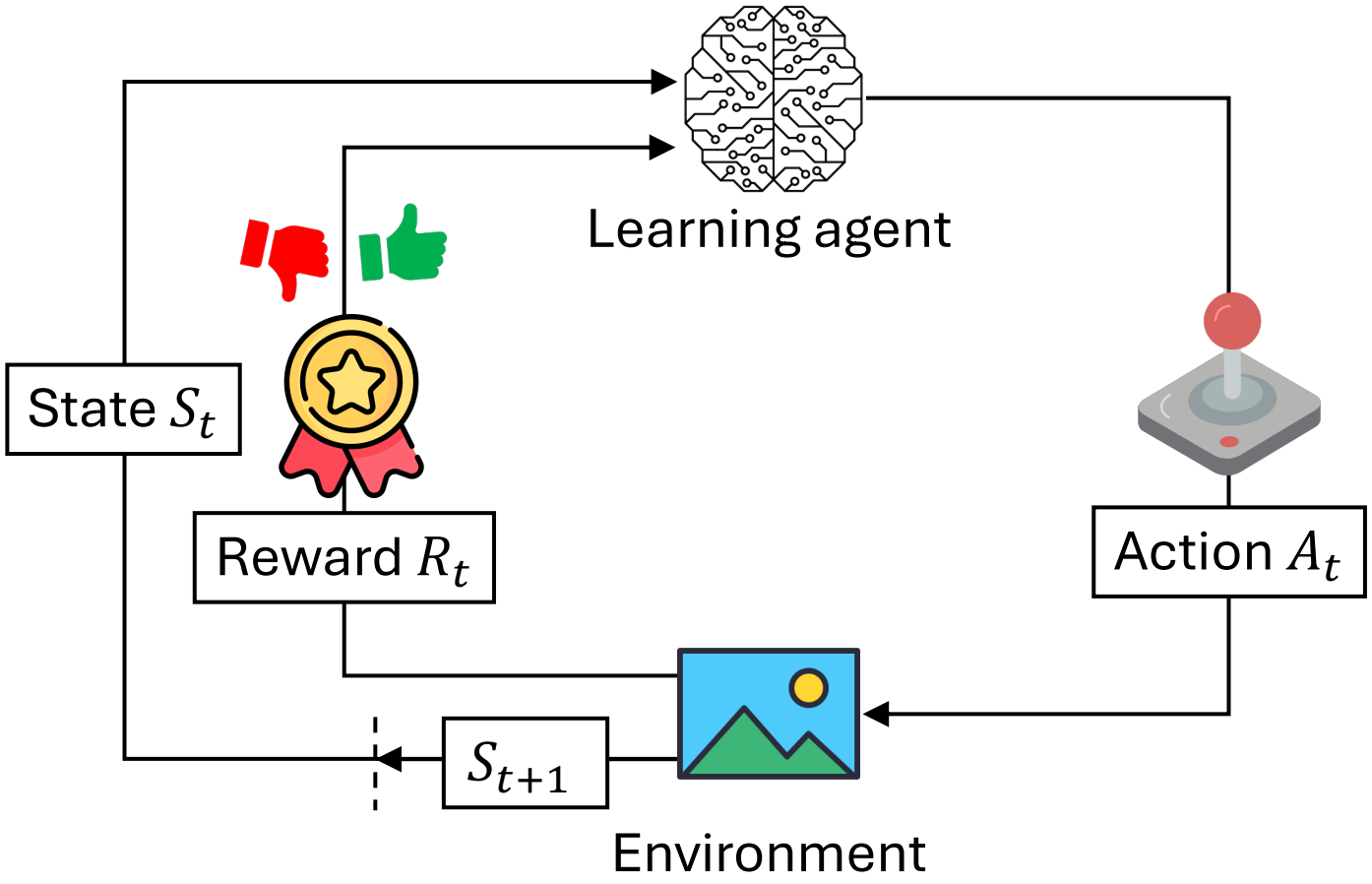
Research focus: Use RL for optimizing the management of cloud resources to balance computational expenses and service quality

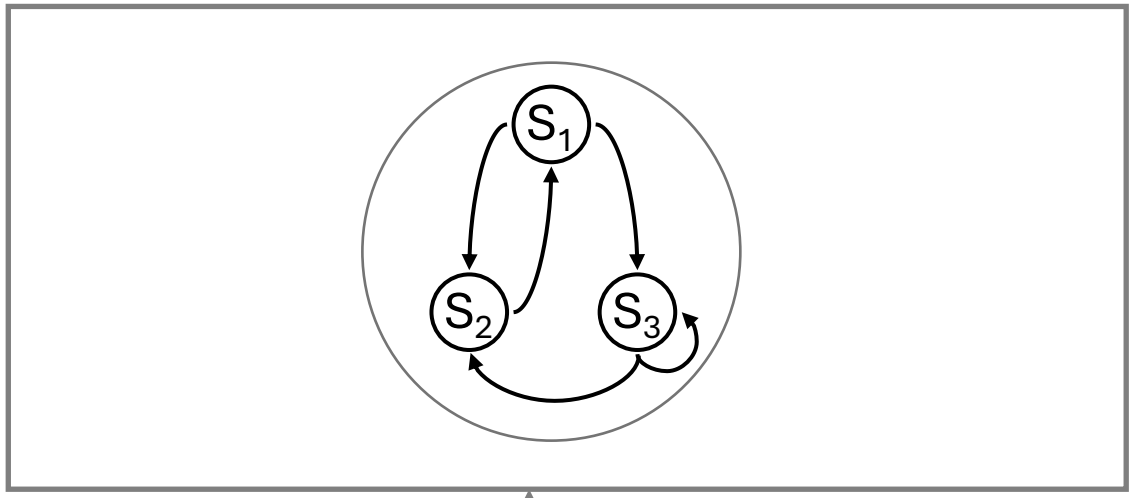
Background - RL



- An **agent observes** the **environment** and **interacts** with it by making **actions**
- Agent receives a **reward signal** based on the **quality** of the action
- Actions **change environment's state**
- **Goal**: maximize expected cumulative reward in a trajectory of T time steps, called an **episode**
- **Transition function:** defines the probability of moving from state s to state s' after taking action a
- **Policy:** maps states into actions
- **Value function:** estimates cumulative reward starting from a specific state

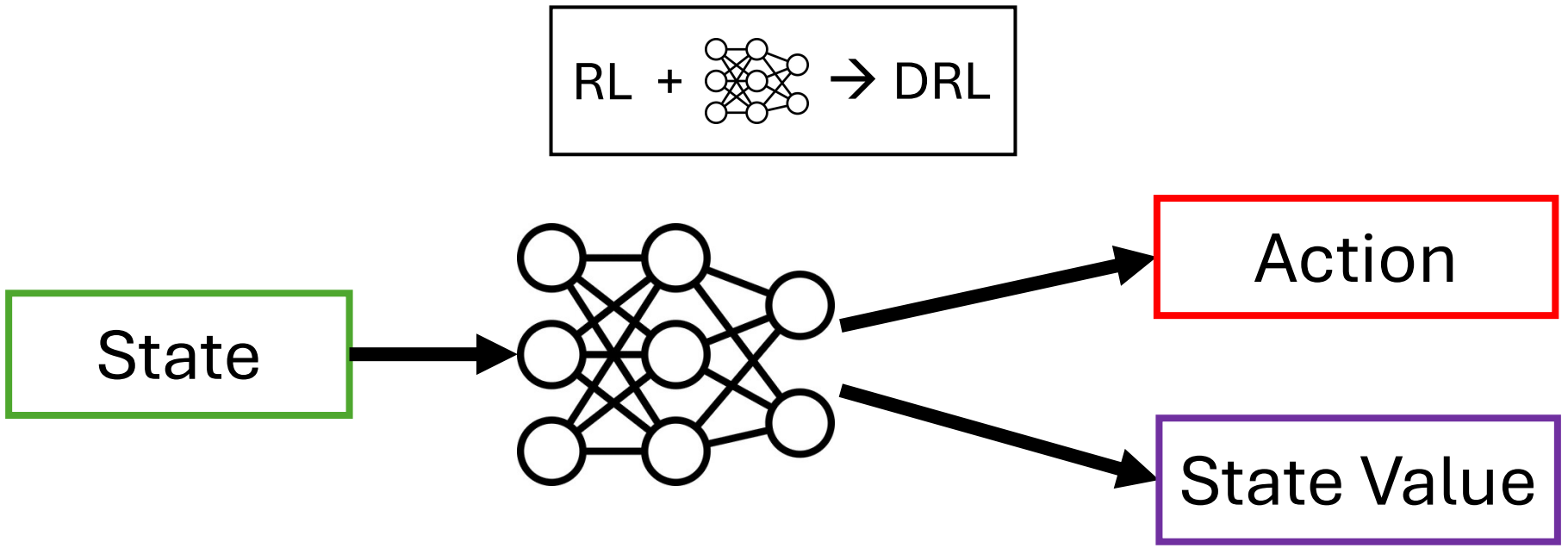
Background - RL





- Actions **change environment's state**
- **Goal**: maximize expected cumulative reward in a trajectory of T time steps, called an **episode**
- **Transition function:** defines the probability of moving from state s to state s' after taking action a
- **Policy:** maps states into actions
- **Value function:** estimates cumulative reward starting from a specific state

Background - DRL



- DNN used as function approximator to map:
  - state to actions (policy)
  - state to state values (value function)

Why Use RL for Resource Management

1. Adaptability to Dynamic Changes:
    - RL overcomes limitations of heuristics and classic ML methods
    - It autonomously **adapts** to evolving environments, handling uncertainties
2. Improved Solutions:
    - RL explores options, finding **global optimal** solutions
    - Enables real-time decision-making and **long-term** planning

Limitations

1. Slow training
2. Difficult debugging of decision-making
3. Handling large and complex state spaces

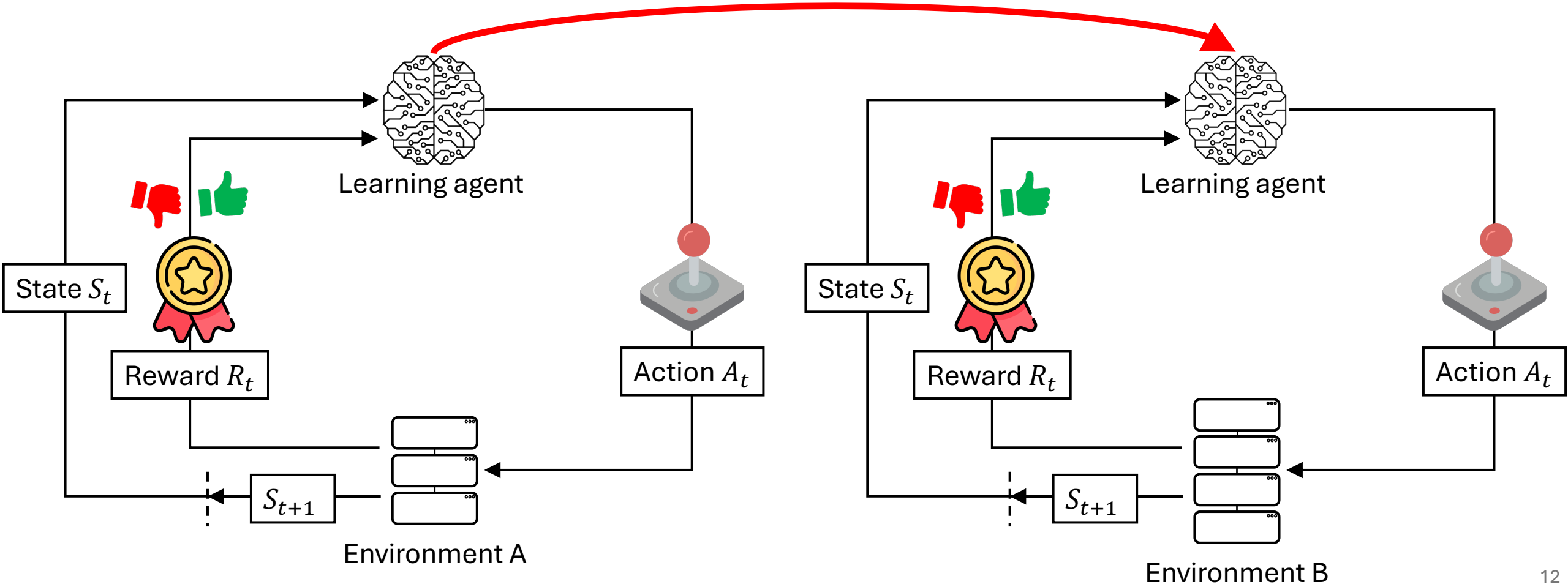
Research Questions

- **RQ1:** What are the capabilities and limitations of the different state-of-the-art methodologies for resource allocation in cloud environments?
  - Comparison in terms of adaptability, computational efficiency, optimality, temporal dependency handling [3]
    - RL is better in terms of adaptability, but still not adaptable enough for real-world scenarios [3]
- **RQ2:** How can RL agents be reused in different cloud environments?

