Ray - Scalability from a Laptop to a Cluster

Dean Wampler - April 20, 2020 - GOTO Chicago
dean@anyscale.com
@deanwampler
https://ray.io
https://anyscale.com

Checkout our online events this Summer:

https://anyscale.com/events

RAY

anyscale

What We'll Talk About:

- Ray demo
  - We'll get into the mechanics of using the Ray API
- Why Ray Is Needed
- ML/AI Ray Libraries
- Ray for Microservices

RAY

Demo

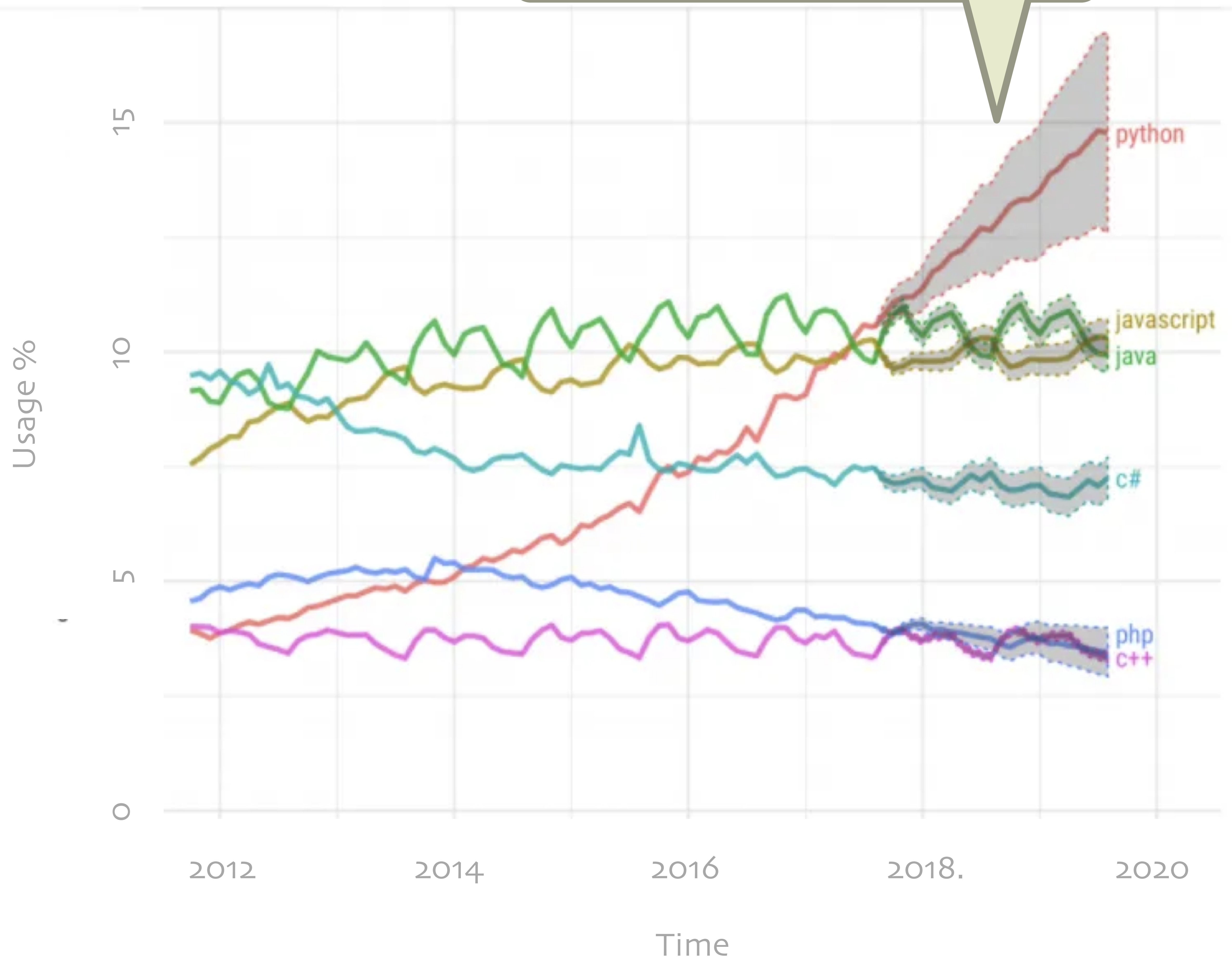
- From forthcoming free tutorials:
    - Anyscale Academy
- Contact Dean for details:
    - dean@anyscale.com

Two Major Trends

Model sizes and therefore compute requirements outstripping Moore's Law

Hence, there is a pressing need for robust, easy to use solutions for distributed Python

Python growth driven by ML/AI and other data science workloads



The ML Landscape Today

All require distributed implementations to scale

Featurization	Streaming	Hyperparam Tuning	Training	Simulation	Model Serving
Apache Spark, hadoop	Apache Spark, kafka	SIGOPT, talos AUTONOMY	PyTorch, scikit learn	MPI, Orleans	

The Ray Vision: Sharing a Common Framework

More libraries coming soon

Ray Community

Community and Resources

- ray.io
- ray.readthedocs.io/en/latest/
- Tutorials (free): Anyscale Academy
- github.com/ray-project/ray.git
- Need help?
    - Ray Slack: ray-distributed.slack.com
    - ray-dev group

Migrating to Ray

If you're already using…

- asyncio
- joblib
- multiprocessing.Pool

For example, from this:

```
from multiprocessing.pool import Pool
```

To this:

```
from ray.util.multiprocessing.pool import Pool
```

- Use Ray's implementations
  - Drop-in replacements
  - Change import statements
  - Break the one-node limitation!

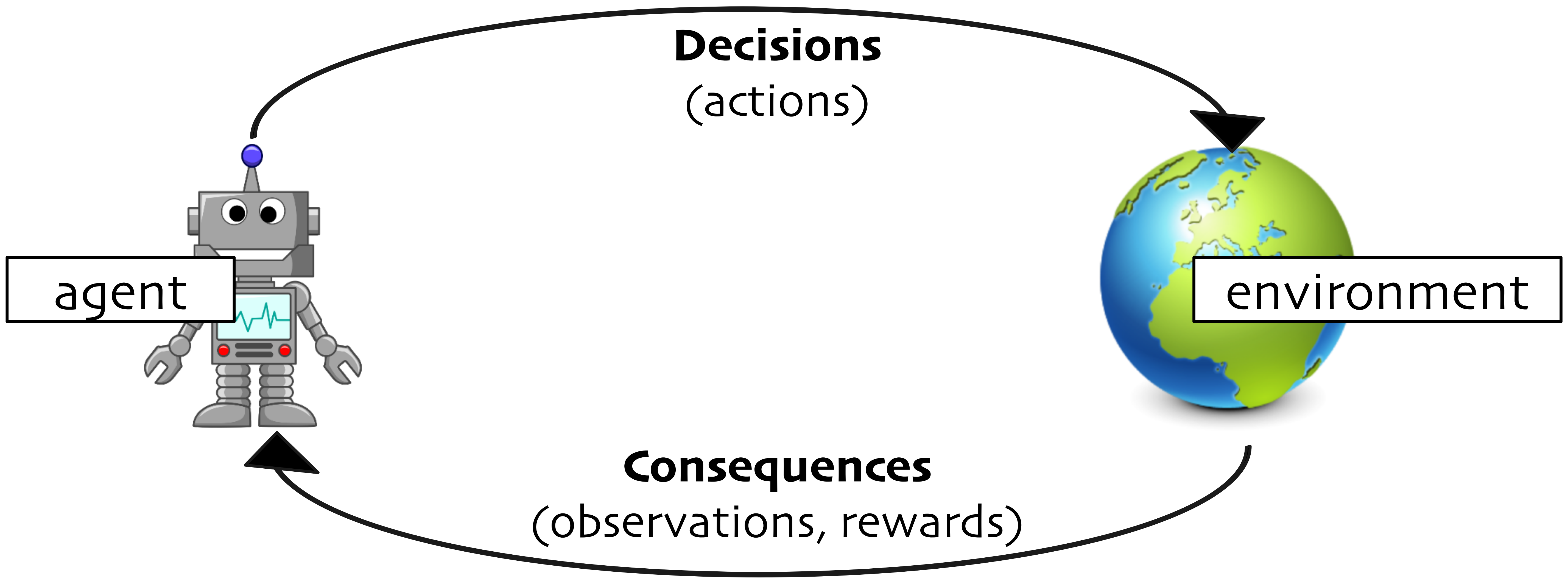
See these blog posts:
https://medium.com/distributed-computing-with-ray/how-to-scale-python-multiprocessing-to-a-cluster-with-one-line-of-code-d19f242f60ff
https://medium.com/distributed-computing-with-ray/easy-distributed-scikit-learn-training-with-ray-54ff8b643b33

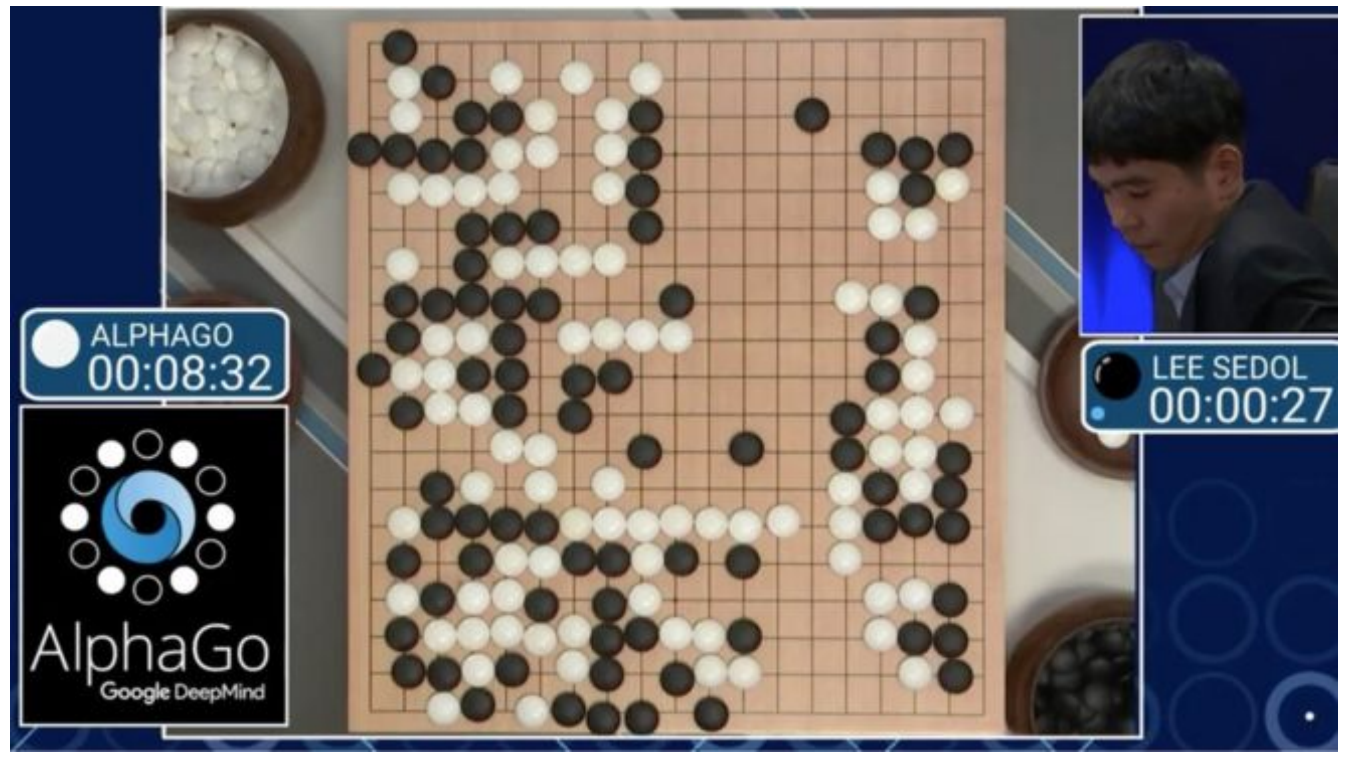
Machine Learning with Ray-based Libraries

