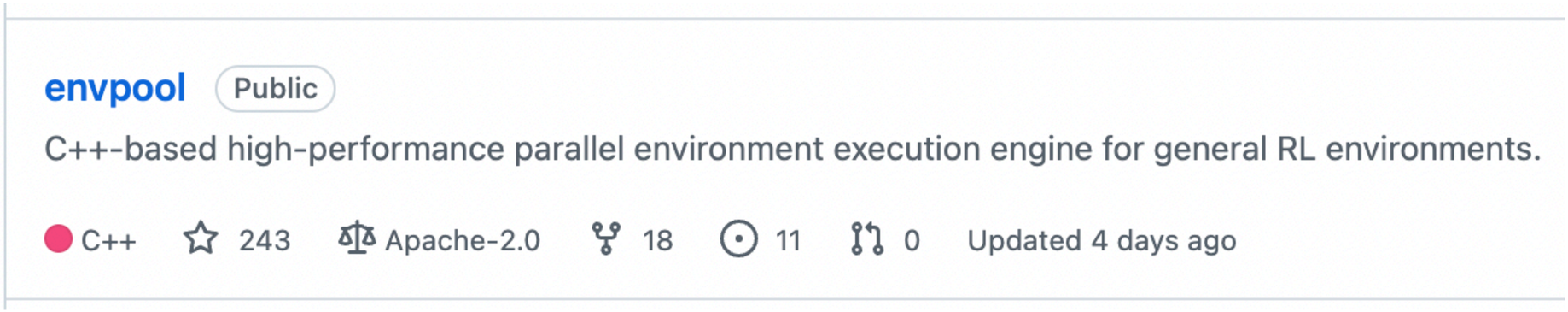
Jiayi Weng

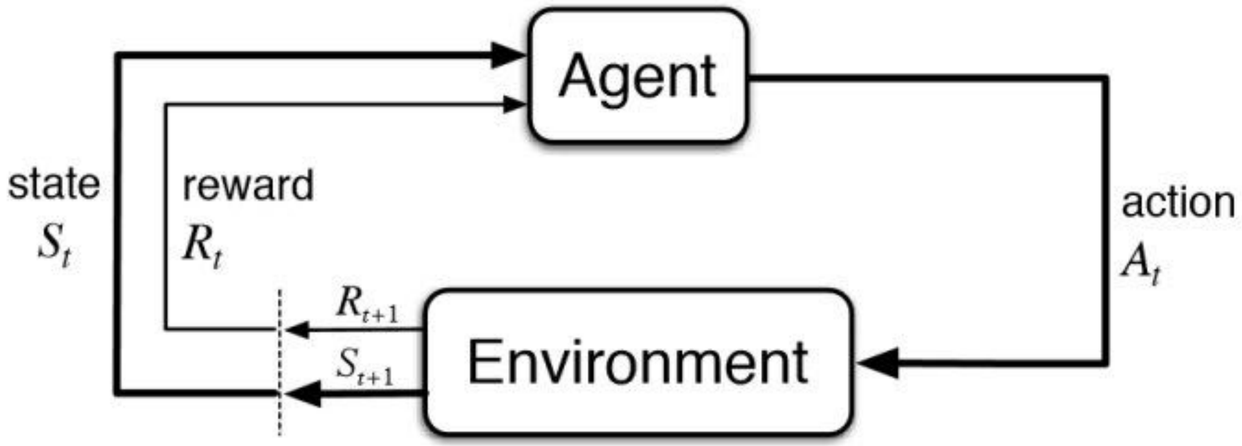
EnvPool has been open-sourced at GitHub on Nov. 11th: https://github.com/sail-sg/envpool



Introduction

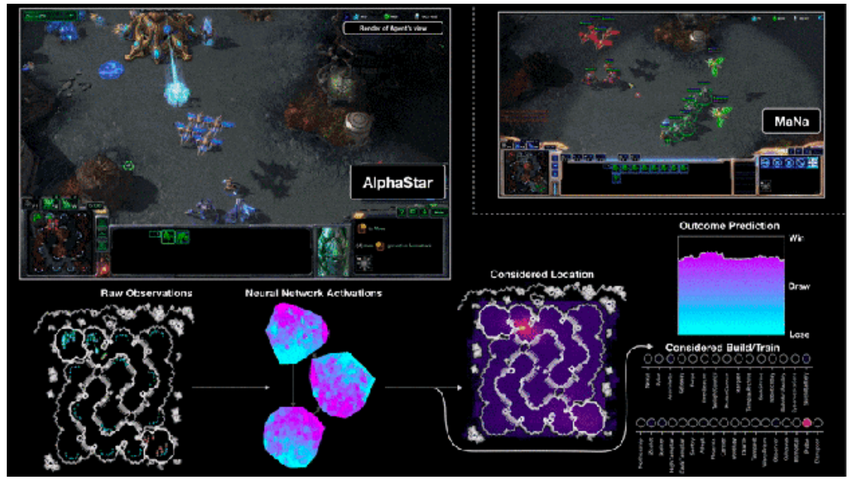
Reinforcement Learning (RL)

- Agent-Environment interaction loop
- Maximize reward-to-go
- Use many transitions of (s, a, r, s') to update agent's parameter

