TWO SIGMA

A 24x Speedup for Reinforcement Learning with RLlib + Ray

Raoul Khouri

5/2021

@buoy_the_samoyed

Raoul Khouri

Two Sigma

Financial Sciences Company

- Investment management
- Other financial data driven endeavours

  (Insurance, Real Estate, Private Equity, …)

Founded in 2001

- CEOs John Overdeck and David Siegel

~2000 employees (~1000 engineers and ~250 researchers)

Offices in NYC, London, Houston, Tokyo, Shanghai

Important Legal Information

This document is being distributed for informational and educational purposes only and is not an offer to sell or the solicitation of an offer to buy any securities or other instruments. The information contained herein is not intended to provide, and should not be relied upon for, investment advice. The views expressed herein are not necessarily the views of Two Sigma Investments, LP or any of its affiliates (collectively, "Two Sigma"). Such views reflect the assumptions of the author(s) of the document and are subject to change without notice. The document may employ data derived from third-party sources. No representation is made by Two Sigma as to the accuracy of such information and the use of such information in no way implies an endorsement of the source of such information or its validity.

The copyrights and/or trademarks in some of the images, logos or other material used herein may be owned by entities other than Two Sigma. If so, such copyrights and/or trademarks are most likely owned by the entity that created the material and are used purely for identification and comment as fair use under international copyright and/or trademark laws. Use of such image, copyright or trademark does not imply any association with such organization (or endorsement of such organization) by Two Sigma, nor vice versa