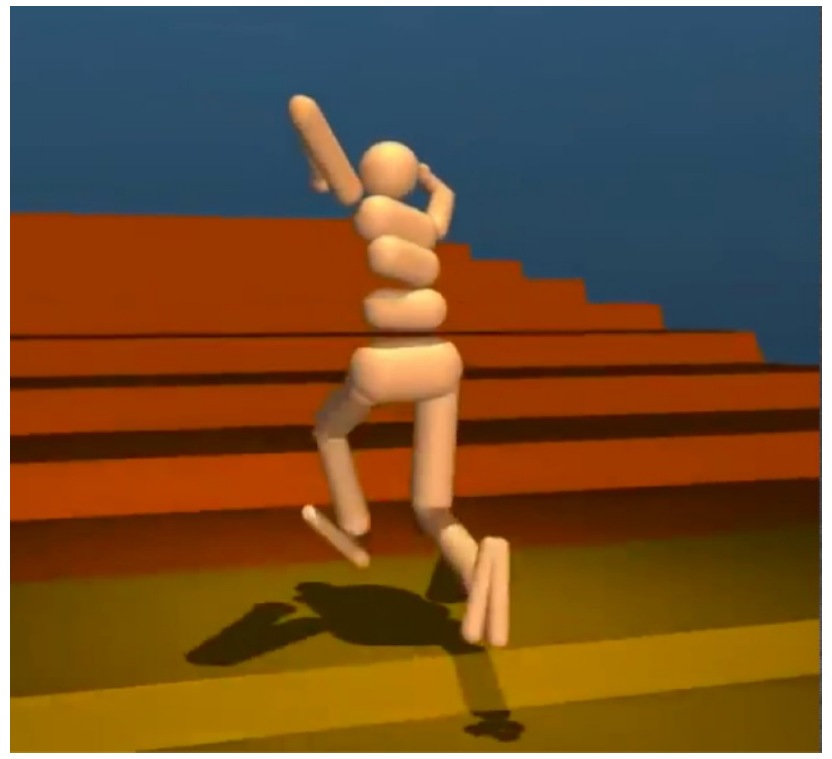
Ray Libraries

Reinforcement Learning - Ray RLlib

Background: Reinforcement Learning



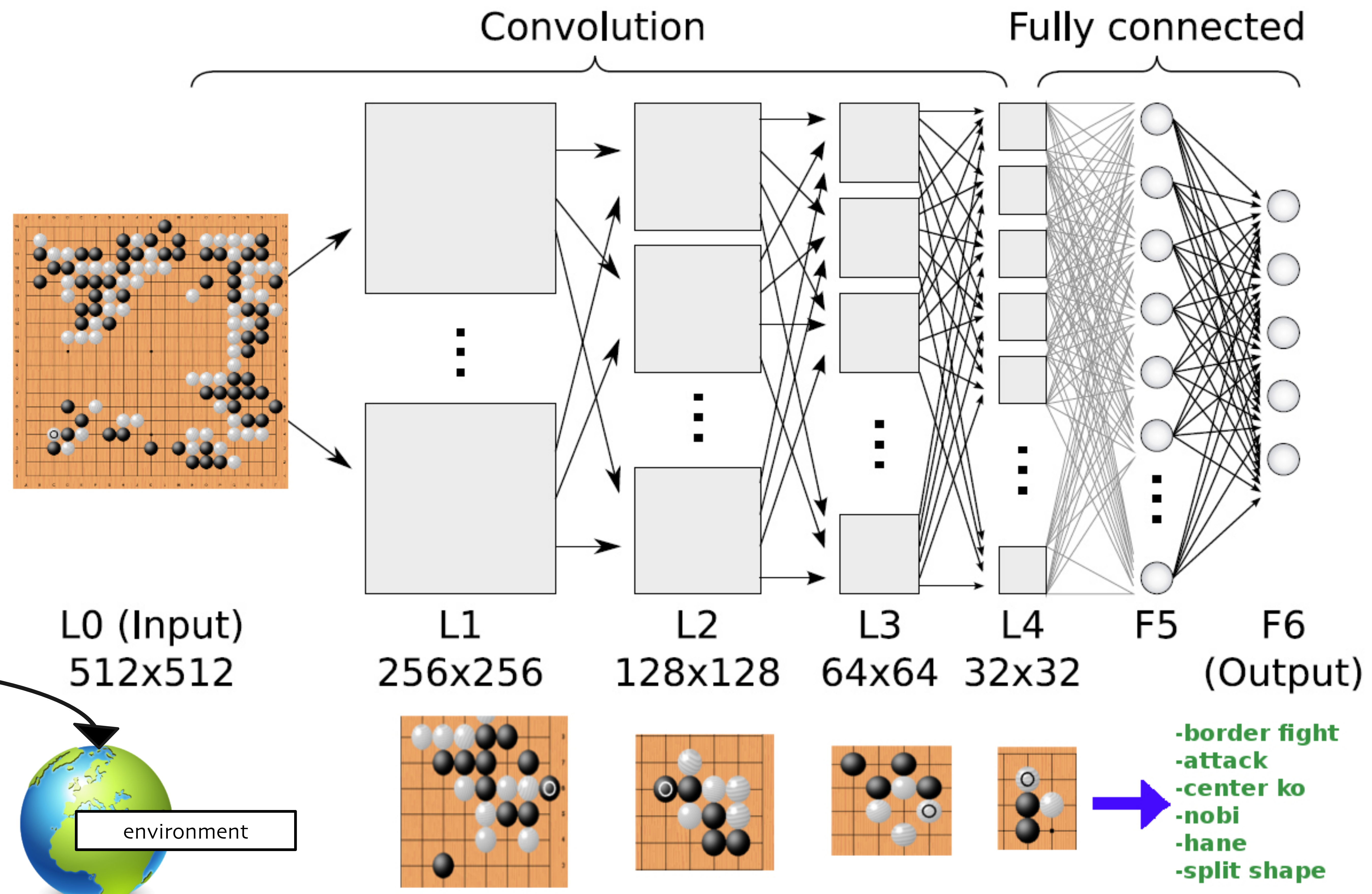


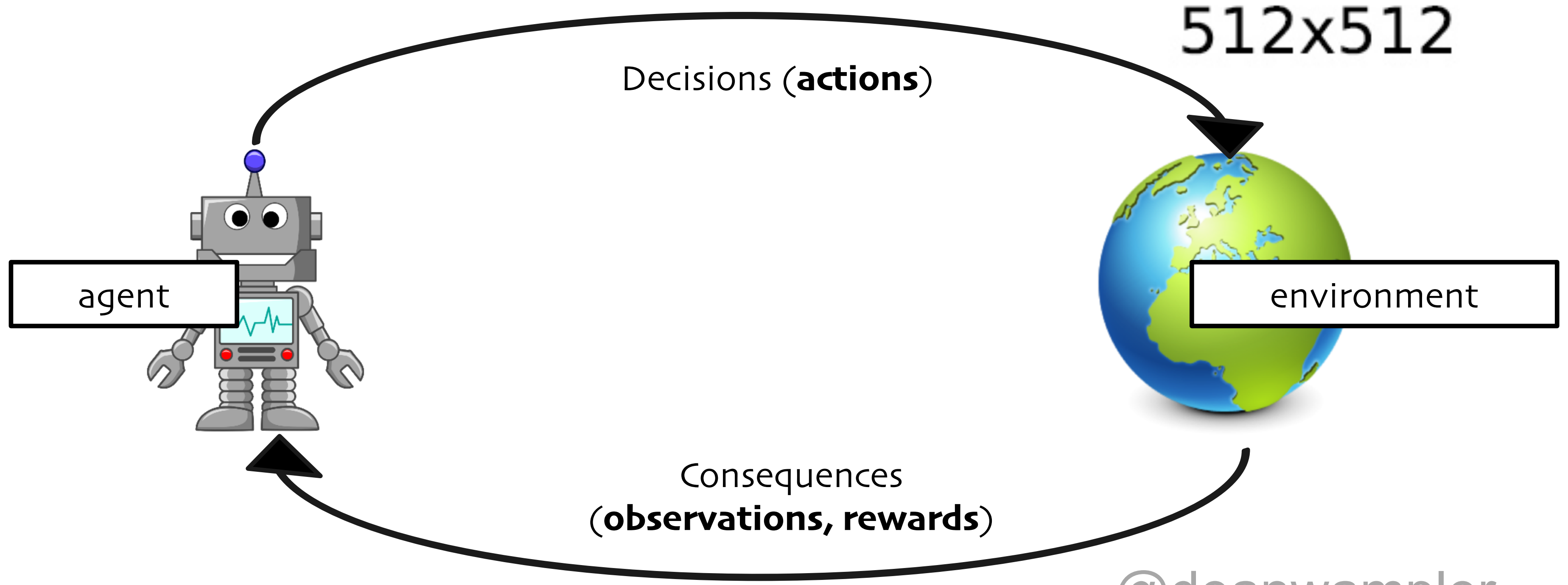


Go as a Reinforcement Learning Problem

AlphaGo (Silver et al. 2016)

- **Observations**:
  - board state
- **Actions**:
  - where to place the stones
- **Rewards**:
  - 1 if win
  - 0 otherwise





Growing Number of RL Applications

Industrial Processes | System Optimization | Advertising | Recommendations | Finance

RL applications

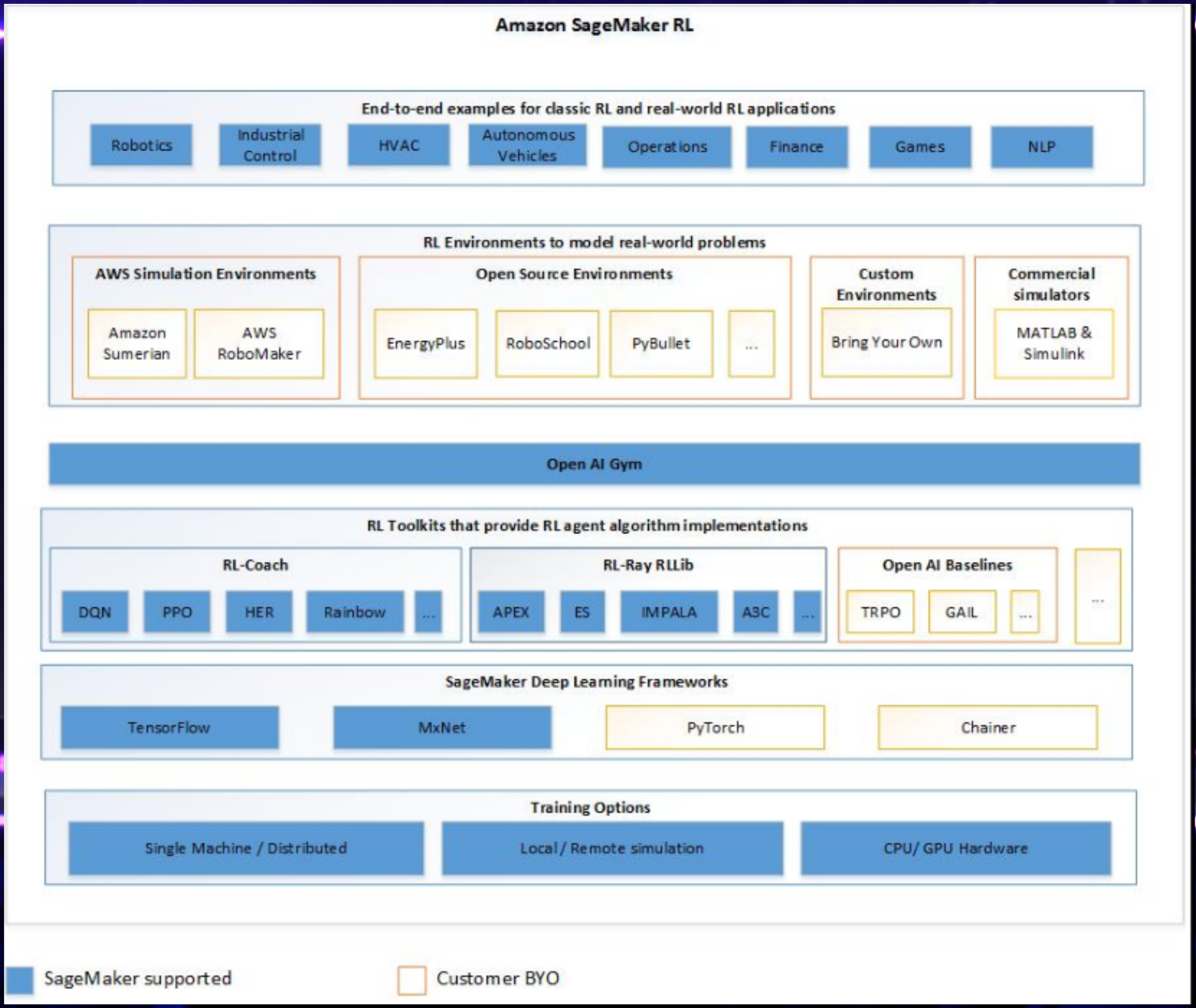
RLlib: A Scalable, Unified Library for RL

@deanwampler

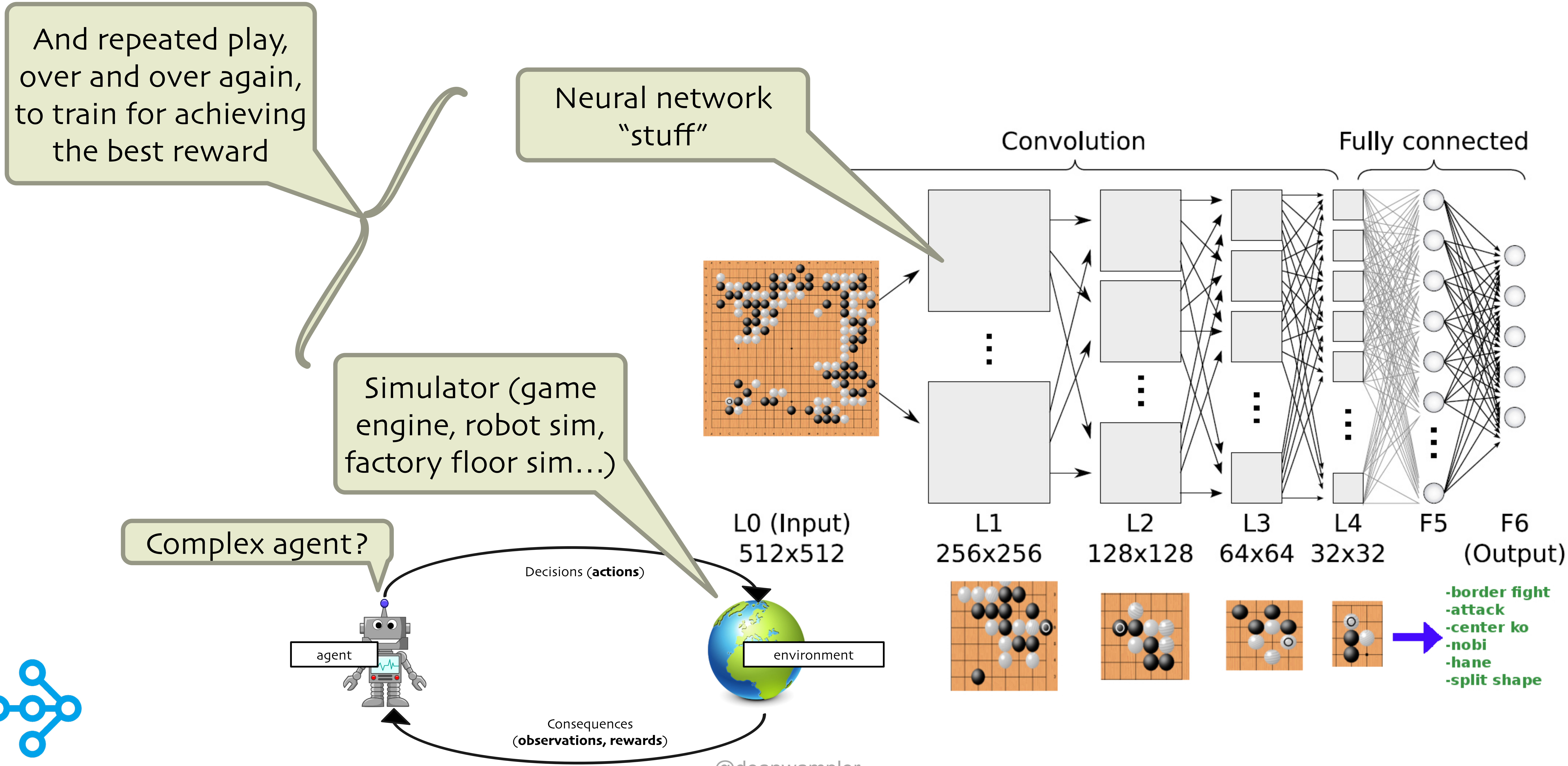
Broad Range of Scalable Algorithms

- High-throughput architectures
  - Distributed Prioritized Experience Replay (Ape-X)
  - Importance Weighted Actor-Learner Architecture (IMPALA)
  - Asynchronous Proximal Policy Optimization (APPO)
- Gradient-based
  - Soft Actor-Critic (SAC)
  - Advantage Actor-Critic (A2C, A3C)
  - Deep Deterministic Policy Gradients (DDPG, TD3)
  - Deep Q Networks (DQN, Rainbow, Parametric DQN)
  - Policy Gradients
  - Proximal Policy Optimization (PPO)
- gradient-free
  - Augmented Random Search (ARS)
  - Evolution Strategies
- Multi-agent specific
  - QMIX Monotonic Value Factorisation (QMIX, VDN, IQN)
- Offline
  - Advantage Re-Weighted Imitation Learning (MARWIL)


Amazon
SageMaker RL
Reinforcement learning for every
develope nd data scientist


Amazon SageMaker RL
End-to-end examples for classic RL and real-world RL applications
Robotics
Industrial Control
HVAC
Autonomous Vehicles
Operations
Finance
Games
NLP
RL Environments to model real-world problems
AWS Simulation Environments
Amazon Sumerian
AWS RoboMaker
Open Source Environments
EnergyPlus
RoboSchool
PyBullet
...
Custom Environments
Bring Your Own
Commercial simulators
MATLAB & Simulink
Open AI Gym
RL Toolkits that provide RL agent algorithm implementations
RL-Coach
DQN
PPO
HER
Rainbow
...
RL-Ray RLLib
APEX
ES
IMPALA
A3C
...
Open AI Baselines
TRPO
GAIL
...
...
SageMaker Deep Learning Frameworks
TensorFlow
MxNet
PyTorch
Chainer
Training Options
Single Machine / Distributed
Local / Remote simulation
CPU/ GPU Hardware
SageMaker supported
Customer BYO

Diverse Compute Requirements Motivated Creation of Ray!