[3] Aslanidis, Theodoros, Chouliaras Andreas, and Chatzopoulos Dimitris. "Reinforcement Learning Techniques for Optimizing System Configuration on the Cloud: A Taxonomy and Open Problems." *Proceedings of the 2023 INTERNATIONAL CONFERENCE ON EMBEDDED WIRELESS SYSTEMS AND NETWORKS*. 2023 [3]

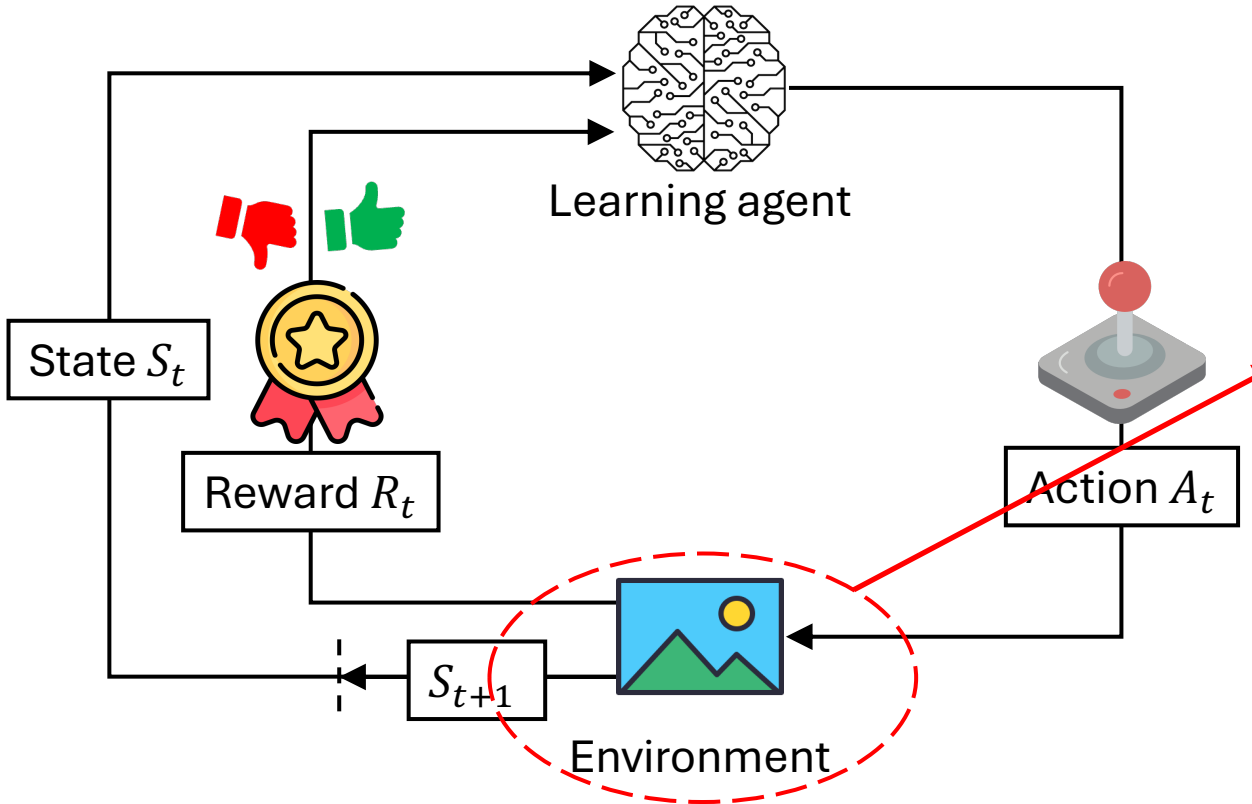
Research Questions

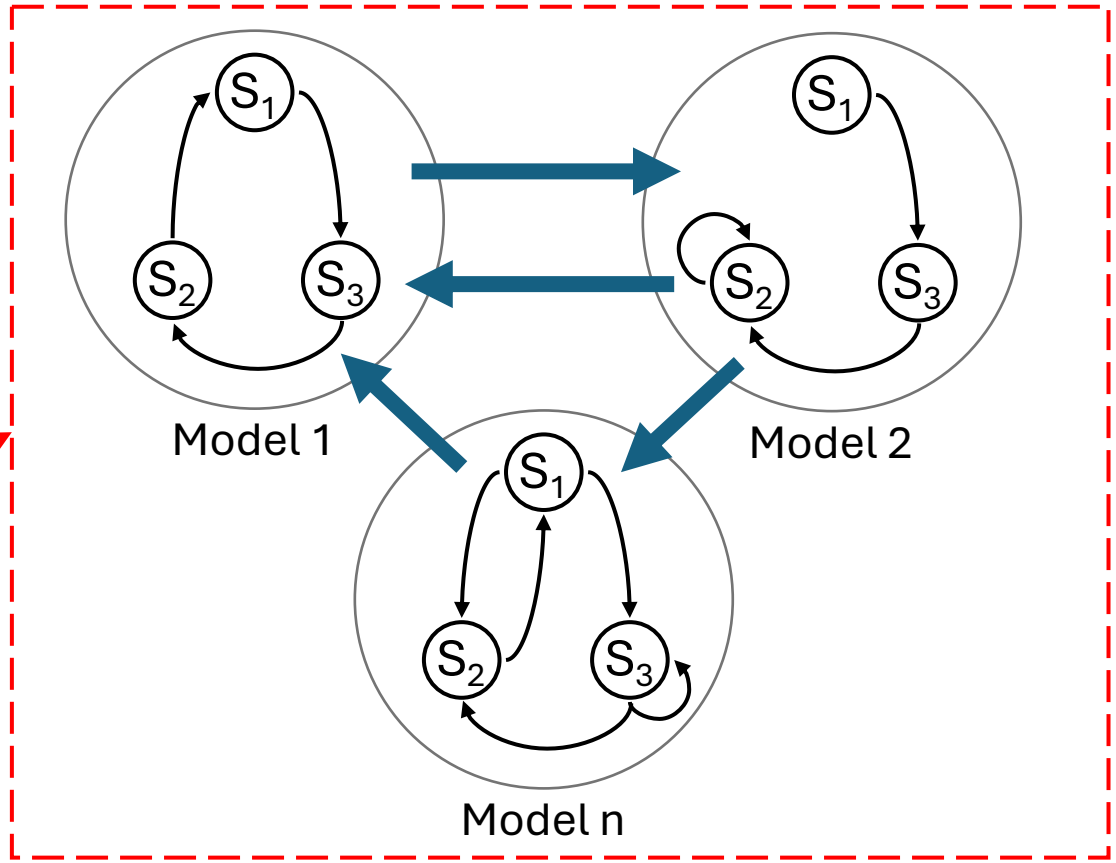
- **RQ1:** What are the capabilities and limitations of the different state-of-the-art methodologies for resource allocation in cloud environments?
- **RQ2:** How can RL agents be reused in different cloud environments?



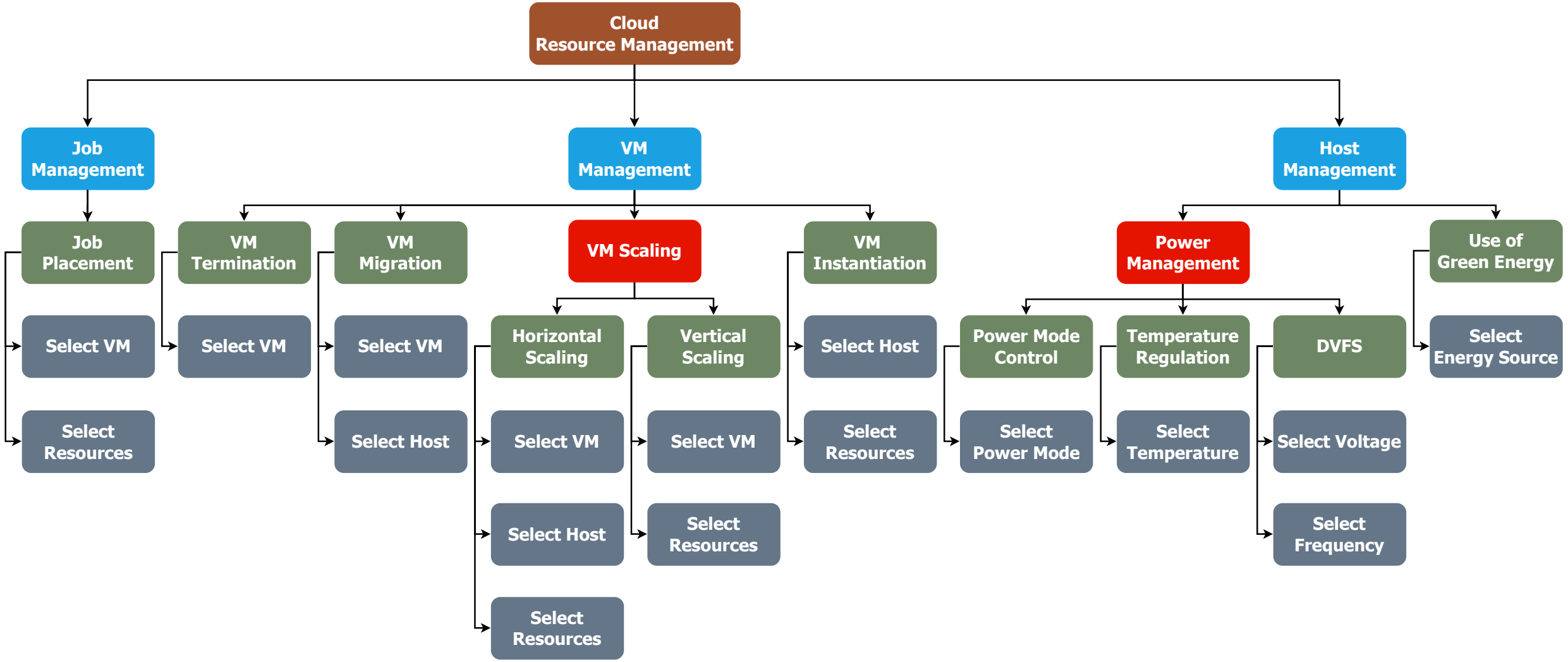
Research Questions

- **RQ1:** What are the capabilities and limitations of the different state-of-the-art methodologies for resource allocation in cloud environments?
- **RQ2:** How can RL agents be reused in different cloud environments?
- **RQ3:** How can RL algorithms adapt to perform efficiently in non-stationary environments?





Taxonomy



[3] Aslanidis, Theodoros, Chouliaras Andreas, and Chatzopoulos Dimitris. "Reinforcement Learning Techniques for Optimizing System Configuration on the Cloud: A Taxonomy and Open Problems." *Proceedings of the 2023 INTERNATIONAL CONFERENCE ON EMBEDDED WIRELESS SYSTEMS AND NETWORKS*. 2023 [3]

Taxonomy

Approach	Computational efficiency	Optimality	Temporal Dependency Handling	Adaptability
Rule-based	●	○	○	○
Heuristics	◕	◔	◔	◑
Queuing theory	◑	◑	◑	◔
Control theory	◔	◕	◕	◕
Game theory	◔	◕	◕	◕
Metaheuristics	◔	◕	◕	◕
Traditional ML	◔	◕	◕	◕
RL and DRL	◔	◕	●	●

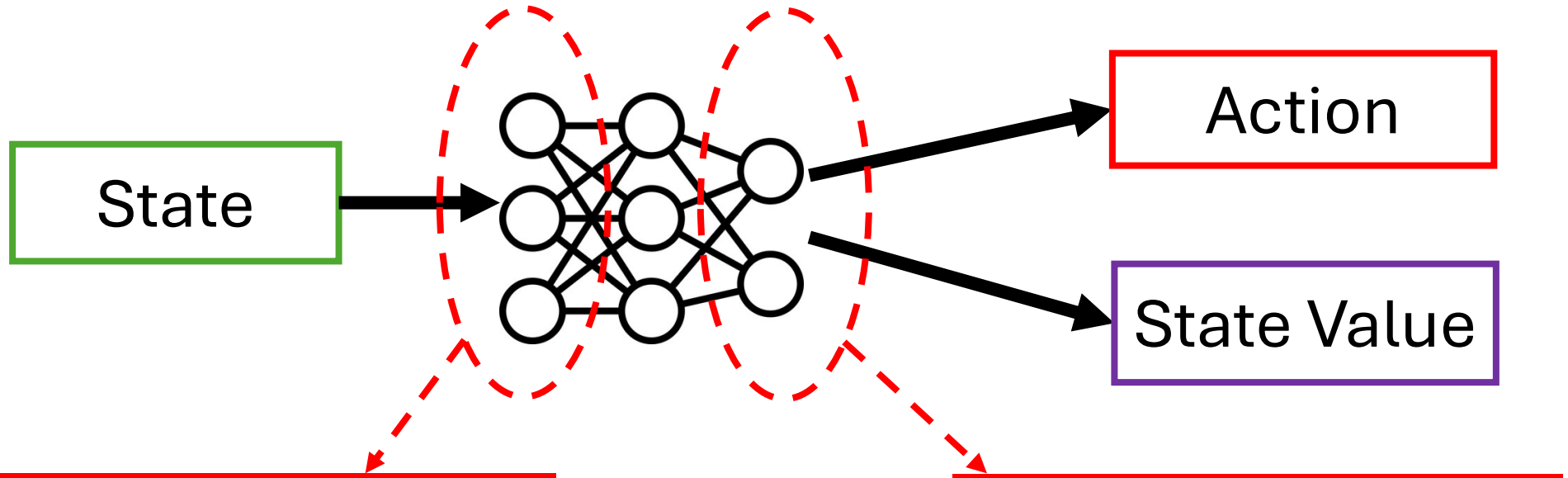
○ : Low ◔ : Low to Medium ◑ : Medium ◕ : Medium to High ● : High

