ENVPOOL

翁家翌

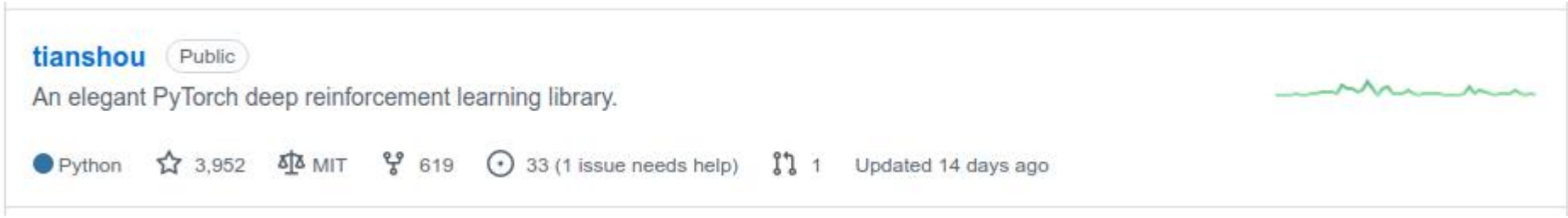
Nov 17, 2021

https://github.com/sail-sg/envpool

About Me

- 清华大学计算机科学与技术系，本科，2016-2020
- 卡内基·梅隆大学（CMU）计算数据科学项目（MCDS），硕士在读，2020-2022

- 强化学习算法库天授（https://github.com/thu-ml/tianshou）作者，开源至今快4k stars



Tianshou

https://github.com/sail-sg/envpool

sea | AI Lab

About Sea AI Lab (SAIL)

- https://sail.sea.com/
- 位于新加坡，成立于2020年12月，由颜水成（Yan Shuicheng）老师带队，专注于前沿突破性基础研究
- SAIL隶属于Sea Ltd., 旗下有Shopee、Garena等众多优秀的子公司，发展前景十分可观
- 人贩贴：SAIL长期招聘 Research Scientist / Research Engineer / Research Intern，新加坡与北京都有名额，详情咨询 HR：kaylajs@sea.com

Acknowledgement

Jiayi Weng

Min Lin

Zhongwen Xu

Shuicheng Yan

https://github.com/sail-sg/envpool

What: 什么是EnvPool？

- 高性能 强化学习 环境 并行 模拟
- vector_env
- 相比基于Python的subprocess解决方案（gym, baselines, ...）能快13倍
- 相比目前最快的通用CPU环境并行实现（Sample Factory）能快近2倍

- 对于任意环境都能进行通用的加速
- 在几乎任意硬件上都能观察到加速
- 兼容已有上下游生态（环境：gym/dm_env API，训练：tianshou, ...）

https://github.com/sail-sg/envpool

性能怎么个高法？

- 以Atari Pong举例
- 在12核的笔记本上，EnvPool能到4w~5w FPS

  gym.vector_env等基于Python subprocess只能跑不到2w FPS
- 在96核机器（HPE-Apollo）上，同步版EnvPool能到16w FPS，异步版能到45w FPS

  gym.vector_env最高不到2w FPS
- 在256核机器（Nvidia DGX-A100）上，同步版EnvPool能到47w FPS，异步版能到84w FPS，开了NUMA之后能106w FPS