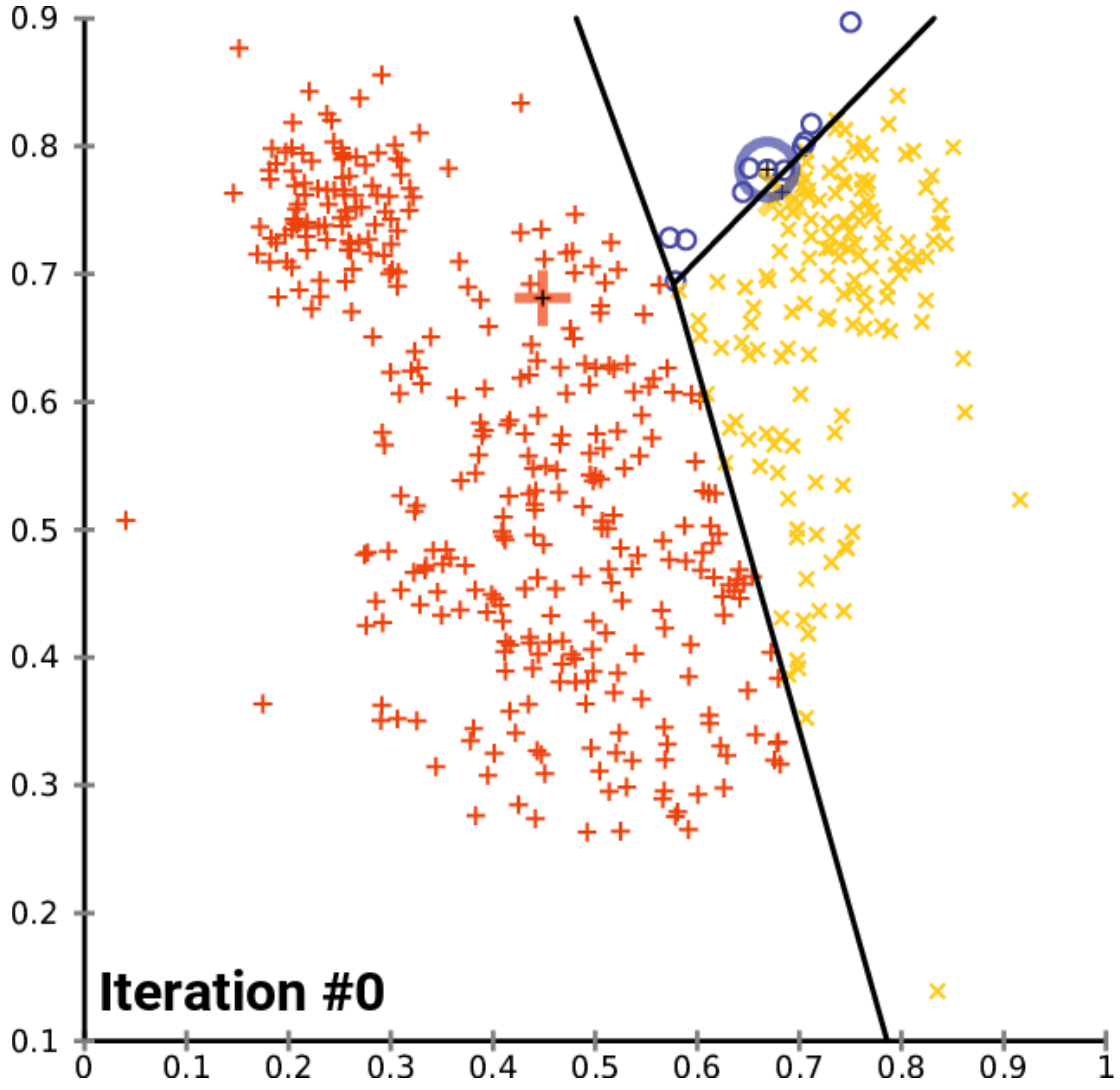


Hyperparameter Tuning - Ray Tune

What Is Hyperparameter Tuning?

Trivial example:

- What’s the best value for “k” in k-means??
  - k is a “hyperparameter”
    - The resulting clusters are defined by “parameters”

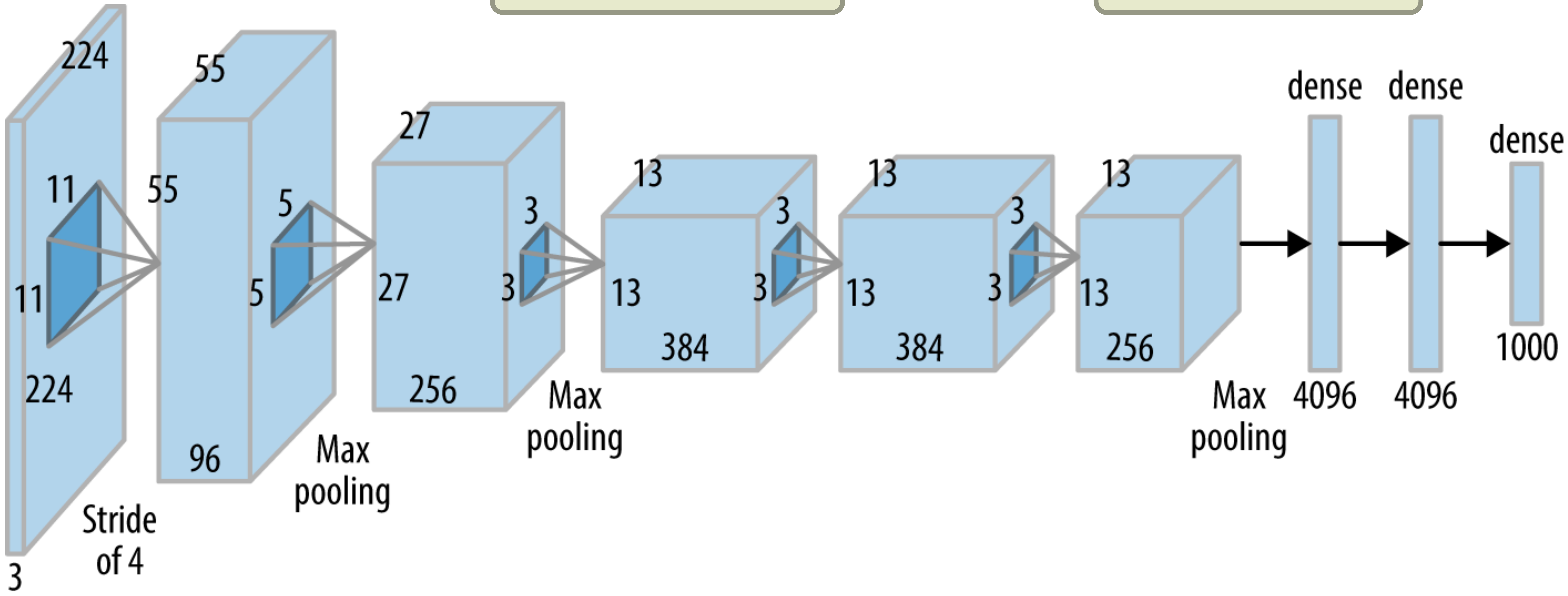


Source: https://commons.wikimedia.org/wiki/File:K-means_convergence.gif

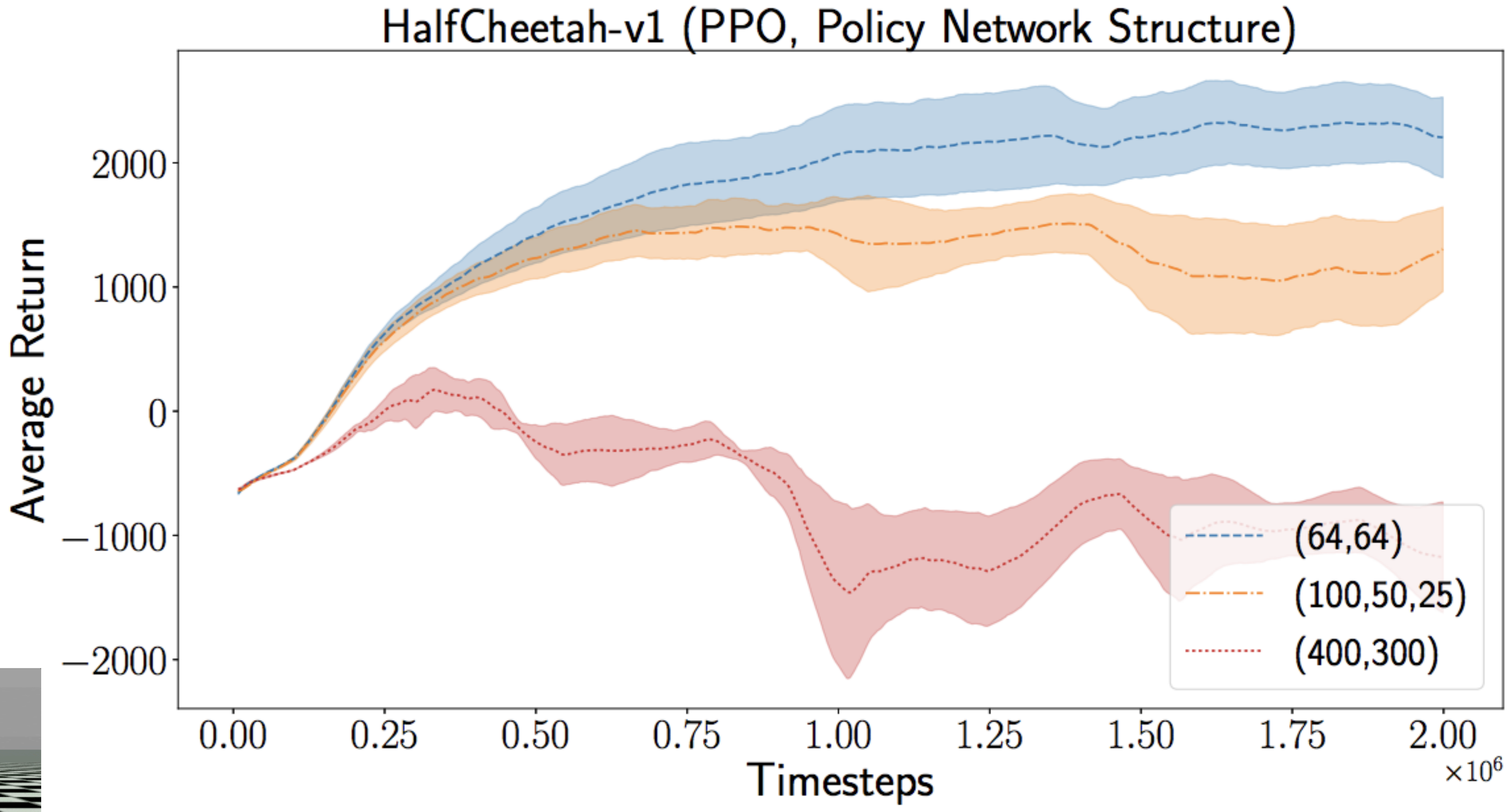
Nontrivial Example - Neural Networks

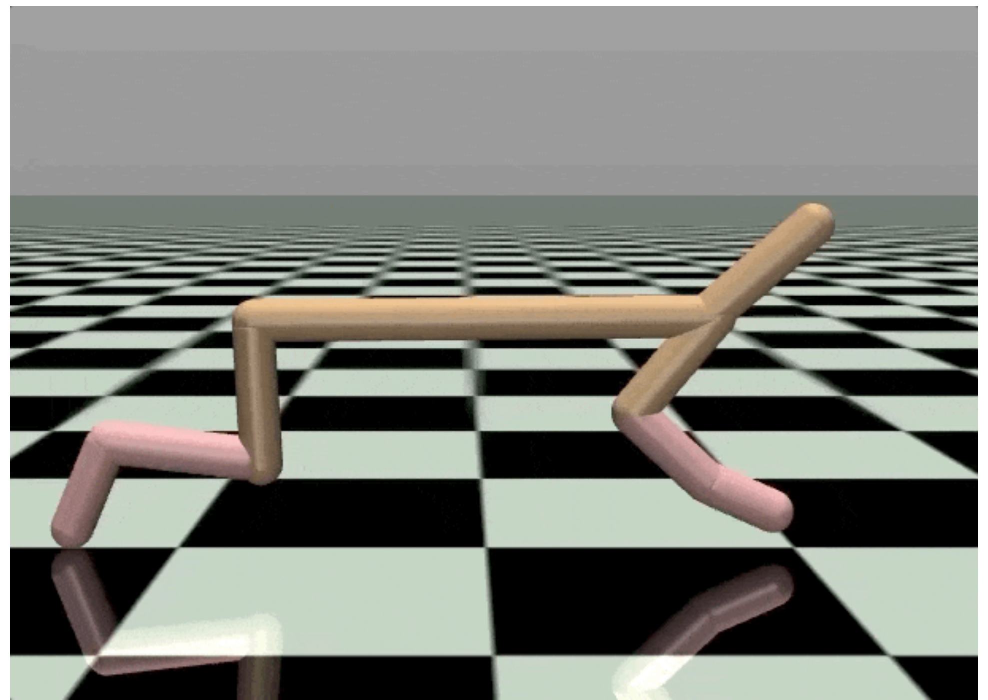
How many layers?
What kinds of layers?

Every number
shown is a
hyperparameter!