RL and DRL approaches struggle with high non-stationarity in real-world scenarios [3]

[3] Aslanidis, Theodoros, Chouliaras Andreas, and Chatzopoulos Dimitris. "Reinforcement Learning Techniques for Optimizing System Configuration on the Cloud: A Taxonomy and Open Problems." *Proceedings of the 2023 INTERNATIONAL CONFERENCE ON EMBEDDED WIRELESS SYSTEMS AND NETWORKS*. 2023 [3]

Reusability Challenge

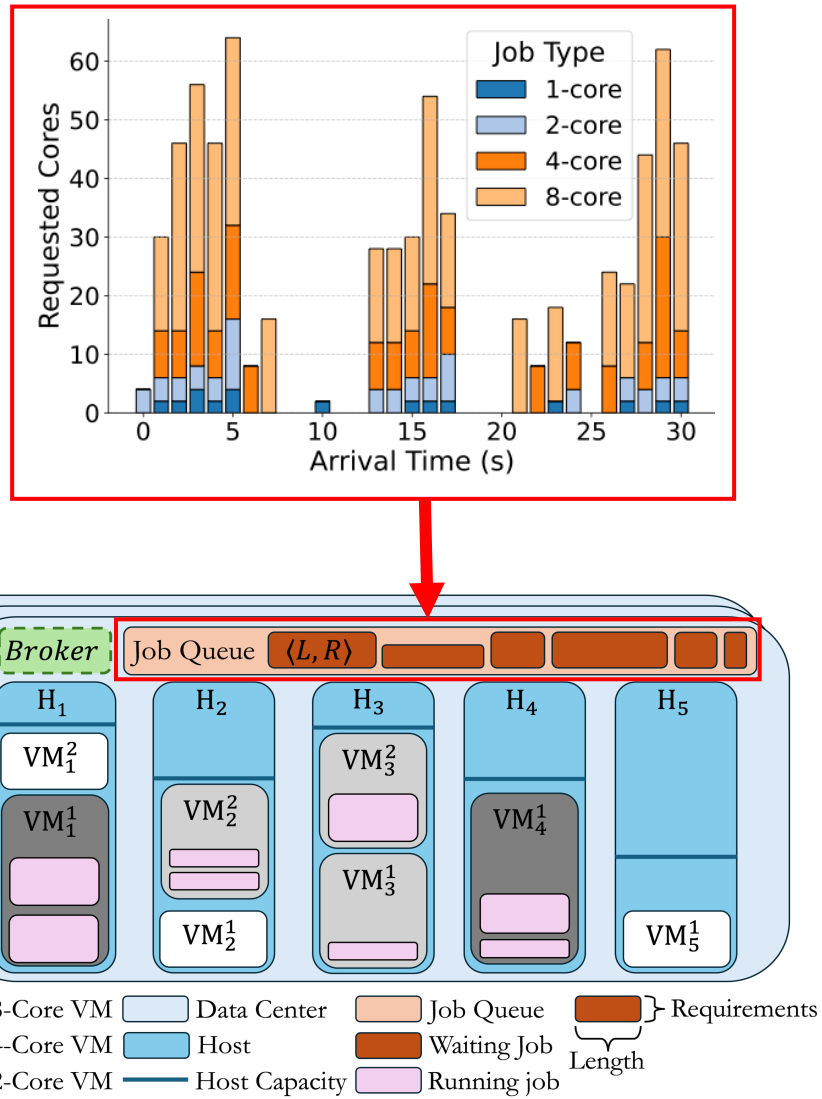
Number of nodes directly affects state and action dimensionality



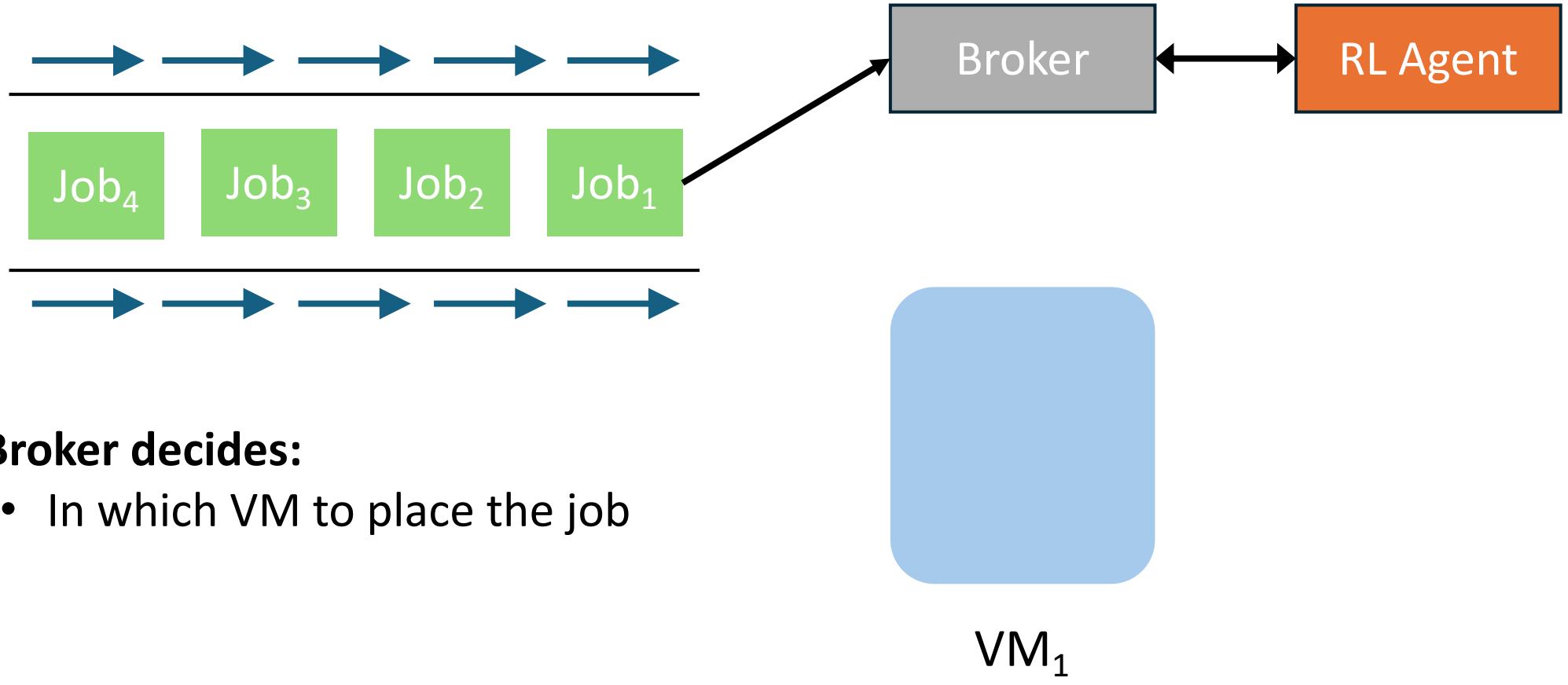
A poorly designed RL agent cannot be reused to a different environments

VM Management: System Model

- A datacenter/cluster receives short-lived jobs/applications that needs to be placed in a VM inside a host/node
- The nodes are fixed, but without an allocated VM in them, cannot run any application
  - A DRL model is responsible in finding the optimal VM allocation strategy inside nodes which minimizes:
    1) Application waiting time
    2) Total allocated VM cores
    3) Total unutilized VM cores

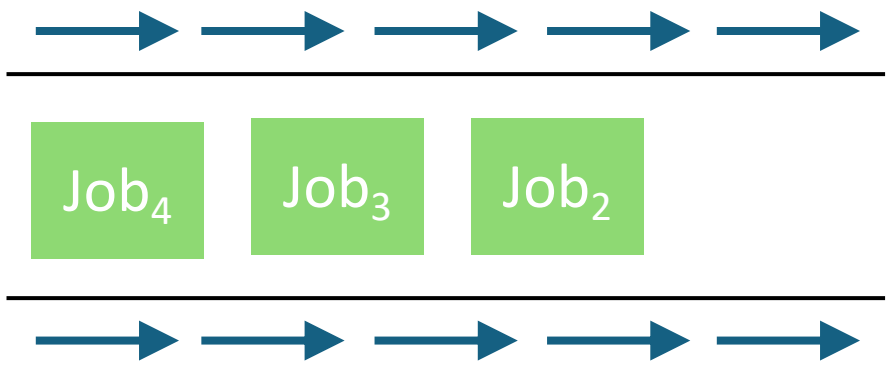


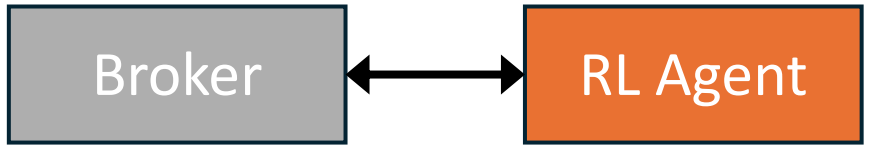
VM Management: Example



- **Broker decides:**
  - In which VM to place the job

VM Management: Example

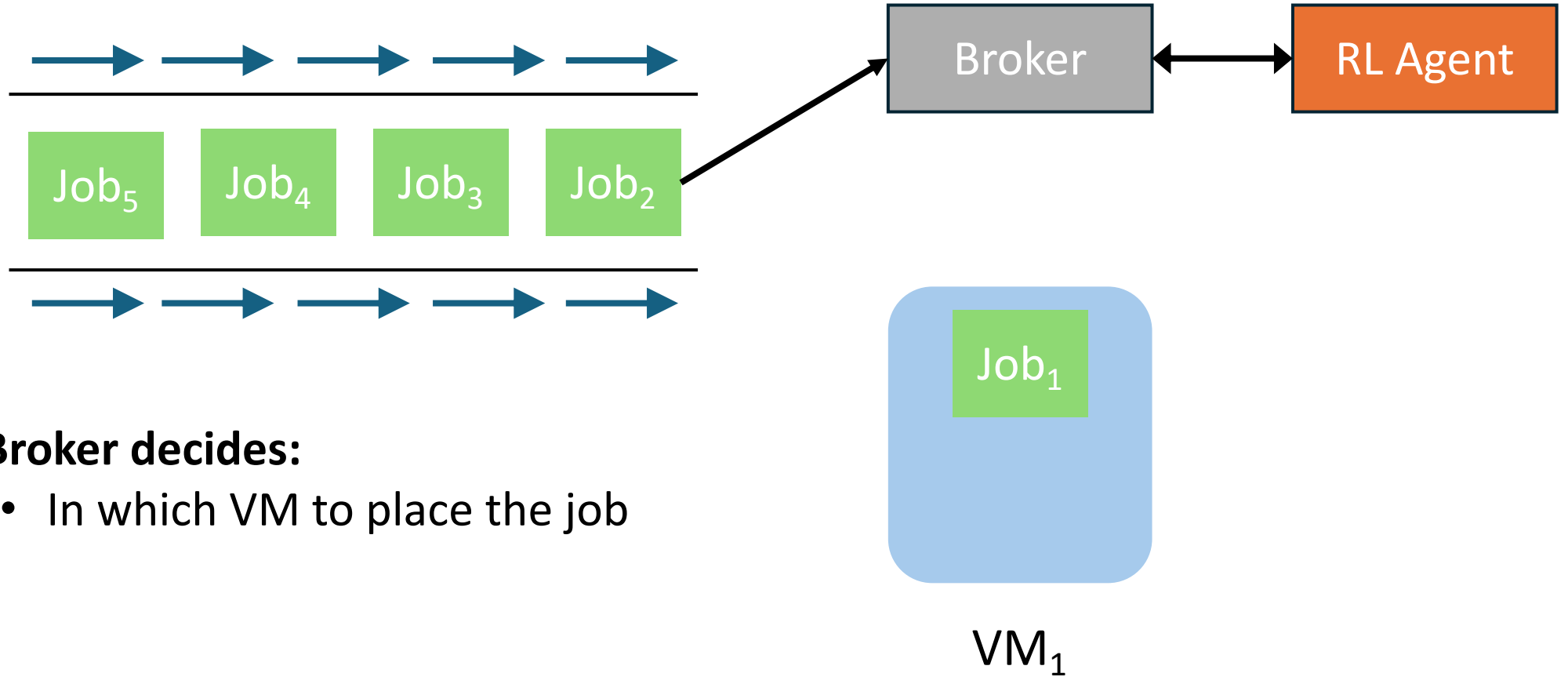




VM_1

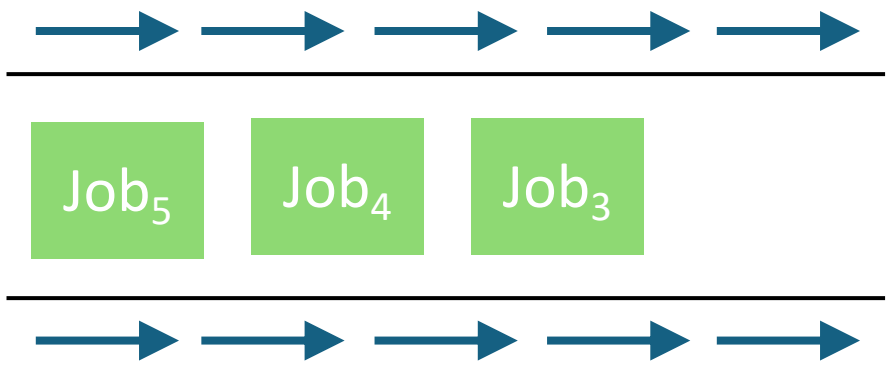
- **Broker decides:**
  - In which VM to place the job

