Ray RLlib

A scalable and unified library for reinforcement learning

https://rllib.io

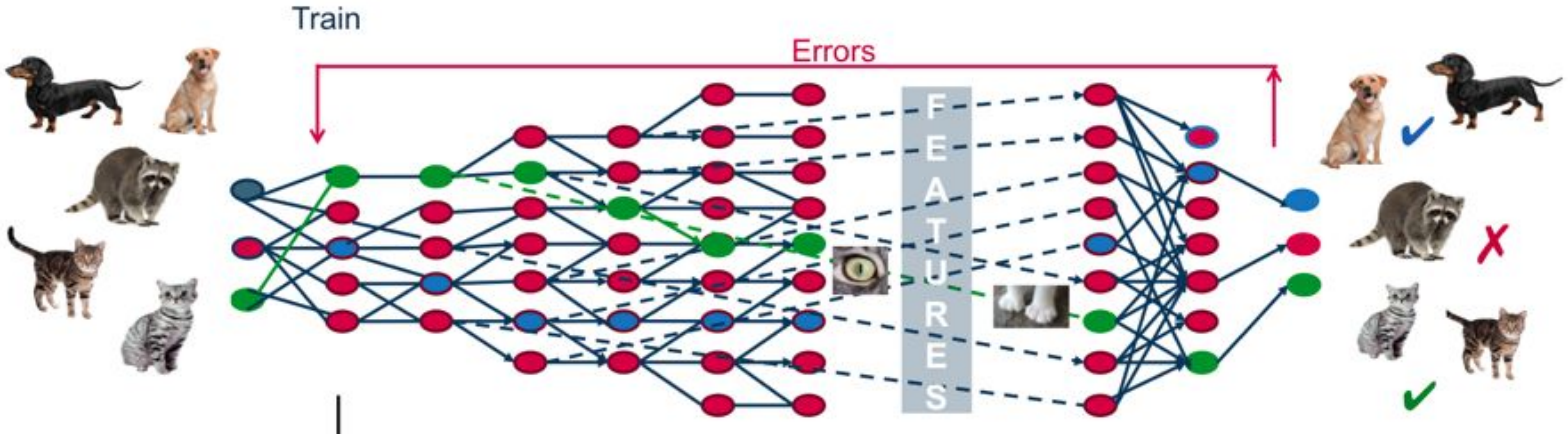
Eric Liang

riselab
UC Berkeley

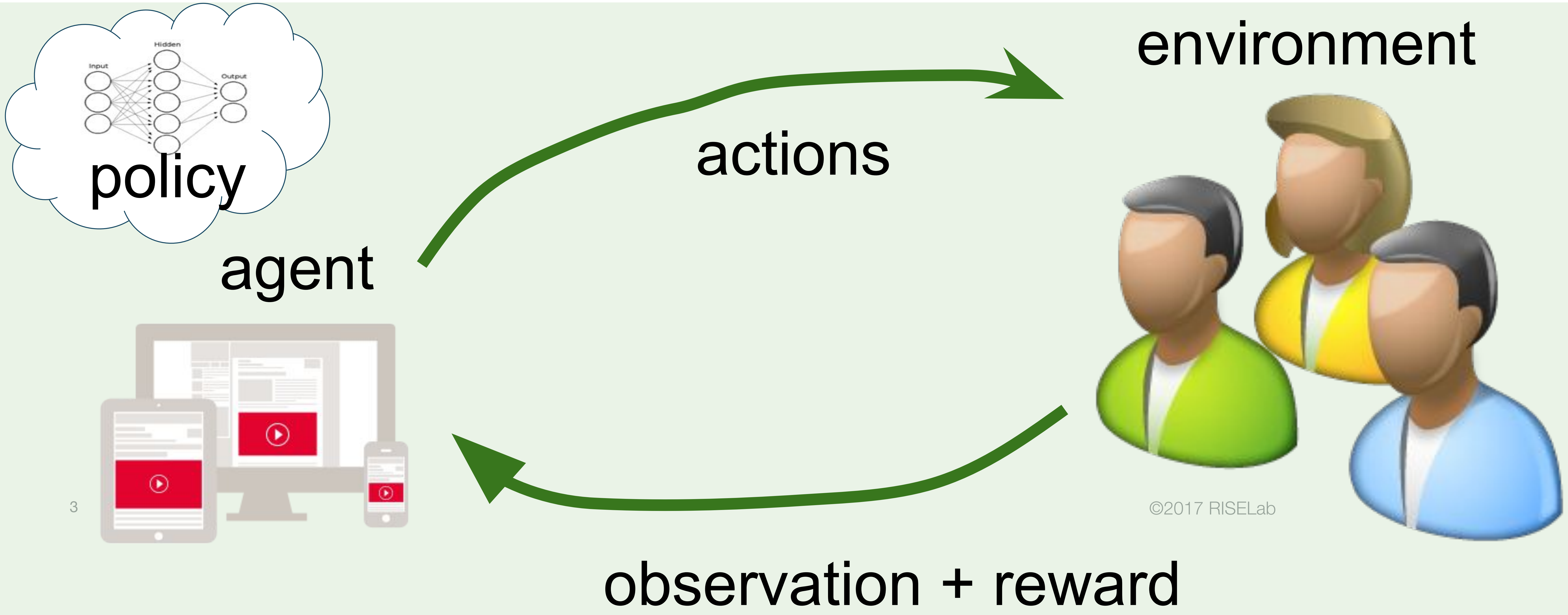
Overview

- RLlib the open source project
- System challenges building a scalable RL library ("abstractions for RL")

Background: What is reinforcement learning?