Hyperparameters Are Important for Performance





@deanwampler

Why We Need a Framework for Tuning Hyperparameters

We want the best model

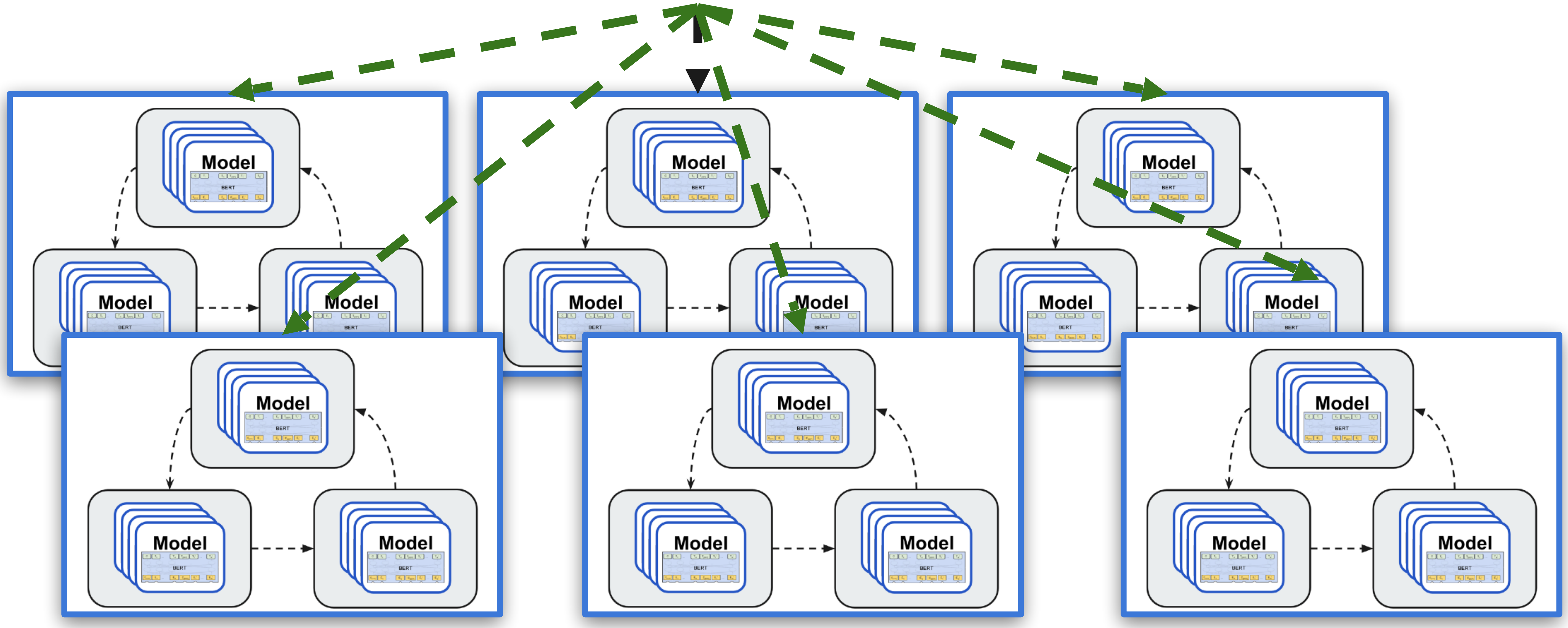
Resources are expensive

Model training is time-consuming

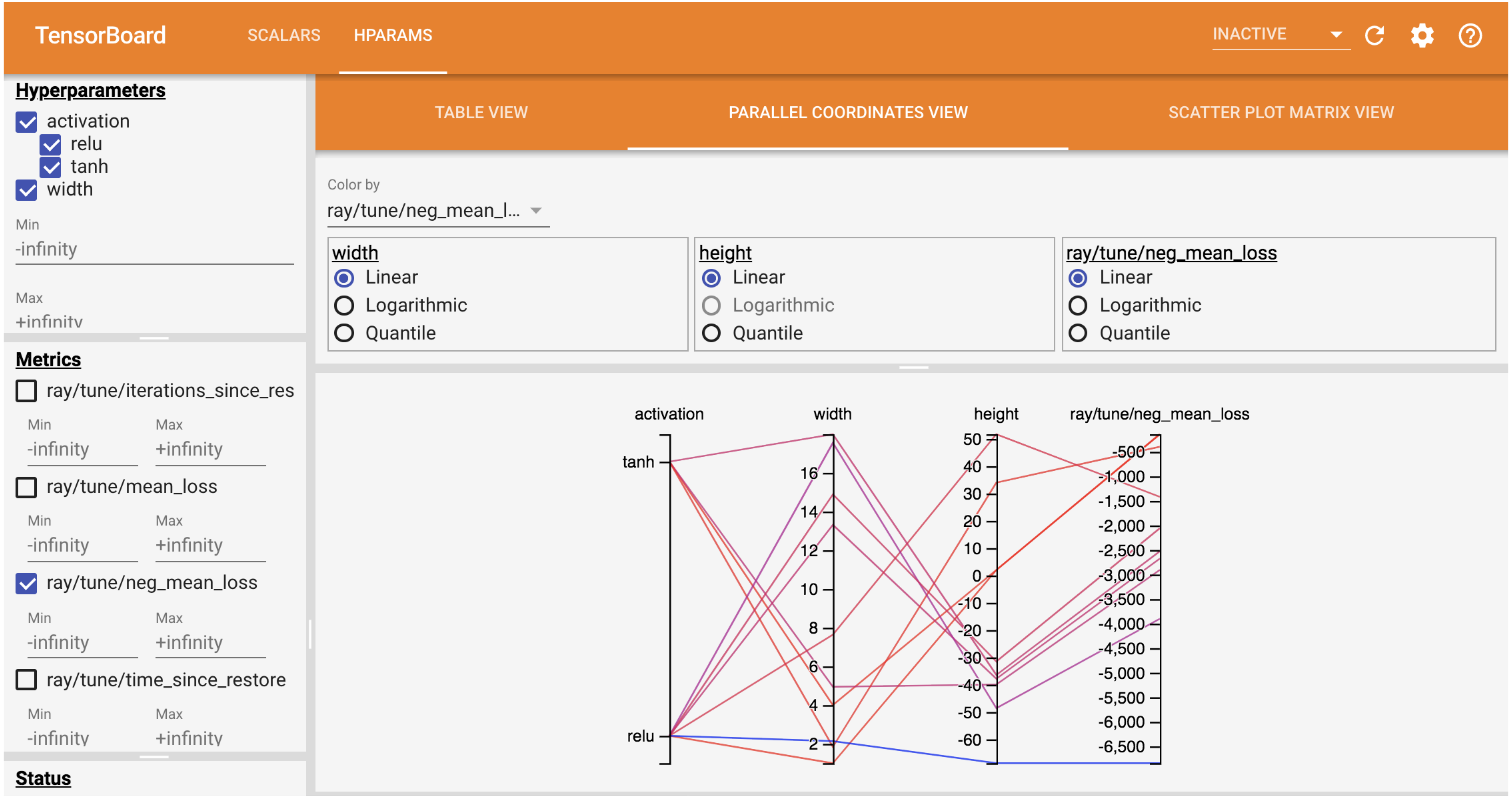
tune

Tuning + Distributed Training

```
tune.run(PytorchTrainable,
      config={
         "model_creator": PretrainBERT,
         "data_creator": create_data_loader,
         "use_gpu": True,
         "num_replicas": 8,
         "lr": tune.uniform(0.001, 0.1)
      },
      num_samples=100,
      search_alg=BayesianOptimization()
)
```



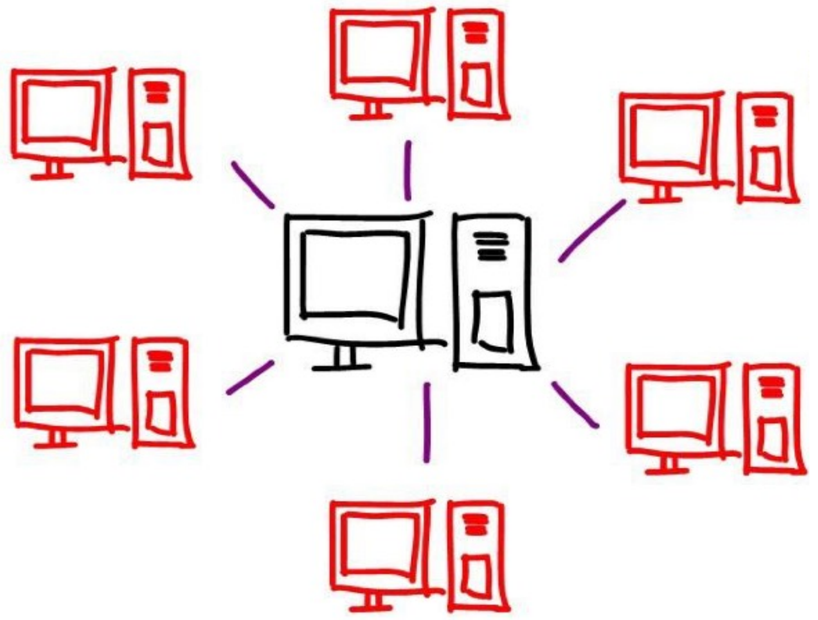
Native Integration with TensorBoard HParams



Tune is Built with Deep Learning as a Priority

Resource Aware Scheduling

Seamless Distributed Execution



Simple API for new algorithms

```
class TrialScheduler:
    def on_result(self, trial, result): ...
    def choose_trial_to_run(self): ...
```

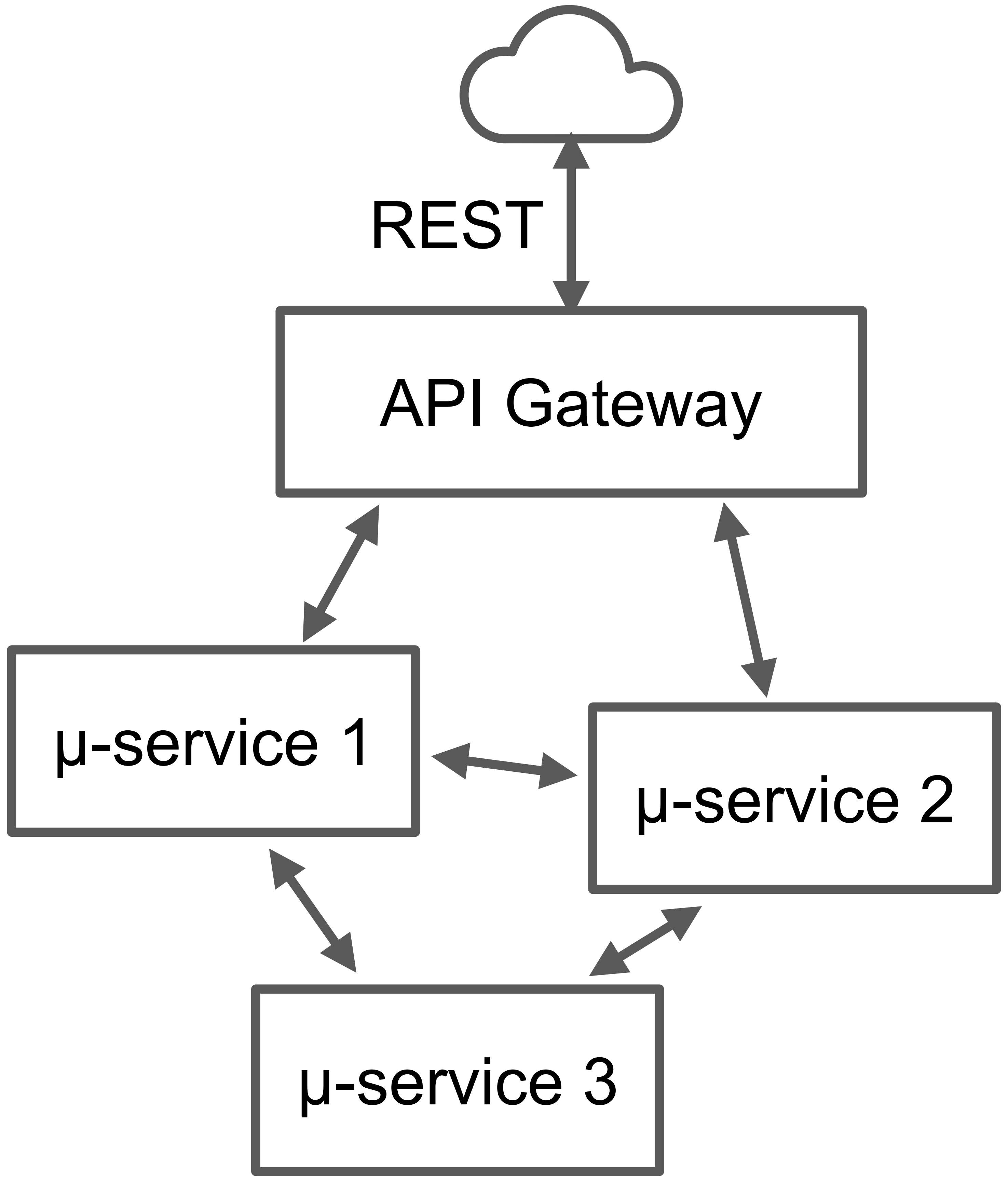
Framework Agnostic

ray.readthedocs.io/en/latest/tune.html

What about Ray for Microservices?

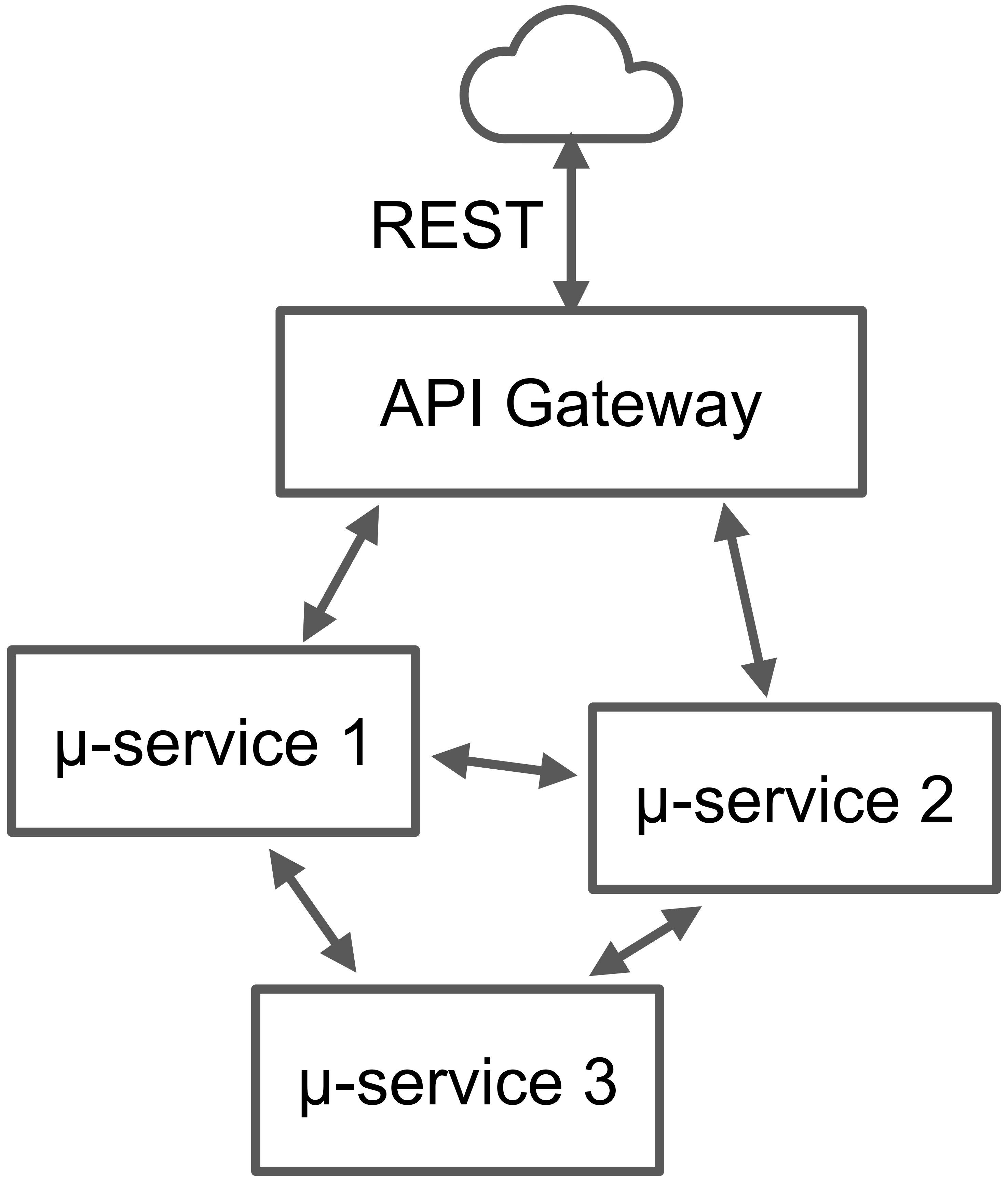
What Are Microservices?