  gym.vector_env最高不到8w FPS

性能怎么个高法？

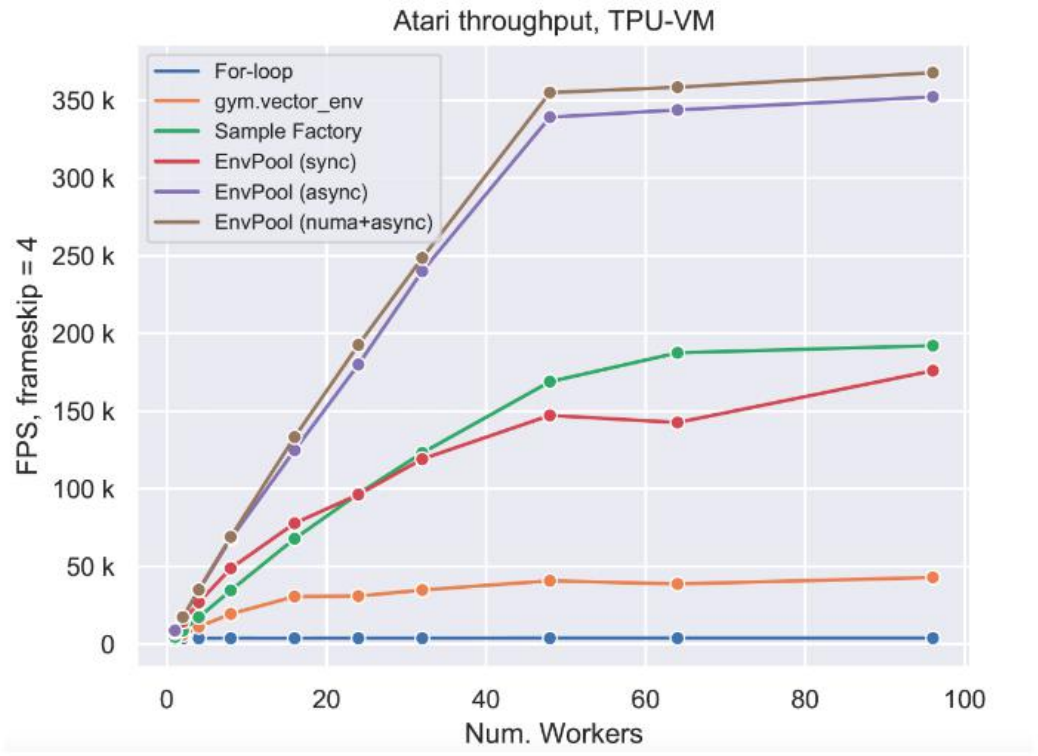
- 比起之前最快的实现Sample-Factory也是有一定的优势

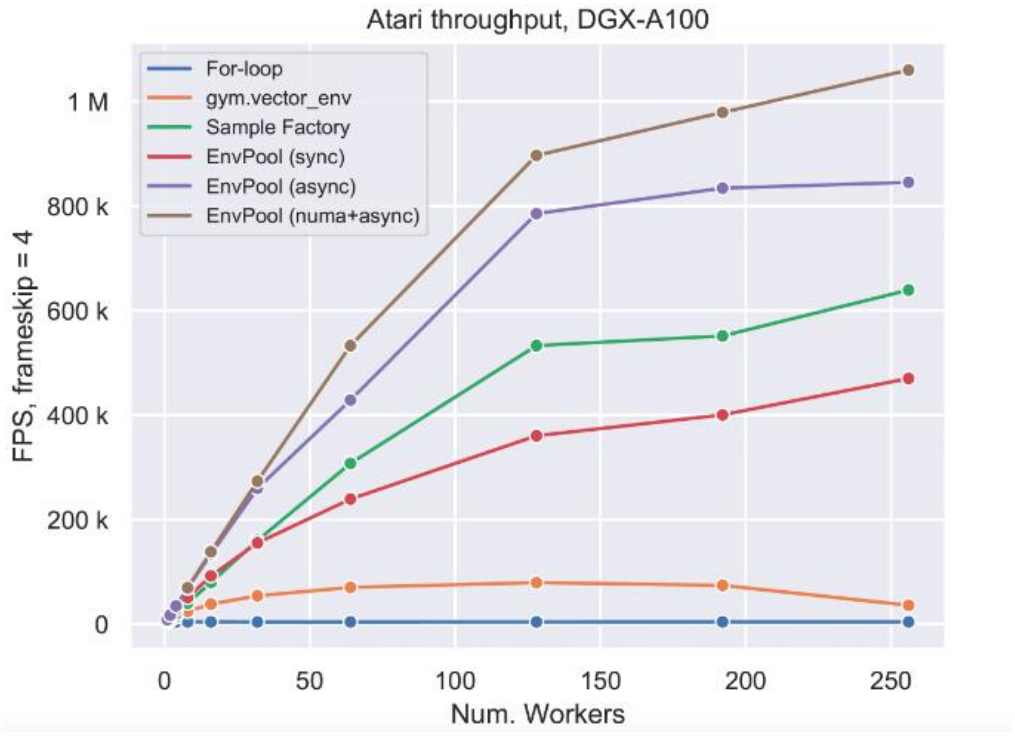
Highest FPS	Laptop (12)	TPU-VM (96)	Apollo (96)	DGX-A100 (256)
For-loop	4,876	3,817	4,053	4,336
Subprocess	18,249	42,885	19,560	79,509
Sample Factory	27,035	192,074	262,963	639,389
EnvPool (sync)	40,791	175,938	159,191	470,170
EnvPool (async)	**50,513**	352,243	410,941	845,537
EnvPool (numa+async)	/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

https://github.com/sail-sg/envpool

性能怎么个高法？

- 比起之前最快的实现Sample-Factory也是有一定的优势





https://github.com/sail-sg/envpool

性能怎么个高法？

- 集成进 天授（tianshou）也能直接跑
- 整体性能有显著提升，即使只有8个Atari Pong环境，在笔记本上跑

- Shmem
- 2:10

```
$ 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
```

- EnvPool (sync)
- 1:42

```
$ 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
```

- EnvPool (async) —— 还在集成中（内部版是可以的）

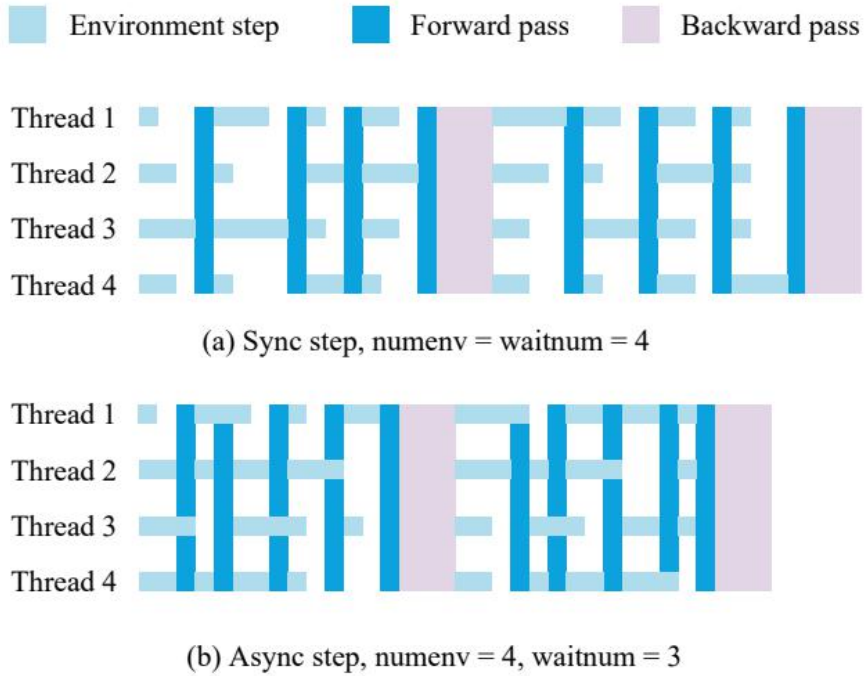
性能怎么个高法？

- 还有新的idea —— EnvPool今后还会更快

概念介绍：同步（sync step）与异步（async step）

- 内部实现不区别对待
- 异步step：8个env（num_envs=8）每次设置只返回其中4个的结果（batch_size=4），每次step的时候把剩下的4个env放在background继续跑
- 同步step：num_envs == batch_size，每次等全部step好再返回

- 当环境数量很多，或者执行时间不均匀时有明显加速效果



https://github.com/sail-sg/envpool

EnvPool用法

```python
import envpool
import numpy as np

# make gym env
env = envpool.make("Pong-v5", env_type="gym", num_envs=100)
# or use envpool.make_gym(...)
obs = env.reset()  # should be (100, 4, 84, 84)
act = np.zeros(100, dtype=int)
obs, rew, done, info = env.step(act)
```

```python
import envpool
import numpy as np

# make asynchronous
num_envs = 64
batch_size = 16
env = envpool.make("Pong-v5", env_type="gym", num_envs=num_envs, batch_size=batch_size)
action_num = env.action_space.n
env.async_reset()  # send the initial reset signal to all envs
while True:
    obs, rew, done, info = env.recv()
    env_id = info["env_id"]
    action = np.random.randint(action_num, size=batch_size)
    env.send(action, env_id)
```