Supervised Learning



Reinforcement Learning

Growing number of RL applications

Robotics

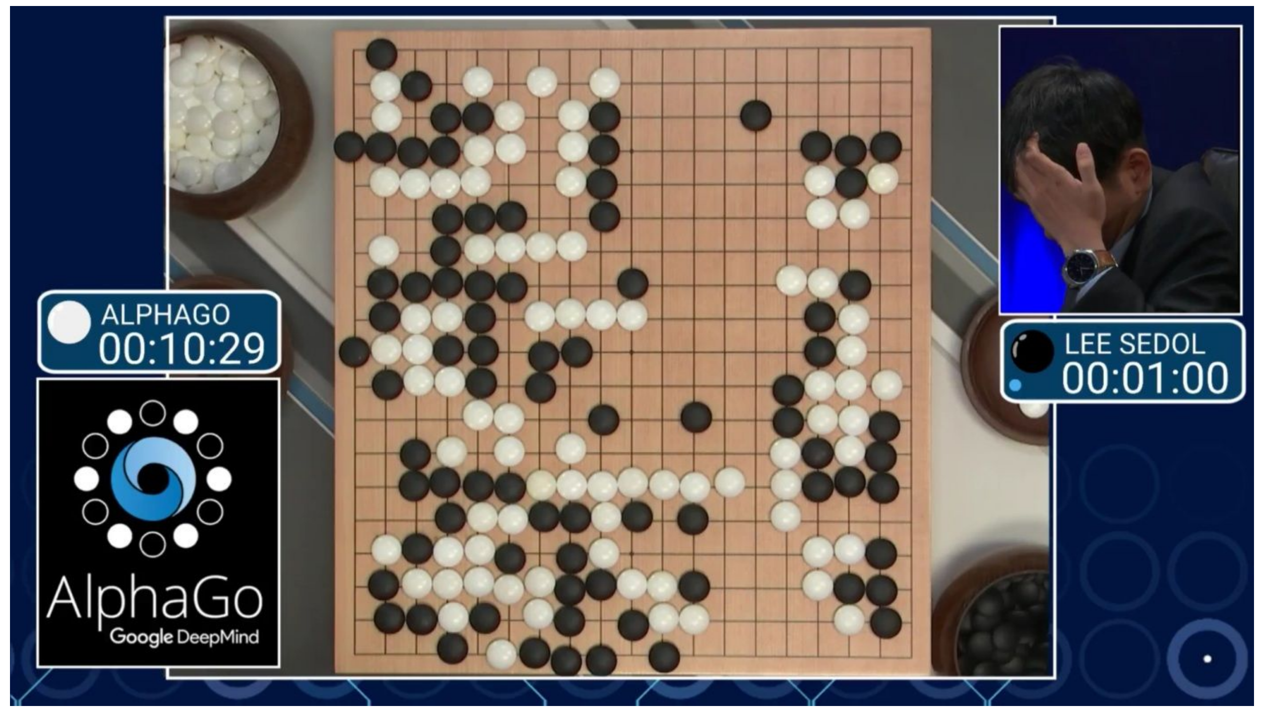
Industrial Control

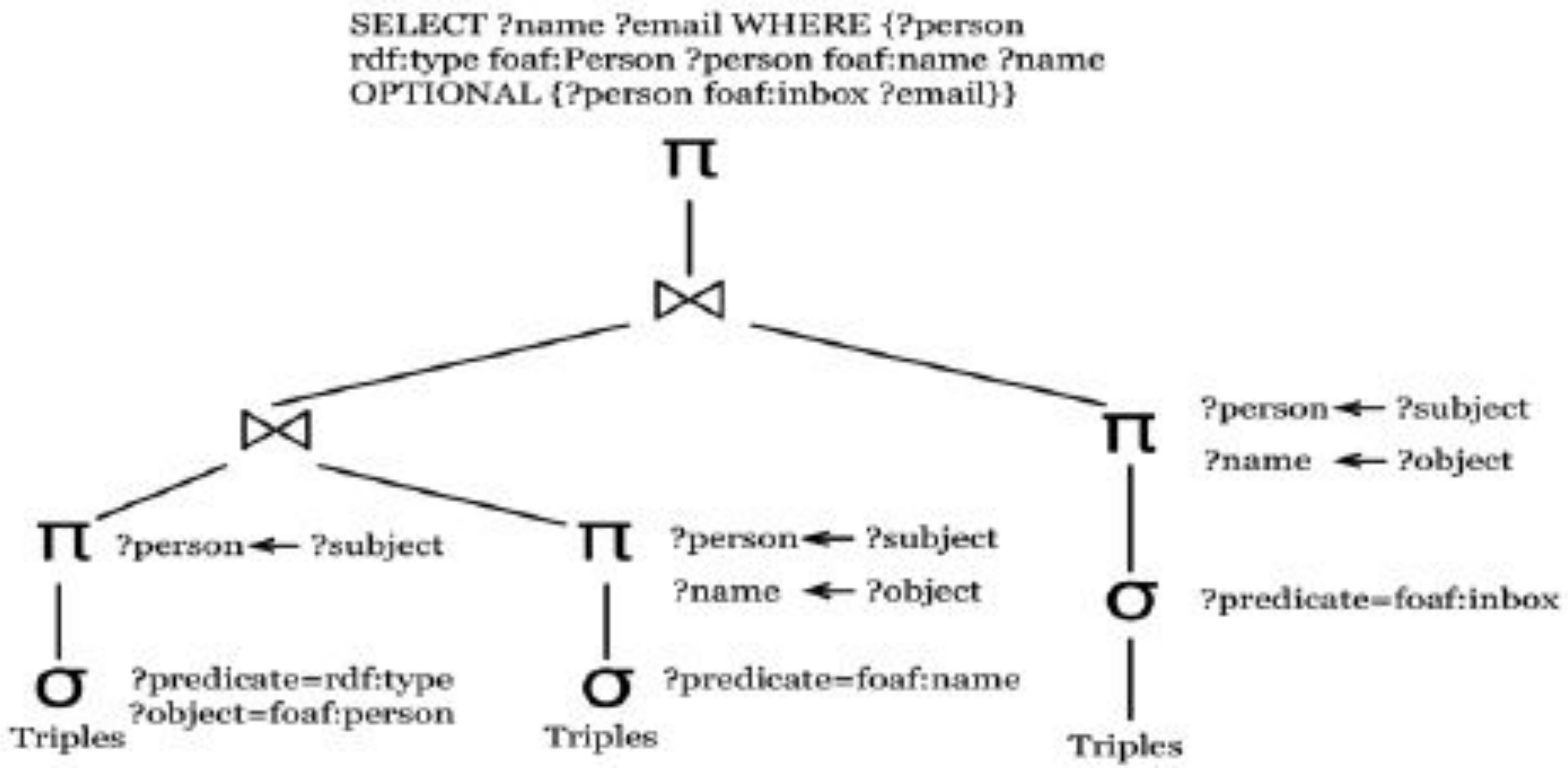
Advertising

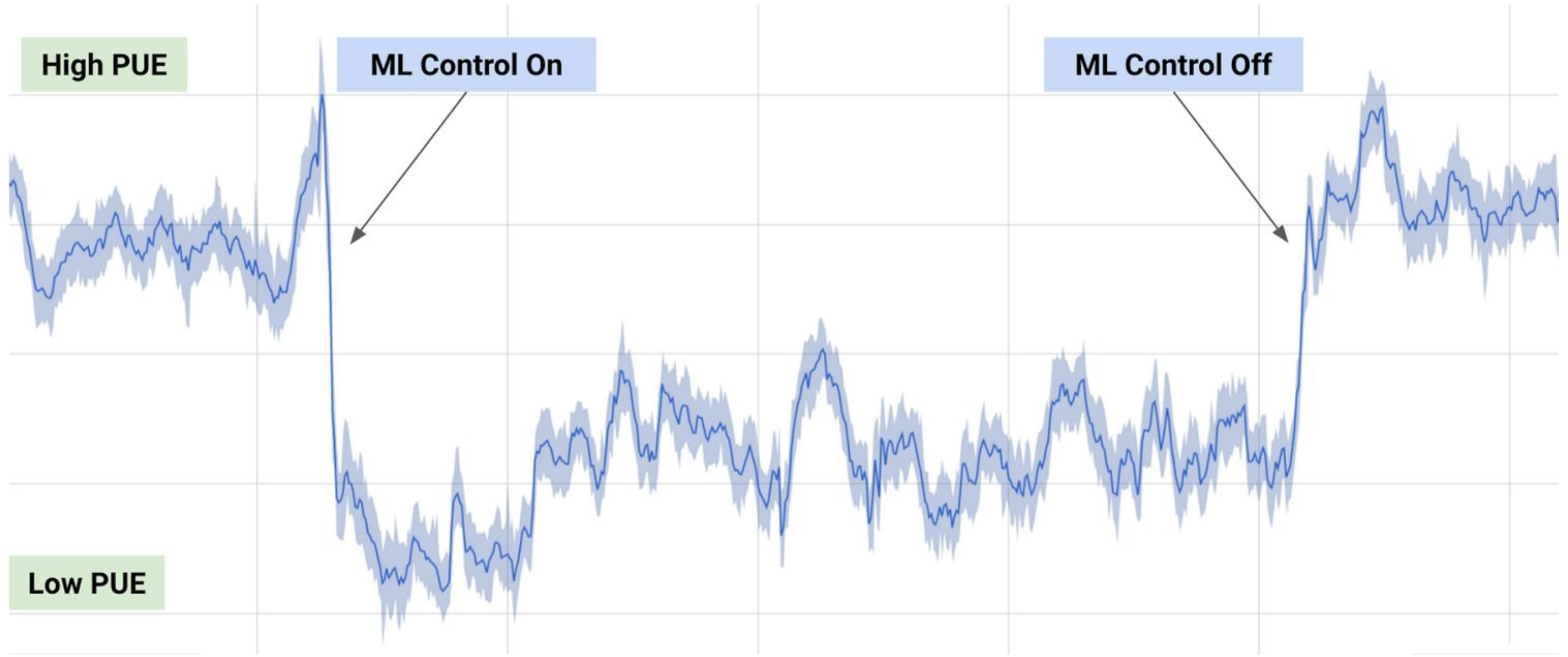
System Optimization

Finance

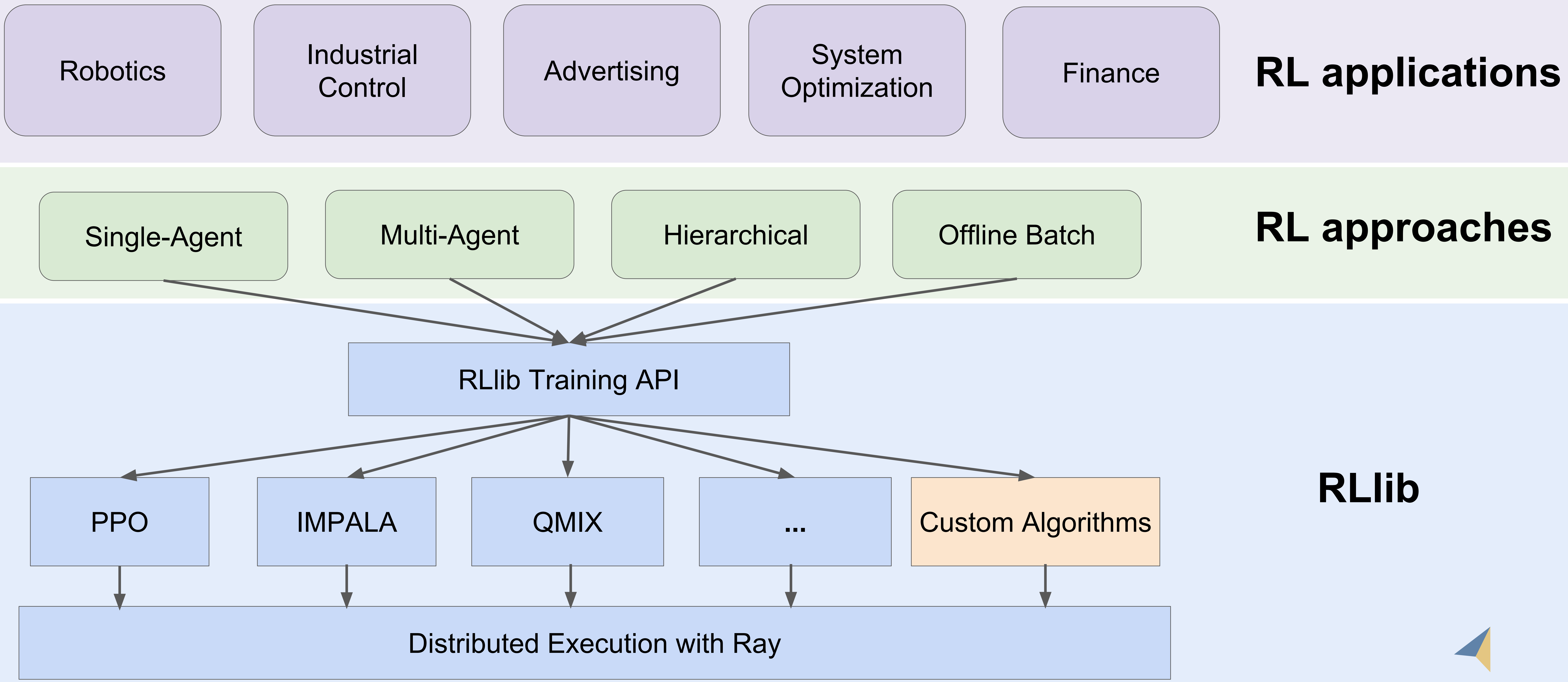
RL applications







A scalable, unified library for reinforcement learning

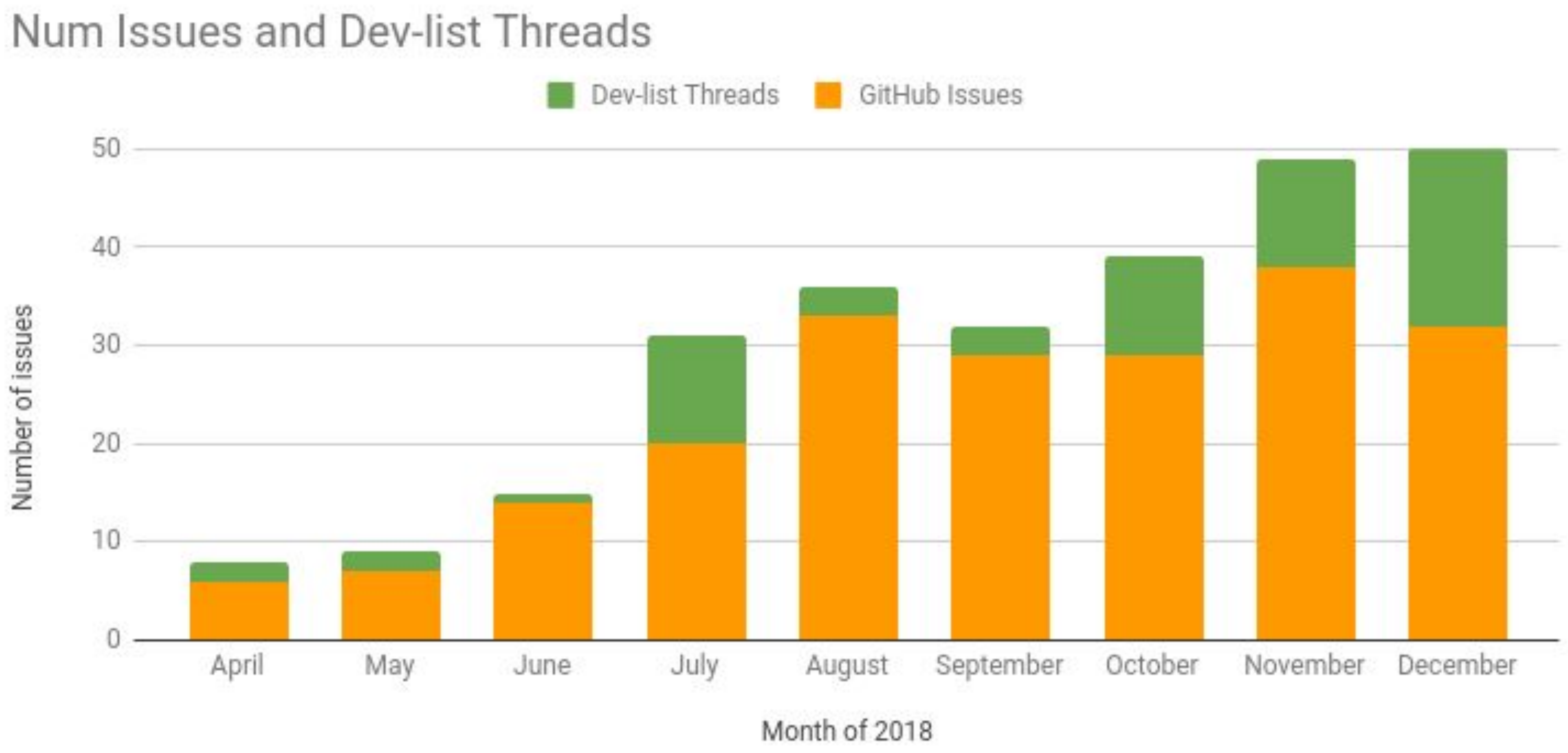


Performance

IMPALA and A2C vs A3C after 1 hour of training:

env	RLlib IMPALA 32-workers	RLlib A2C 5-workers	Mnih et al A3C 16-workers
BeamRider	3181	874	~1000
Breakout	538	268	~10
QBert	10850	1212	~500