- They partition the domain
  - Conway's Law - Embraced
  - Separate responsibilities
- Separate management



Conway's Law - Embraced

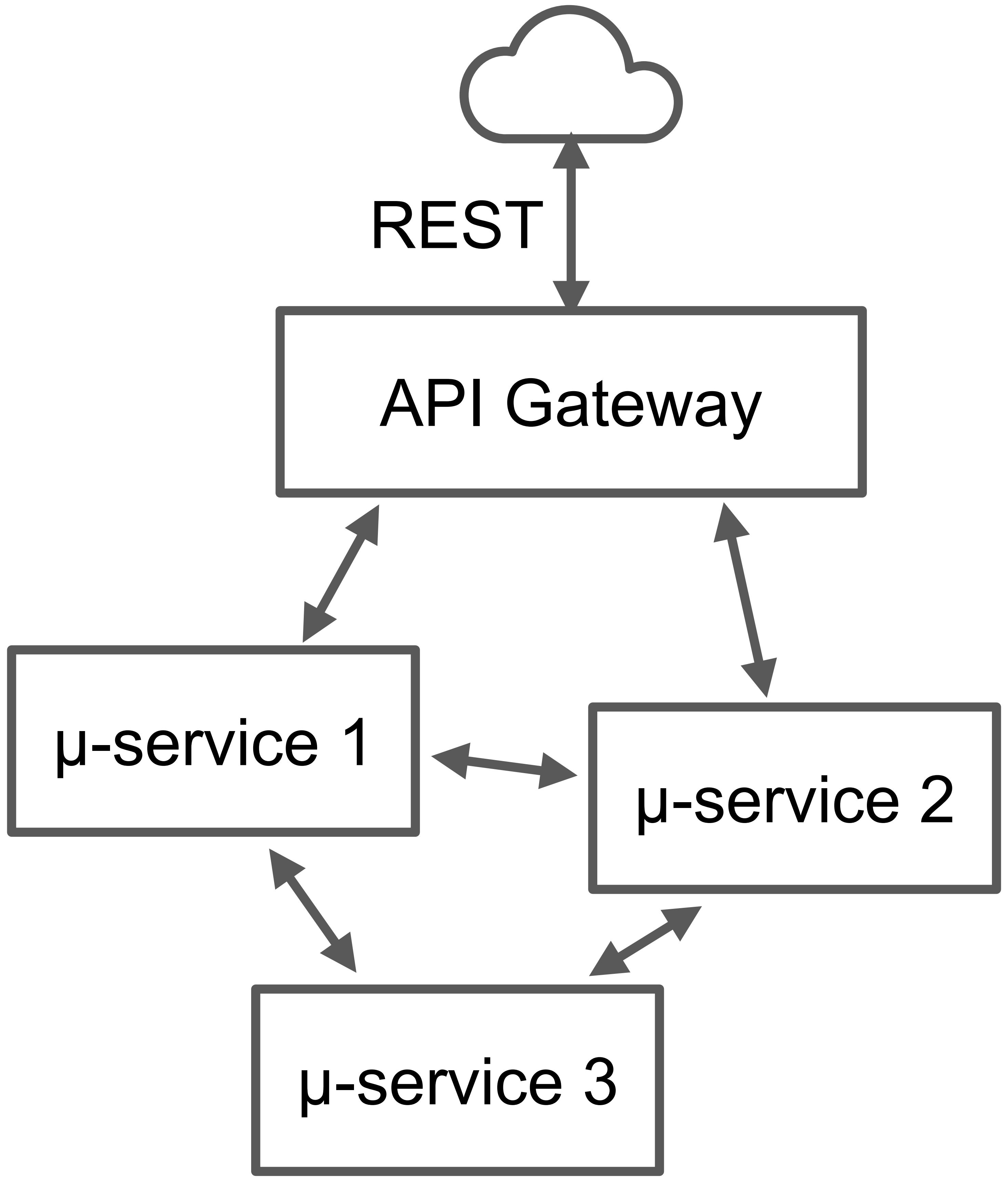
- "Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure"
- Let each team own and manage the services for its part of the domain



en.wikipedia.org/wiki/Conway's_law

Separate Responsibilities

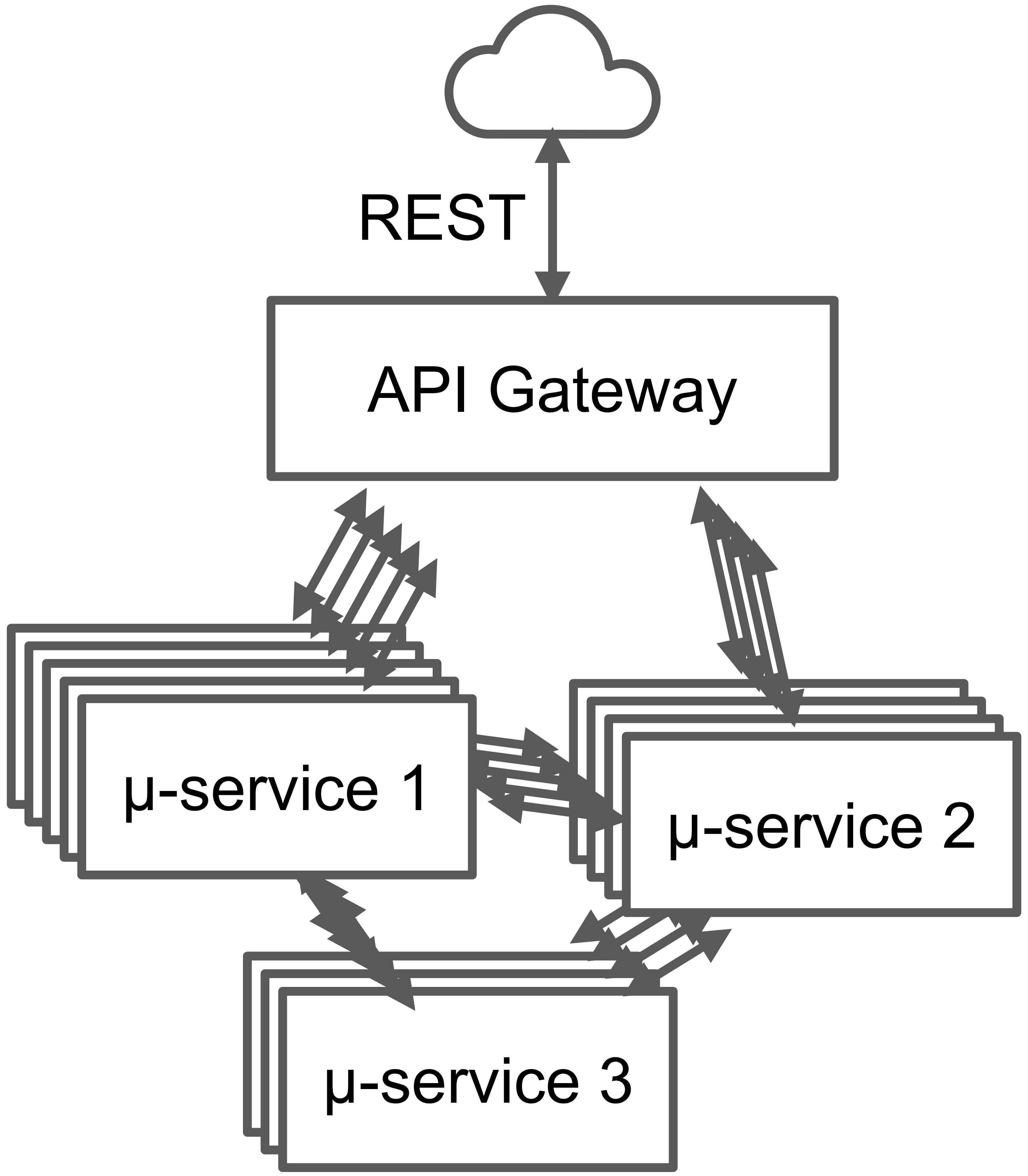
- Each microservice does "one thing", a single responsibility with minimal coupling to the other microservices
- (Like, hopefully, the teams are organized, too…)



wikipedia.org/wiki/Single-responsibility_principle

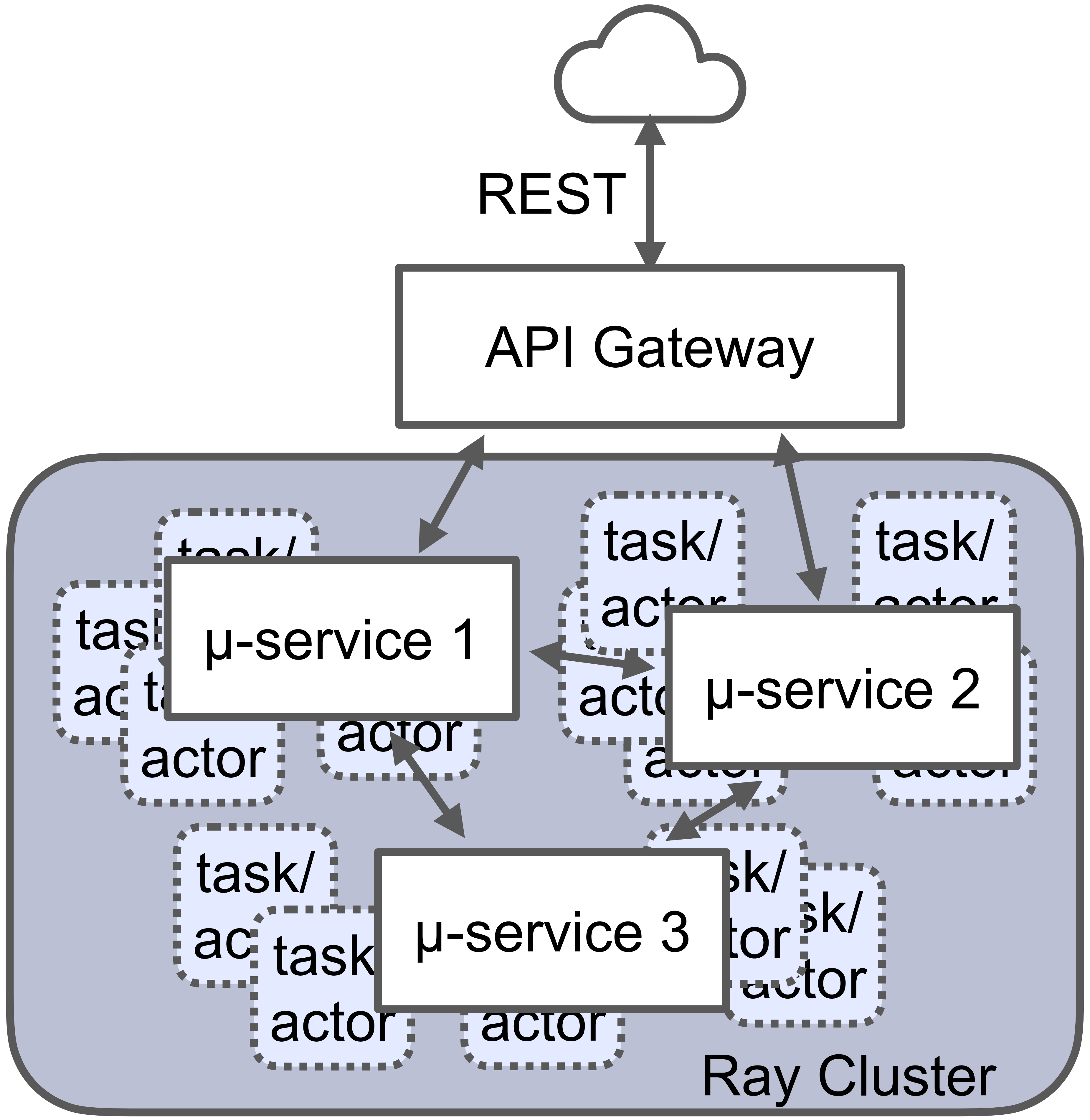
Separate Management

- Each team manages its own instances
- Each microservice has a different number of instances for scalability and resiliency
- But they have to be managed **explicitly**



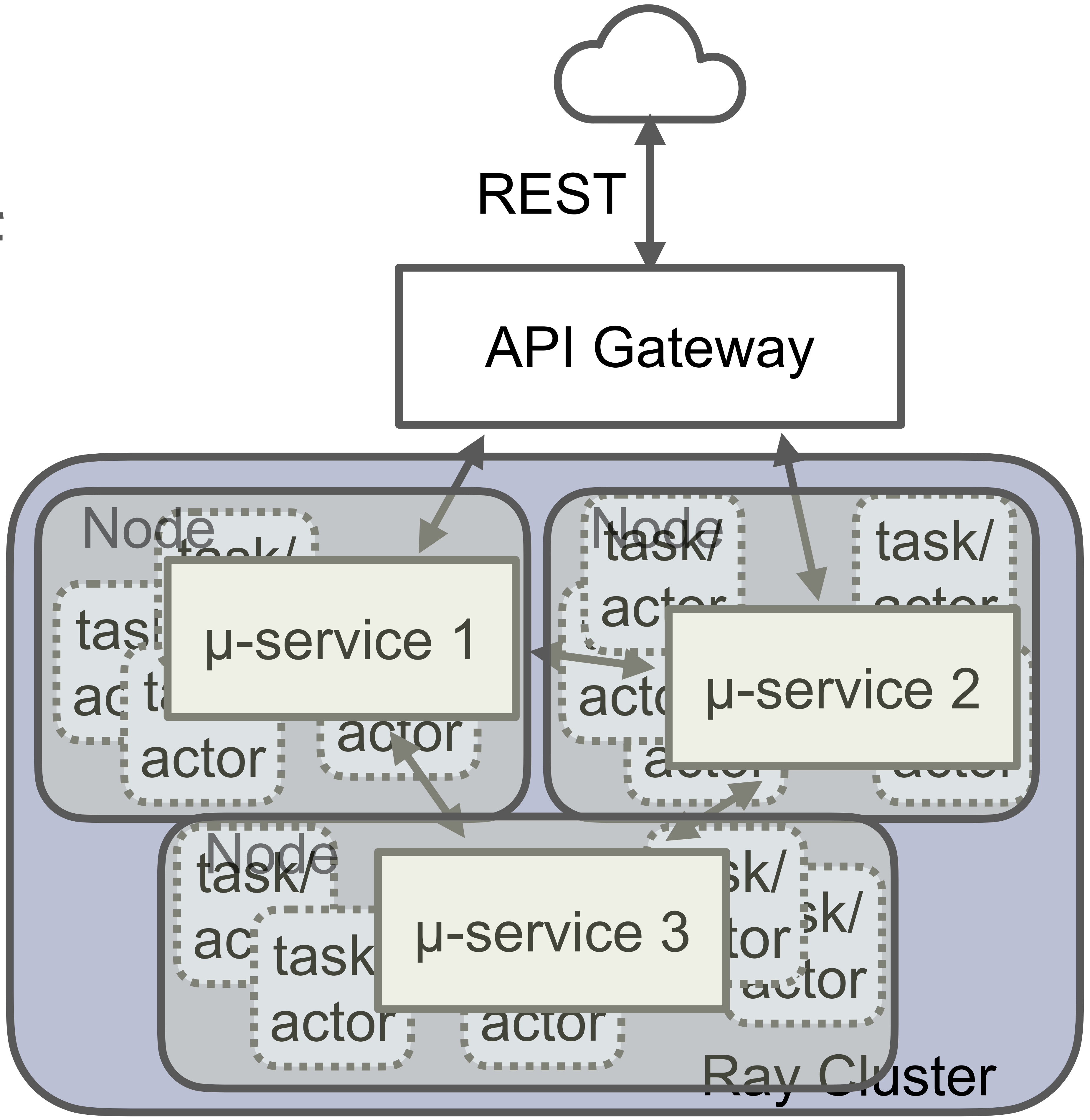
Management - Simplified

- With Ray, you have one "logical" instance to manage and Ray does the cluster-wide scaling for you.