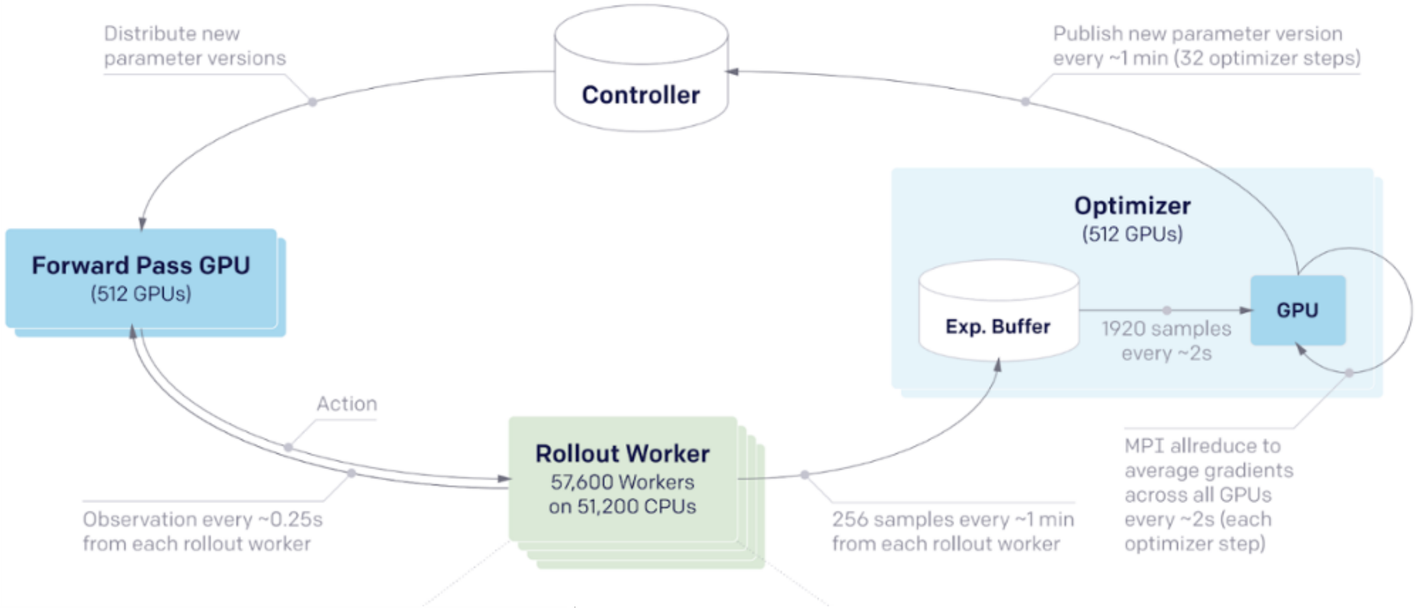


Huge Success of RL

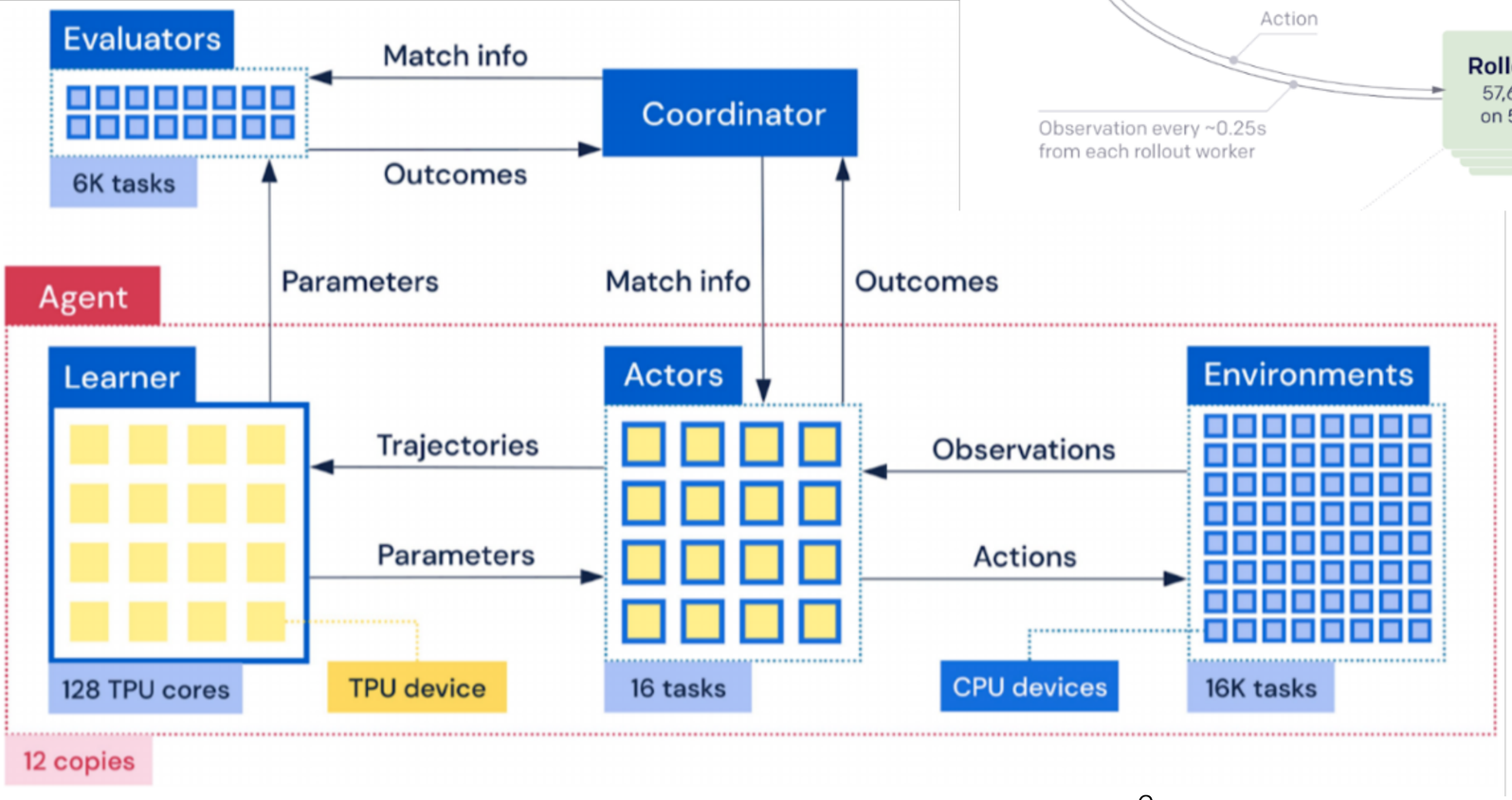
- Atari (2013)
- AlphaGo (2016)
- OpenAI Five (2019)
- AlphaStar (2019)
- …



More and More Computation



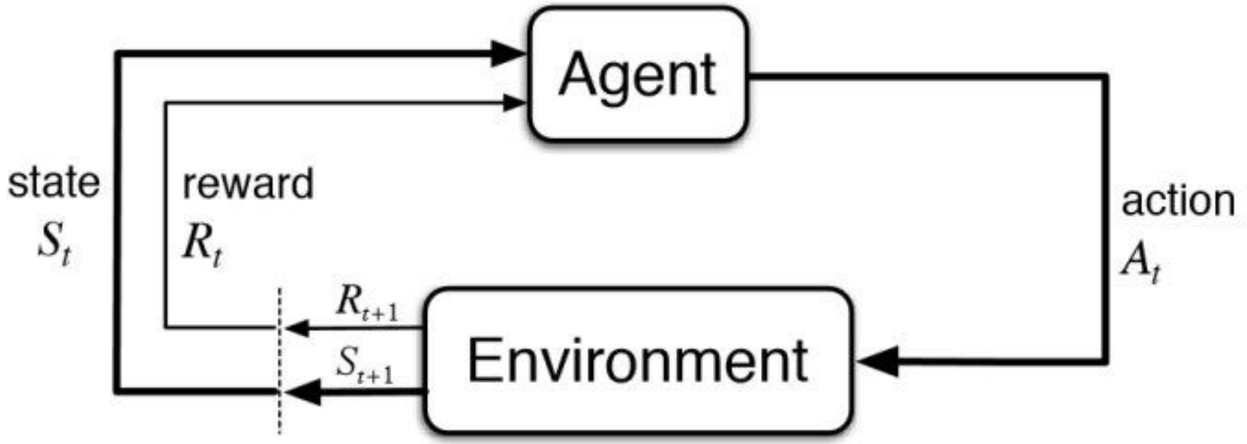
OpenAI Five, 50k+ CPUs



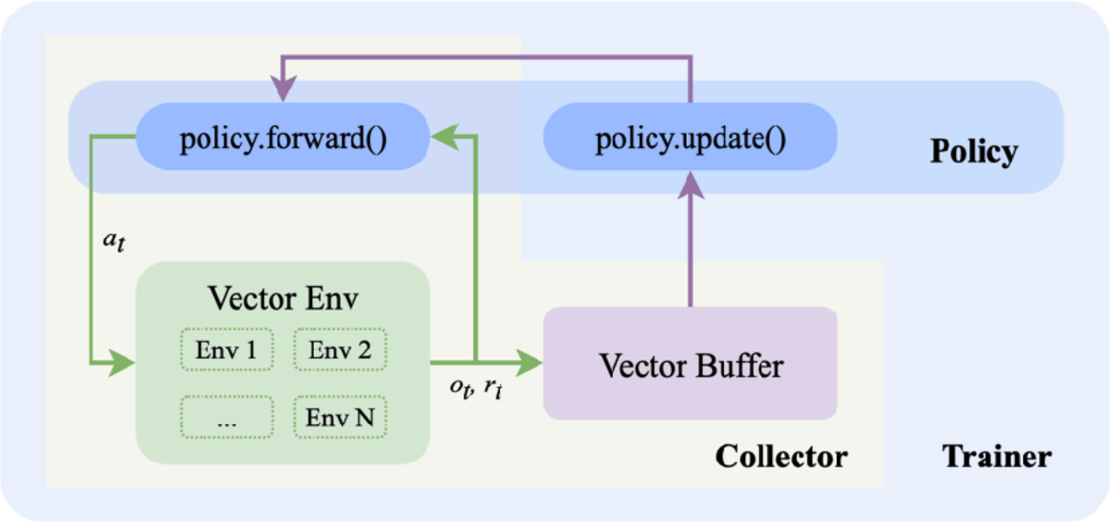
AlphaStar 16k tasks

Vectorized Environment

- 1 env - 1 agent
  - GPU with batch size = 1 ?