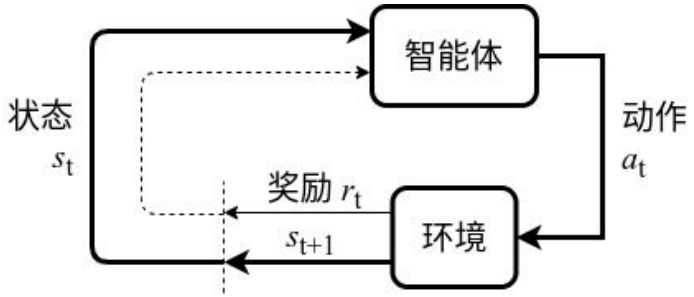
https://github.com/sail-sg/envpool

EnvPool安装

- 目前只支持Linux，python >= 3.7
- pip3 install envpool

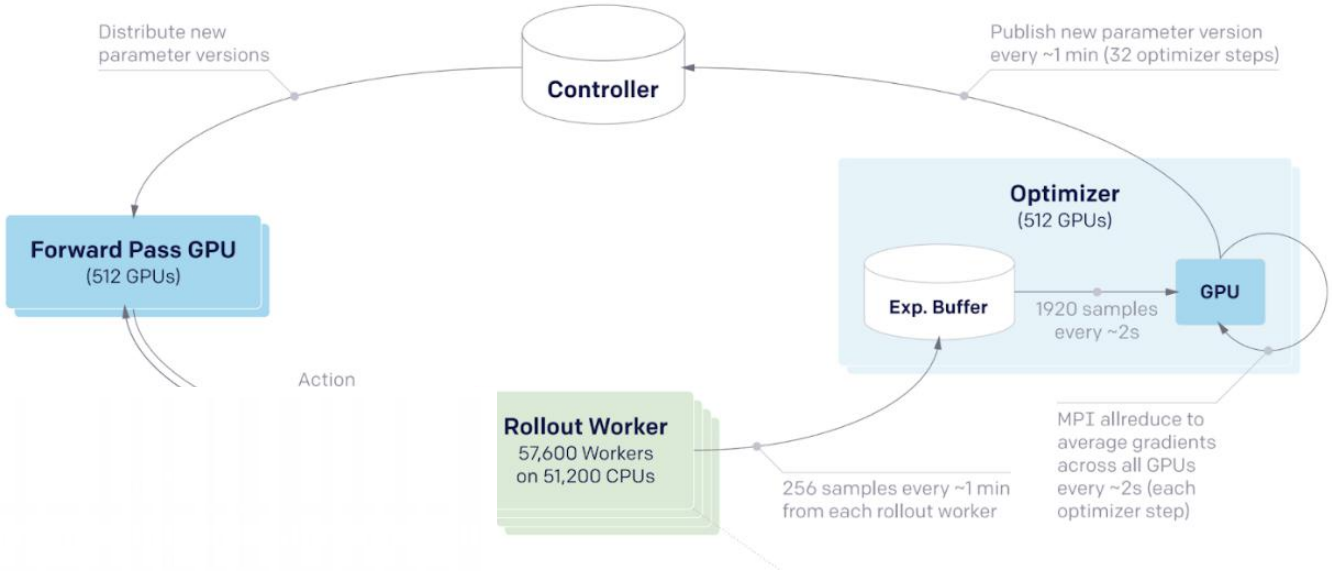
Why：为什么要做这个项目？

- 传统设置：1 env with 1 agent
- vector_env: n env with 1 agent
- gpu: batch learning / batch inference
- batch 意味着计算资源利用率提高