Disclaimer

What we will talk about today

Applied research currently includes Reinforcement Learning

Migrated from **Stable Baselines** to **RLlib + Ray**

Lessons learned

Case study of an experiment

The RL Pipeline

Overview of the RL pipeline

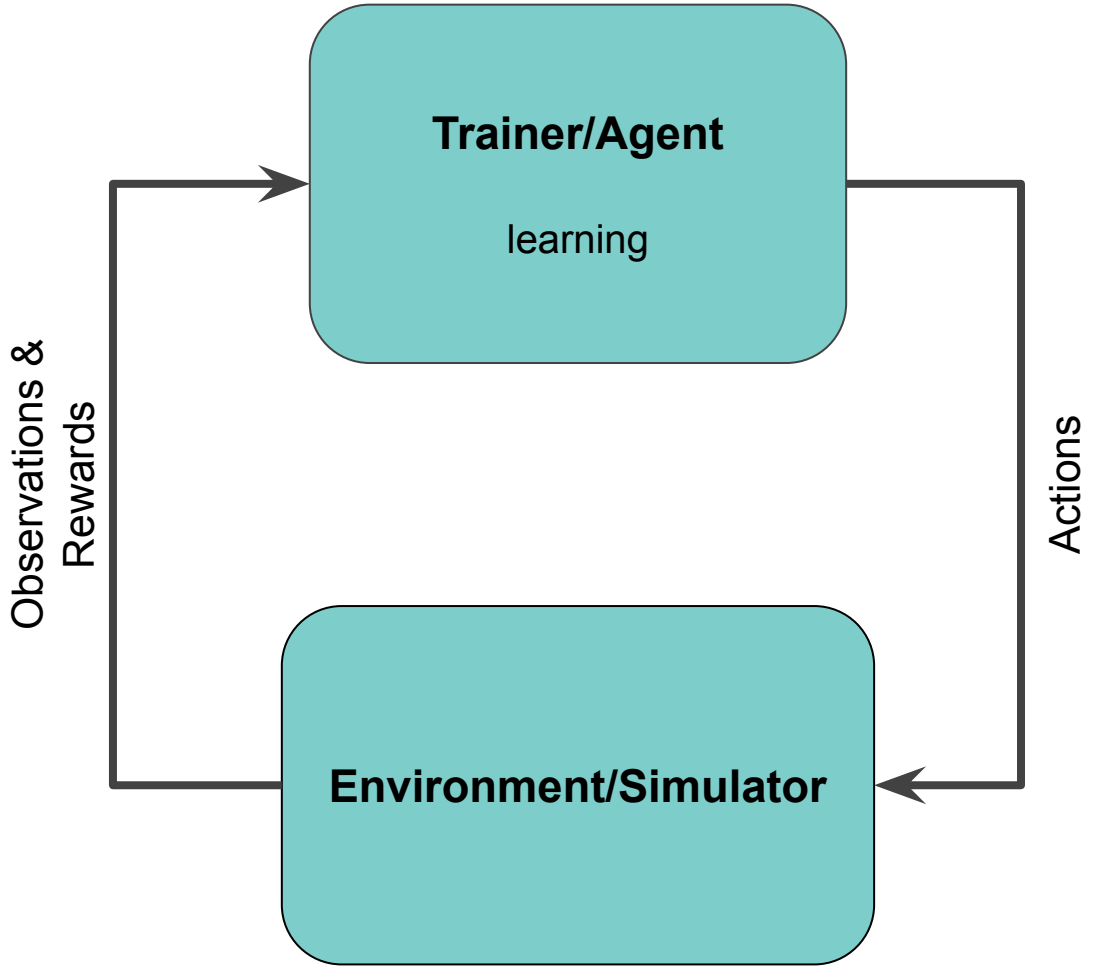
Trainer

- In charge of learning a policy

Environment

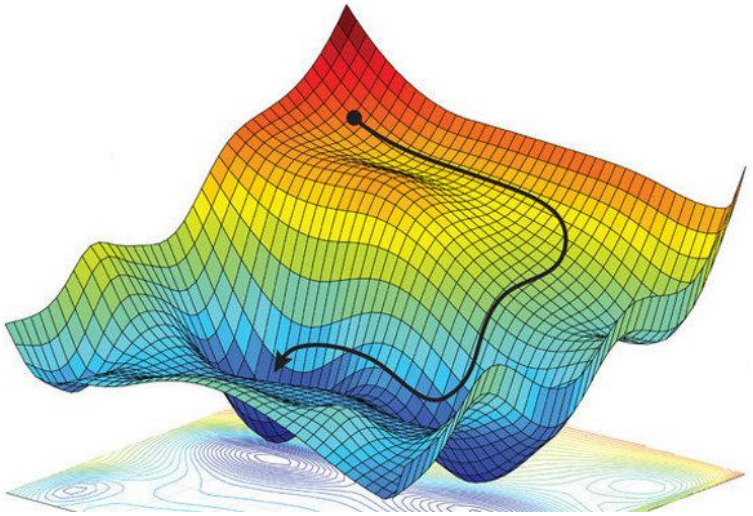
- State machine
- Actions go in
- Observations and rewards come out

Many interactions with the environment needed!