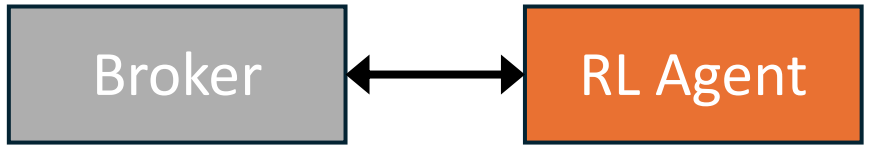
VM Management: Example

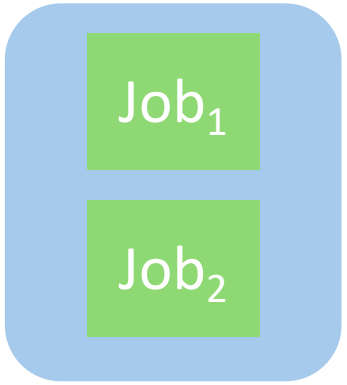


- **Broker decides:**
  - In which VM to place the job

VM Management: Example



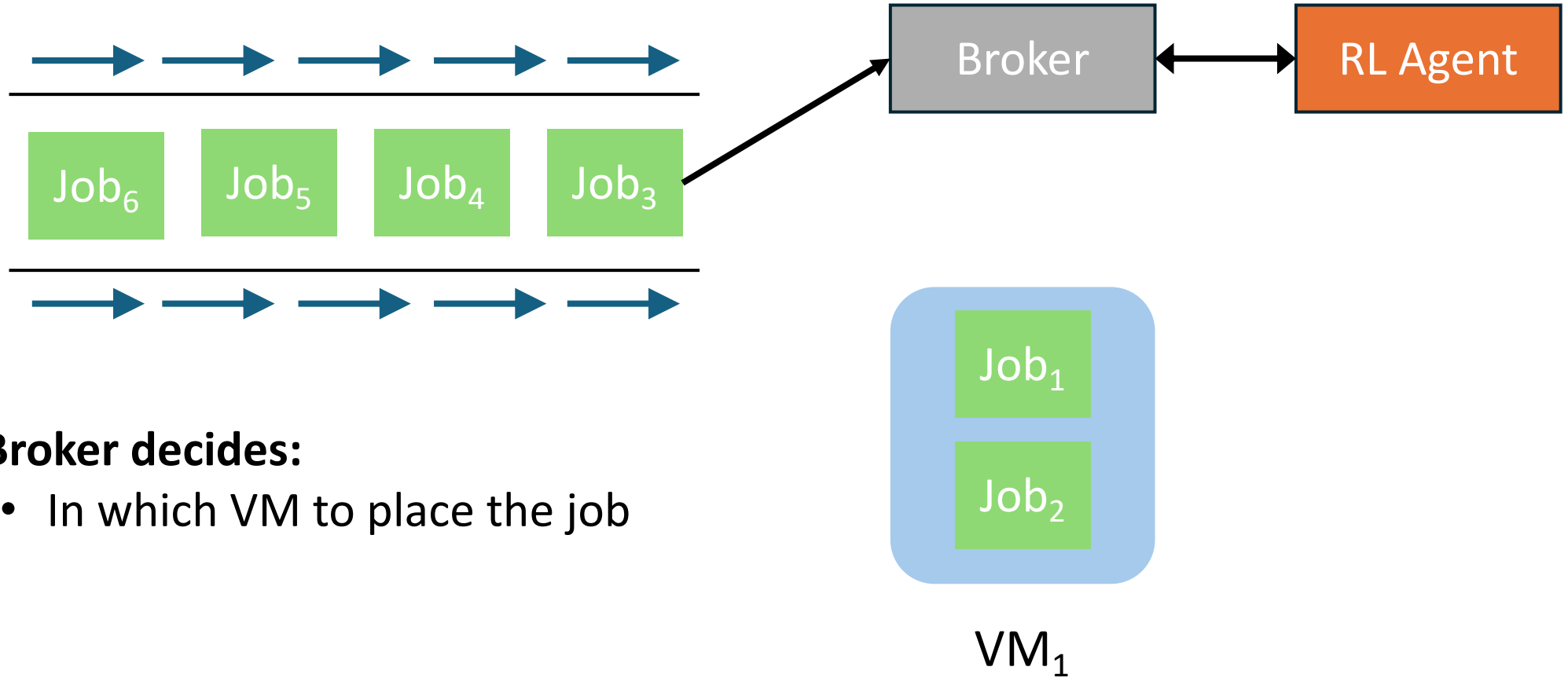




VM_1

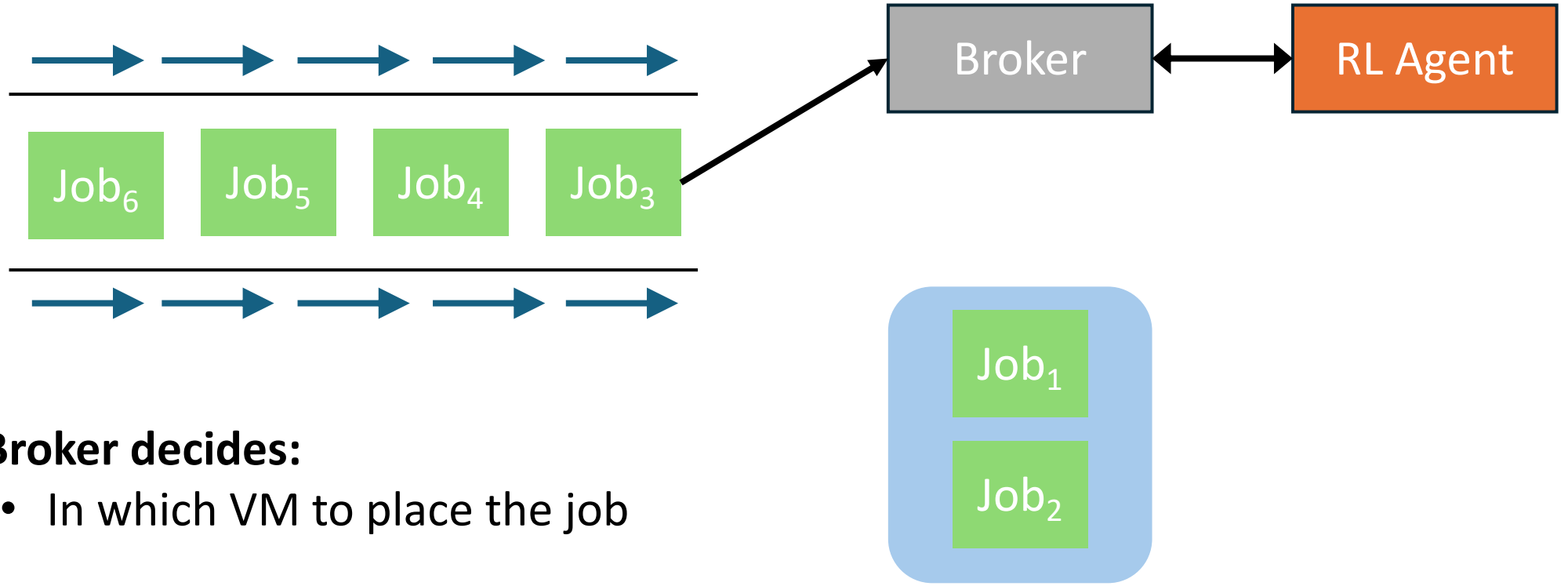
- **Broker decides:**
  - In which VM to place the job

VM Management: Example



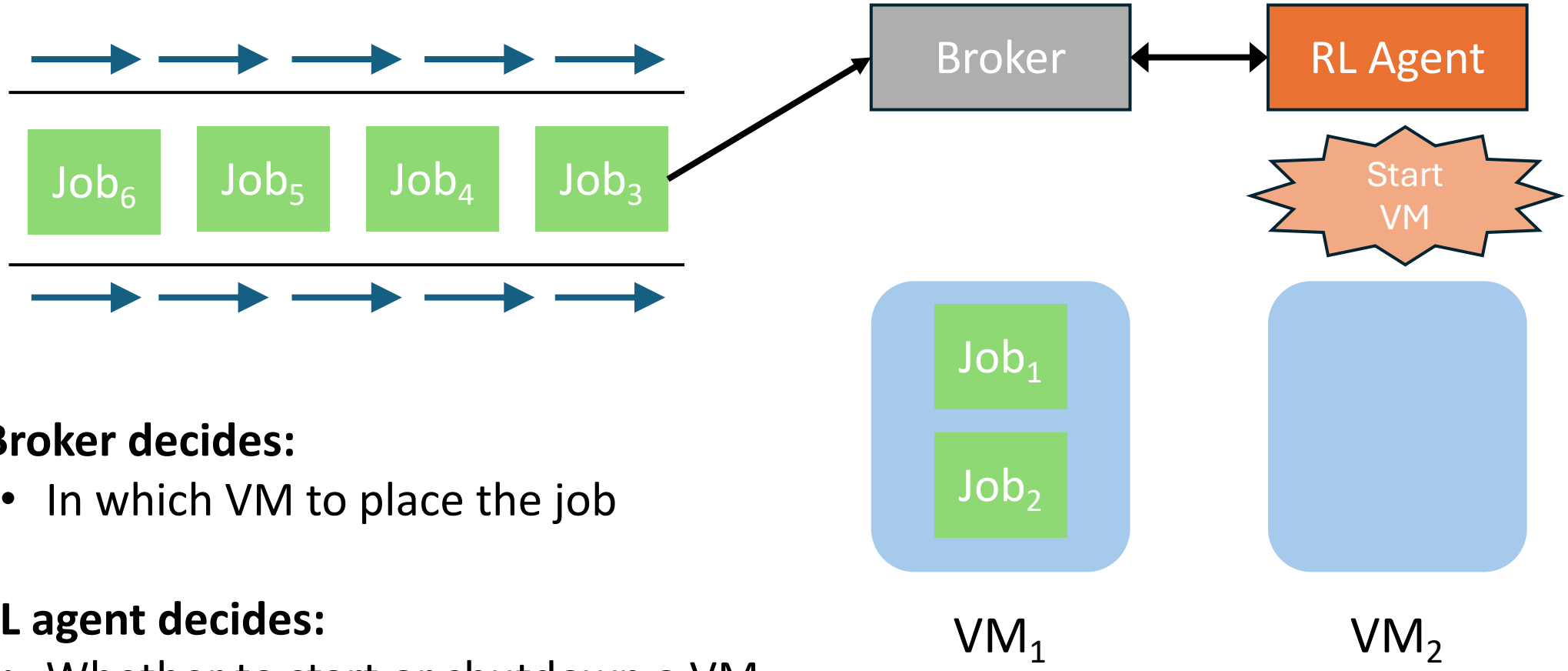
- **Broker decides:**
  - In which VM to place the job

VM Management: Example



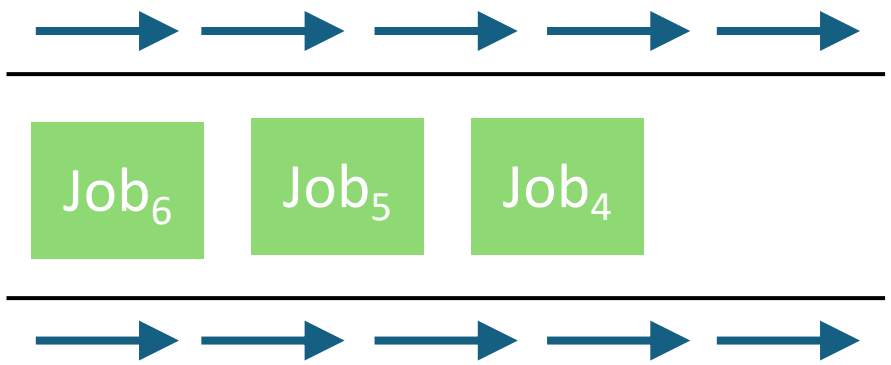
- **Broker decides:**
  - In which VM to place the job
- **RL agent decides:**
  - Whether to start or shutdown a VM