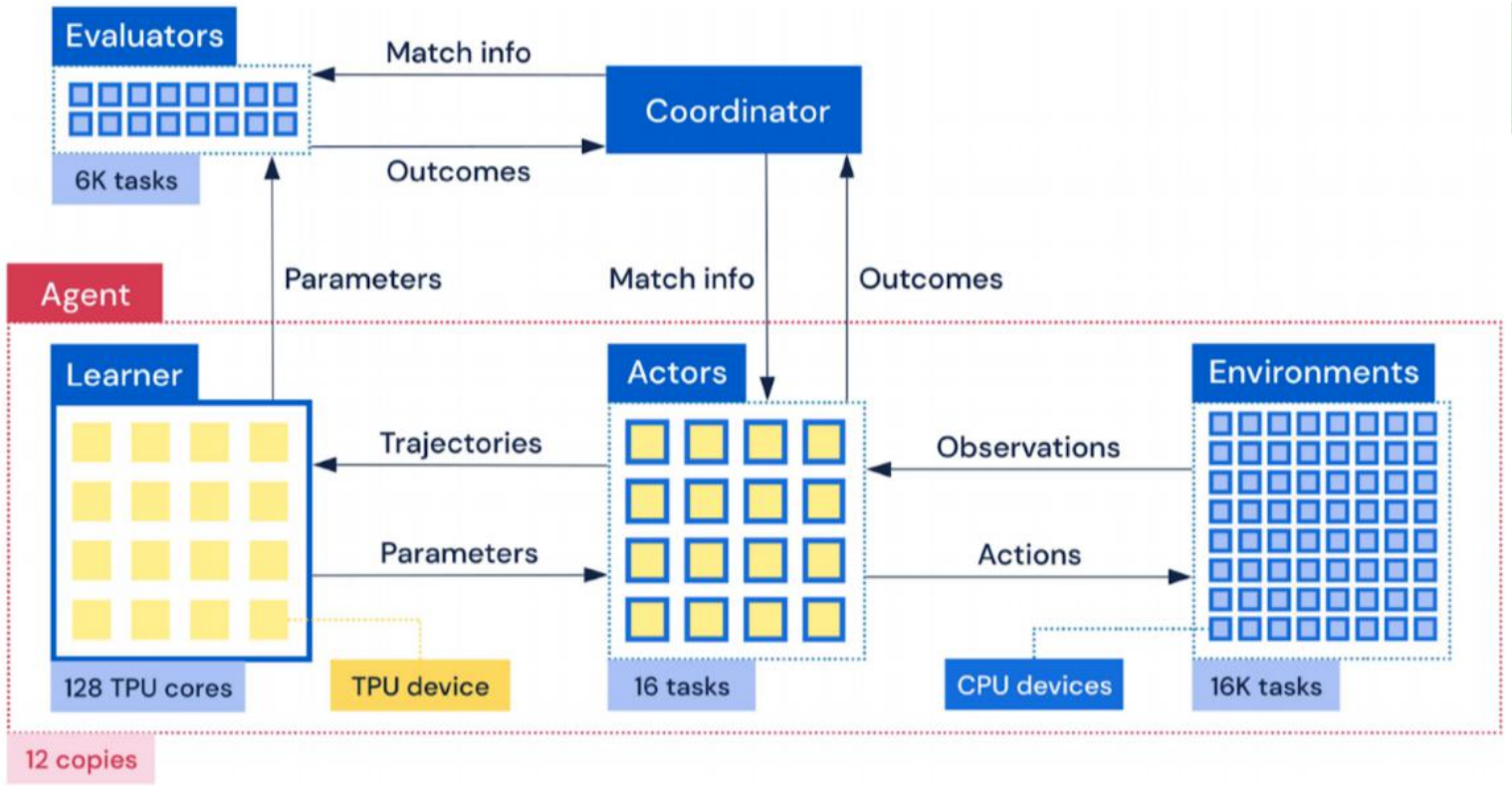


https://github.com/sail-sg/envpool

Why：为什么要做这个项目？



OpenAI Five



AlphaStar

https://github.com/sail-sg/envpool

在哪里并行？

- 目前最主流解决方案是基于python的subprocess

  openai baselines, gym.vector_env, ...