- n envs - 1 agent
  - More efficient



Problem Definition

What is a good vector_env?

- Scalability
- User-friendly — integrate new env
- Compatibility

Contribution

What is special in EnvPool?

- Fast: 1M FPS on DGX-A100 with Atari video games (fastest), 13x speed-up
- General: support any C++ env, sync/async, single/multi agent env
- Compatibility: gym/dm_env API, downstream training

Related Works

How to implement vector_env?

- Python subprocess
- Distributed Execution
- GPU CUDA
- Modern C++ std::thread

Python Subprocess

- OpenAI Baselines / gym.vector_env
- + share memory
- Pros: Very easy to integrate new environments
- Cons: Slow
  - GIL / Context switching / Inefficient data movement / Python language

Python Asynchronous Execution

- Sample-Factory
- Pros: Fast
- Cons: Optimized for Async PPO, cannot generalize to other algorithms

build passing | codecov 54% | license MIT | downloads 6k

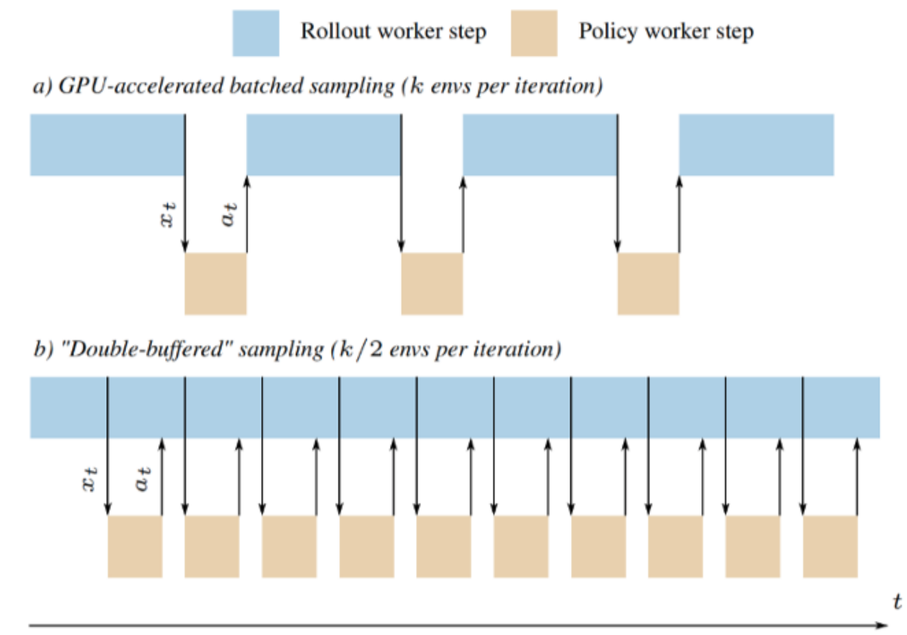
Sample Factory

Codebase for high throughput asynchronous reinforcement learning.

Paper: https://arxiv.org/abs/2006.11751

Talk: https://youtu.be/lLG17LKKSZc

Videos: https://sites.google.com/view/sample-factory

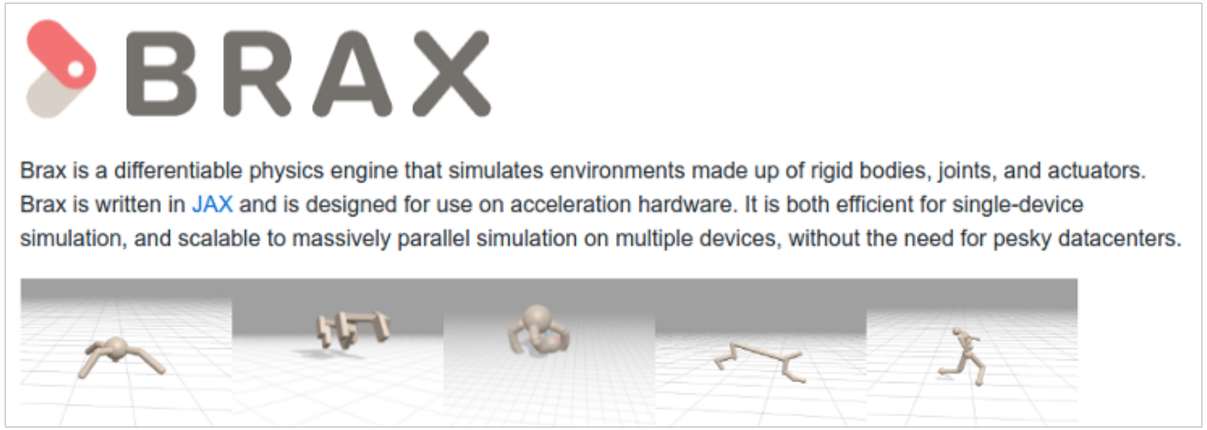


Distributed Execution

- Ray
- Pros: Can use many machines
- Cons: The underlying parallelization mechanism limits overall performance

CUDA Environments on GPU

- Brax / Isaac-gym
- Pros: Super fast
- Cons: Need to rewrite whole env



C++ Parallelization

- OpenAI Procgen / gym3: don’t achieve good performance
- DeepMind PodRacer: no open-source
- **EnvPool**

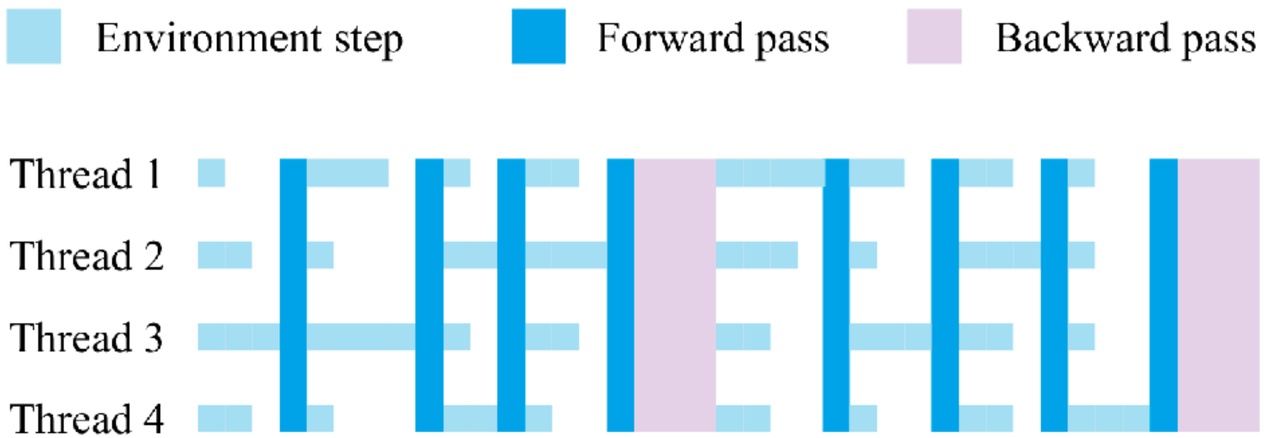
Why EnvPool choose C++-level parallelization?