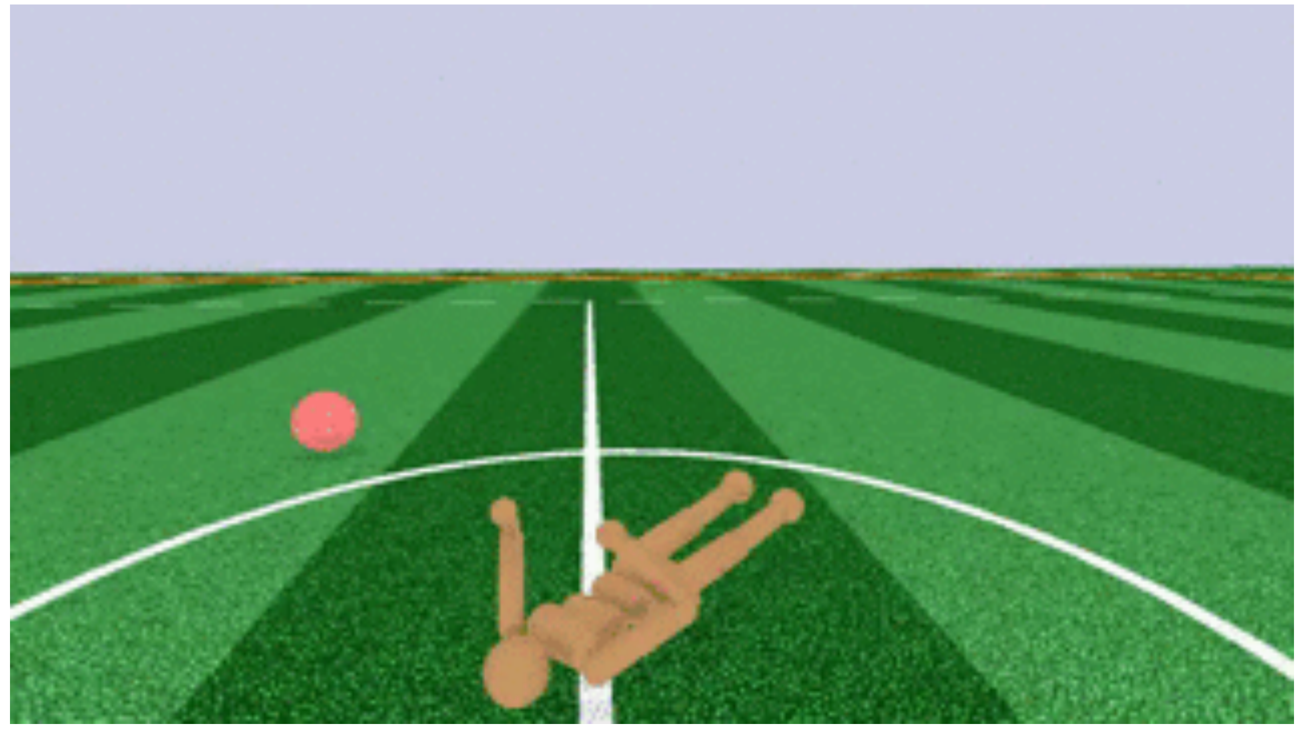
What about Kubernetes (and others…)?

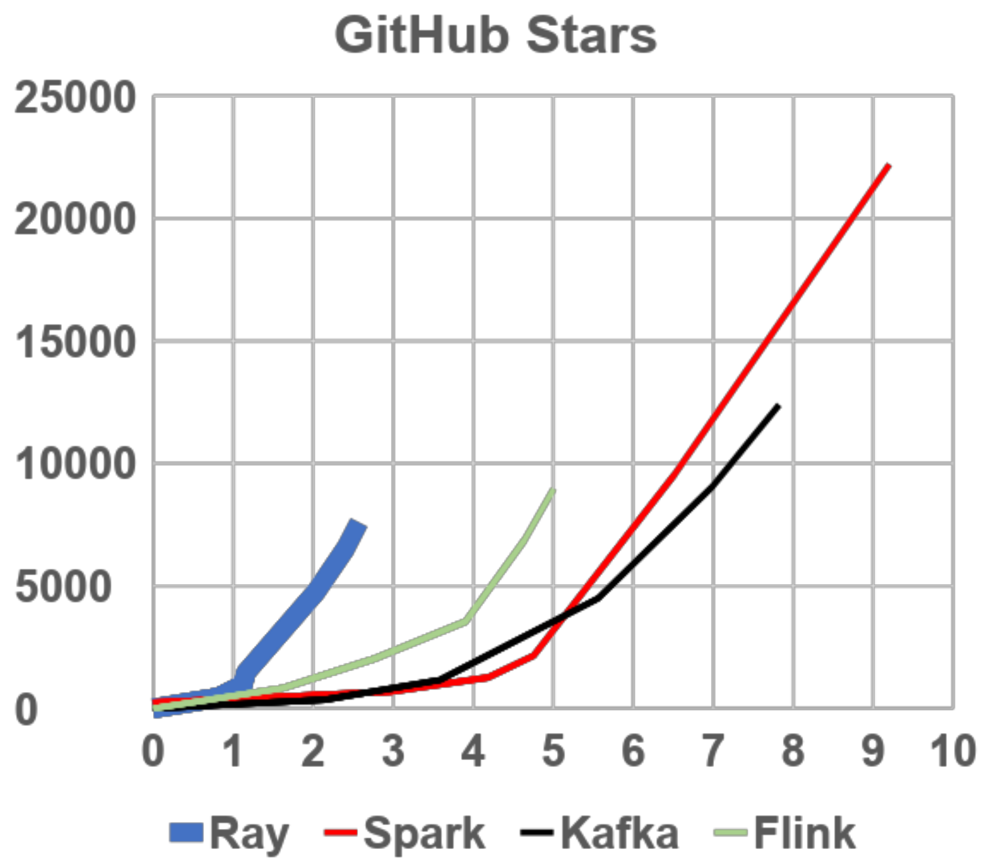
- Ray scaling is very fine grained.
- It operates within the “nodes” of coarse-grained managers
    - Containers, VMs, or physical machines

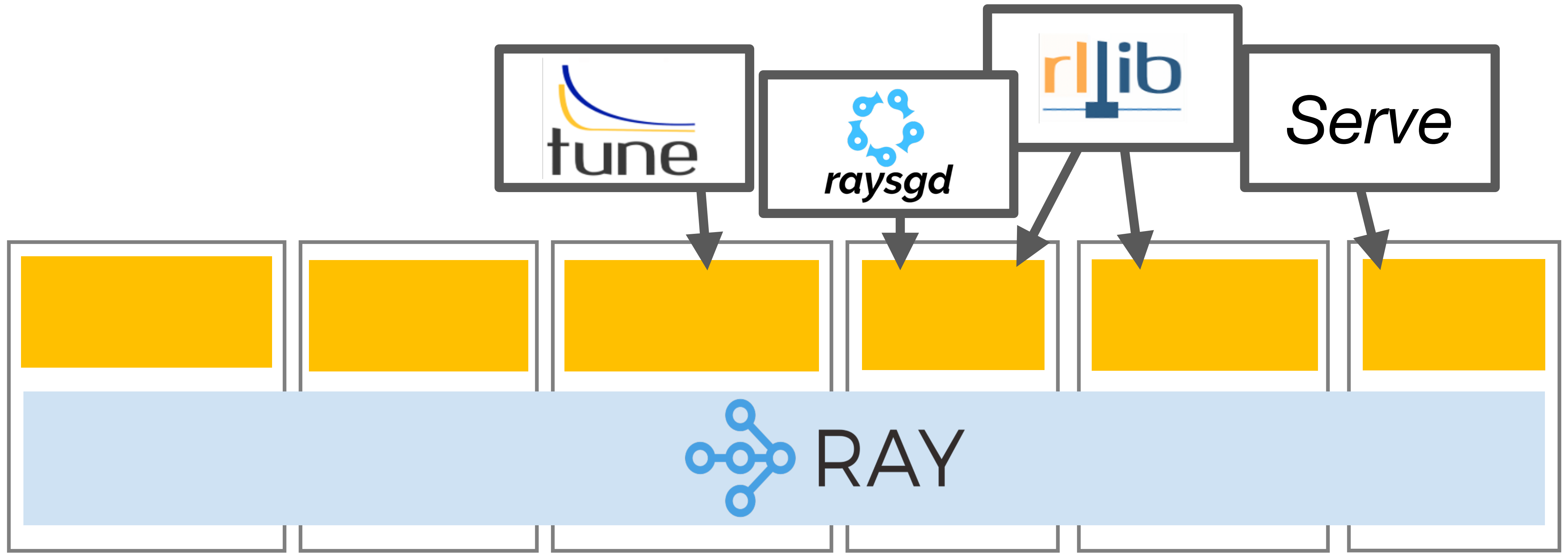


Conclusion

- Ray is the new state-of-the-art for distributed computing
  - The shortest path from your laptop to the cloud
  - Run complex distributed tasks on large clusters from simple code on your laptop







About Anyscale, Inc

- Spun out of U.C. Berkeley
- Making Ray the standard for distributed computing
- We are hiring!
- https://anyscale.com

anyscale

Questions?

ray.io
anyscale.com - We're Hiring!
anyscale.com/events
raysummit.org
dean@anyscale.com

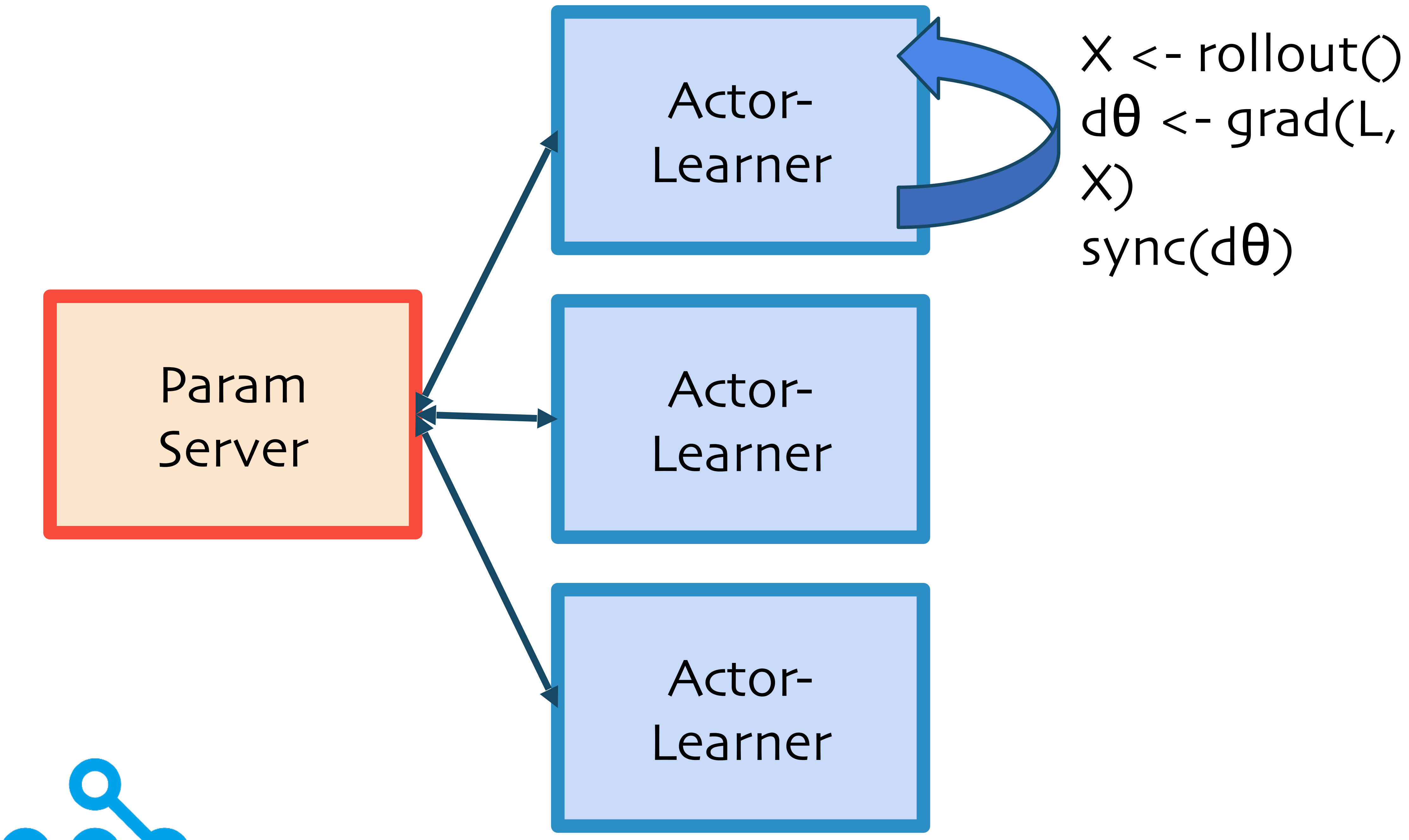
RAY SUMMIT
CONNECT

Extra Slides

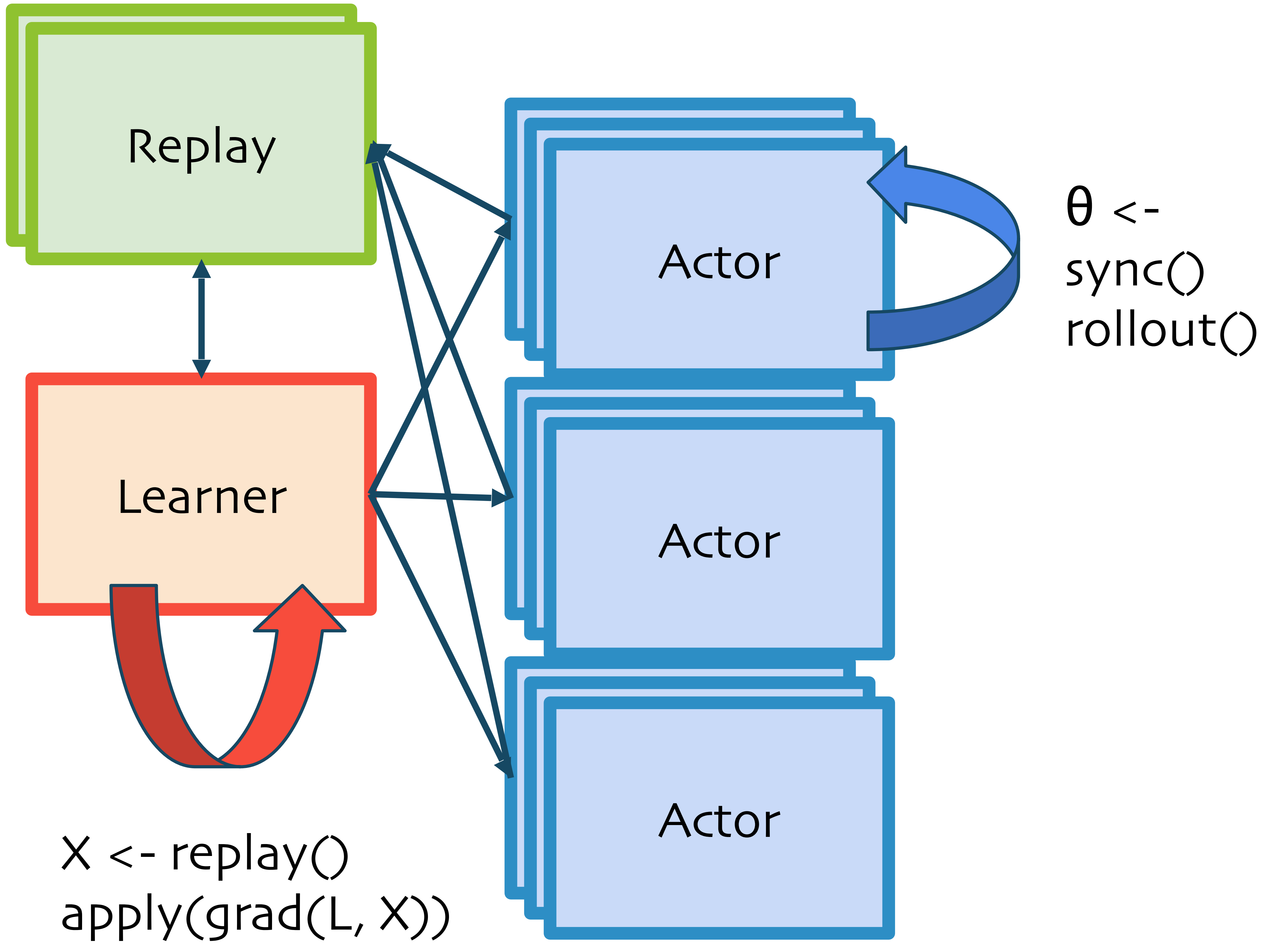
Returning to RLlib…

Many Practical RL Loop Decompositions

Async DQN (Mnigh et al, 2016)