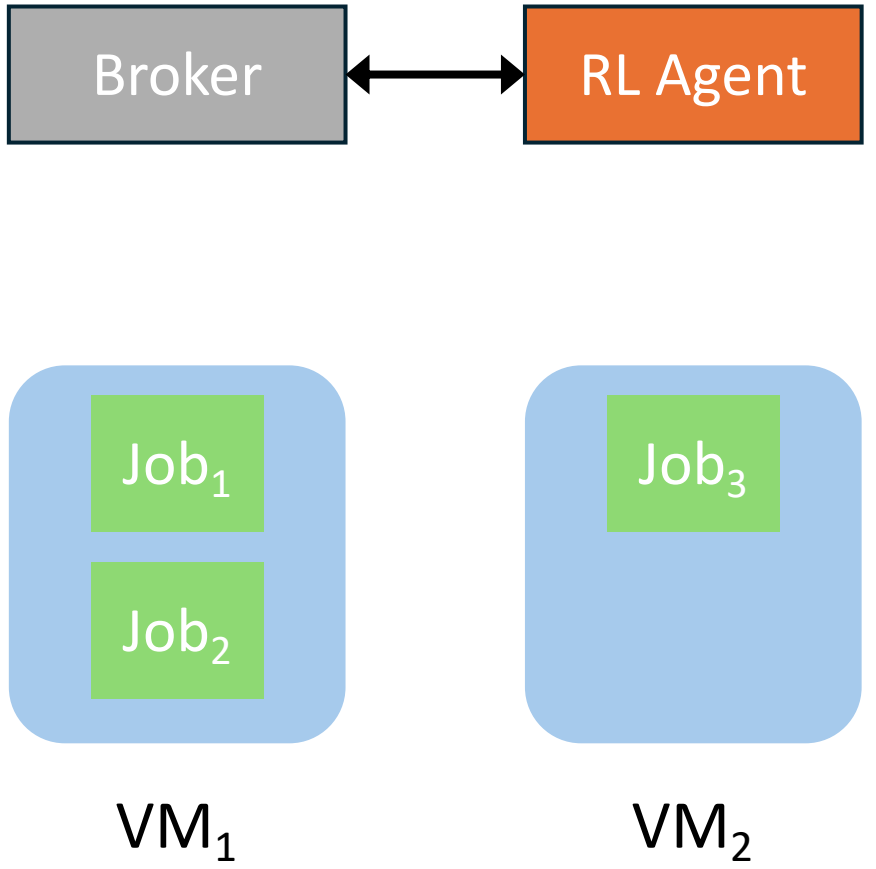
VM Management: Example



- **Broker decides:**
  - In which VM to place the job
- **RL agent decides:**
  - Whether to start or shutdown a VM

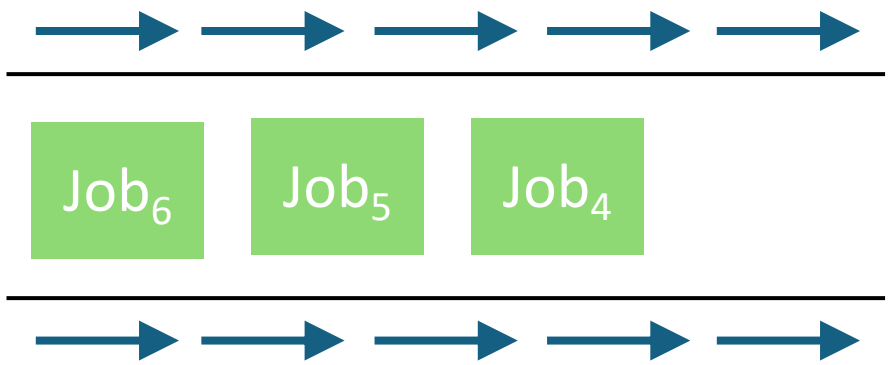
VM Management: Example

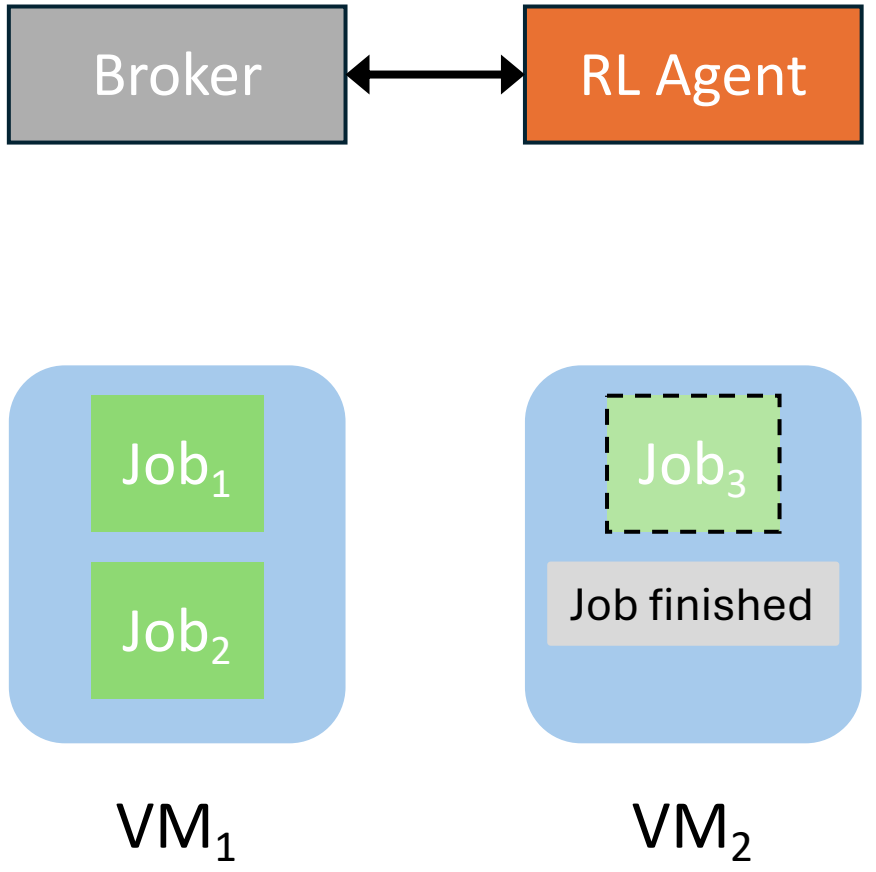




- **Broker decides:**
  - In which VM to place the job
- **RL agent decides:**
  - Whether to start or shutdown a VM

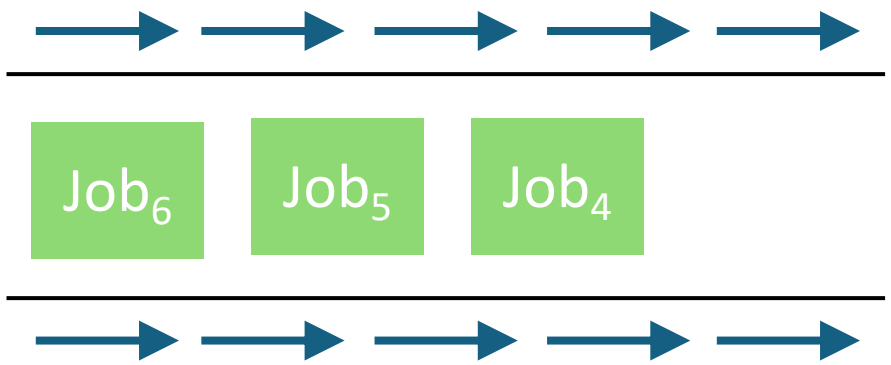
VM Management: Example

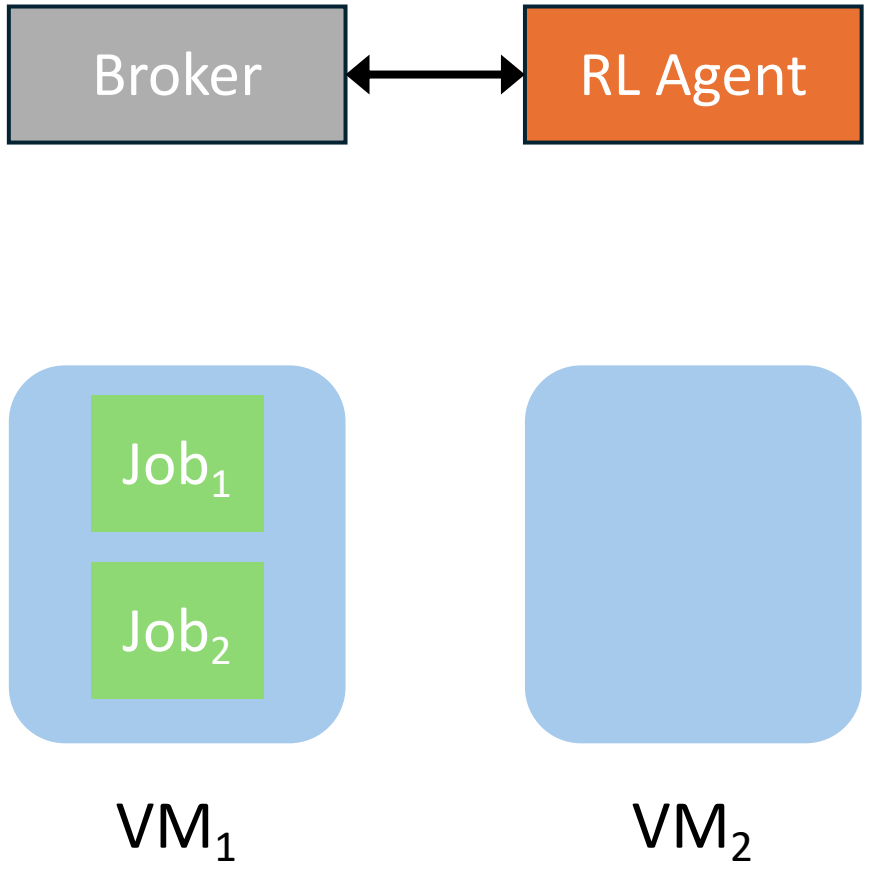




- **Broker decides:**
  - In which VM to place the job
- **RL agent decides:**
  - Whether to start or shutdown a VM

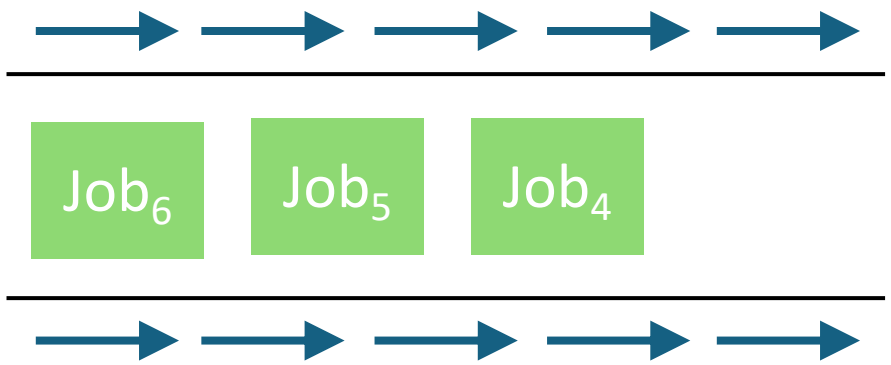
VM Management: Example

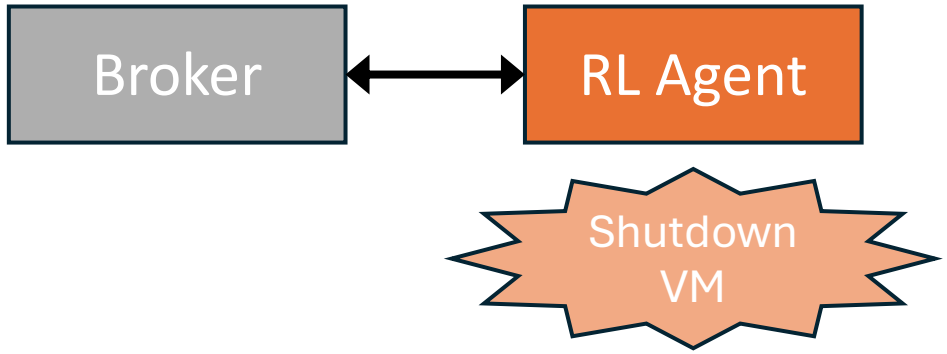


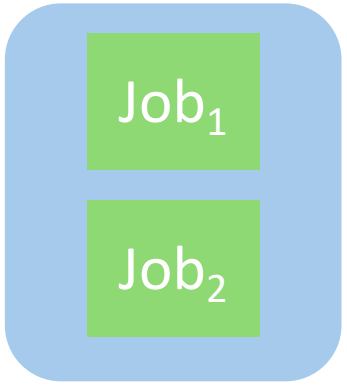


- **Broker decides:**
  - In which VM to place the job
- **RL agent decides:**
  - Whether to start or shutdown a VM