SpaceInvaders	843	518	~300

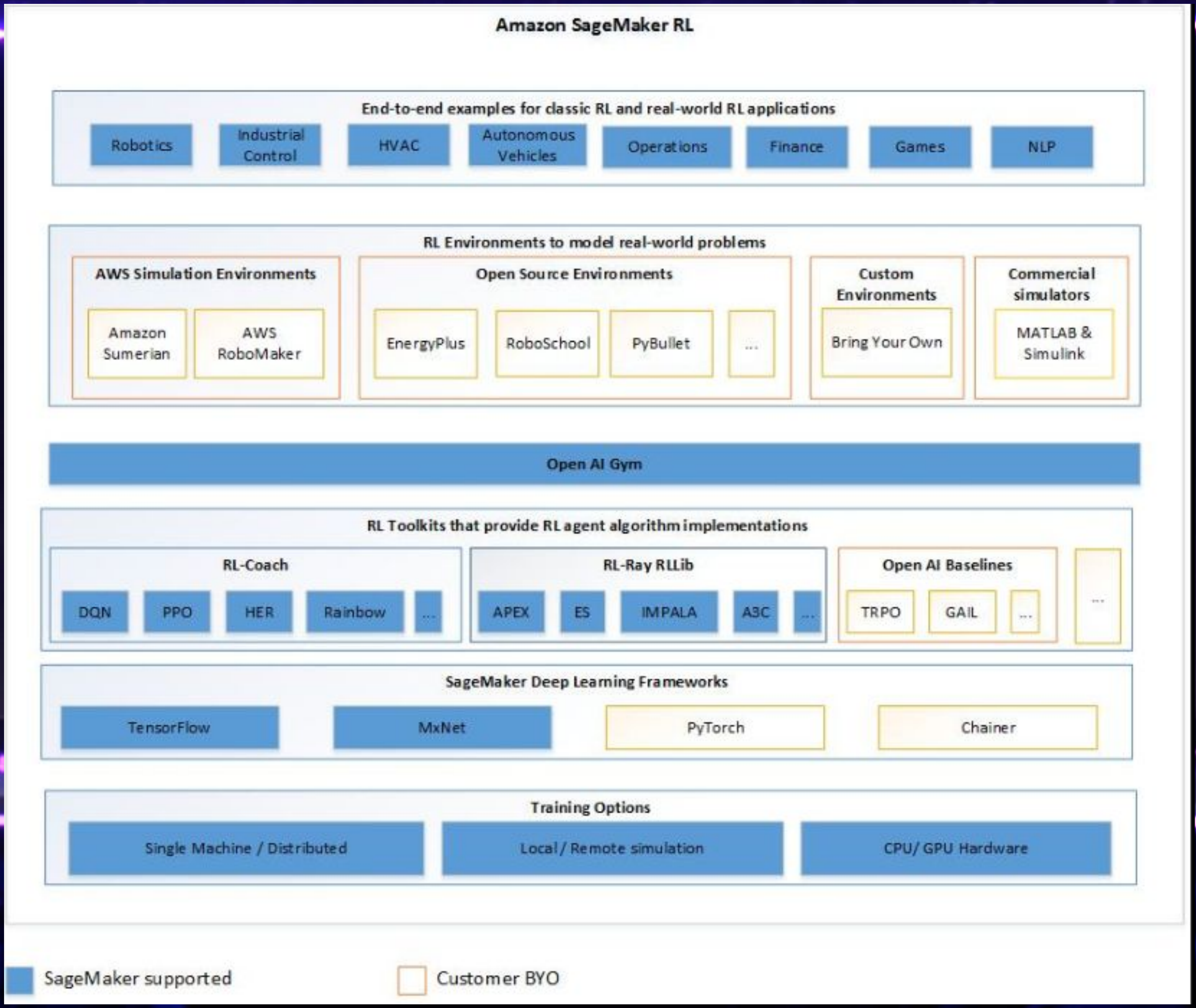
User growth in 2018



Filtering GitHub and ray-dev@ issues for "rllib":

- user engagement is increasing
- couple dozen companies and research labs using RLlib!


Amazon
SageMaker RL
Reinforcement learning for every
developer and 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

Project status

- **Goal: be the best library for RL applications and RL applications research**
- Continuing development (https://github.com/ray-project/ray)
  - new algorithms
  - cross-cutting features (env modeling, AutoRL)
  - better performance
- Documentation at https://rllib.io