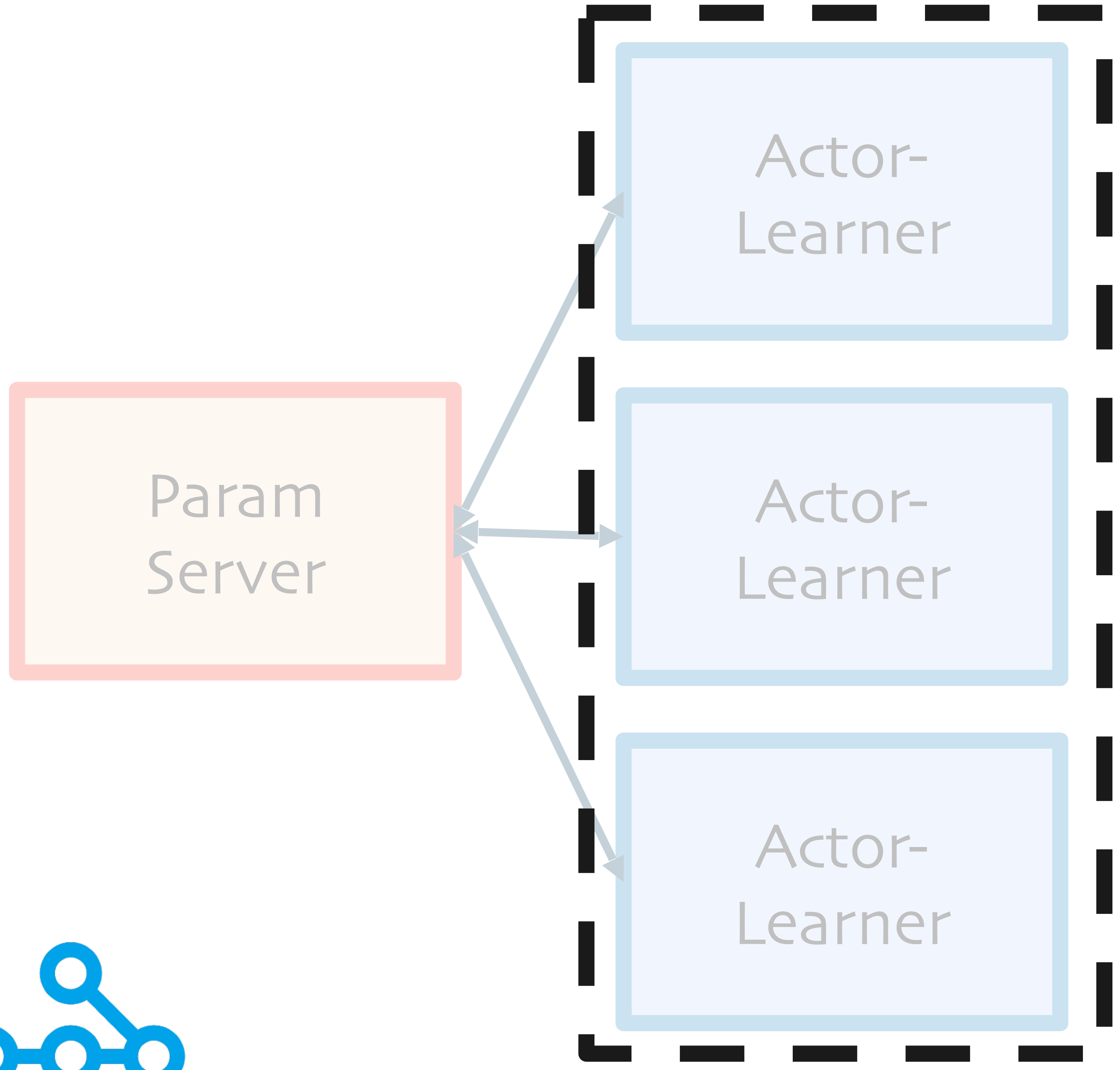


Ape-X DQN (Horgan et al, 2018)



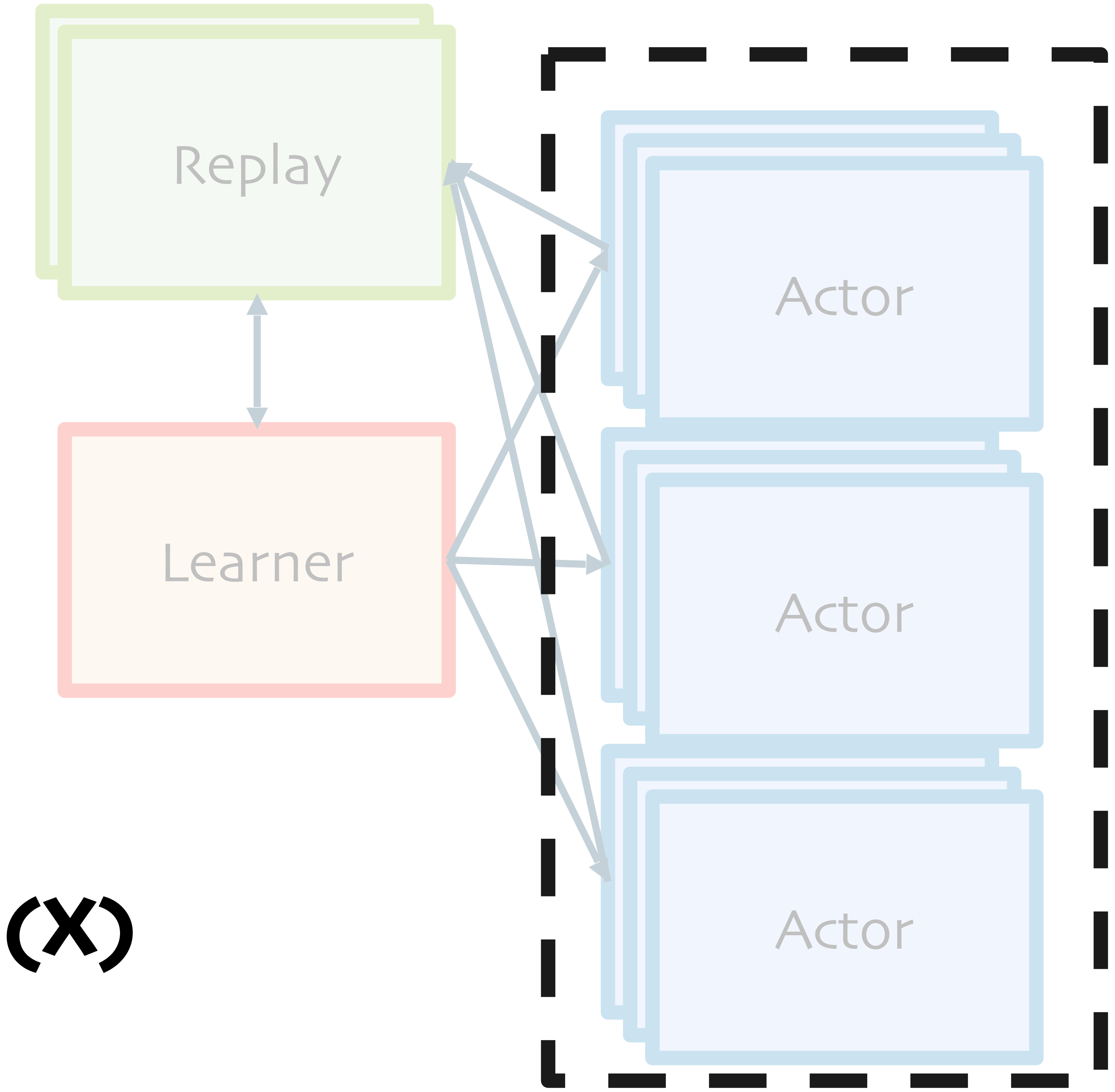
Many Practical RL Loop Decompositions

Async DQN (Mnigh et al, 2016)



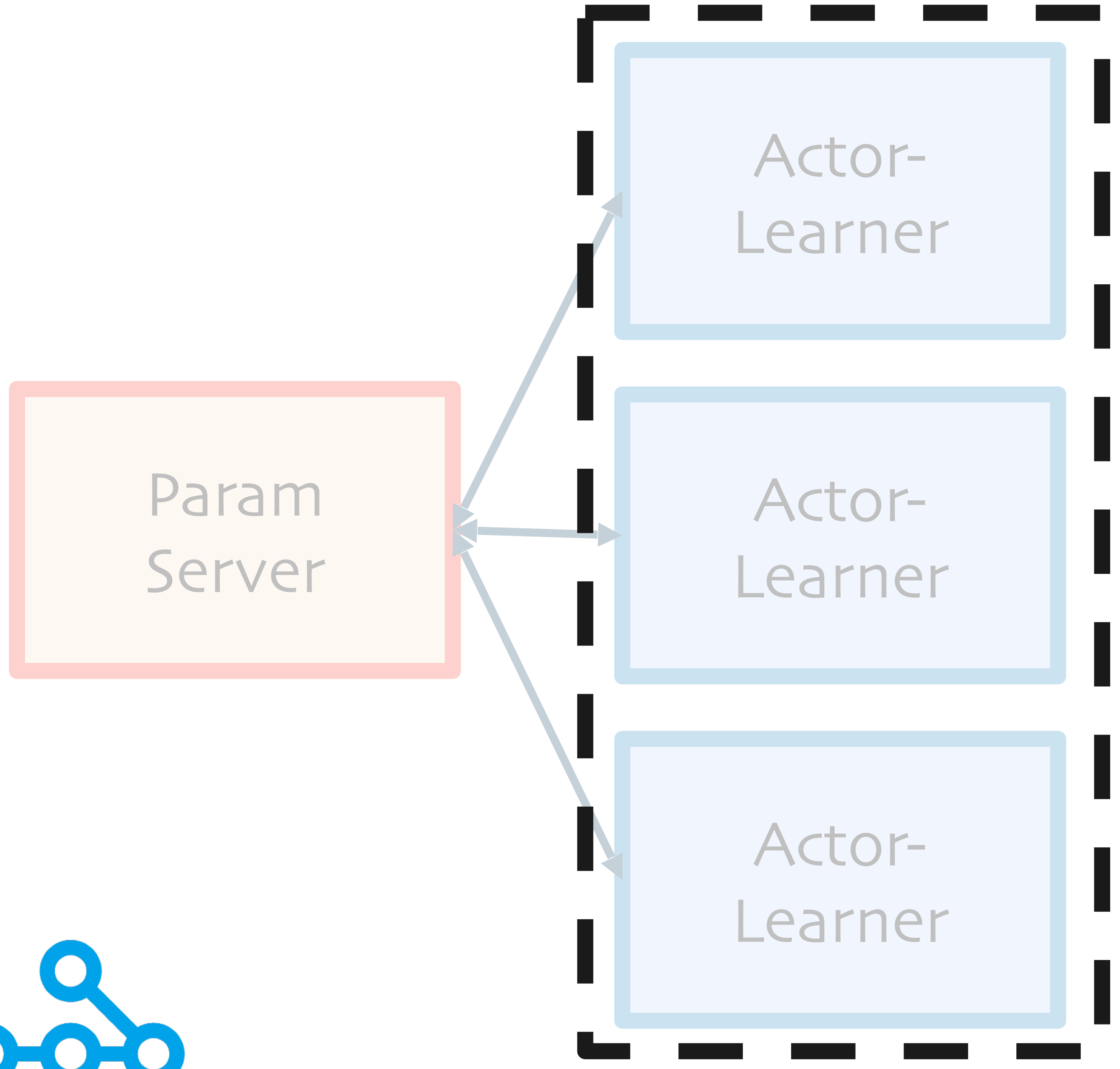
Policy $\pi_\theta(o_t)$
Trajectory
postprocessor $\rho_\theta(X)$
Loss $L(\theta,X)$

Ape-X DQN (Horgan et al, 2018)



Many Practical RL Loop Decompositions

Async DQN (Mnigh et al, 2016)



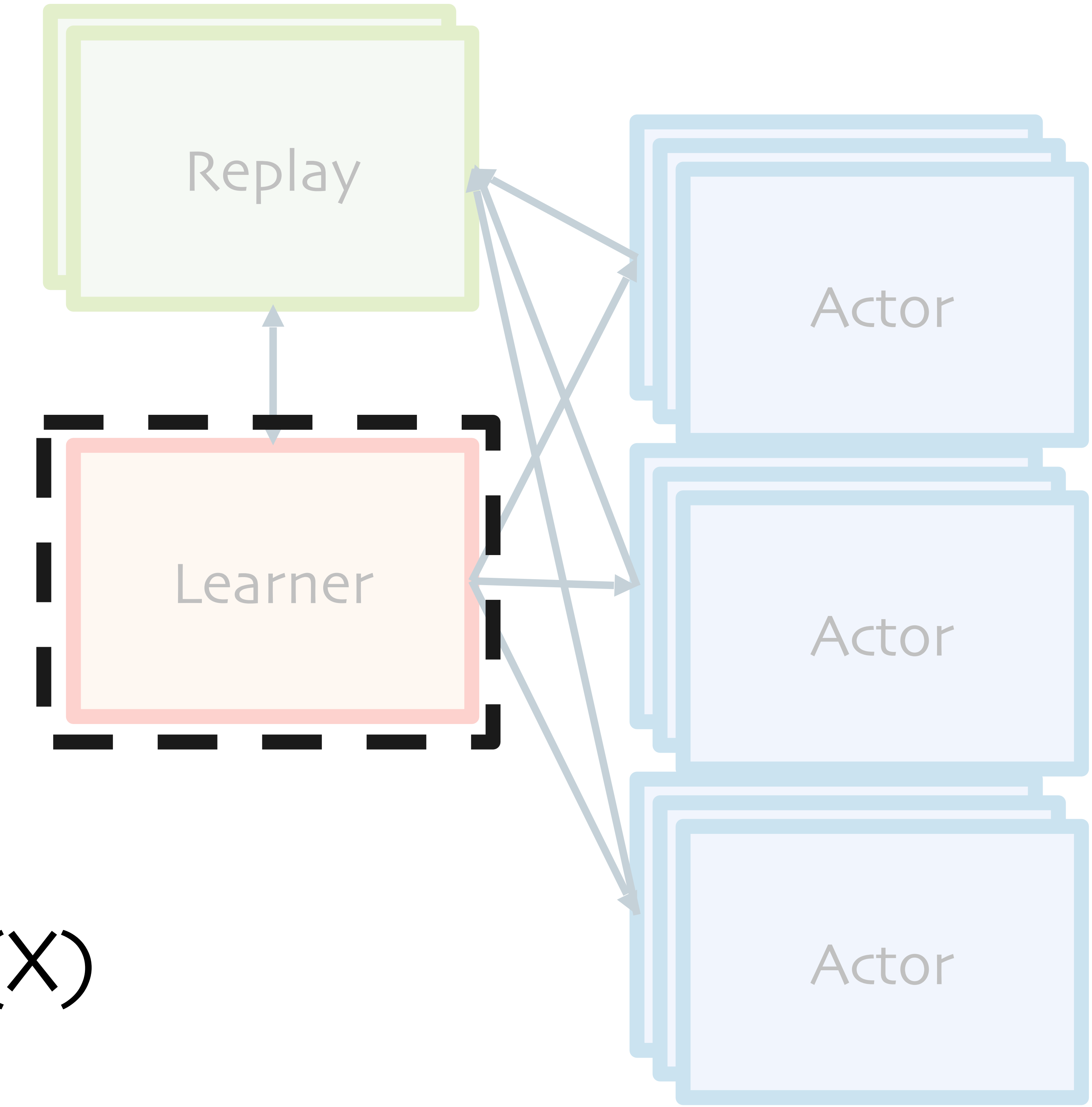
Policy $\pi_\theta(o_t)$

Trajectory

postprocessor $\rho_\theta(X)$

Loss $L(\theta,X)$

Ape-X DQN (Horgan et al, 2018)



Many Practical RL Loop Decompositions

A big motivation for Ray:
No existing system effectively met all the varied demands of RL workloads.