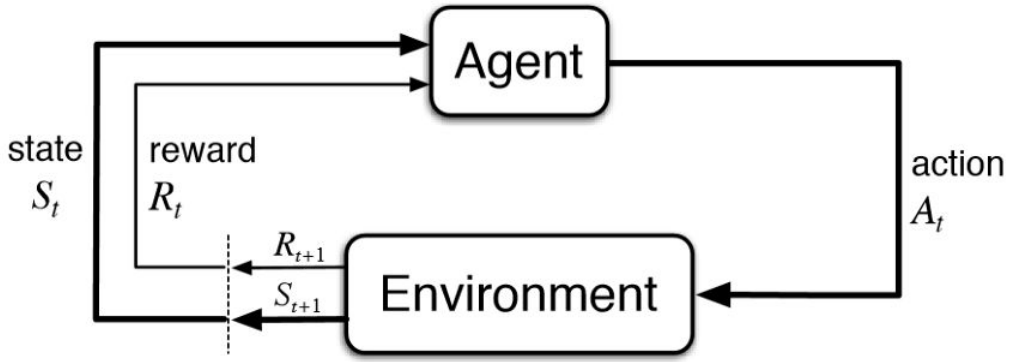


Bottlenecks in the RL pipeline

Learning

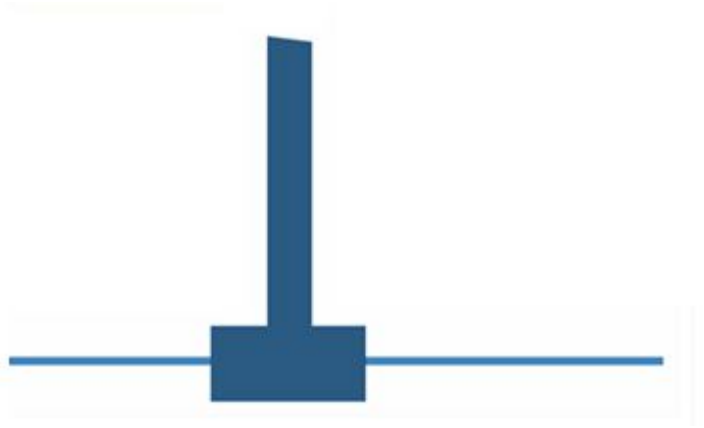


Sampling



[RLlib Scaling Guide]

Our Experiment



Financial data

- Lots of data
- Low signal to noise ratio

Experiment taking 7-10 hrs to complete

- **24 CPUs using Stable Baselines**

Learning taking <10% of total time

- **~90%** generating samples

What we tried

GPUs

- Yielded little to no speed up
  - Simulators don’t use GPUs
- GPUs are expensive
- “I got a machine with a larger GPU but I see no speed up.”

More CPUs

- Tried machines with >24 CPUs on **Stable Baselines**
- Little to no speed up

Off-Policy

- Lower solution quality vs on-policy
- Still need many samples due to noisy data