Abstractions for Distributed Reinforcement Learning

RL research scales with compute

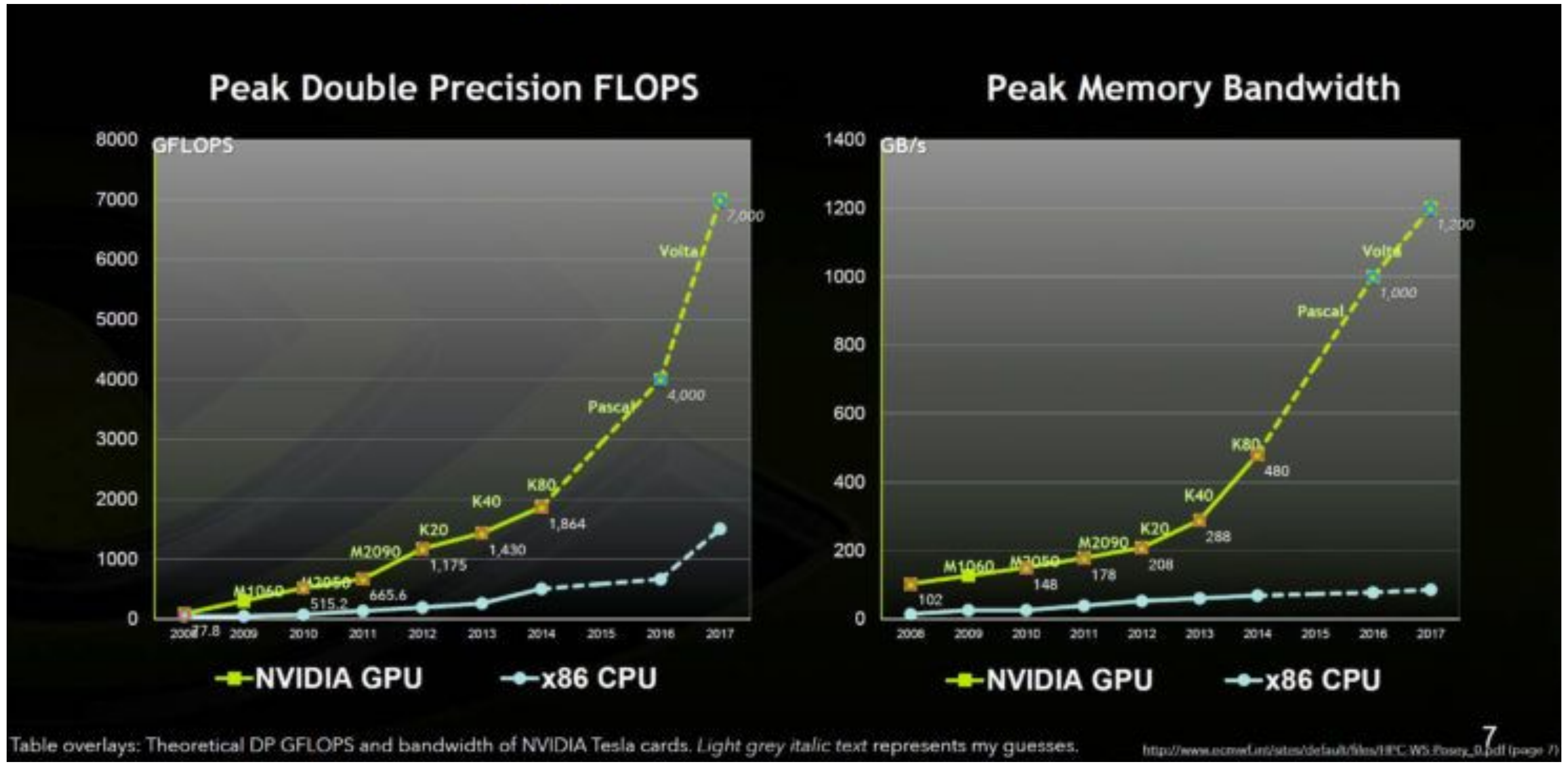


Fig. courtesy NVidia Inc.

CPU GPU TPU

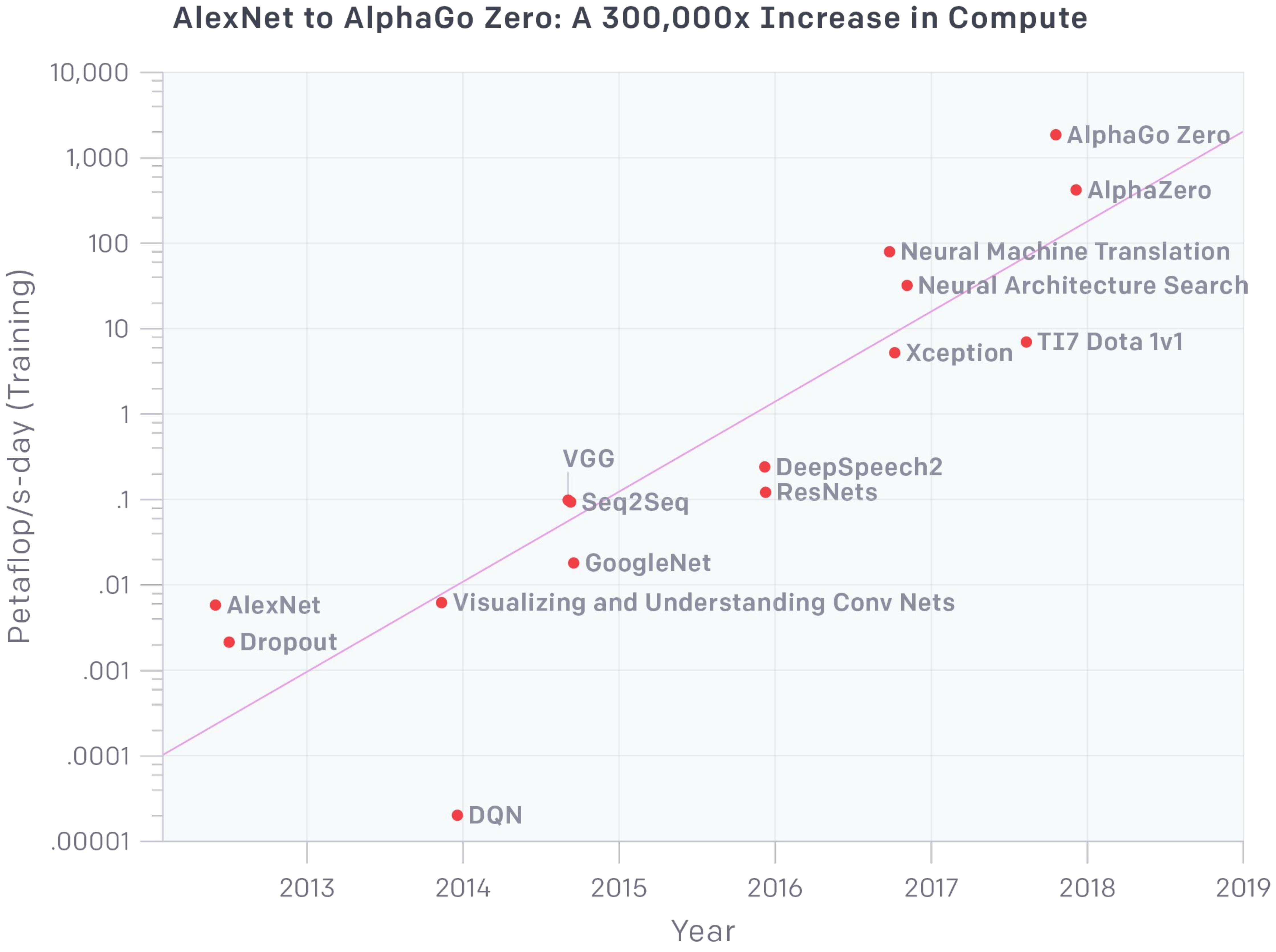
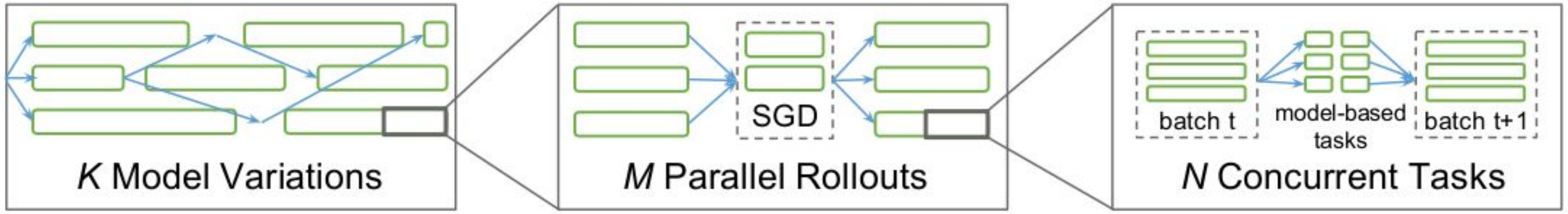


Fig. courtesy OpenAI

How do we leverage this hardware?

(a) Supervised Learning



(b) Reinforcement Learning

scalable abstractions for RL?

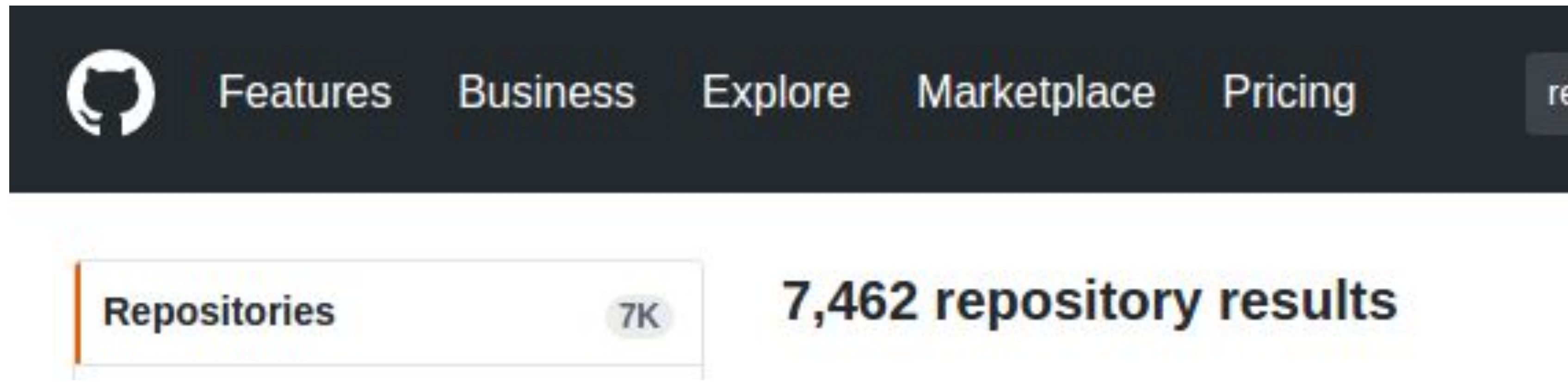
Example

```
rllib train --run=PPO --env=Pong-v0 --config='{"num_workers": 1}'
```

```
rllib train --run=PPO --env=Pong-v0
          --config='{"num_workers": 4, "num_gpus": 1}'
```

```
rllib train --run=PPO --env=Pong-v0
          --config='{"num_workers": 256, "num_gpus": 8}'
          --redis-address=localhost:6379
```

Systems for RL today



- Many implementations (7000+ repos on GitHub!)
  – how general are they (and do they scale)?