- Speed: GPU >> C++ >> Python
- Env Integration: Python > C++ >> GPU
  - Many RL envs are written in C++ for faster speed
- Current satisfying C++-level parallelization solution for vector_env? No

Method

Sync step vs Async step

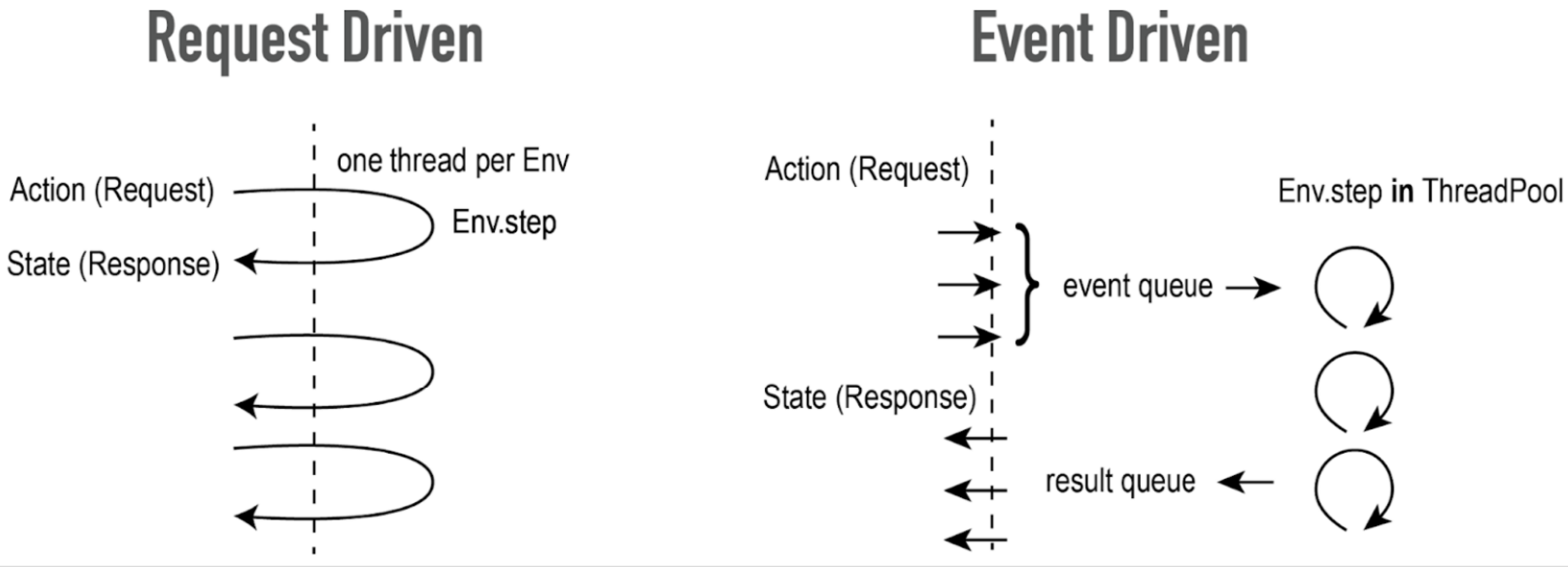
- num_envs (N): N x RL envs
- batch_size (M): Every step returns top M finished result
- Sync step: N == M
- Async step: N > M
- Huge speedup when N is large or uneven env.step execution time, see —>



(a) Sync step, num_envs = batch_size = 4

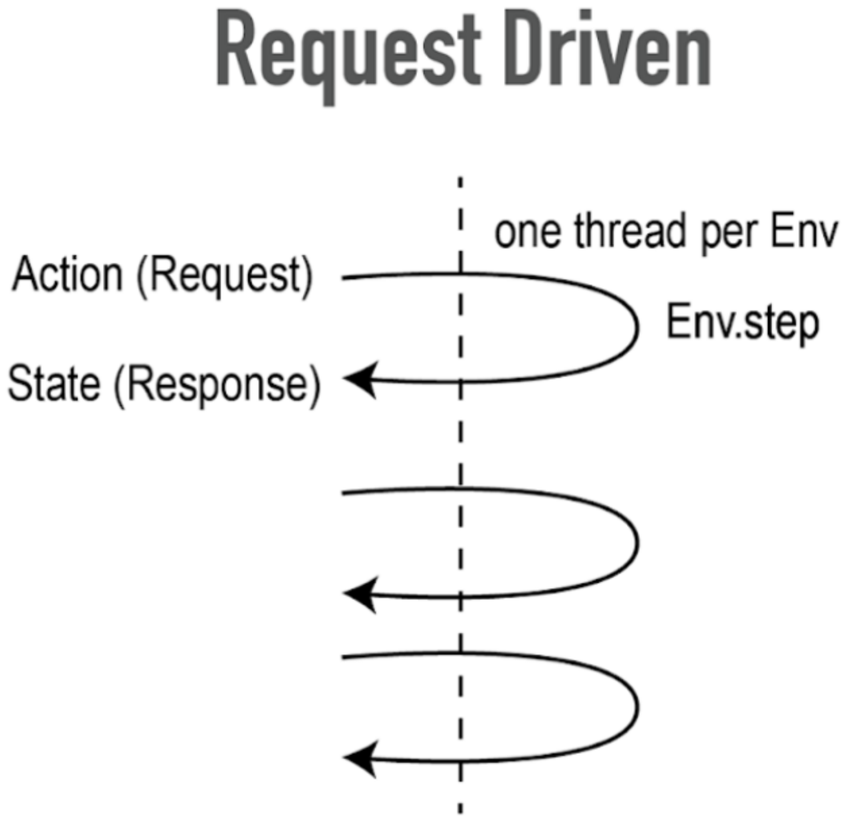
(b) Async step, num_envs = 4, batch_size = 3

Analogy to Web Service



Sync vs Async API

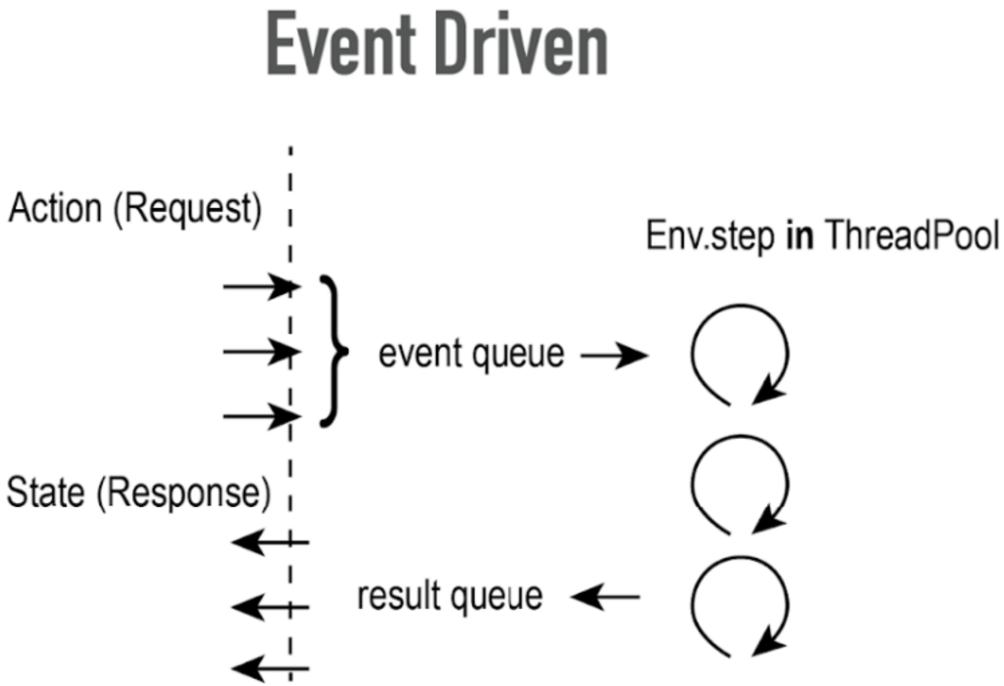
```
# Synchronous API
state = env.step(action)
```



Sync vs Async API

They can happen in different threads!