RLlib

Migrating to RLlib

Generic Gym API

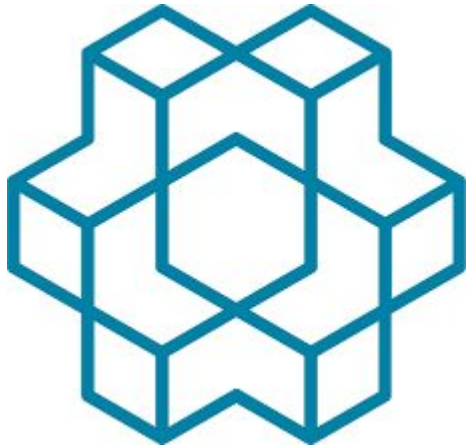
- Easy environment migration

RLlib has a superset of algorithms vs **Stable Baselines**

- Small changes in hyper parameters
- Same solution quality

Tune managed our experiments

~1 week to migrate a project



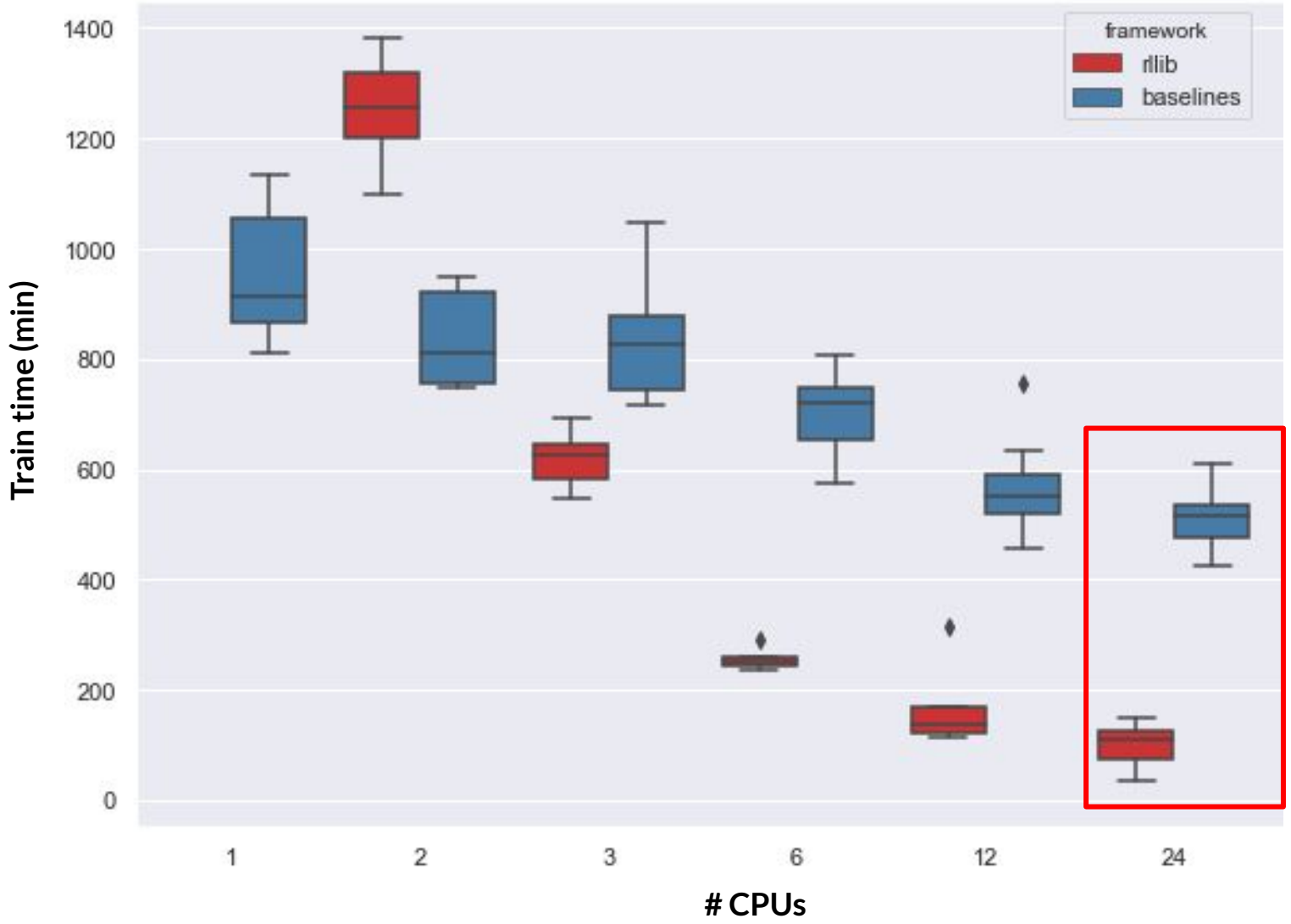
Great community support + **RLlib** examples in the GitHub Repository

RLlib vs Stable Baselines Experiment Time (20M samples)

Stable Baseline 7-10 Hrs

RLlib 1.5-2.5 Hrs

~4x speed up

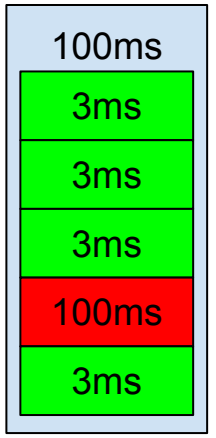


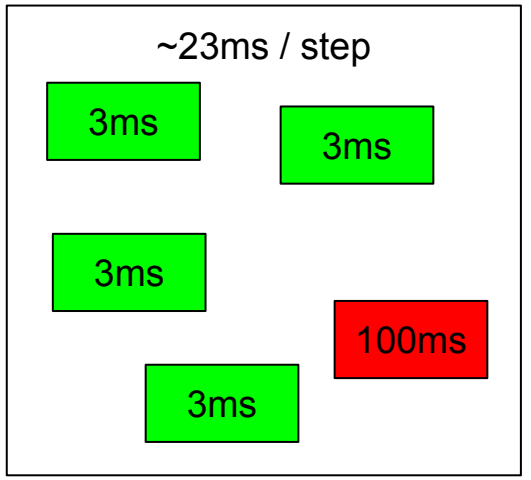
lower is better!

Why does RLlib parallelize better?