  PPO: multiprocessing, MPI

  AlphaZero: custom systems

  Evolution Strategies: Redis

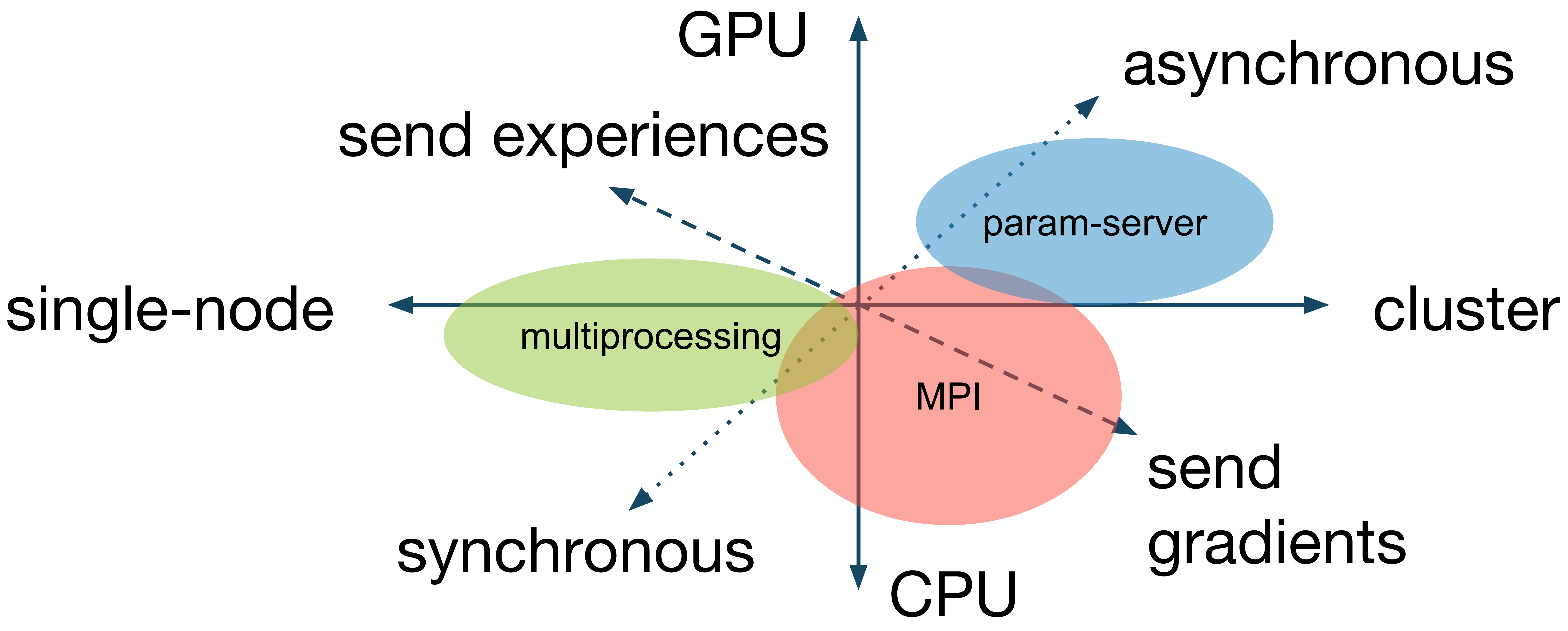
  IMPALA: Distributed TensorFlow

  A3C: shared memory, multiprocessing, TF

- Huge variety of algorithms and distributed systems used to implement, but little unification of different architectures

Challenges to unification

1. Wide range of physical execution strategies for one "algorithm"



Challenges to unification

2. Tight coupling with deep learning frameworks

Different parallelism paradigms:

– Distributed TensorFlow vs TensorFlow + MPI?

Challenges to unification

3. Large variety of algorithms with different structures

Algorithm Family	Policy Evaluation	Replay Buffer	Gradient-Based Optimizer	Other Distributed Components
DQNs	X	X	X	
Policy Gradient	X		X	
Off-policy PG	X	X	X	
Model-Based/Hybrid	X		X	Model-Based Planning
Multi-Agent	X	X	X	
Evolutionary Methods	X			Derivative-Free Optimization
AlphaGo	X	X	X	MCTS, Derivative-Free Optimization

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 compare and reproduce algorithms