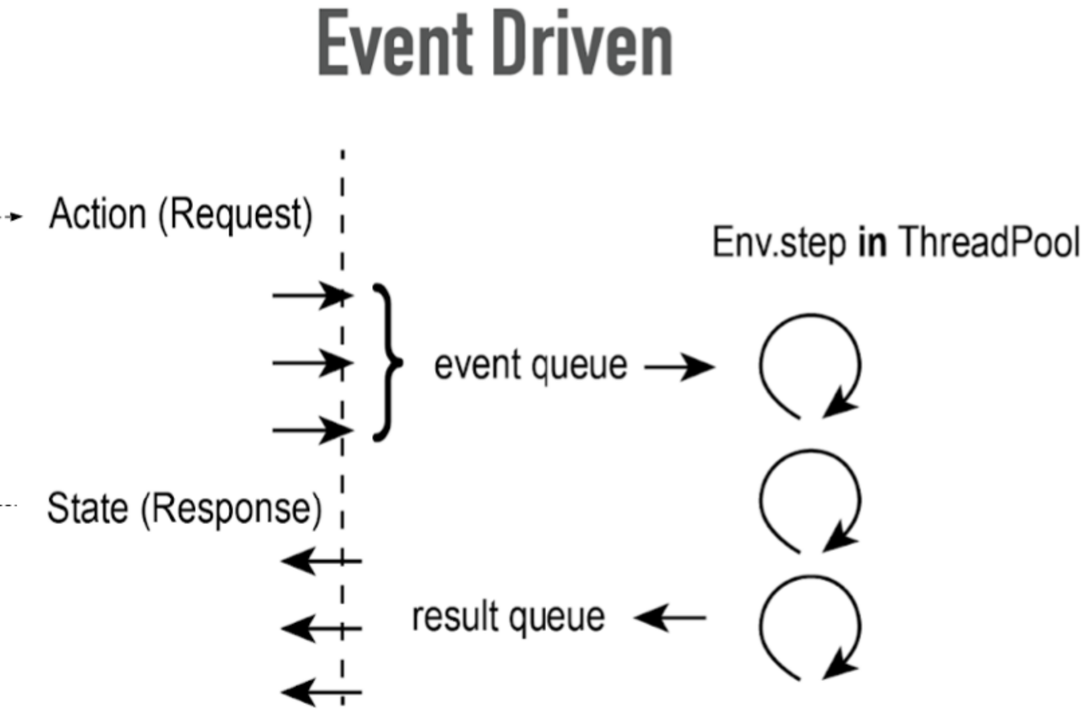
```
# Asynchronous API
env.send(action)
state = env.recv()
```



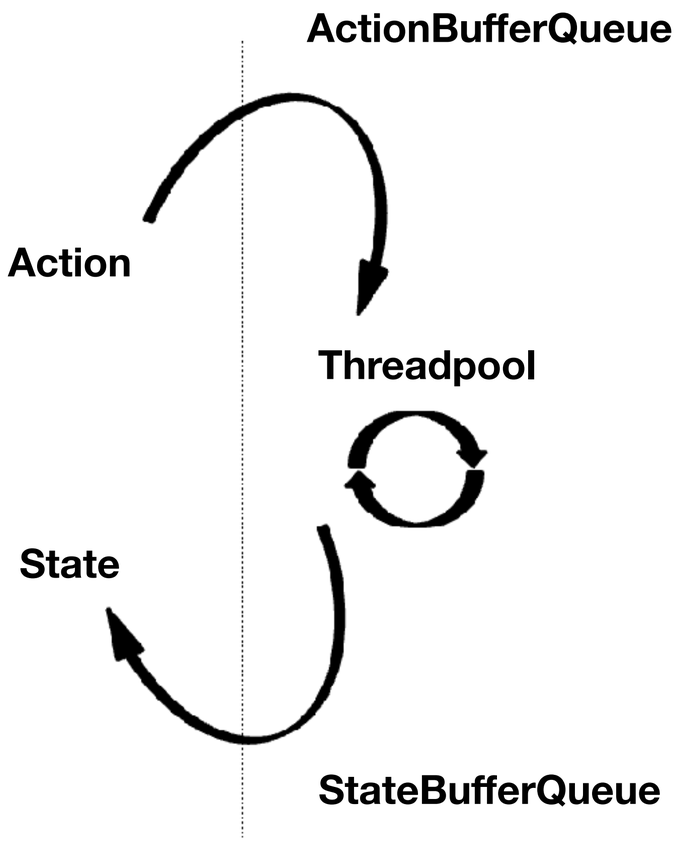
Sync vs Async API

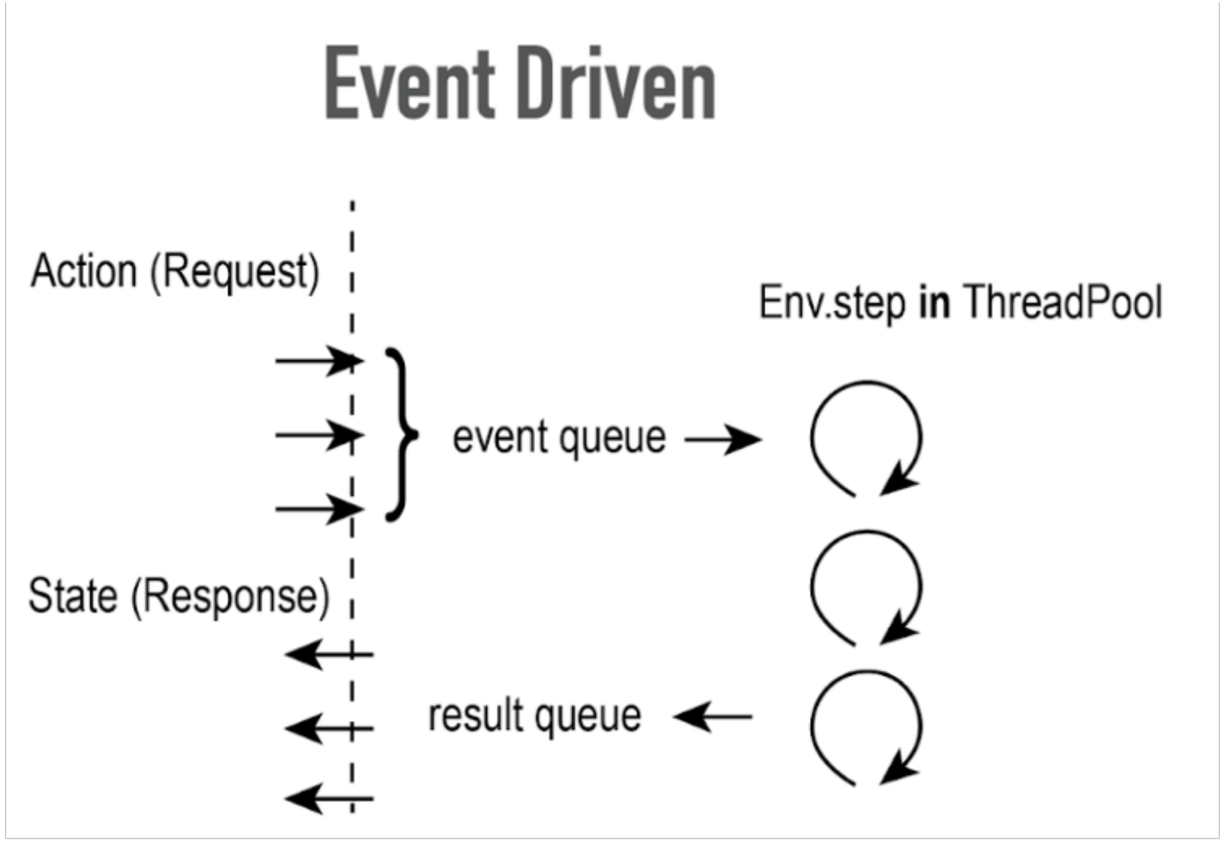
Interacting in Batch

```
# Batched
env.send(batch_action)
batch_state = env.recv()
```