Types of RL parallelization

- Vectorized environments
  - Efficient inference
  - Everything must step together
    - If large variance in step time this can be very inefficient!
    - We call this the **“lock-step”** issue
  - Constrained to this by many RL libraries
- Batched rollouts
  - Slower inference
  - Sync at end of rollouts
    - requires much less waiting
    - avoids the **“lock-step”** issue





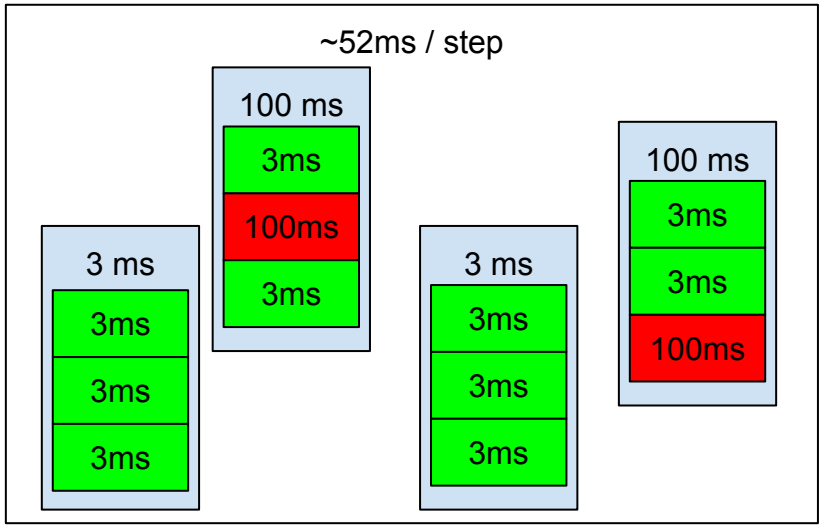
The “**lock-step**” issue is what caused our **Stable Baselines** experiments to not parallelize well.

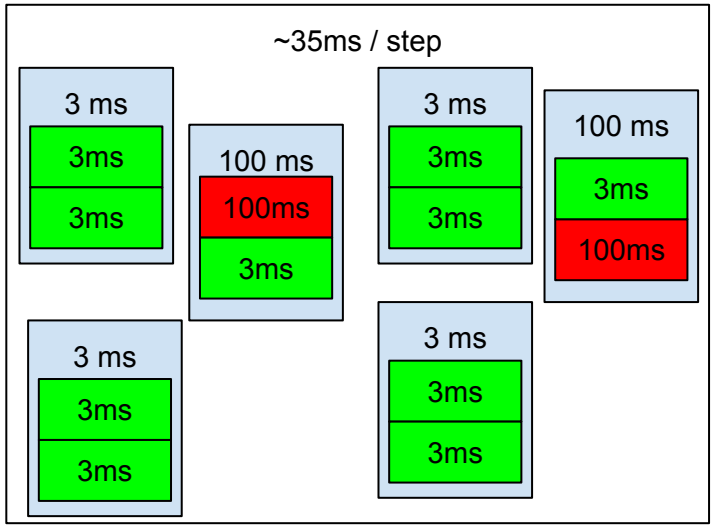
The RLlib Solution to Parallelization

Hybrid Solution

- rollout workers -> batched rollouts
- envs per worker -> vectors

Very customizable!

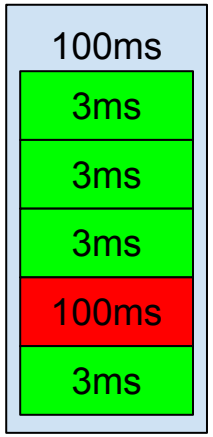




Diagnostic of our Experiment

	Overall Performance		Sampling Performance		
	Sampling	**Optimizing**	**Reset**	**Inference**	**Step**
Stable Baselines (Vectorized)	150s	**13s**	**100ms**	**0.8ms**	24ms
RLlib (Batched)	**20s**	**13s**	**100ms**	1ms	**3ms**

Wasting 6 hours here!