Structure of RL computations

Agent

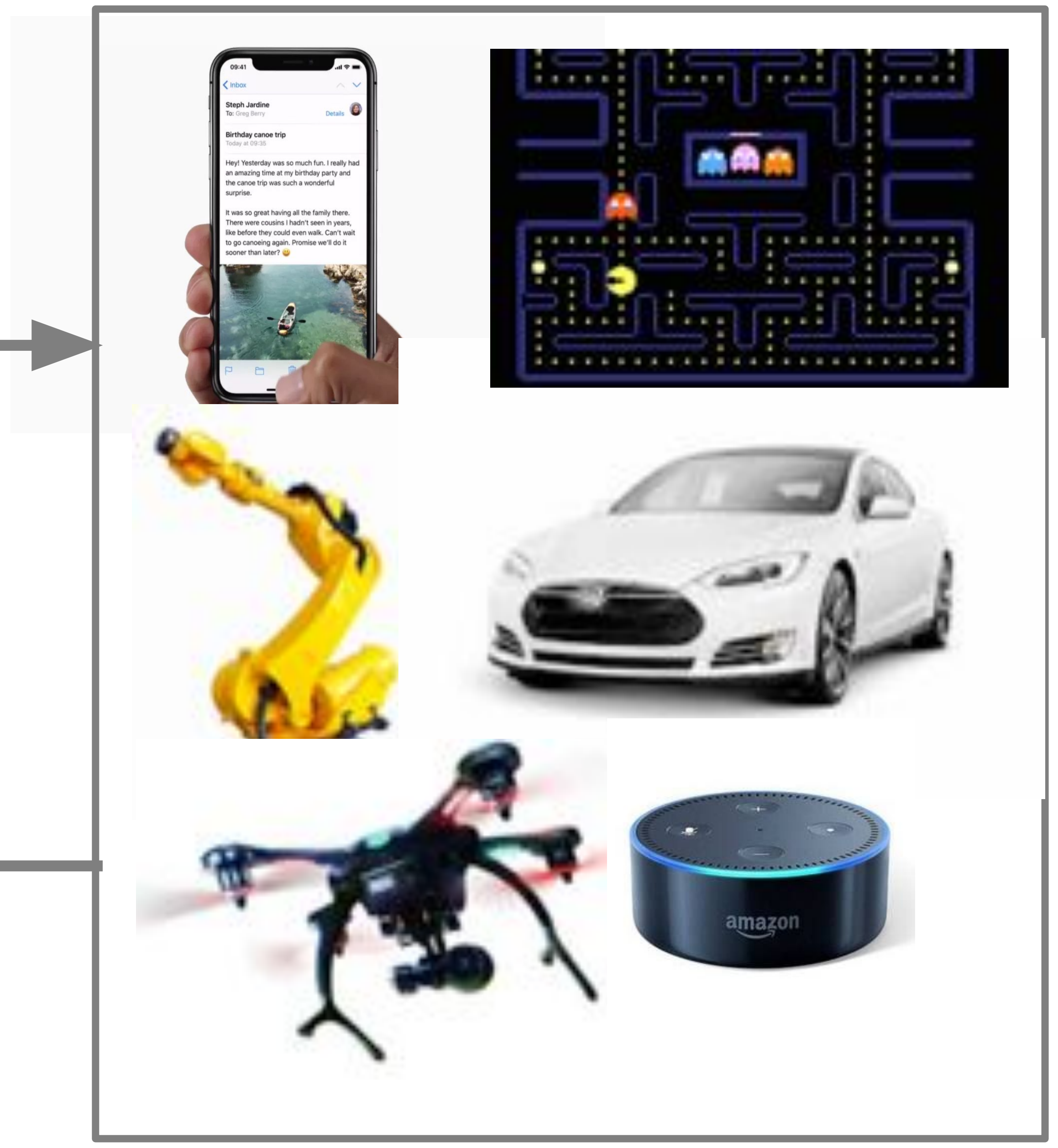
Environment

Policy:
state → action

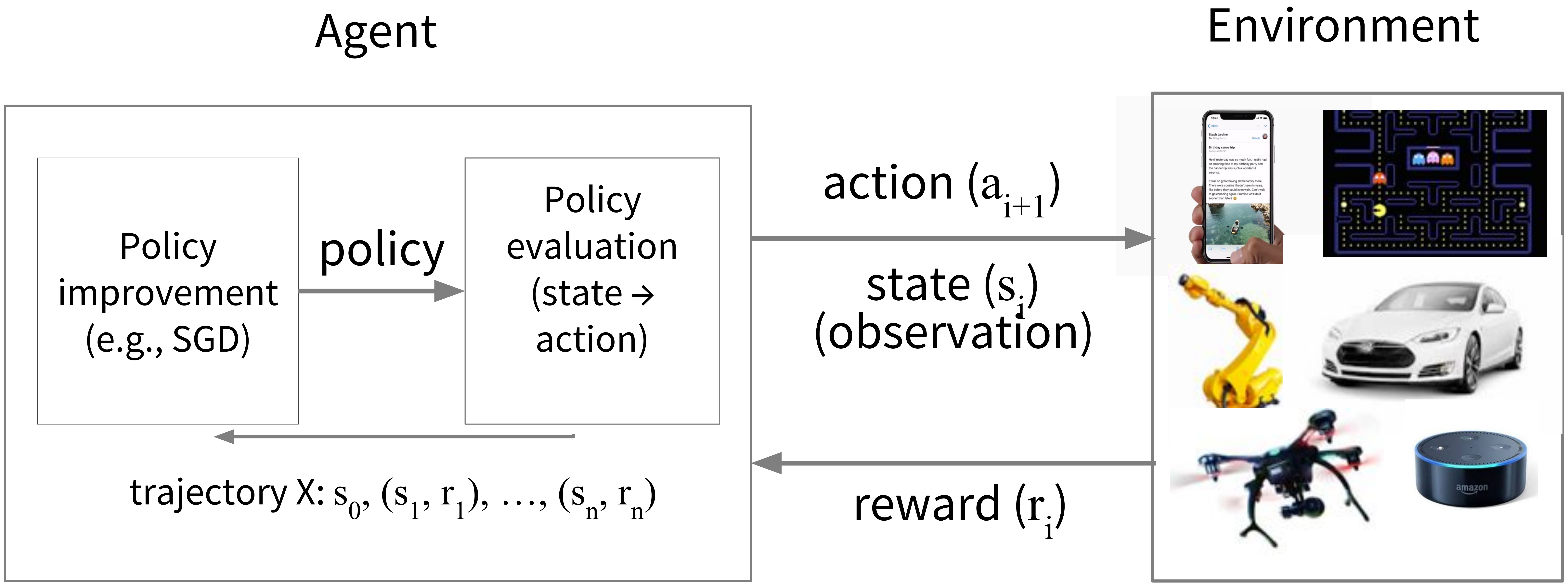
action (a_{i+1})

state (s_i)
(observation)

reward (r_i)



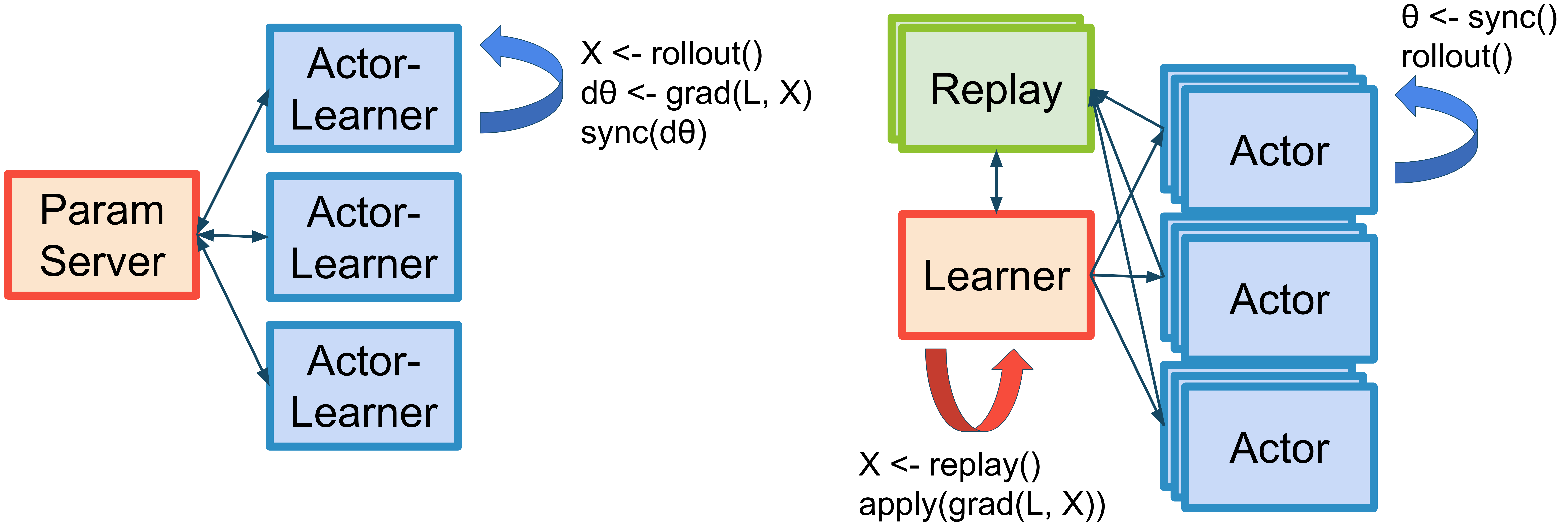
Structure of RL computations



Many RL loop decompositions

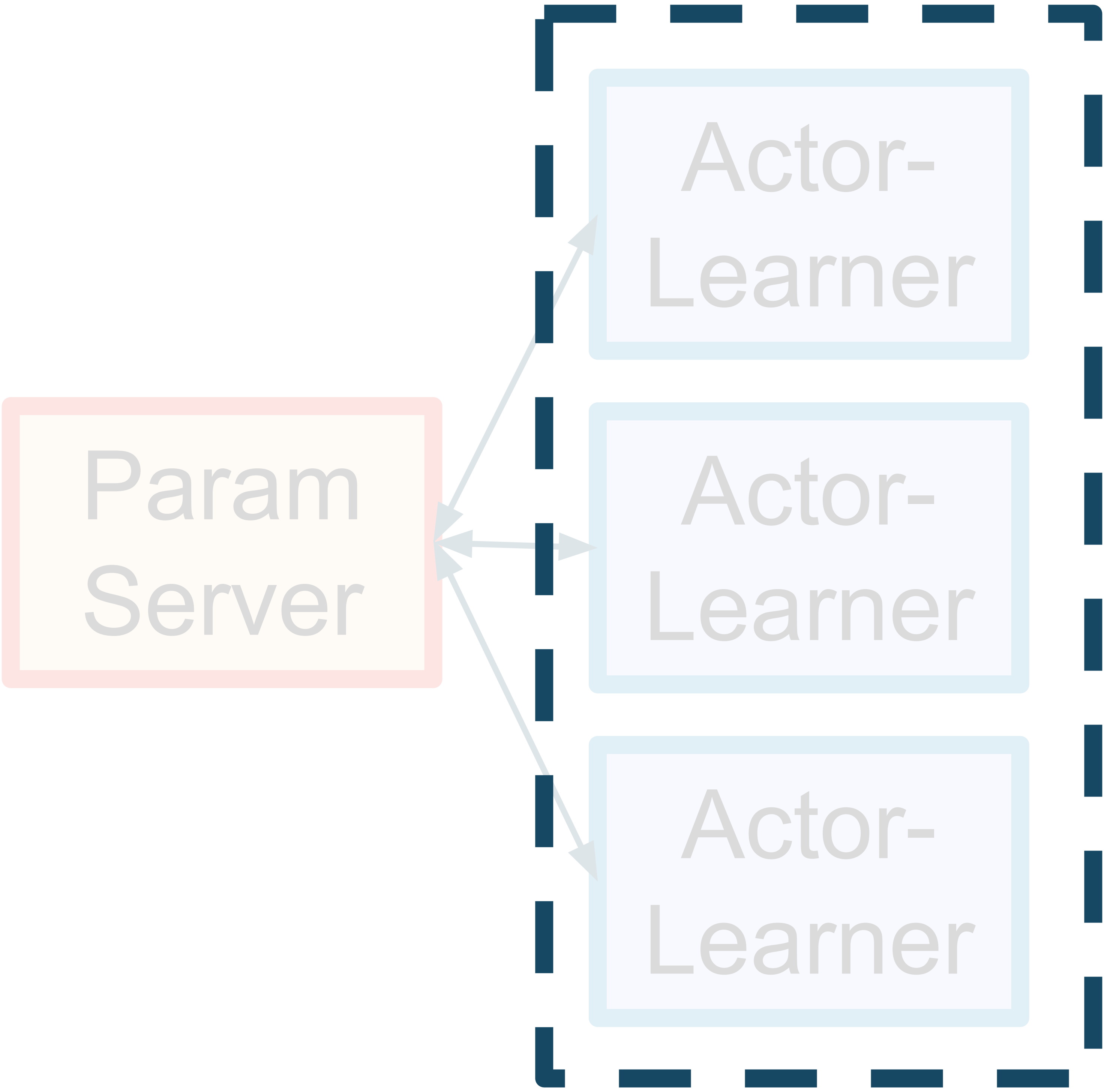
Async DQN (Mnih et al; 2016)

Ape-X DQN (Horgan et al; 2018)



Common components

Async DQN (Mnih et al; 2016)