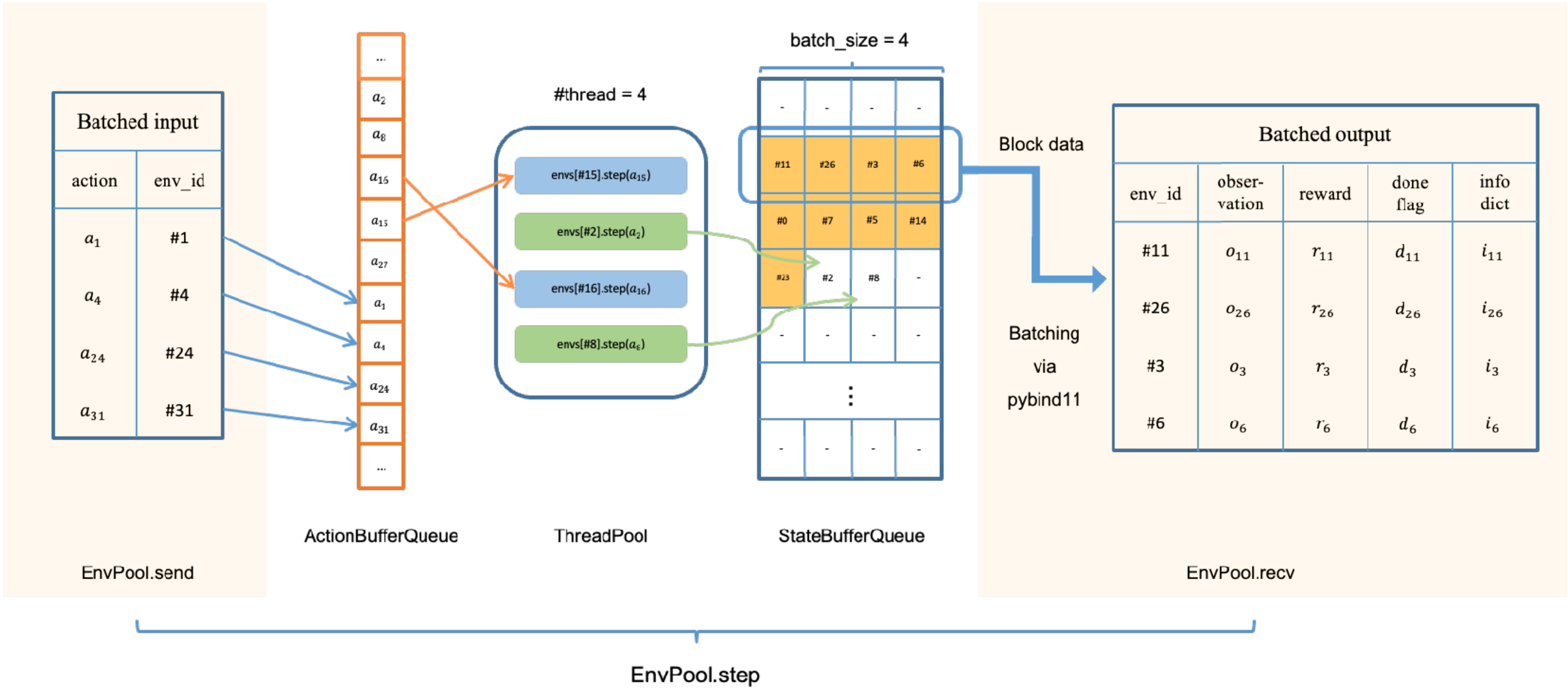


Key Components



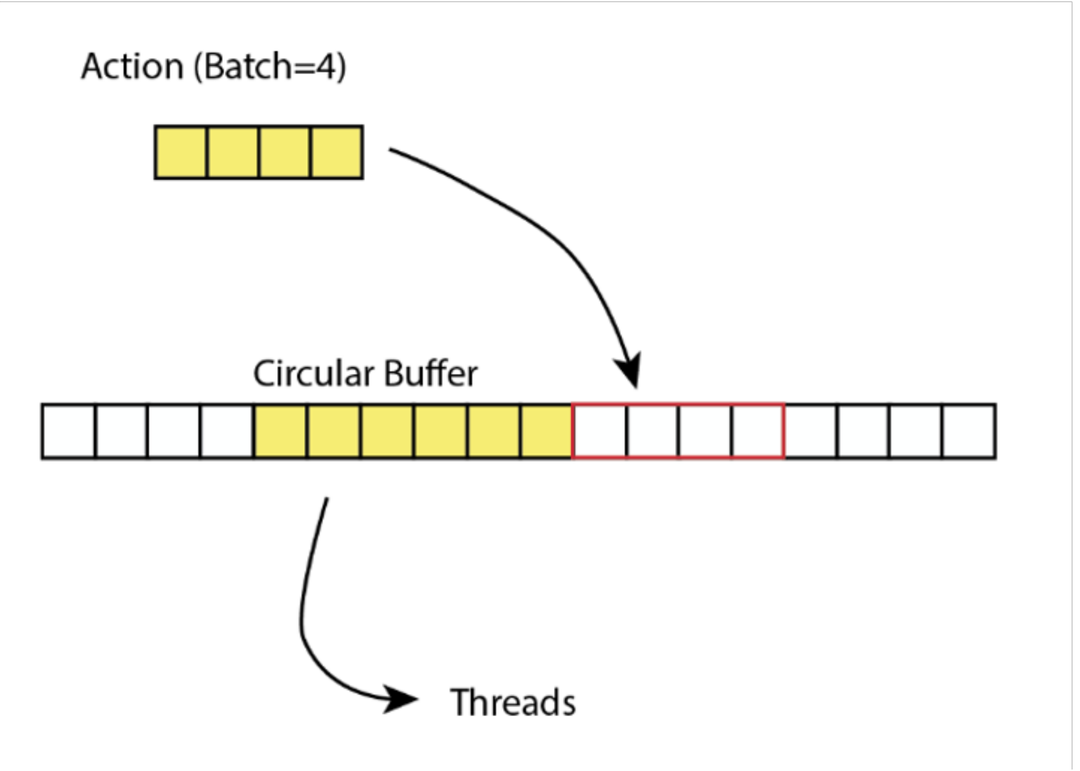


System Overview



ActionBufferQueue (event queue)

- Lock free circular buffer
- Fixed buffer size (> num_envs)
- Lock free: Semaphore to coordinate read/write; atomic pointer for indexing

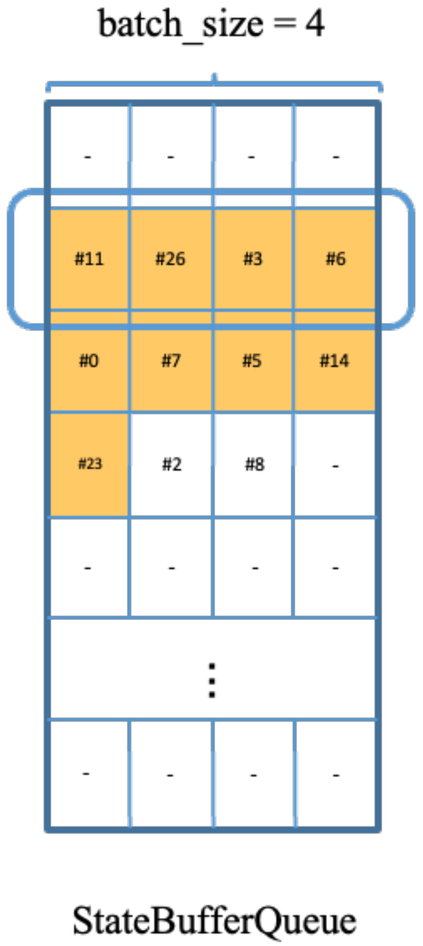


ThreadPool

- No context switching
  - num_threads == num_cores
  - Thread affinity turned on
- To keep threads busy
  - num_envs > num_threads
  - num_envs > batch_size (asynchronous)

StateBufferQueue (result queue)

- Lock free circular queue with preallocated buffer
  - Preallocated buffer —> batching costs zero copy
  - Lock free —> no competition for mutex