When the “Lock-Step” Issue Matters

Non-uniform time spent in steps or resets:

- **4-5x speed up**

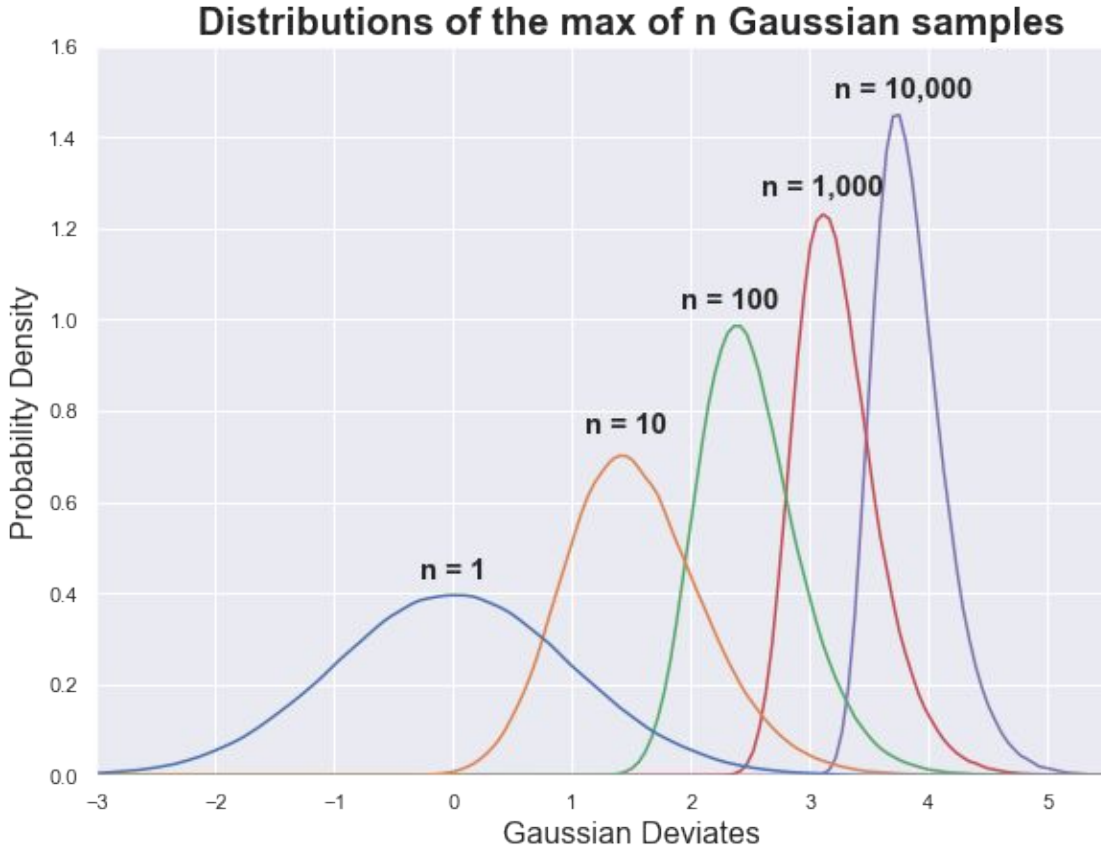
Large parallelization (if inference is marginal):

- **~ 10-50%**

Expected Speed up at 24 CPUs:

$$2\frac{std(step\_time)}{mean(step\_time)}$$

assuming gaussian step/reset variance



RLlib + Ray Clusters

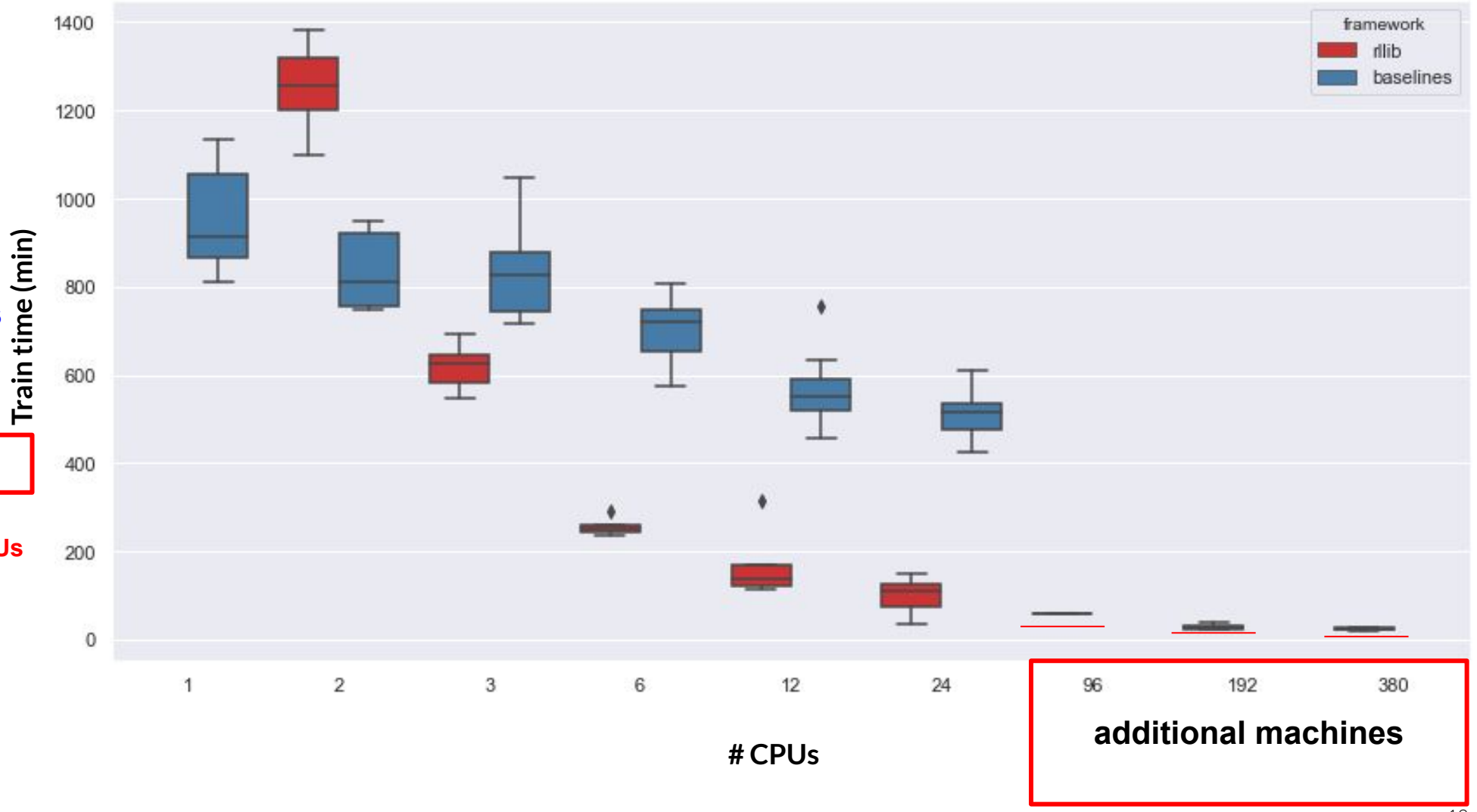
RLlib + Ray vs Stable Baselines Experiment Time (20M samples)