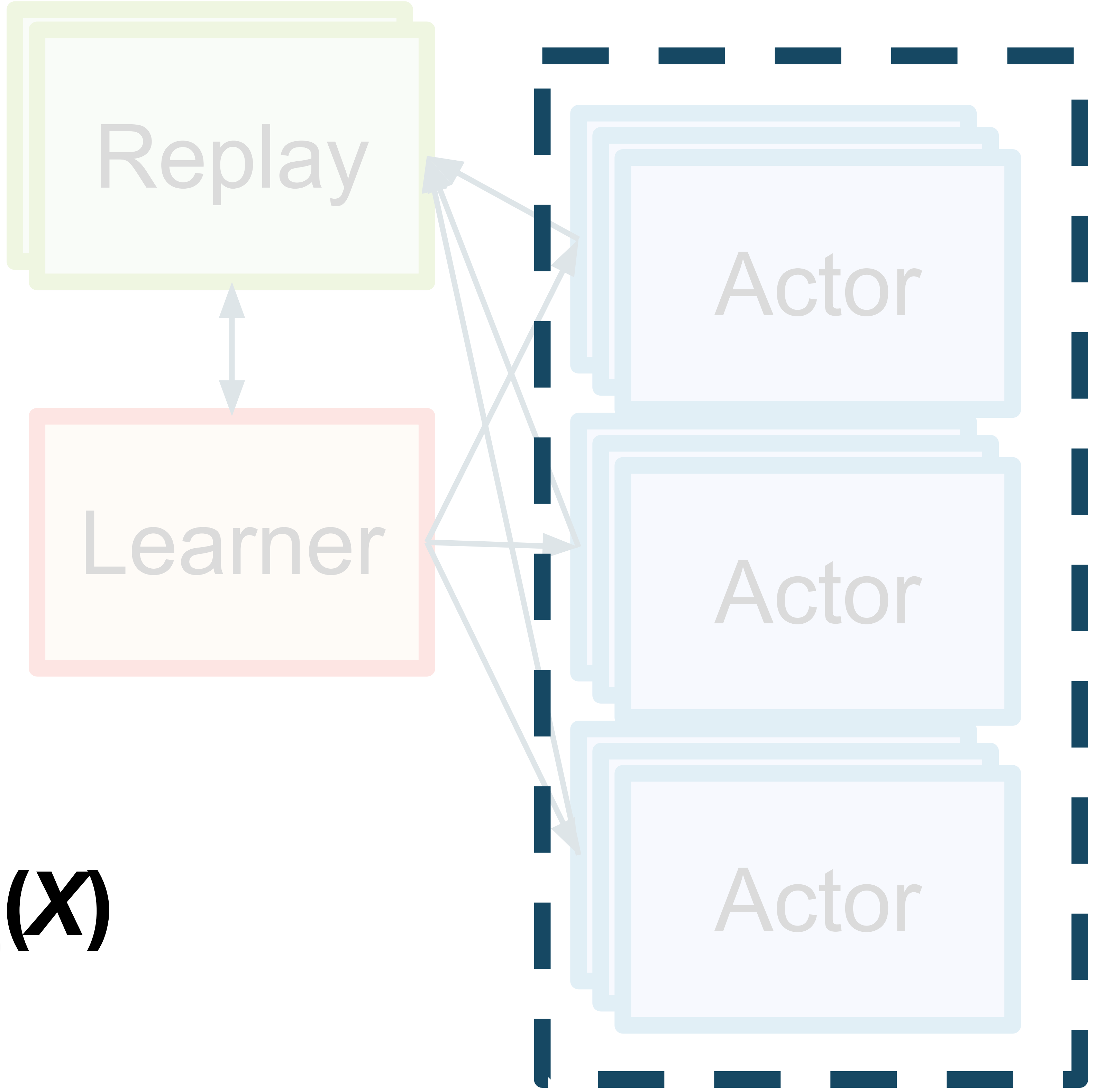


Policy $\pi_\theta(o_t)$

Trajectory postprocessor $\rho_\theta(X)$

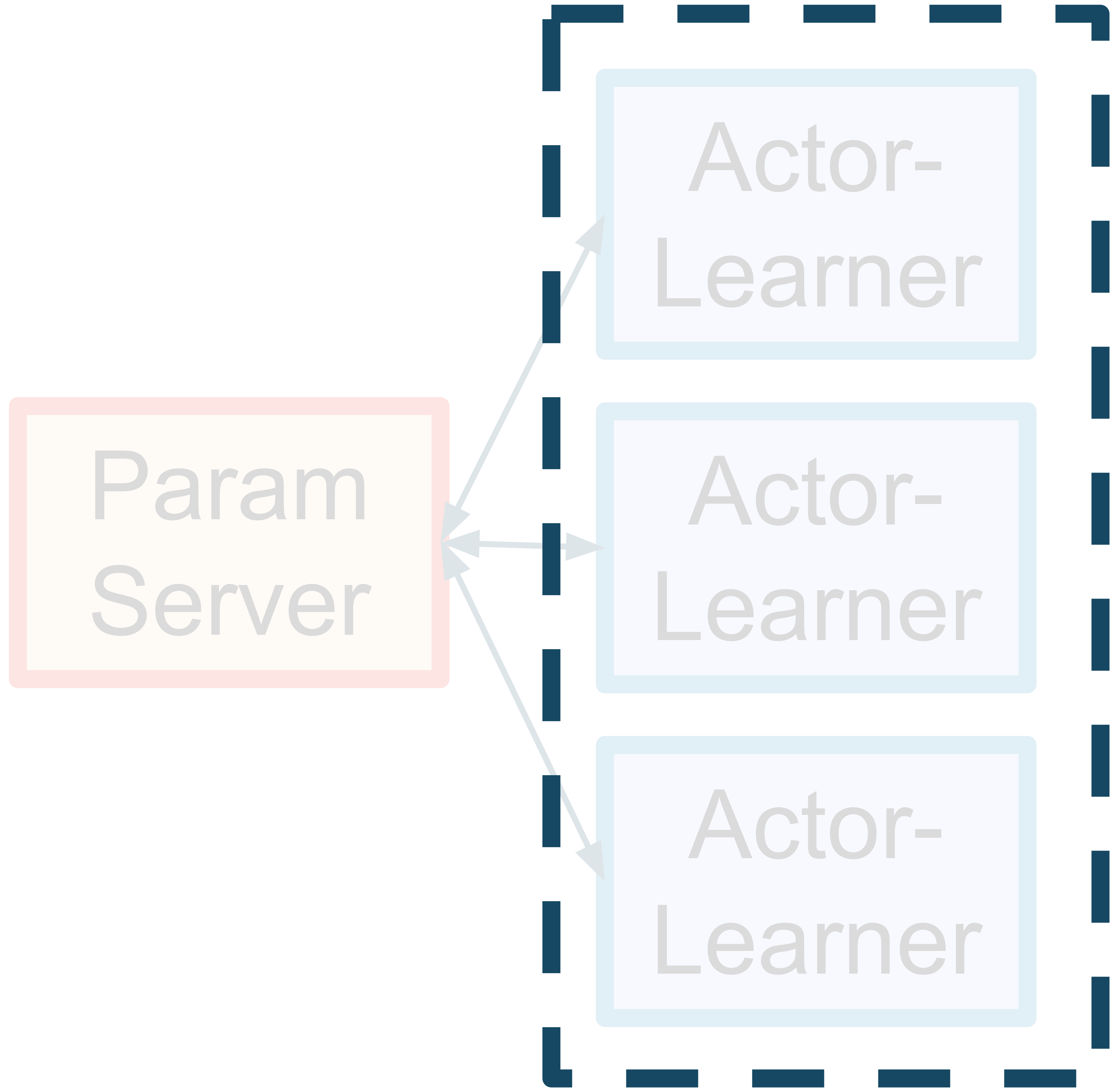
Loss $L(\theta,X)$

Ape-X DQN (Horgan et al; 2018)

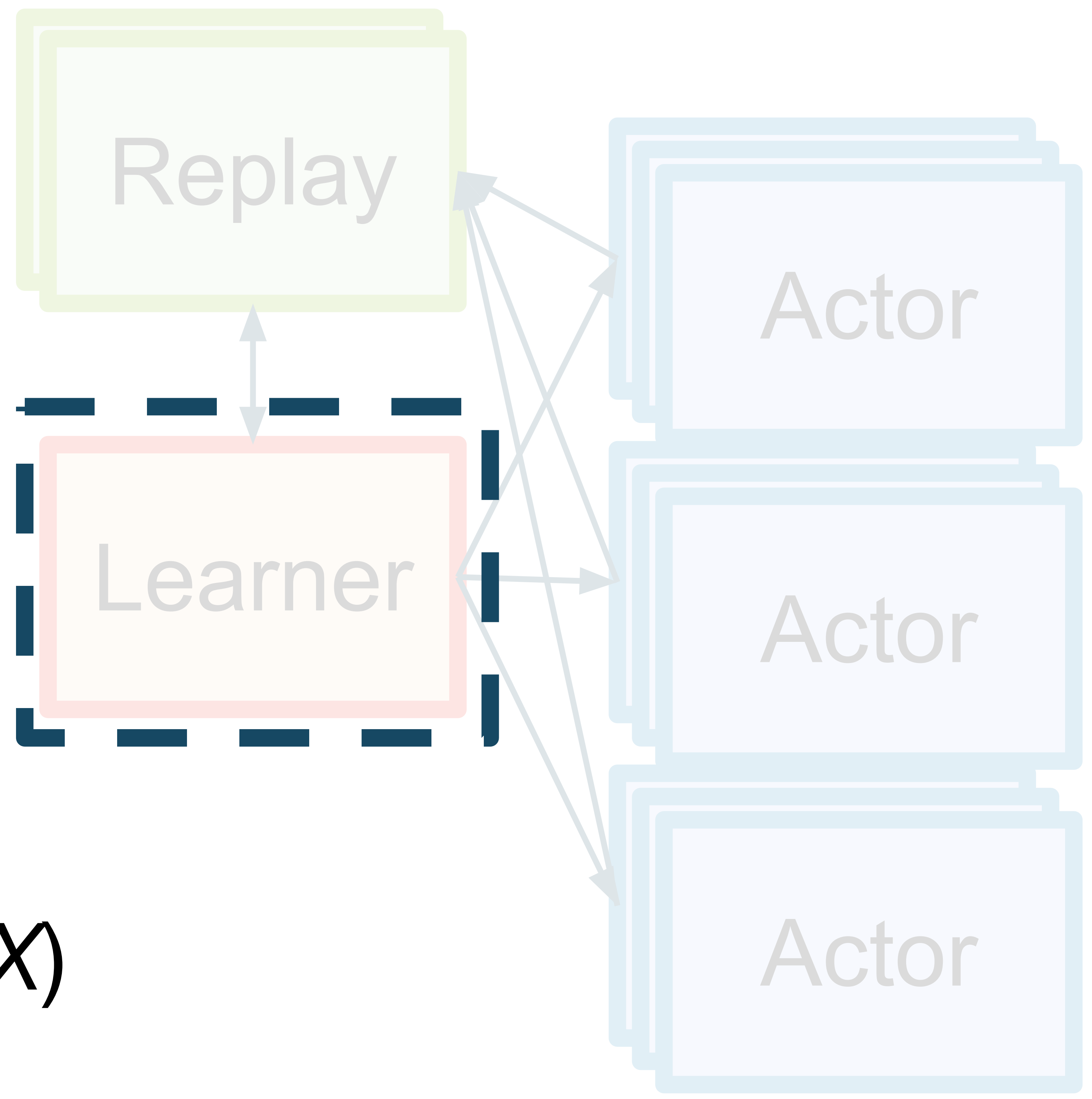


Common components

Async DQN (Mnih et al; 2016)



Ape-X DQN (Horgan et al; 2018)



Policy $\pi_\theta(o_t)$

Trajectory postprocessor $\rho_\theta(X)$

Loss $L(\theta,X)$

Structural differences

Async DQN (Mnih et al; 2016)

- Asynchronous optimization
- Replicated workers
- Single machine

Ape-X DQN (Horgan et al; 2018)

- Central learner
- Data queues between components
- Large replay buffers
- Scales to clusters

+ Population-Based Training (Jaderberg et al; 2017)

- Nested parallel computations
- Control decisions based on intermediate results

...and this is just one family!