We Need Abstractions for RL

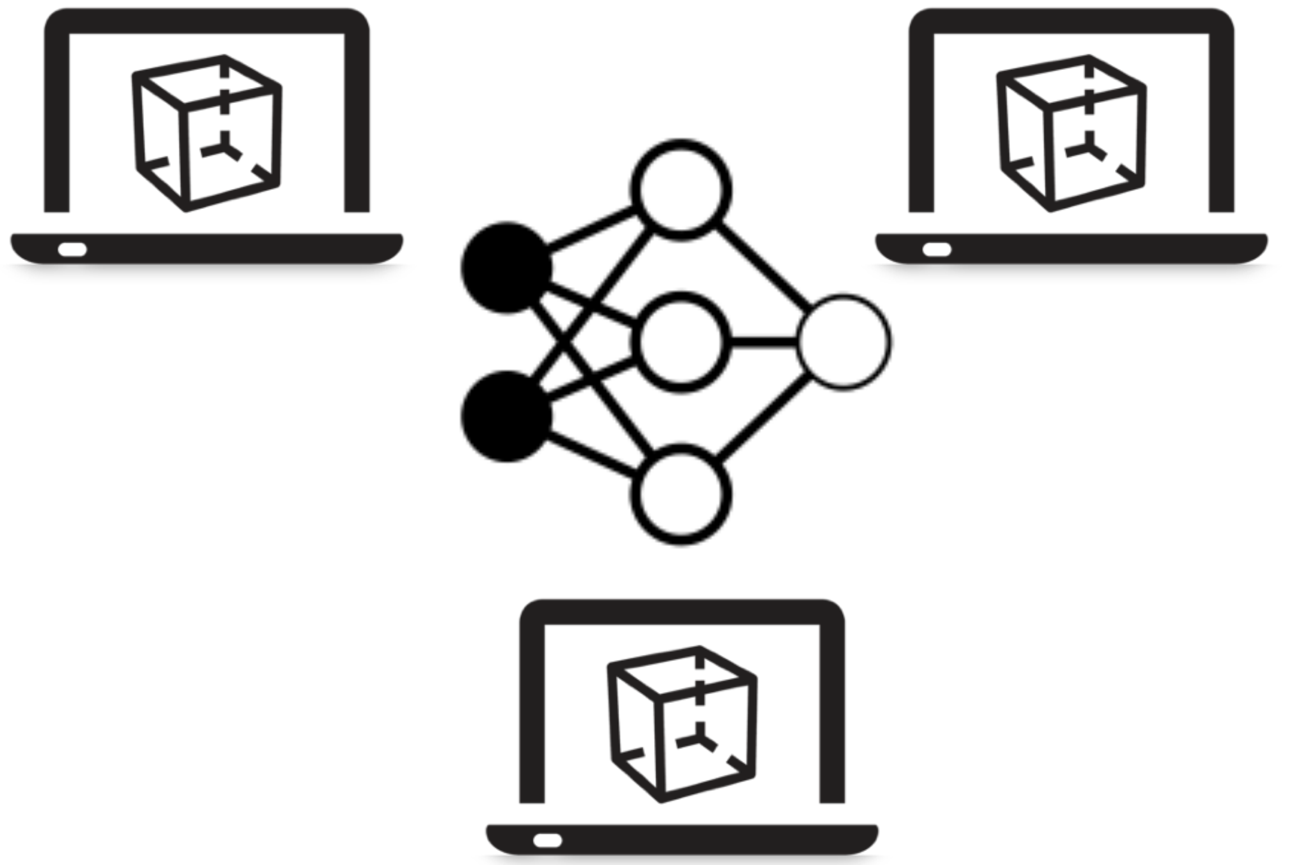
Good abstractions decompose RL algorithms into reusable components.

Goals:

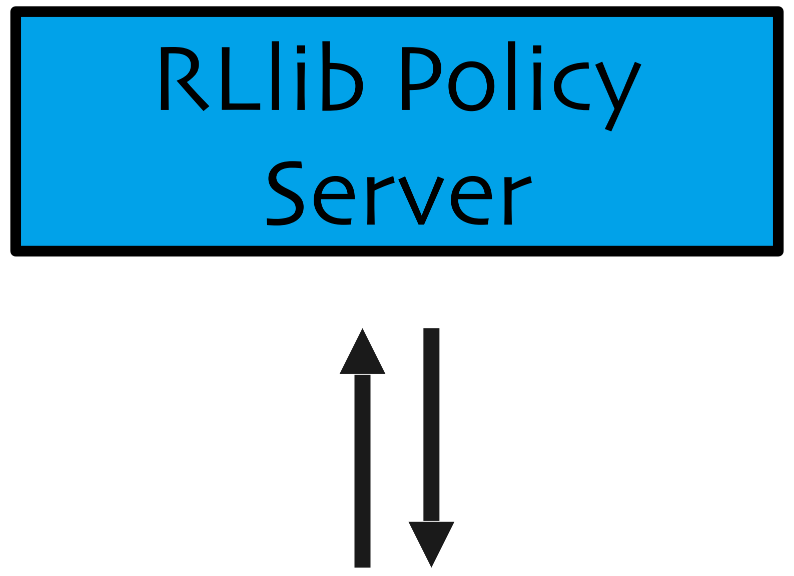
- Code reuse across deep learning frameworks
- Scalable execution of algorithms
- Easily implement, compare, and reproduce algorithms

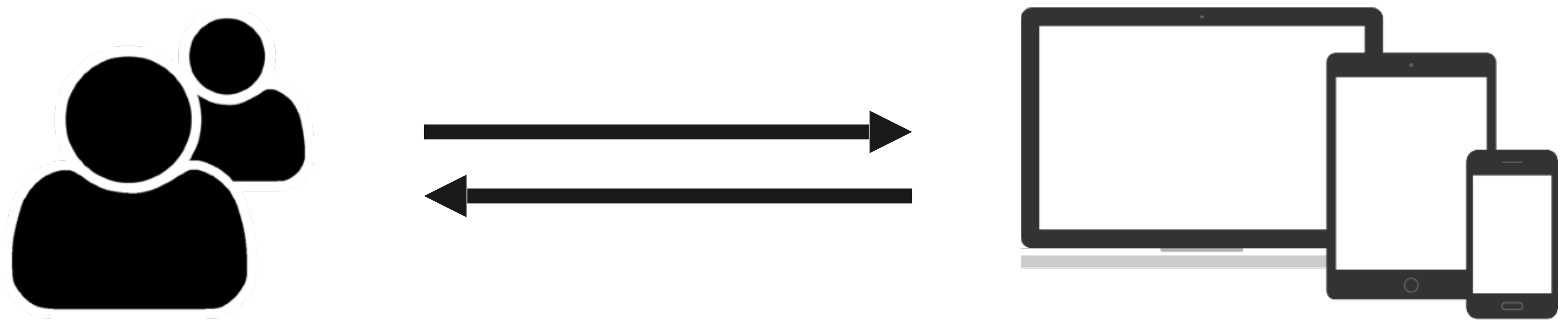
General Purpose APIs: Even More Requirements

Training in Simulation

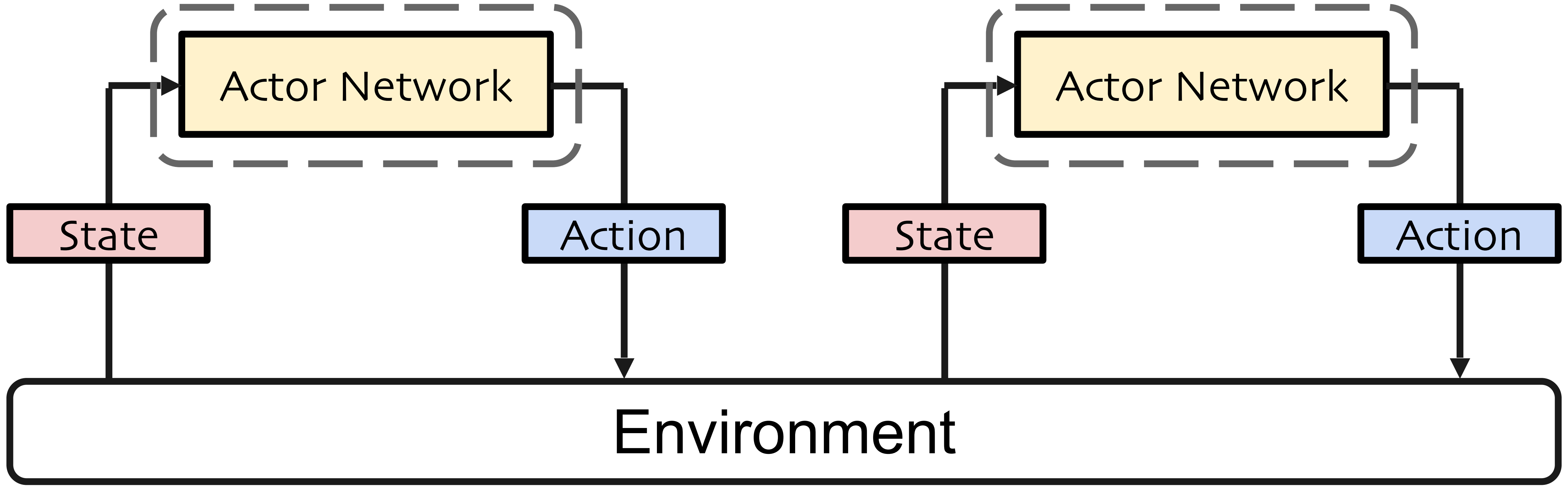


Policy Serving



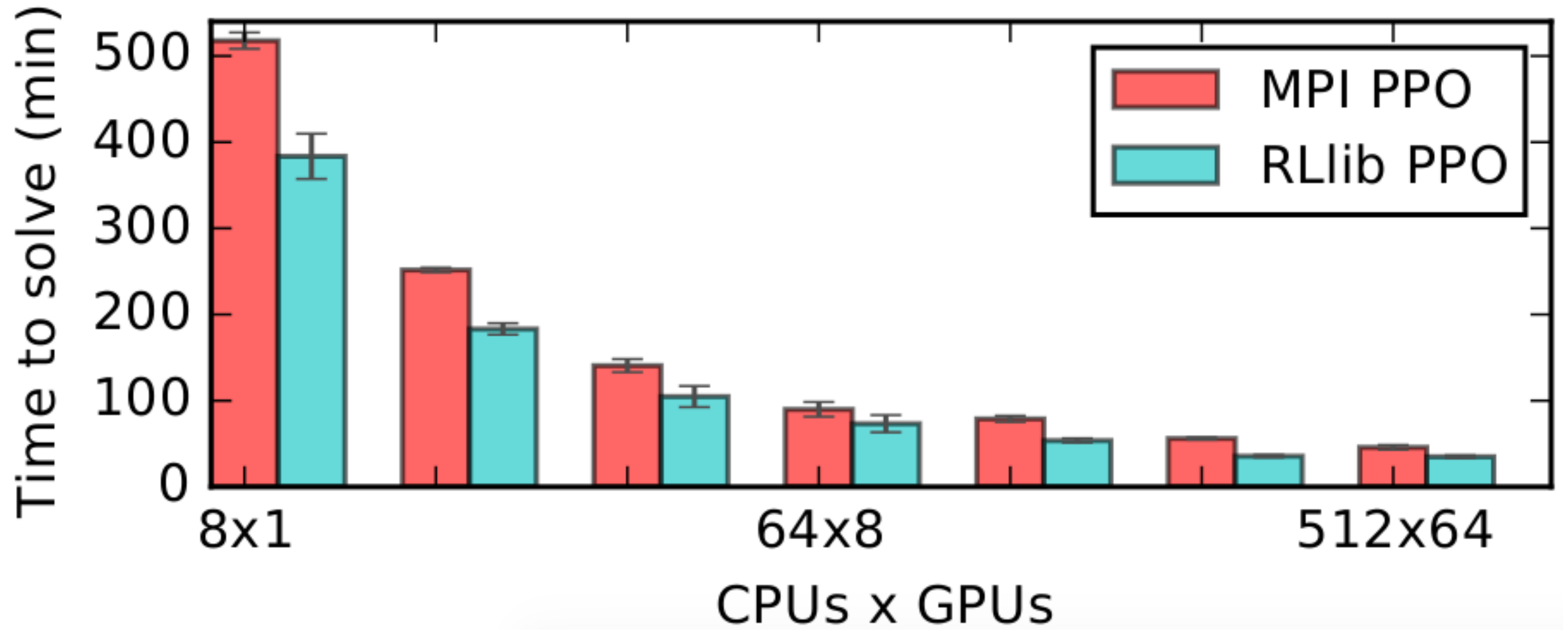


Multi-Agent

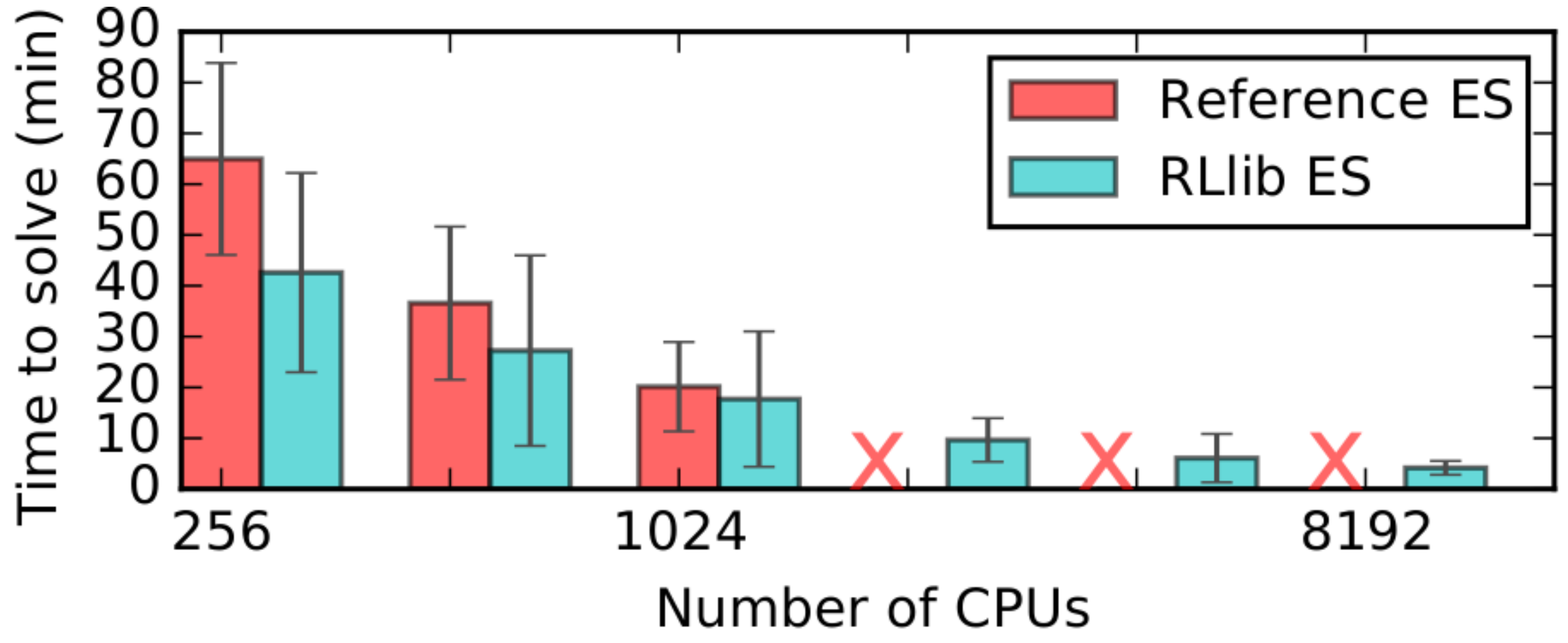


Unified Framework for Scalable RL

Distributed PPO



Evolution Strategies



Ape-X Distributed DQN, DDPG