Experiments

Hardware Configurations

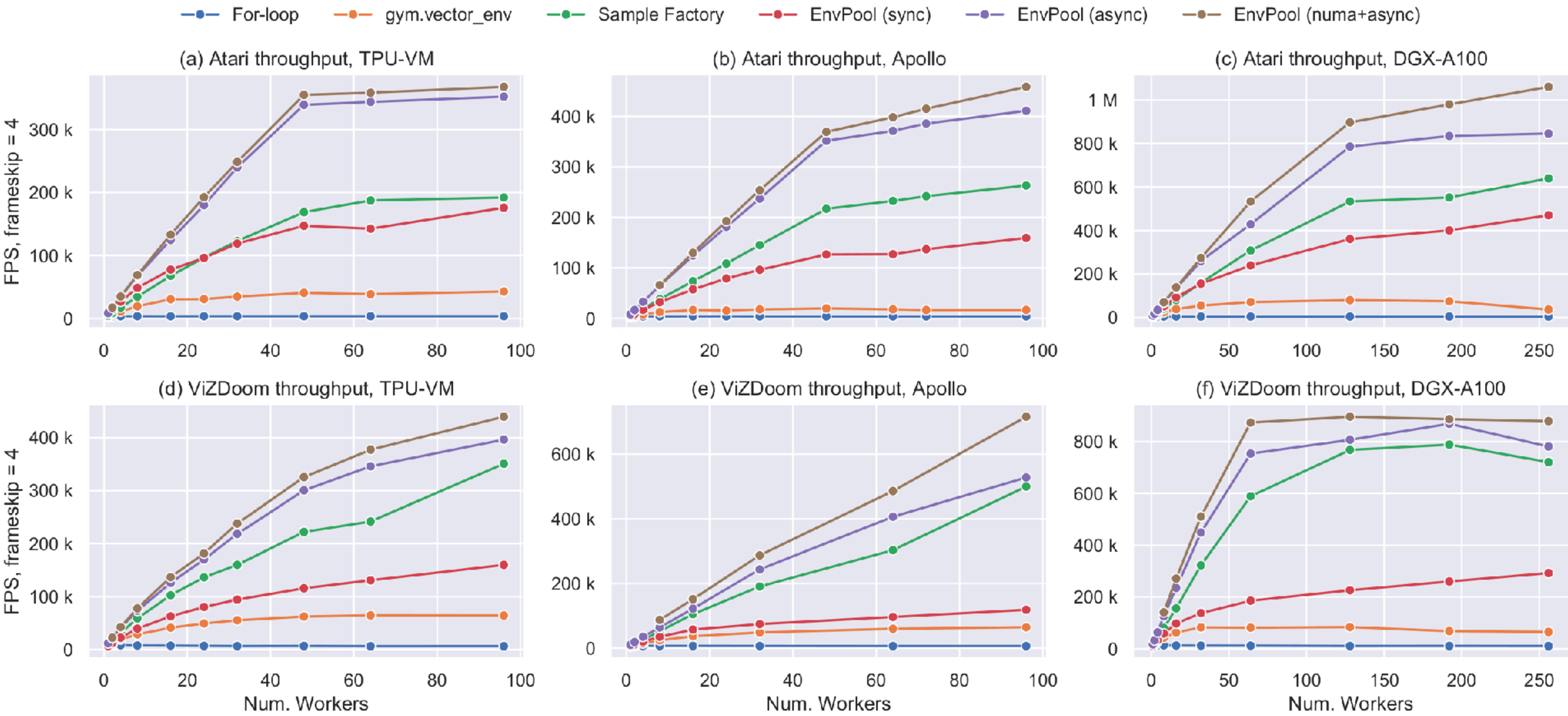
- Personal laptop: **12** core **Intel**(R) Core(TM) i7-8750H CPU @ 2.20GHz
- TPU-VM: **96** core **Intel**(R) Xeon(R) CPU @ 2.00GHz
- Apollo: **96** core **AMD** EPYC 7352 24-Core Processor
- DGX-A100: **256** core **AMD** EPYC 7742 64-Core Processor

Numerical Result

- 1M FPS in DGX-A100; 13x with Subprocess, ~2x with Sample Factory

Highest FPS	Laptop (12)	TPU-VM (96)	Apollo (96)	DGX-A100 (256)
For-loop	4,876	3,817	4,053	4,336
Subprocess	18,249	42,885	19,560	79,509
Sample Factory	27,035	192,074	262,963	639,389
EnvPool (sync)	40,791	175,938	159,191	470,170
EnvPool (async)	**50,513**	352,243	410,941	845,537
EnvPool (numa+async)	/	**367,799**	**458,414**	**1,060,371**

Details


For-loop
gym.vector_env
Sample Factory
EnvPool (sync)
EnvPool (async)
EnvPool (numa+async)
(a) Atari throughput, TPU-VM
(b) Atari throughput, Apollo
(c) Atari throughput, DGX-A100
(d) ViZDoom throughput, TPU-VM
(e) ViZDoom throughput, Apollo
(f) ViZDoom throughput, DGX-A100
FPS, frameskip = 4
Num. Workers
0
100 k
200 k
300 k
400 k
600 k
800 k
1 M
0
20
40
60
80
100
0
50
100
150
200
250

Even on laptop…

- Using tianshou's ShmemVectorEnv (num_envs = 8), 2:10 per 100k updates

```
$ python3 atari_ppo.py
Observations shape: (4, 84, 84)
Actions shape: 6
Epoch #1:   0%|                                                                                                | 0/100000 [00:00<?, ?it/s]
/home/trinkle/github/tianshou-new/tianshou/data/collector.py:413: UserWarning: Using async setting may collect extra transitions into buffer.
  warnings.warn("Using async setting may collect extra transitions into buffer.")
Epoch #1: 100352it [02:10, 748.46it/s, env_step=99328, len=1553, loss=0.001, loss/clip=-0.011, loss/ent=0.906, loss/vf=0.086, n/ep=1, n/st=10
```