VM Management: Example



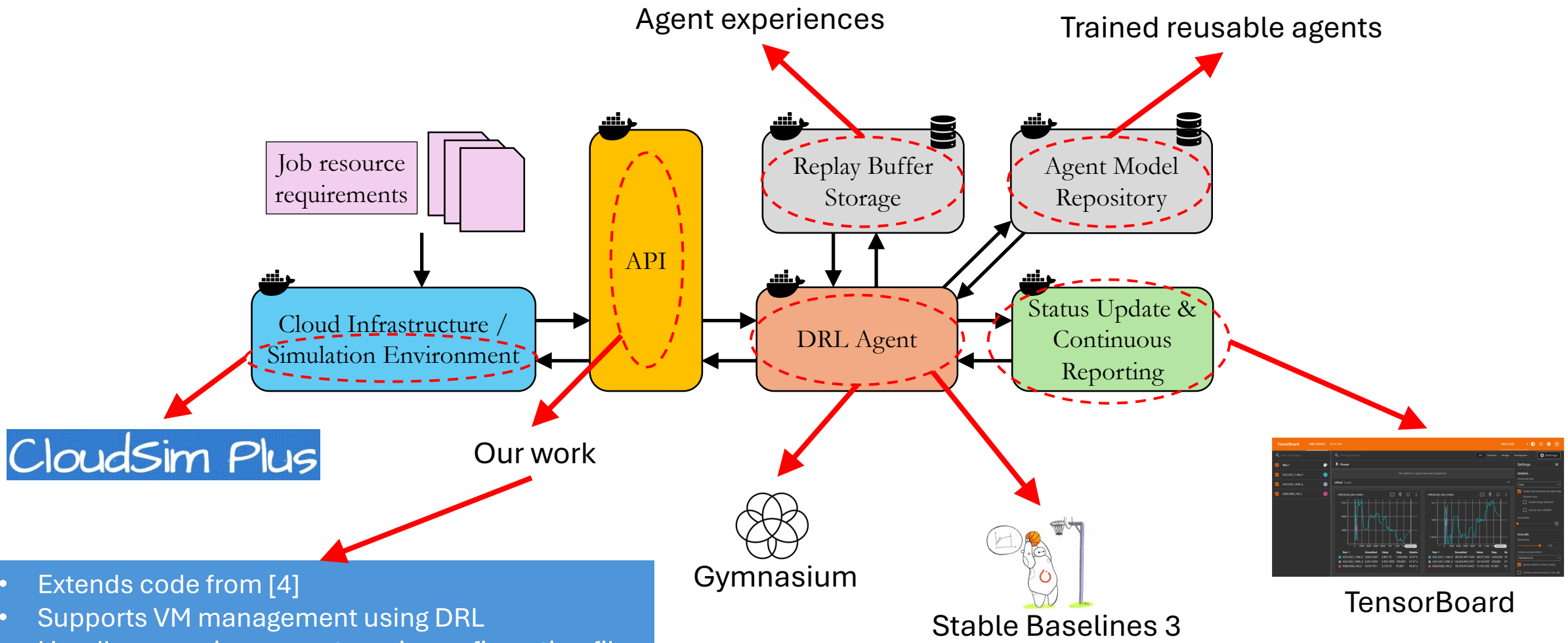




VM_1

- **Broker decides:**
  - In which VM to place the job
- **RL agent decides:**
  - Whether to start or shutdown a VM

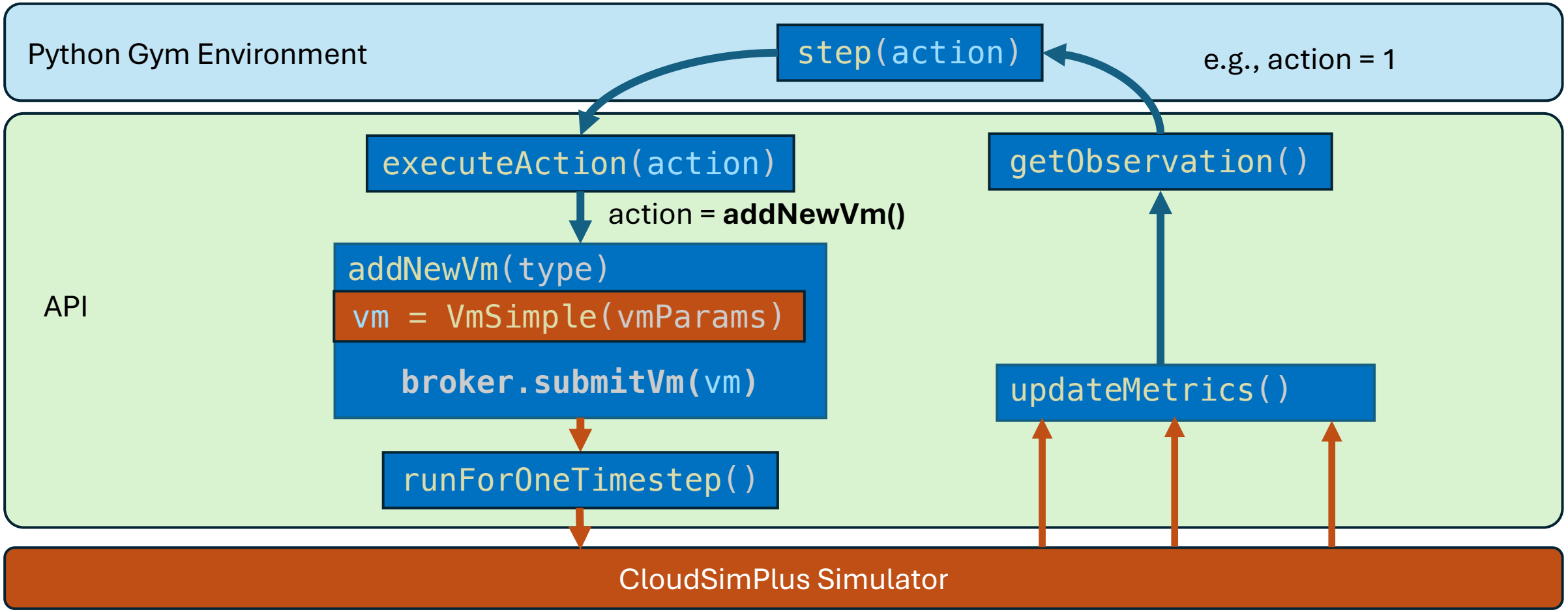
Implementation



- Extends code from [4]
- Supports VM management using DRL
- Handle scenario parameters via configuration file
  - Infrastructure parameters (hosts, etc.)
  - DRL training parameters (time steps, etc.)
- 2500+ lines of code in Java & Python

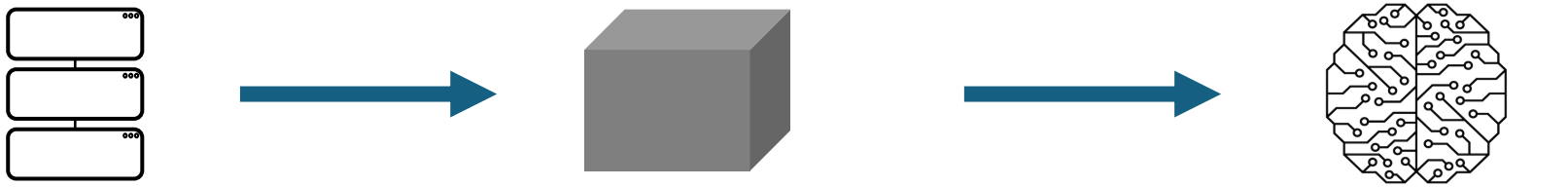
[4] Funika, Włodzimierz, Paweł Koperek, and Jacek Kitowski. "Repeatable experiments in the cloud resources management domain with use of reinforcement learning." *Cracow Grid Workshop*. 2018

RL Loop


Python Gym Environment
step(action)
e.g., action = 1
executeAction(action)
getObservation()
action = addNewVm()
addNewVm(type)
vm = VmSimple(vmParams)
broker.submitVm(vm)
API
updateMetrics()
runForOneTimestep()
CloudSimPlus Simulator

Reusable Agent Design

Reusability: agent able to interact with different cloud environments without redesign



Environment State

State Space Representation

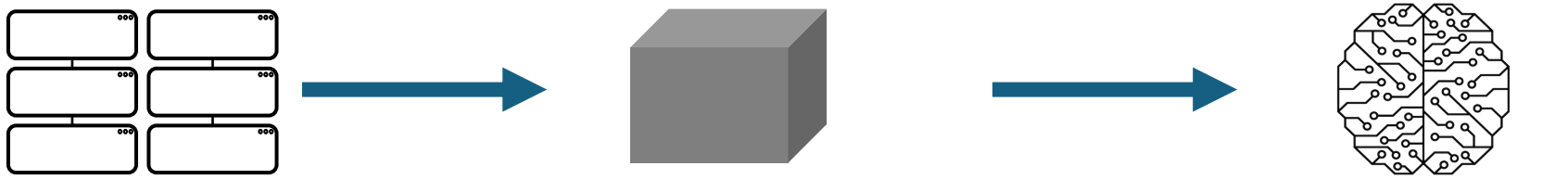
Learning Agent

Benefits of Efficient State Space Representation

- Reduce training time by minimizing state space size
- Outperform models with poor state representations
- Enable agent transfer between different infrastructures

Reusable Agent Design

Reusability: agent able to interact with different cloud environments without redesign



Environment State

State Space Representation

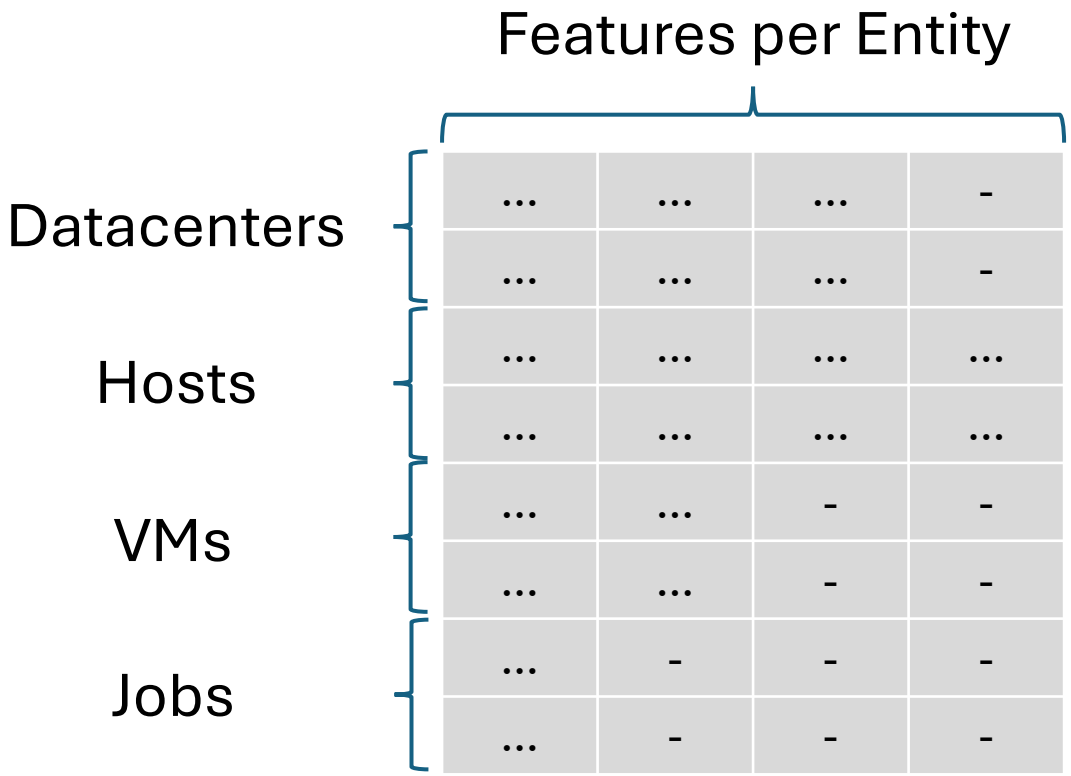
Learning Agent

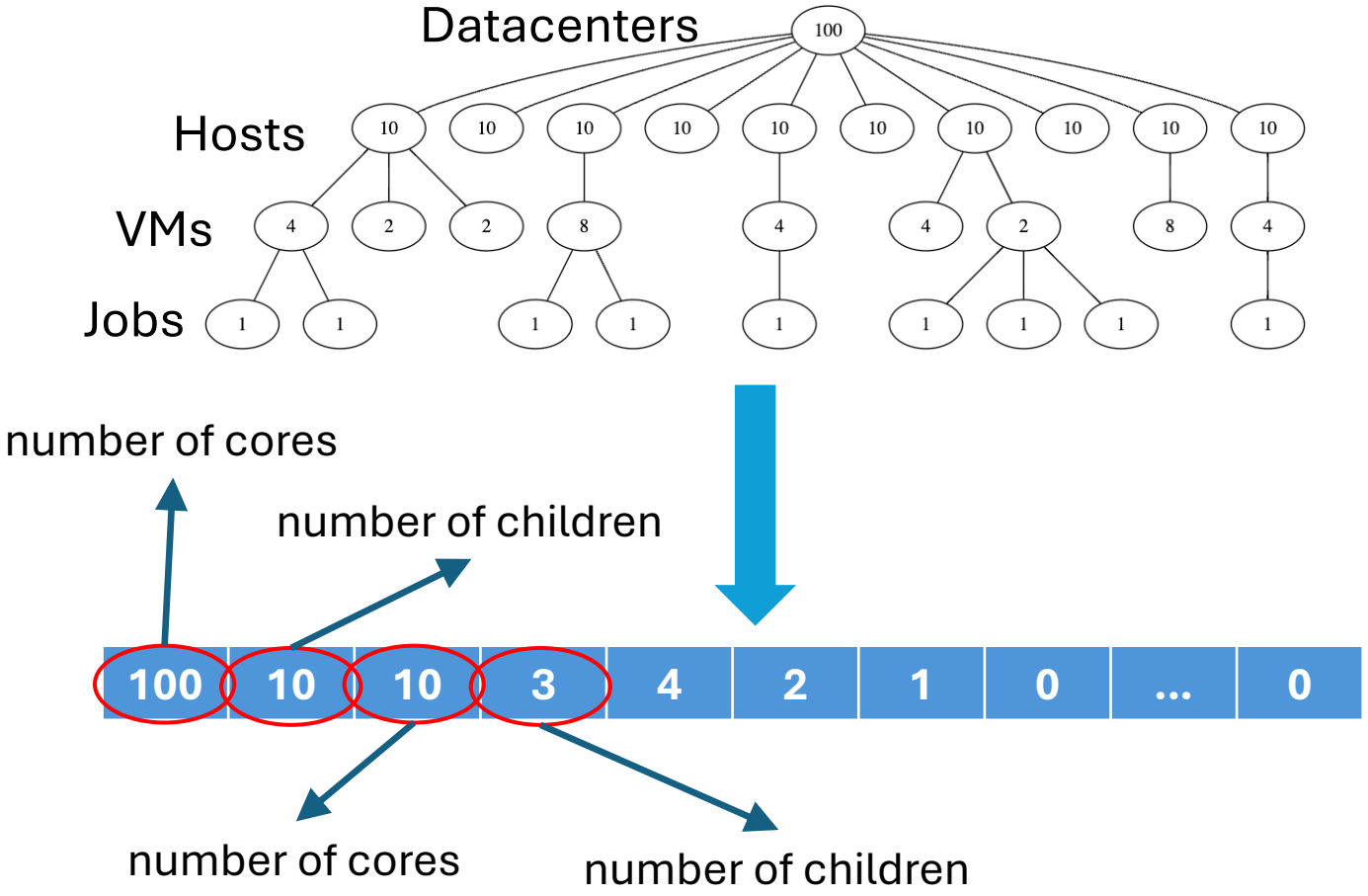
Benefits of Efficient State Space Representation