  +share memory 优化数据传输

- 缺点：

  进程切换需要context switching

  python GIL

  数据传输开销

  python本身慢

在哪里并行？

- Python-level subprocess
- 分布式：Ray —— 计算资源利用率不高
- Python async: Sample-Factory

  不能适用于所有RL算法，针对Async PPO特殊优化环境异步执行

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

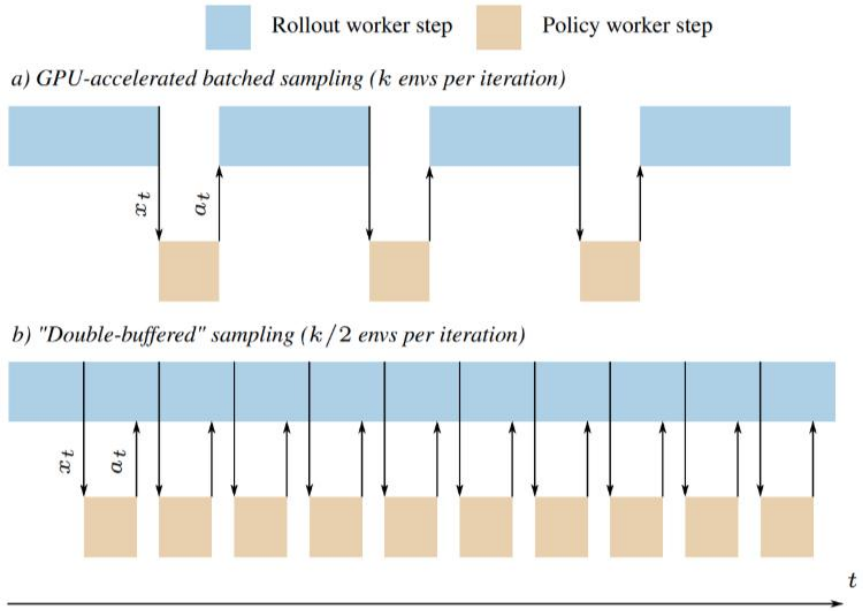
Sample Factory

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



https://github.com/sail-sg/envpool

在哪里并行？

- GPU：Brax / Isaac-gym

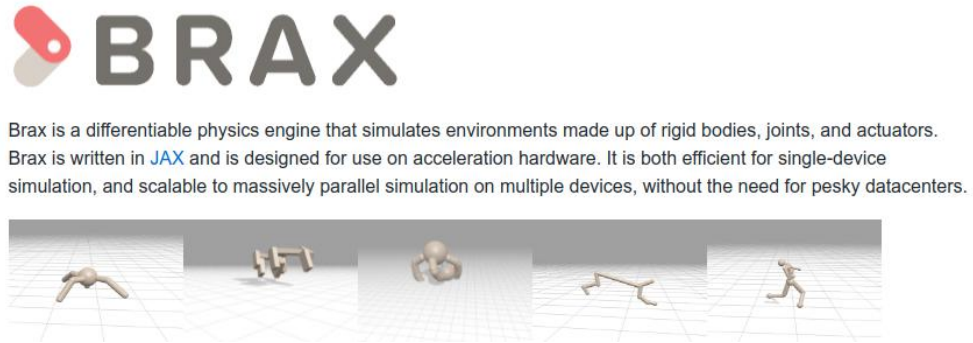
- Tradeoff

  快，能达到千万FPS级别的速度

  只能在CUDA上写纯计算，不够通用

  重写env而不是重写env接口

  生态？



https://github.com/sail-sg/envpool