→ No existing system can effectively meet all the varied demands of RL workloads.

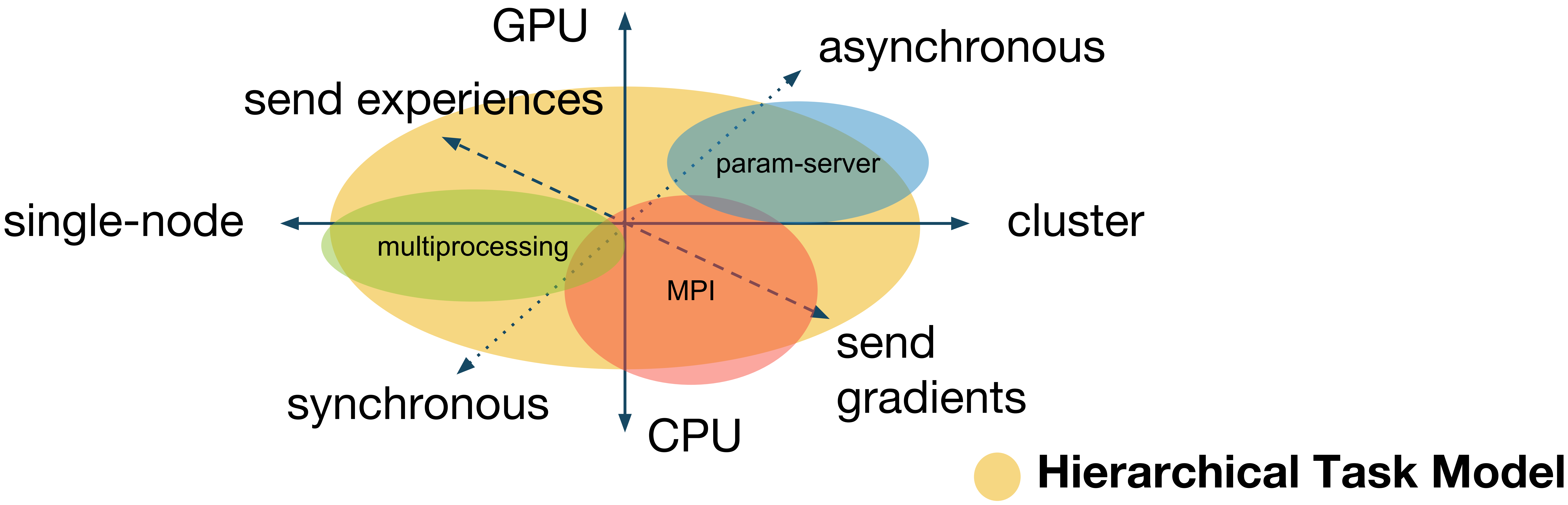
Requirements for a new system

Goal: Capture a broad range of RL workloads with high performance and substantial code reuse

1. Support stateful computations
   - e.g., simulators, neural nets, replay buffers
   - big data frameworks, e.g., Spark, are typically stateless
2. Support asynchrony
   - difficult to express in MPI, esp. nested parallelism
3. Allow easy composition of (distributed) components

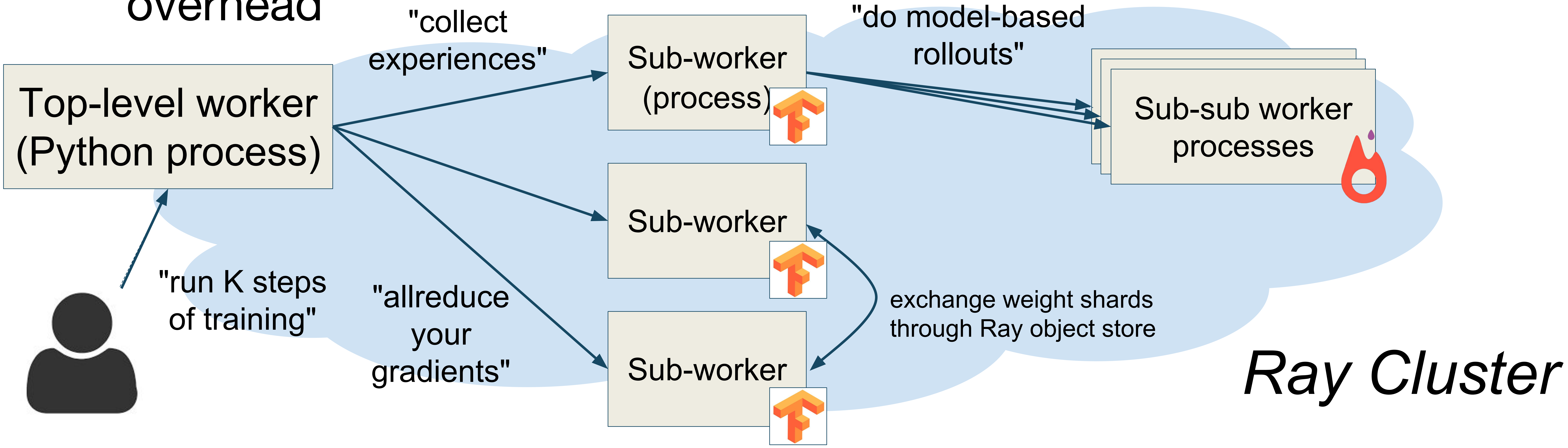
Ray System Substrate

- RLlib builds on Ray to provide higher-level RL abstractions
- Hierarchical parallel task model with stateful workers
  - flexible enough to capture a broad range of RL workloads (vs specialized sys.)



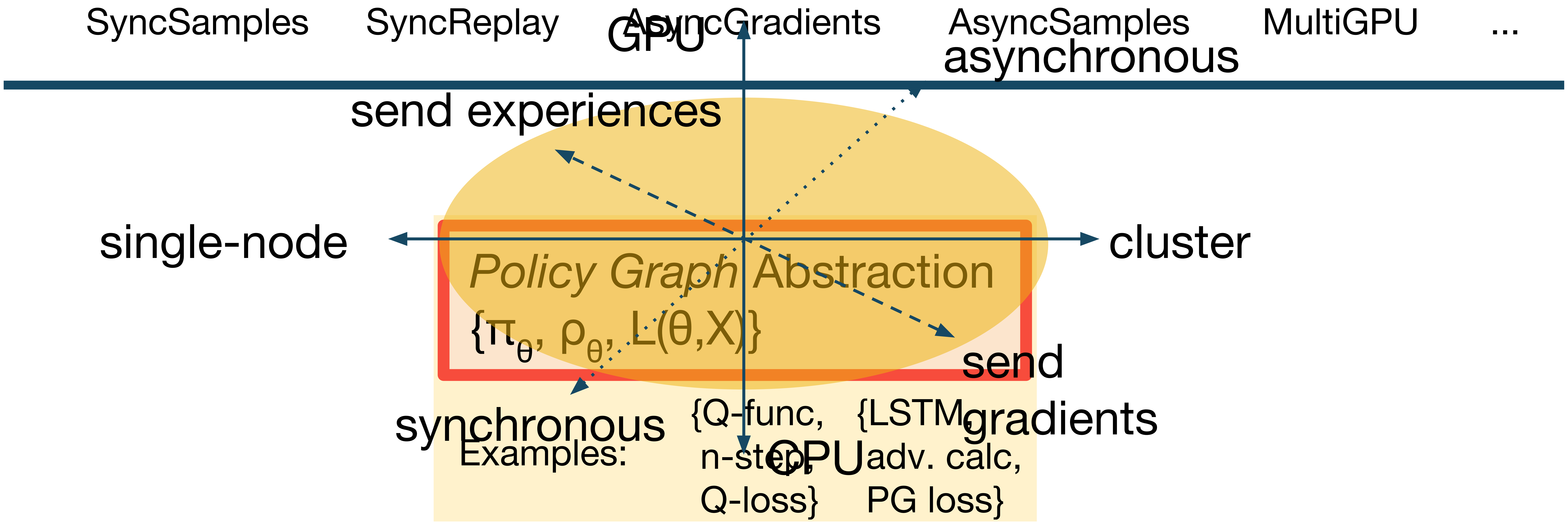
Hierarchical Parallel Task Model

1. Create Python class instances in the cluster (stateful workers)
2. Schedule short-running tasks onto workers
   – Challenge: High performance: 1e6+ tasks/s, ~200us task overhead



Unifying system enables RL Abstractions

Policy Optimizer Abstraction



Hierarchical Task Model

RLlib Abstractions in Action

Summary: RLlib addresses challenges in providing scalable abstractions for reinforcement learning.

RLlib is open source and available at http://rllib.io

Thanks!