Stable Baseline 7-10Hrs

RLlib 1.5-2.5 Hrs

RLlib + Ray ~20 min

~6x speedup with ~16x the CPUs



Other nice things about RLlib + Ray

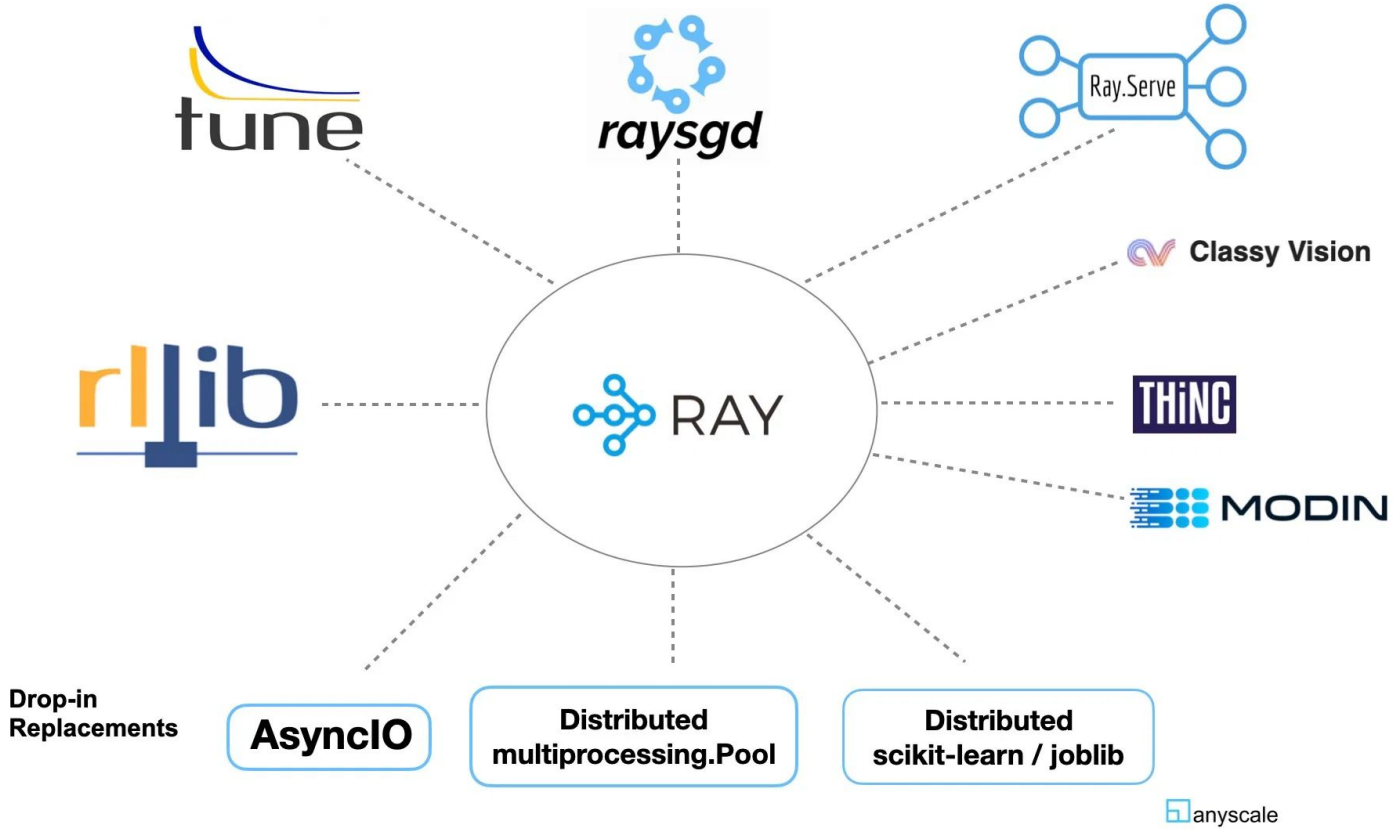
Bonus features

Tune

a. Hyper parameter optimization
b. Experiment management

Large number of supported algorithms in **RLlib**

Ray distributed compute



Conclusion

Stable Baselines -> **RLlib**

- Mileage may vary depending on the experiment
  - 7-10 Hrs -> 1.5-2.5 Hrs ~4x
- Be aware of the environment parallelization!

+ Ray Clusters

- Mileage may vary depending on the experiment
  - 1.5-2.5 Hrs -> **~20 mins** ~6x
- Commodity CPU-only machines are cheap

Together **~24x**

The **Ray** ecosystem is a nice bonus

Questions?