在哪里并行？

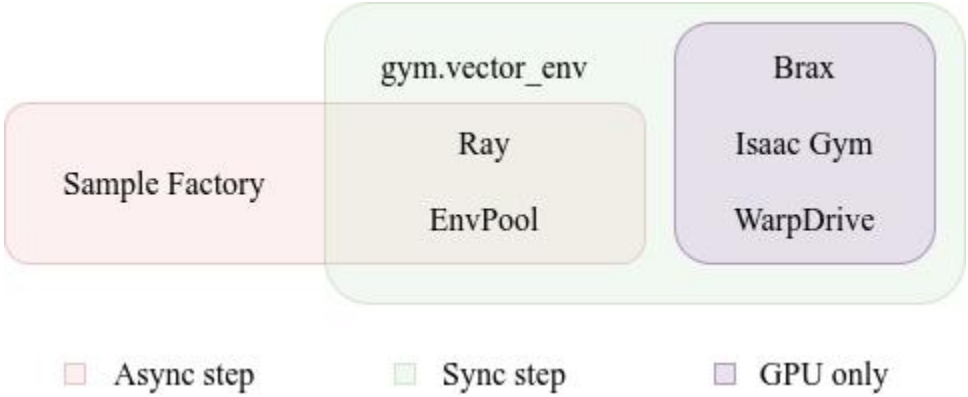
- CPU

  正常的RL环境为了追求性能几乎都是用C/C++写的 —— API比较容易接

  OpenAI procgen / gym3：效率不高：https://github.com/thu-ml/tianshou/issues/409

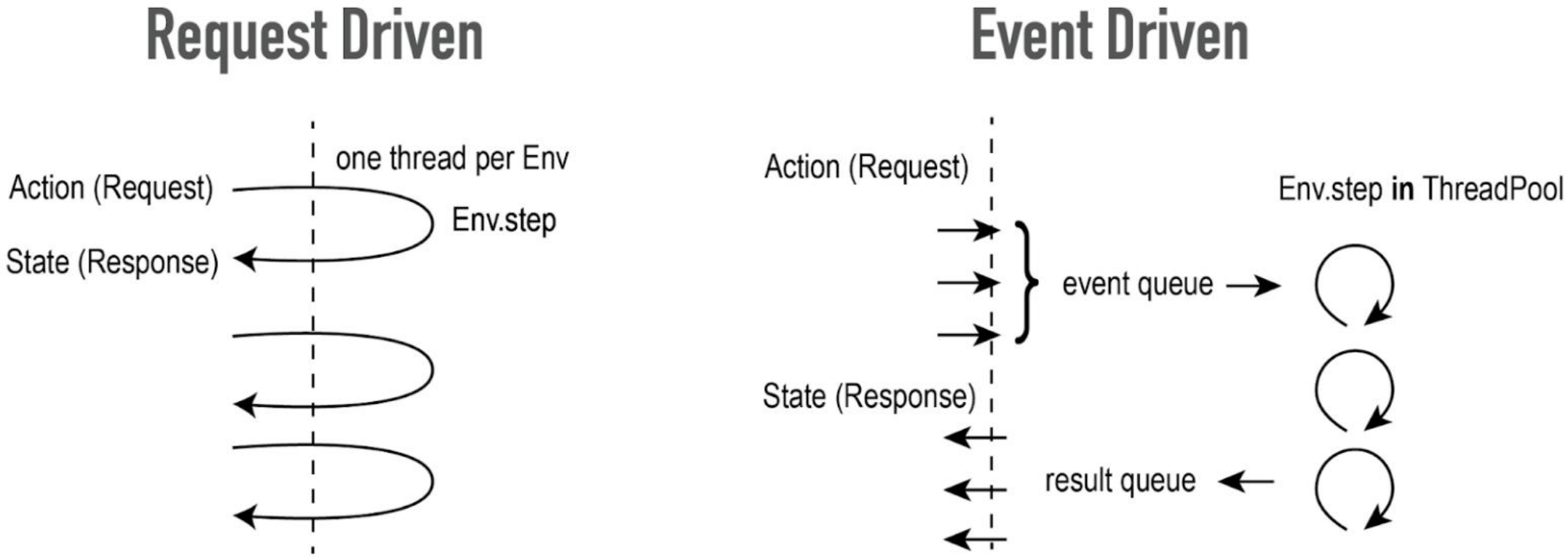
  DeepMind PodRacer：同步step+未开源

  EnvPool



https://github.com/sail-sg/envpool

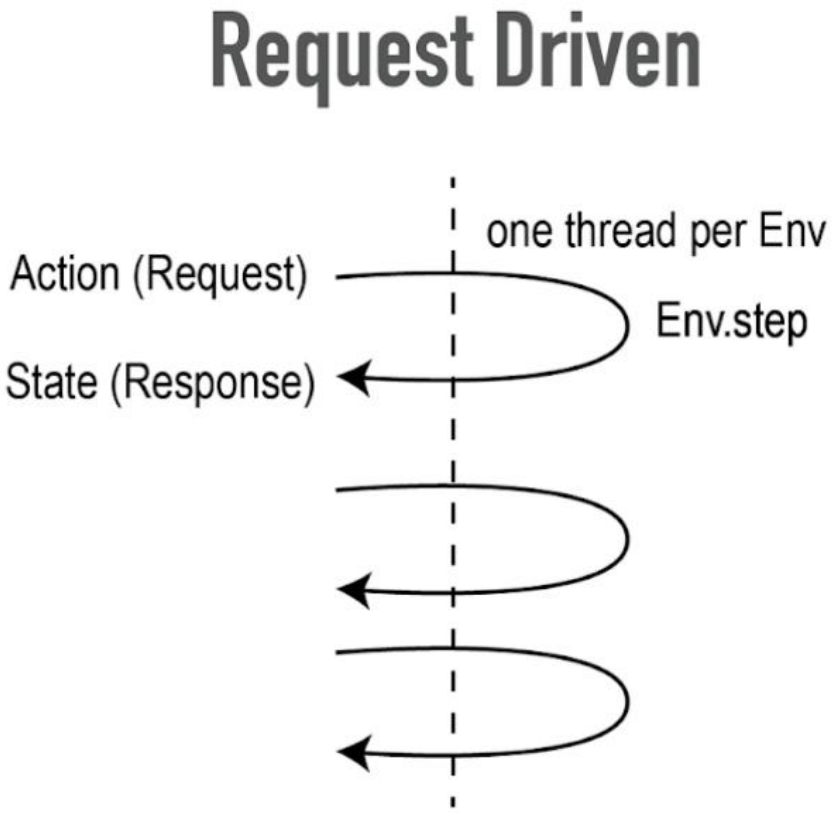
How：EnvPool内部设计与实现



https://github.com/sail-sg/envpool

How：EnvPool内部设计与实现

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

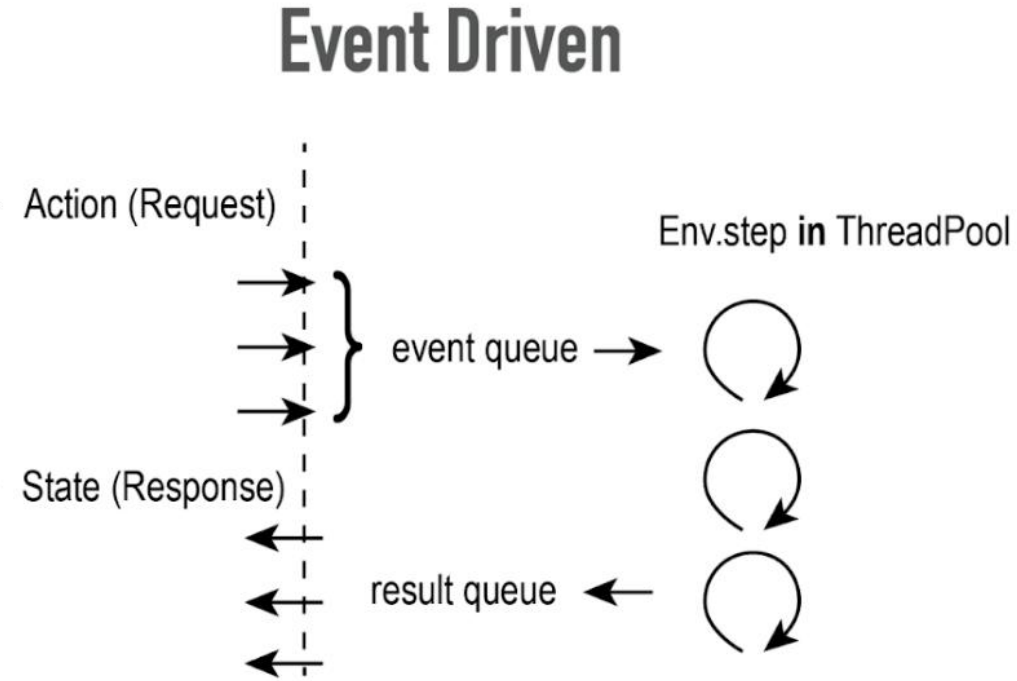


https://github.com/sail-sg/envpool

How：EnvPool内部设计与实现

They can happen in different threads!