- Reduce training time by minimizing state space size
- Outperform models with poor state representations
- Enable agent transfer between different infrastructures

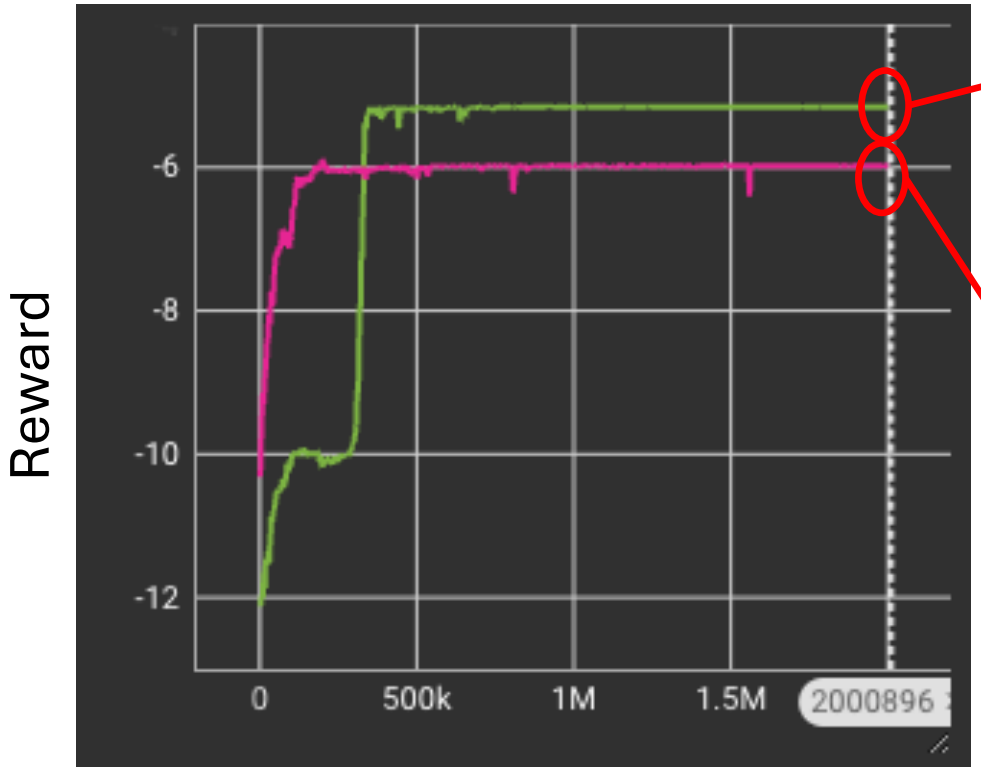
State Representation Comparison

Unoptimized Sparse 2D Array State **vs.** **Optimized Tree Traversal State**





State Representation Comparison

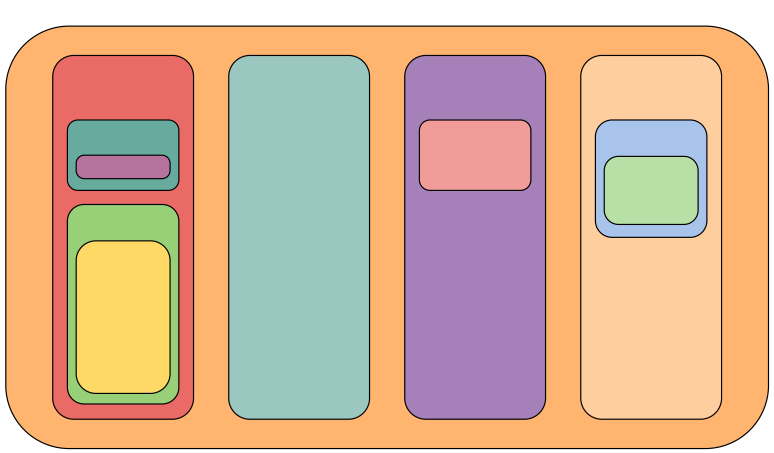


Optimized State
Time: 5.946 hr
Reward: -5.154

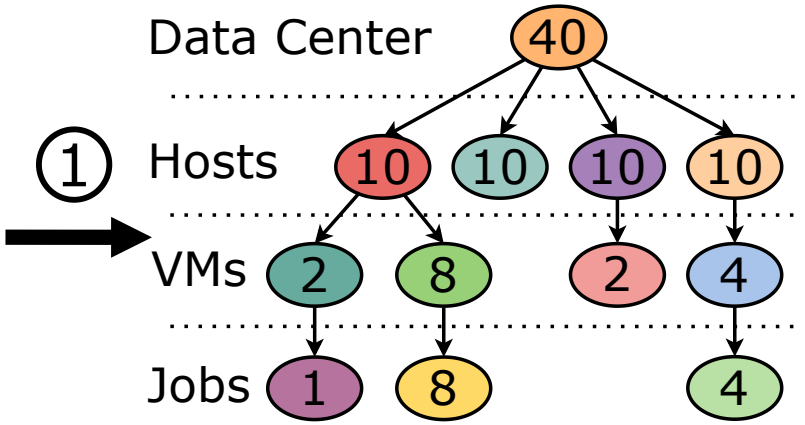
Unoptimized State
Time: 17.32 hr
Reward: -5.976

Optimized state yields:
2.91x convergence time acceleration
13.76% reward performance boost

Tree Traversal Representation Visualized



Infrastructure



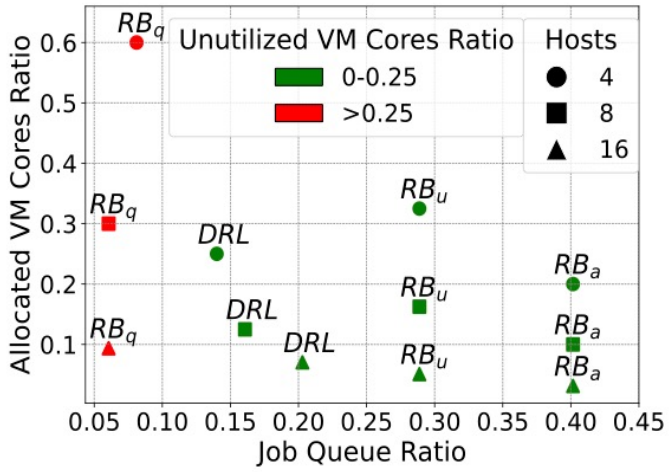
Tree Representation

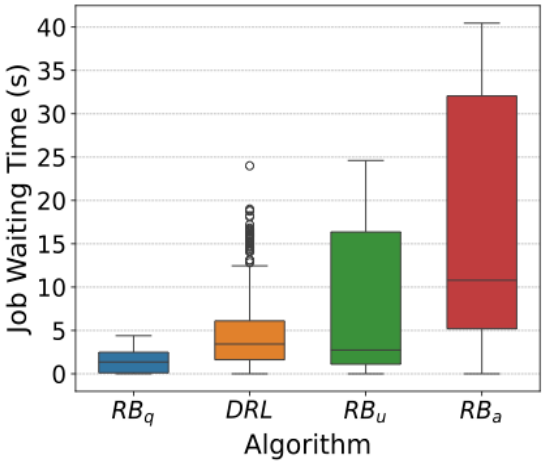
40	4	10	2	2	1
1	0	8	1	8	0
10	0	10	1	2	0
10	1	4	1	4	0

Preorder Edge-Count Array
Infrastructure State (s_I)

VM Management Results: DRL vs Rule-based

- Comparison between DRL and three rule-based approaches
  - RB_q minimizes job queue
  - RB_a minimizes allocated VM cores
  - RB_u minimizes unutilized VM cores
- **DRL balances all metrics outperforming rule-based solutions**

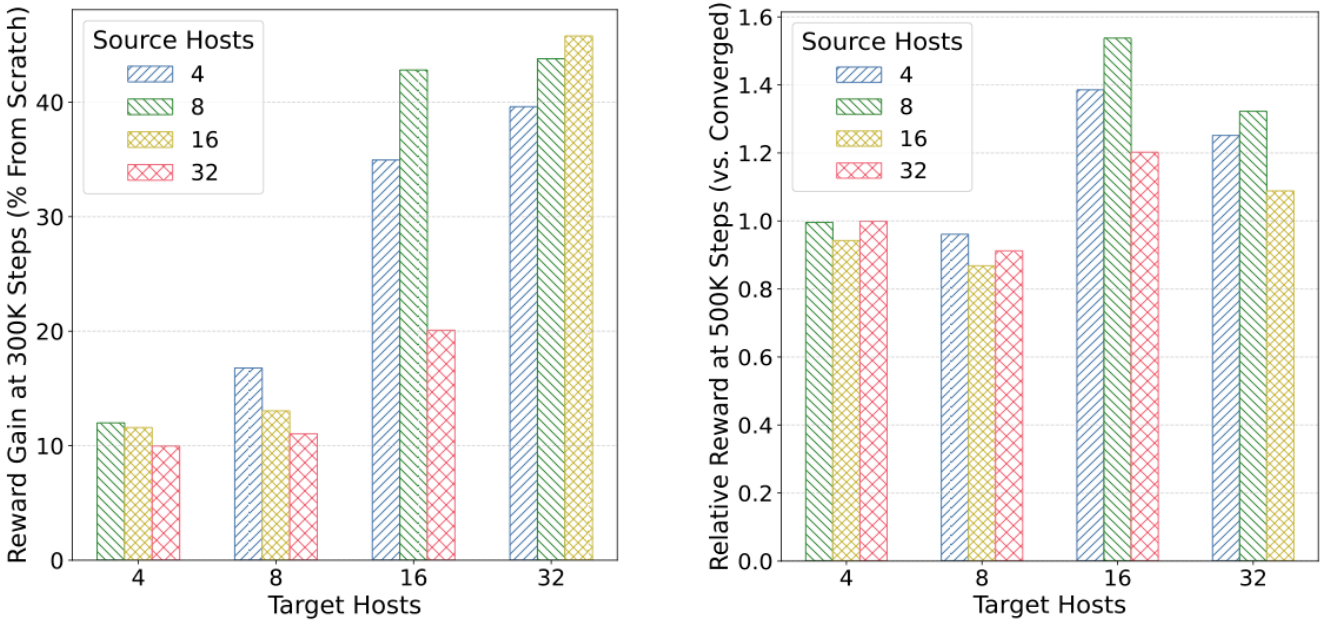




Transferring DRL Agents Between Infrastructures

- Trained DRL agents for 4 different cluster sizes (4, 8, 16, 32 nodes)
- Transferred agents between clusters and compared with training from scratch
- Pre-trained agents consistently outperformed agents trained from scratch when comparing performance over the same training duration (0–300k timesteps, figure a).
- In some cases, pre-trained agents outperformed the converged from-scratch performance in less than 25% of the training time (figure b)
- Performance improved by up to **54%** (e.g., 8 → 16 hosts, figure b)

Transferred Agents vs. Agents Trained from Scratch



(a) (b)

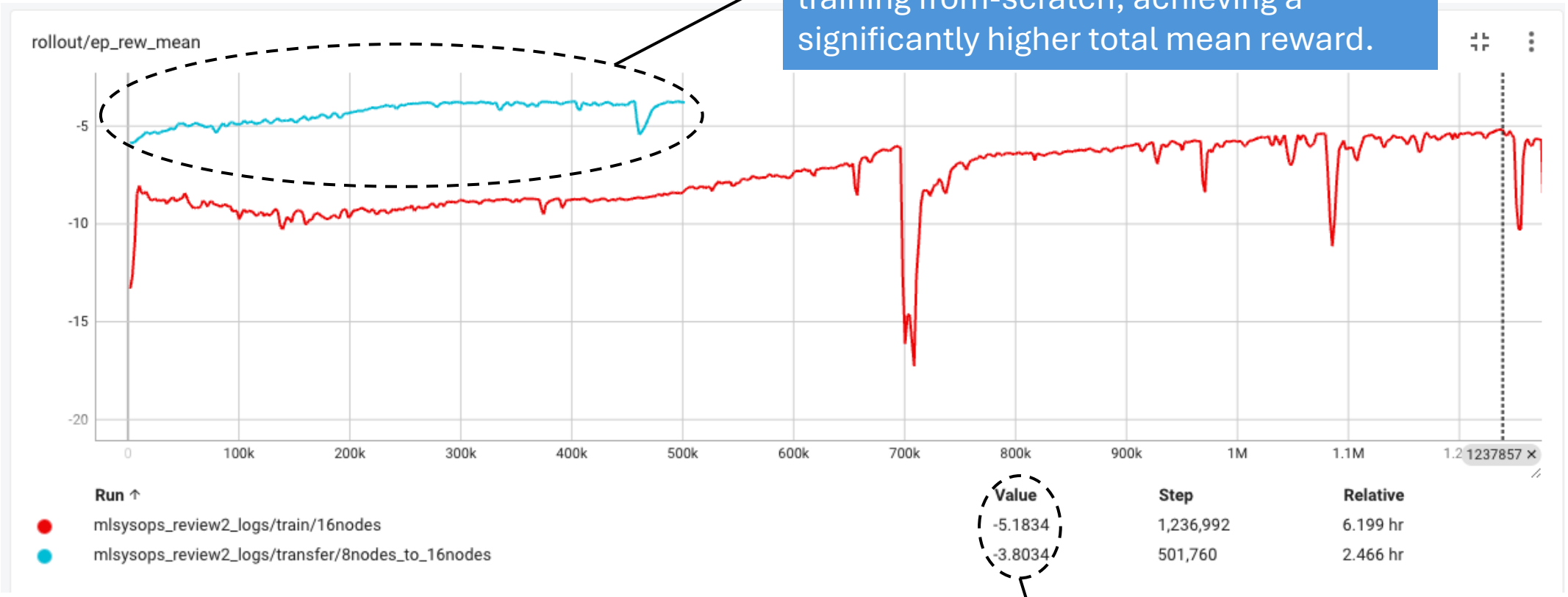
Insight 1: Larger target infrastructures benefit more from transferred agents due to fast **adaptation** to **complex** state spaces.