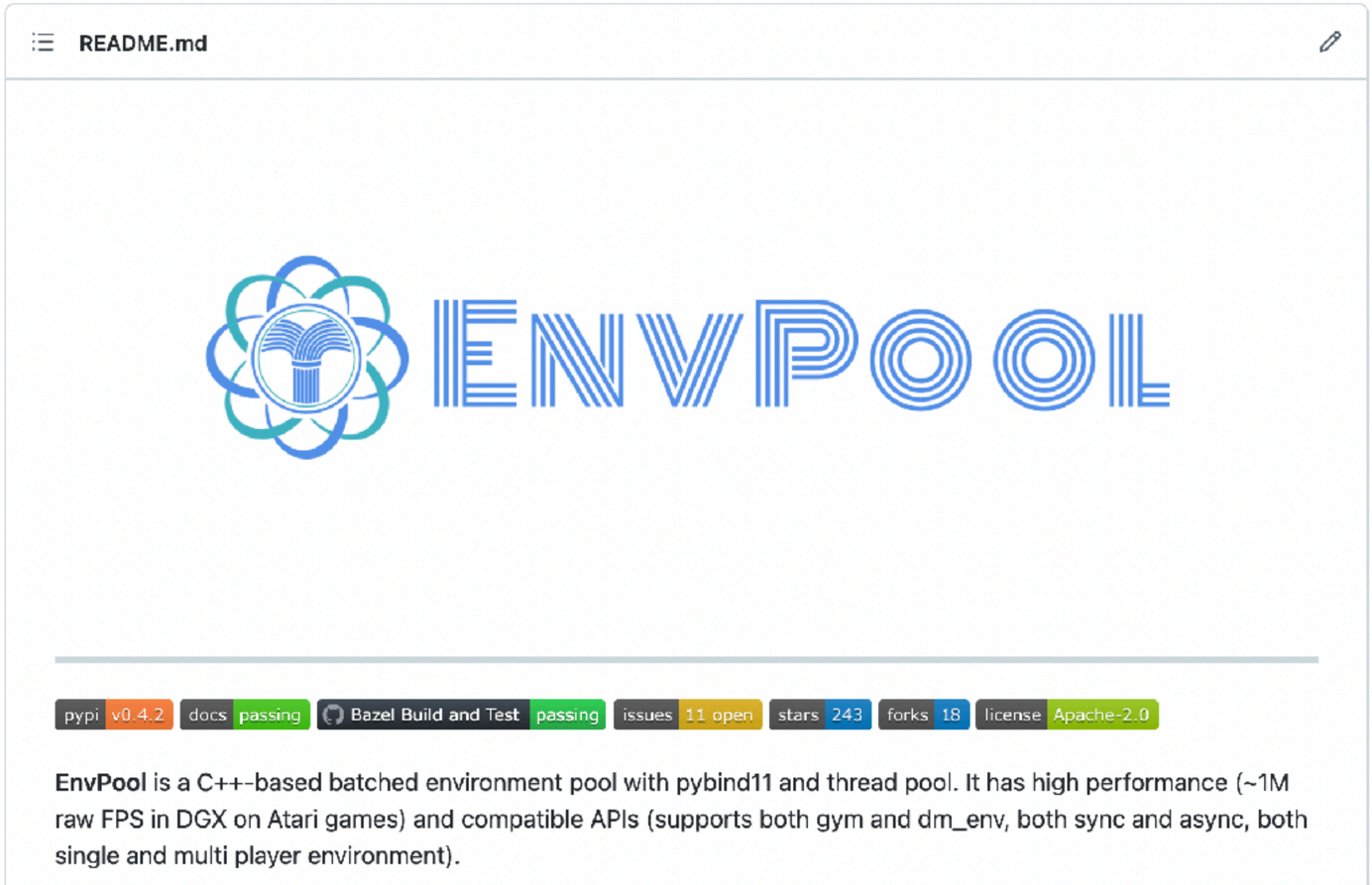
- Replace with EnvPool, 1:42 per 100k updates, 20% improvement in overall system

```
$ python3 atari_ppo.py
Observations shape: (4, 84, 84)
Actions shape: 6
Epoch #1:   0%|                                                                                                | 0/100000 [00:00<?, ?it/s]
/home/trinkle/github/tianshou-new/tianshou/data/collector.py:413: UserWarning: Using async setting may collect extra transitions into buffer.
  warnings.warn("Using async setting may collect extra transitions into buffer.")
Epoch #1: 100352it [01:42, 956.53it/s, env_step=99328, len=1683, loss=0.005, loss/clip=-0.009, loss/ent=0.663, loss/vf=0.083, n/ep=1, n/st=10
```

Miscellaneous

Open-source at GitHub

- https://github.com/sail-sg/envpool
- Impact: 200+ stars now



Software Engineering Standards

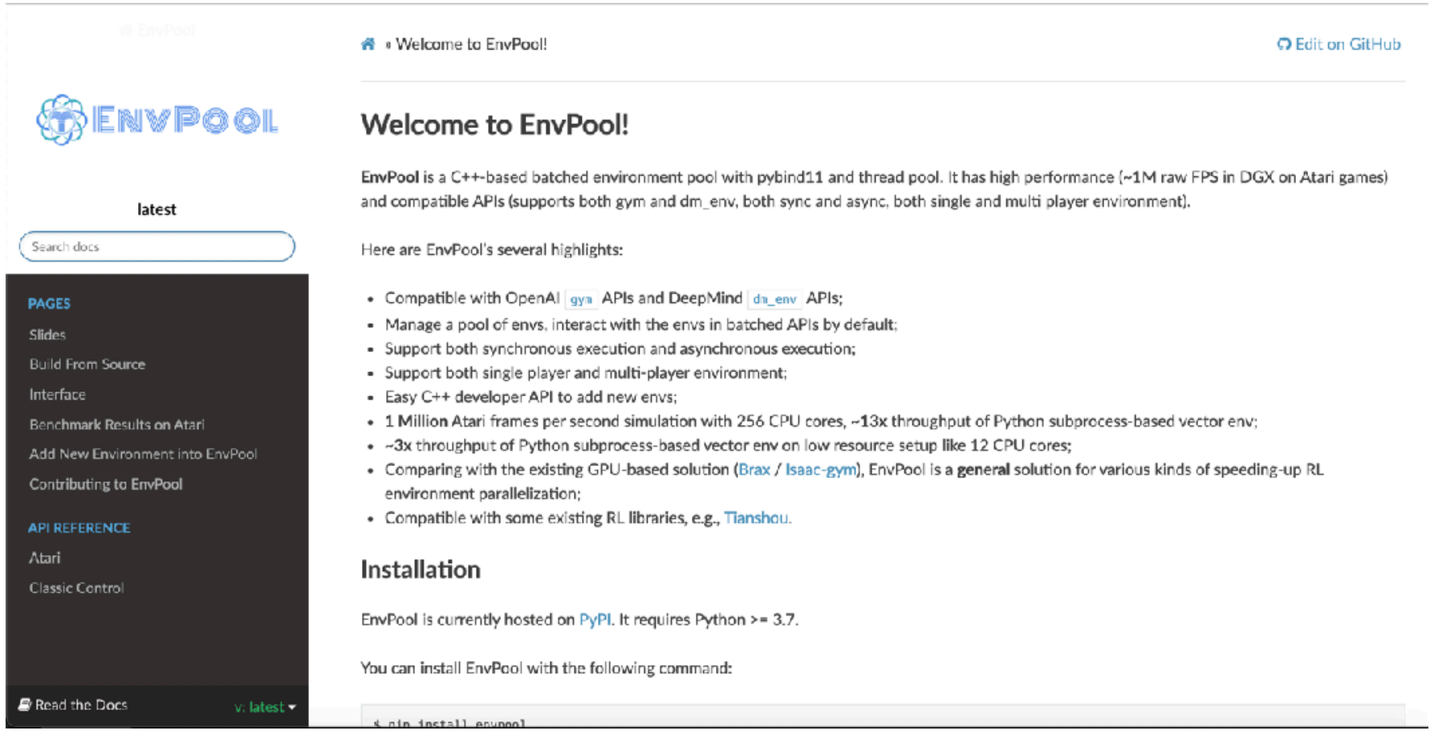
- CI/CD: lint, test (based on Bazel), release
- Issues with Issue Templates
- Pull requests with approval or comments

Community Support

- Upstream (environment API): support gym and dm_env
- Downstream (RL agent training): support tianshou and stable-baselines3
- PyPI installation: (need python >= 3.7) pip install envpool

Documentation

- https://envpool.readthedocs.io/en/latest/
- Including tutorials and how to integrate new environments into EnvPool



Reference

- Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).
- Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *nature* 529.7587 (2016): 484-489.
- Berner, Christopher, et al. "Dota 2 with large scale deep reinforcement learning." *arXiv preprint arXiv:1912.06680* (2019).
- Vinyals, Oriol, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." *Nature* 575.7782 (2019): 350-354.
- Brockman, Greg, et al. "Openai gym." *arXiv preprint arXiv:1606.01540* (2016).
- Cobbe, Karl, et al. "Leveraging procedural generation to benchmark reinforcement learning." *International conference on machine learning*. PMLR, 2020.

Reference

- Moritz, Philipp, et al. "Ray: A distributed framework for emerging AI applications." *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 2018.
- Freeman, C. Daniel, et al. "Brax-A Differentiable Physics Engine for Large Scale Rigid Body Simulation." (2021).
- Makoviychuk, Viktor, et al. "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning." *arXiv preprint arXiv:2108.10470* (2021).
- Petrenko, Aleksei, et al. "Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning." *International Conference on Machine Learning*. PMLR, 2020.
- Hessel, Matteo, et al. "Podracer architectures for scalable Reinforcement Learning." *arXiv preprint arXiv:2104.06272* (2021).
- Weng, Jiayi, et al. "Tianshou: A Highly Modularized Deep Reinforcement Learning Library." *arXiv preprint arXiv:2107.14171* (2021).
- Raffin, Antonin, et al. "Stable baselines3." *GitHub repository* (2019).