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

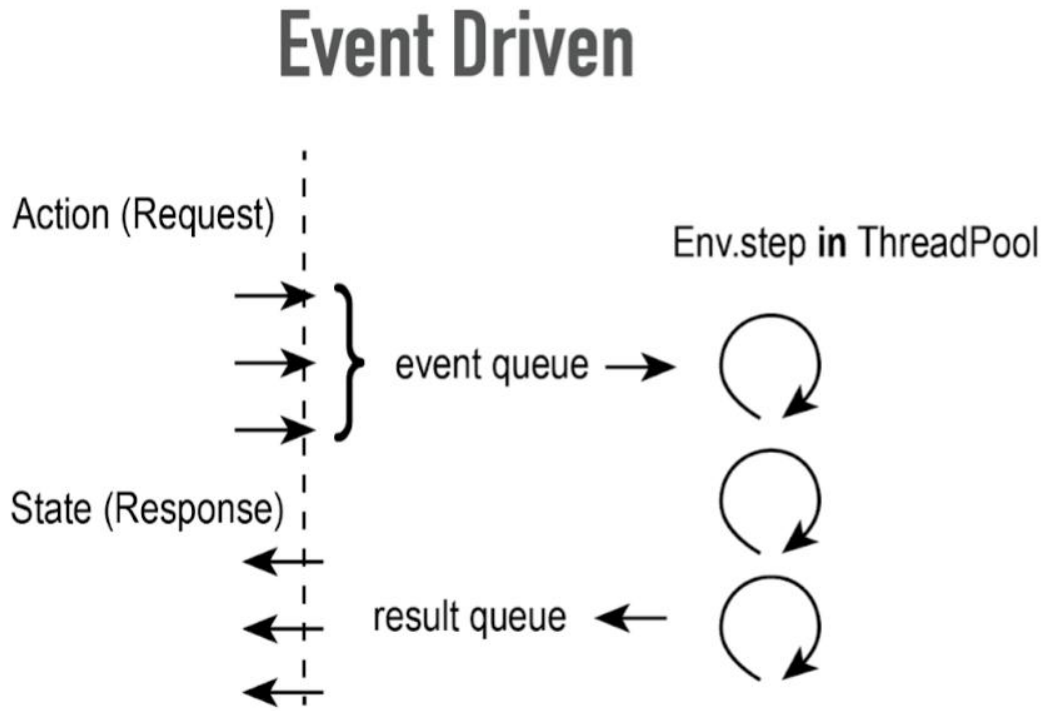


https://github.com/sail-sg/envpool

How：EnvPool内部设计与实现

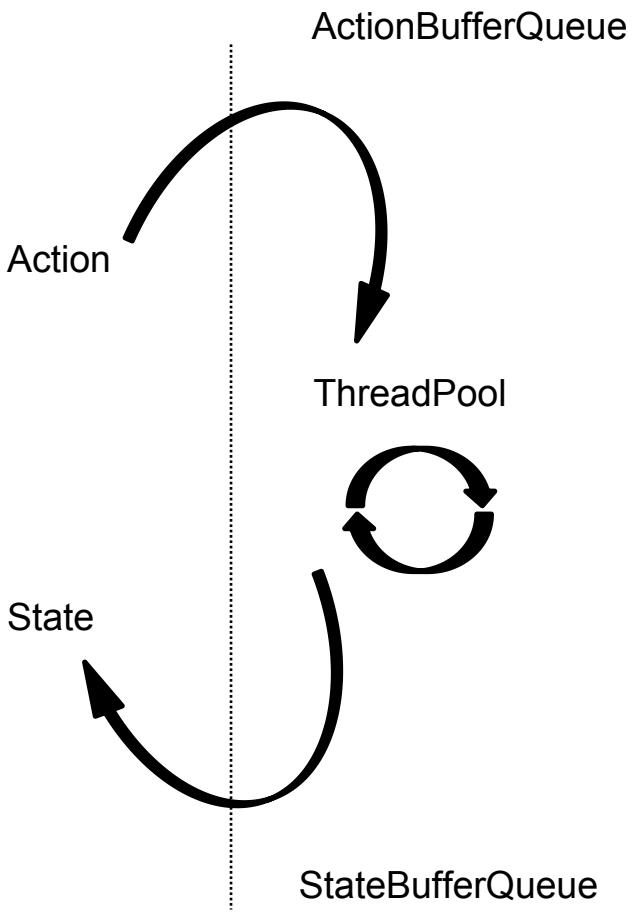
Interacting in Batch

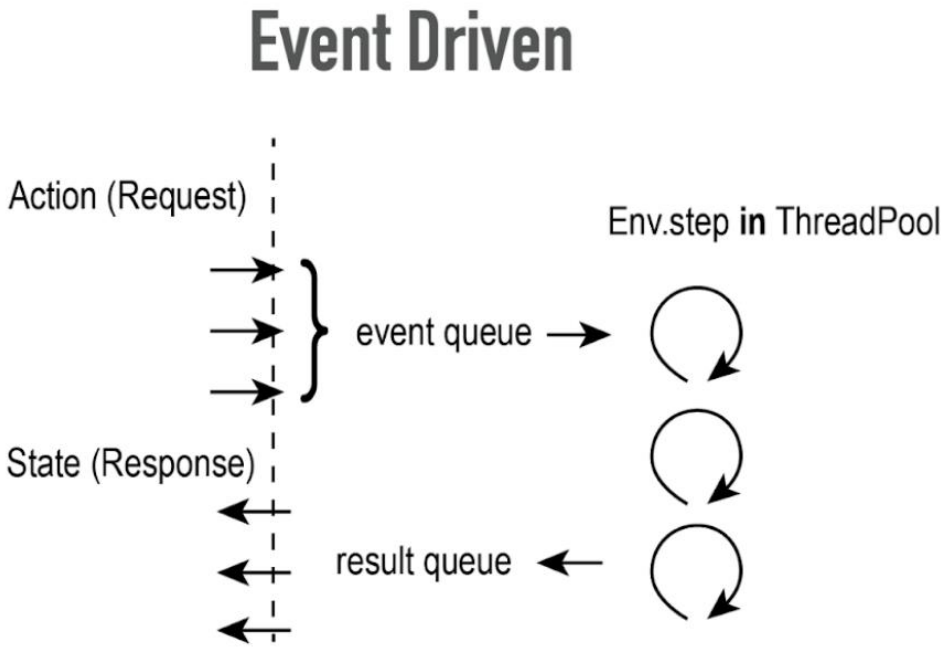
```
# Batched
env.send(batch_action)
batch_state = env.recv()
```