Insight 2: Source-target **similarity** affects performance, with closer matches (e.g., 8 → 16) yielding better results.

Insight 3: Small-to-large transfers (incremental learning) **outperform** converged models.

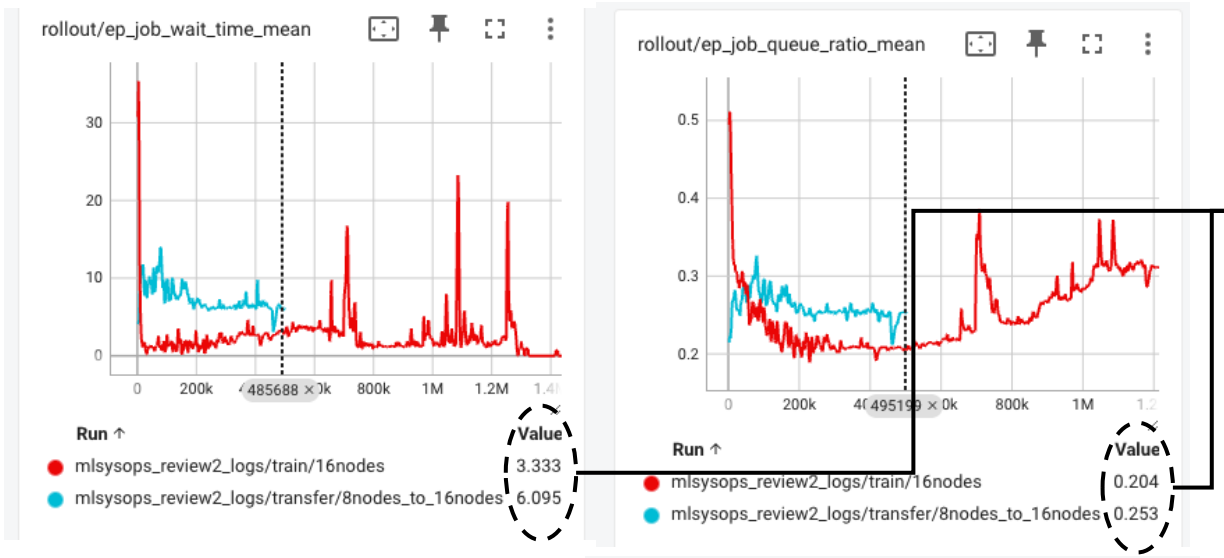
Transferred Agent vs. Agent Trained from Scratch

The transferred DRL agent consistently **outperforms** the agent that starts training from-scratch, achieving a significantly higher total mean reward.



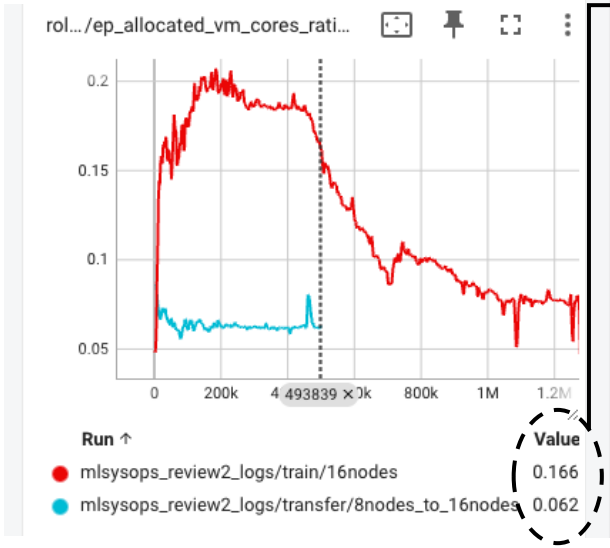
The transferred DRL agent achieves a significantly higher best reward, **-3.8 compared to -5.1** for the from-scratch DRL agent, **despite** the latter **interacting** with the environment **for more than twice as long**.

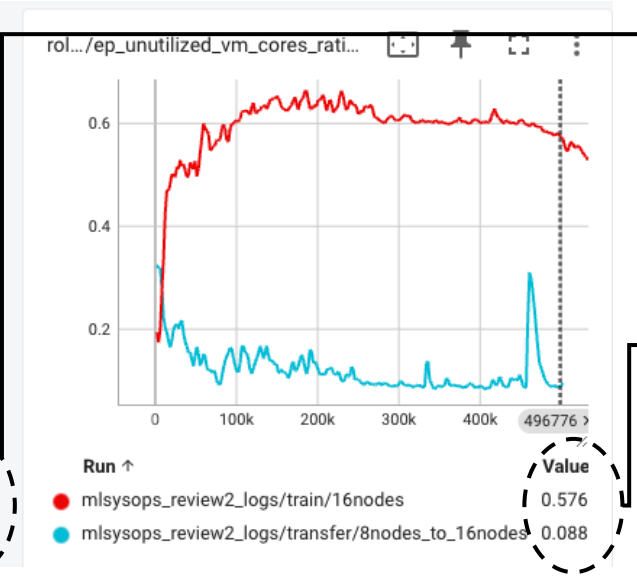
Transferred Agent vs. Agent Trained from Scratch



The transferred DRL agent achieves a **higher total reward** at the cost of slightly increasing the average job waiting time

- **6 seconds vs. 3.3 seconds**





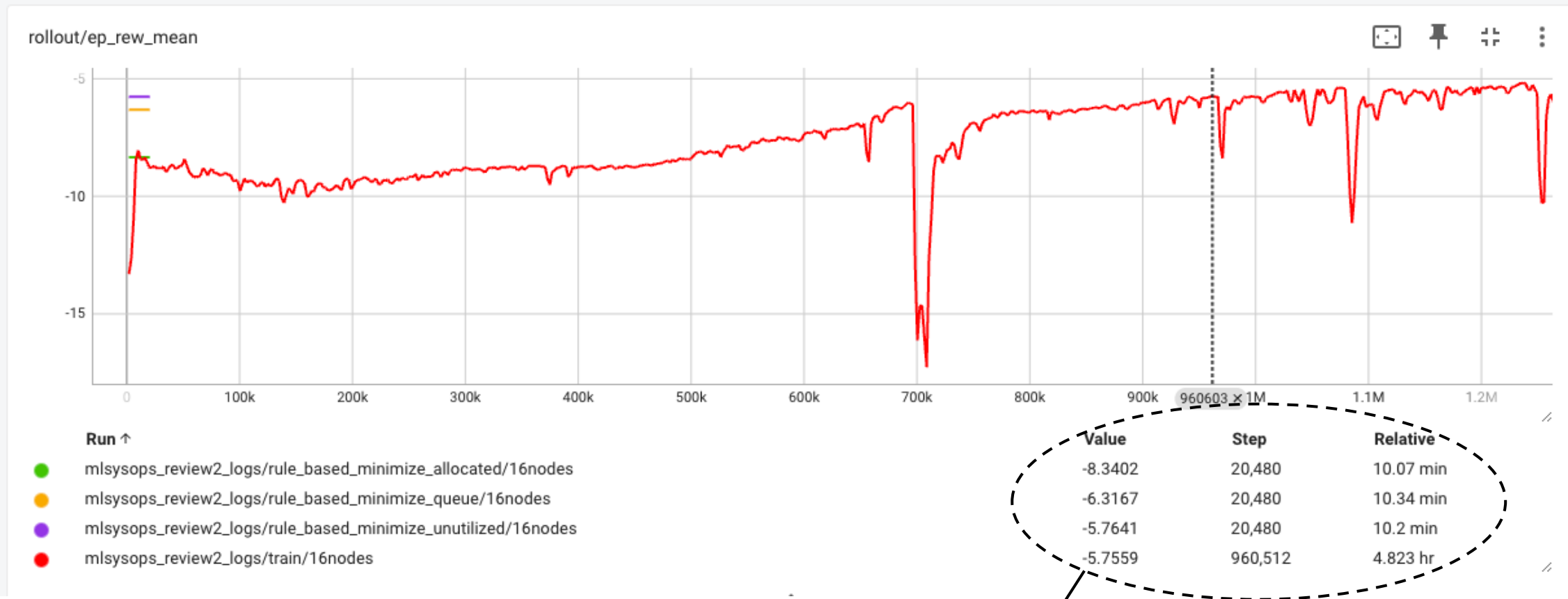
... while significantly reducing overall resource usage

- **6% vs. 16%** of total datacenter cores allocated to VMs.

... and simultaneously minimizing over-provisioning

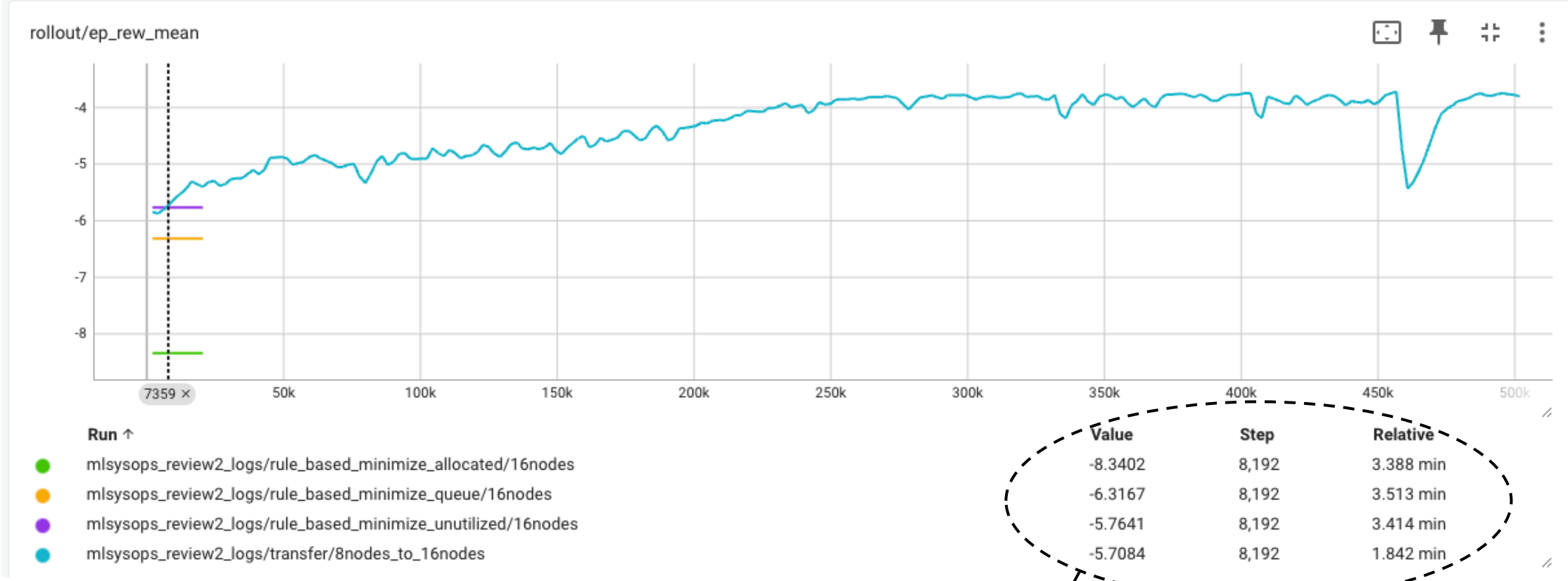
- **9% vs. 57%** of unutilized VM cores.

Transferred Agent vs. Agent Trained from Scratch



- When compared to rule-based approaches, the from-scratch DRL agent only surpasses the best rule-based method after approximately **960K environment interactions**,
- which equals to around **five hours of training**.

Transferred Agent vs. Agent Trained from Scratch



- In contrast, the transferred DRL agent outperforms the best rule-based approach after just **8K environment interactions**, taking **less than two minutes!**
- This demonstrates that with efficient retraining, we can eliminate the need for training from scratch, significantly saving resources, energy, and ultimately, costs.

Thank you!