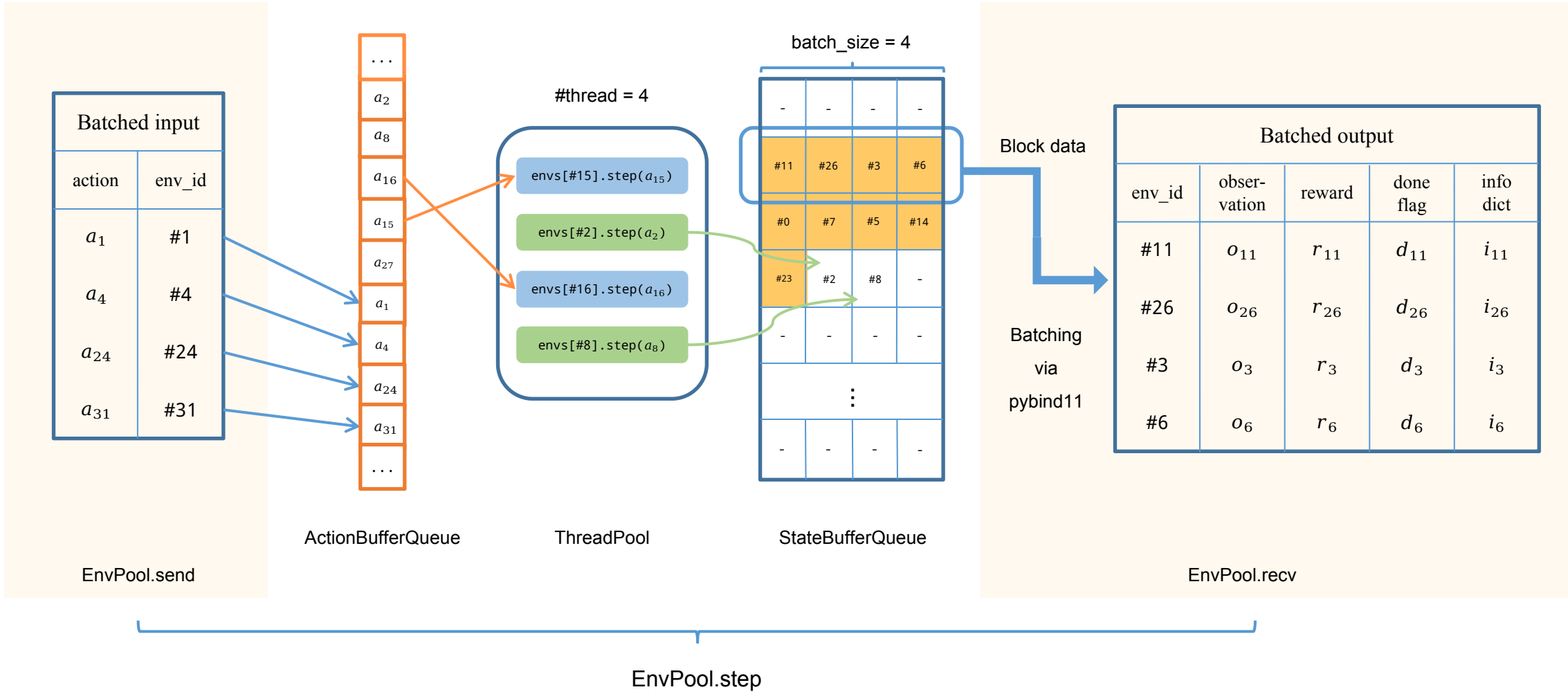
https://github.com/sail-sg/envpool

How：EnvPool内部设计与实现





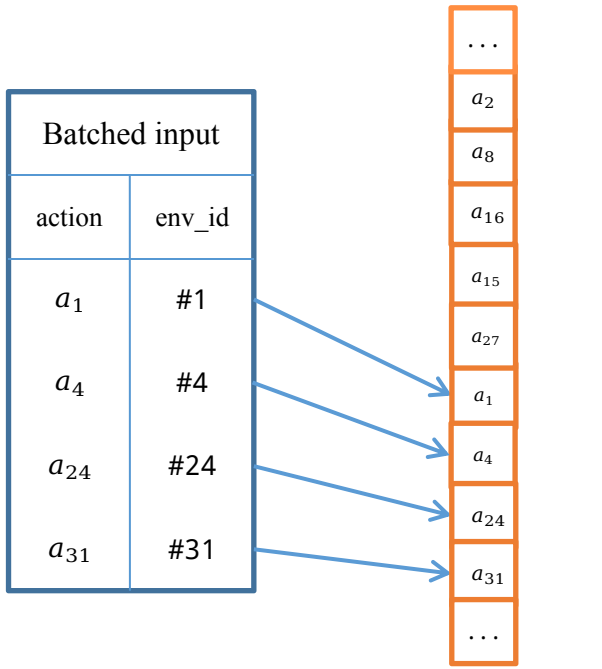
https://github.com/sail-sg/envpool



https://github.com/sail-sg/envpool

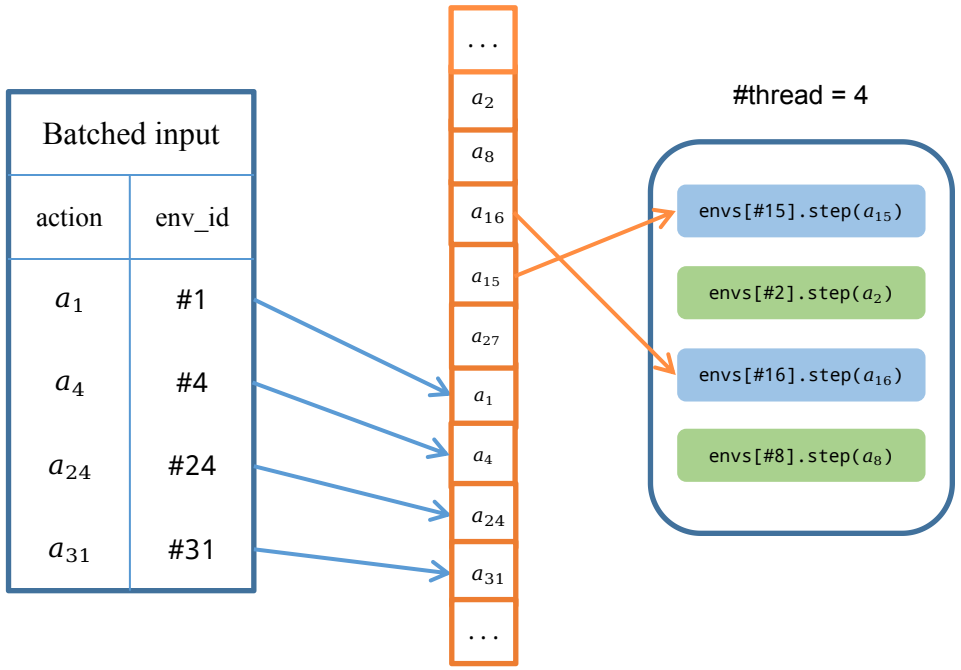
Batched input	
action	env_id
a_1	#1
a_4	#4
a_{24}	#24
a_{31}	#31

EnvPool.send

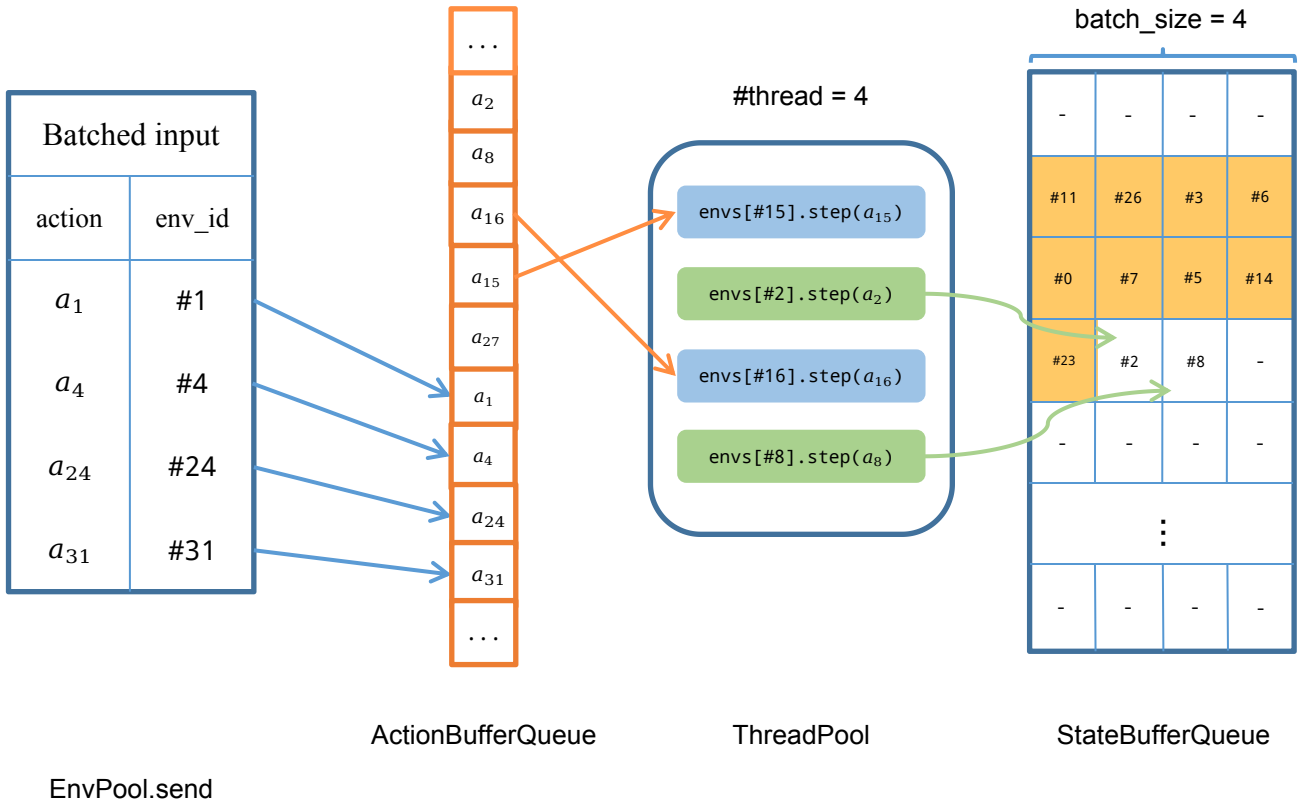


ActionBufferQueue

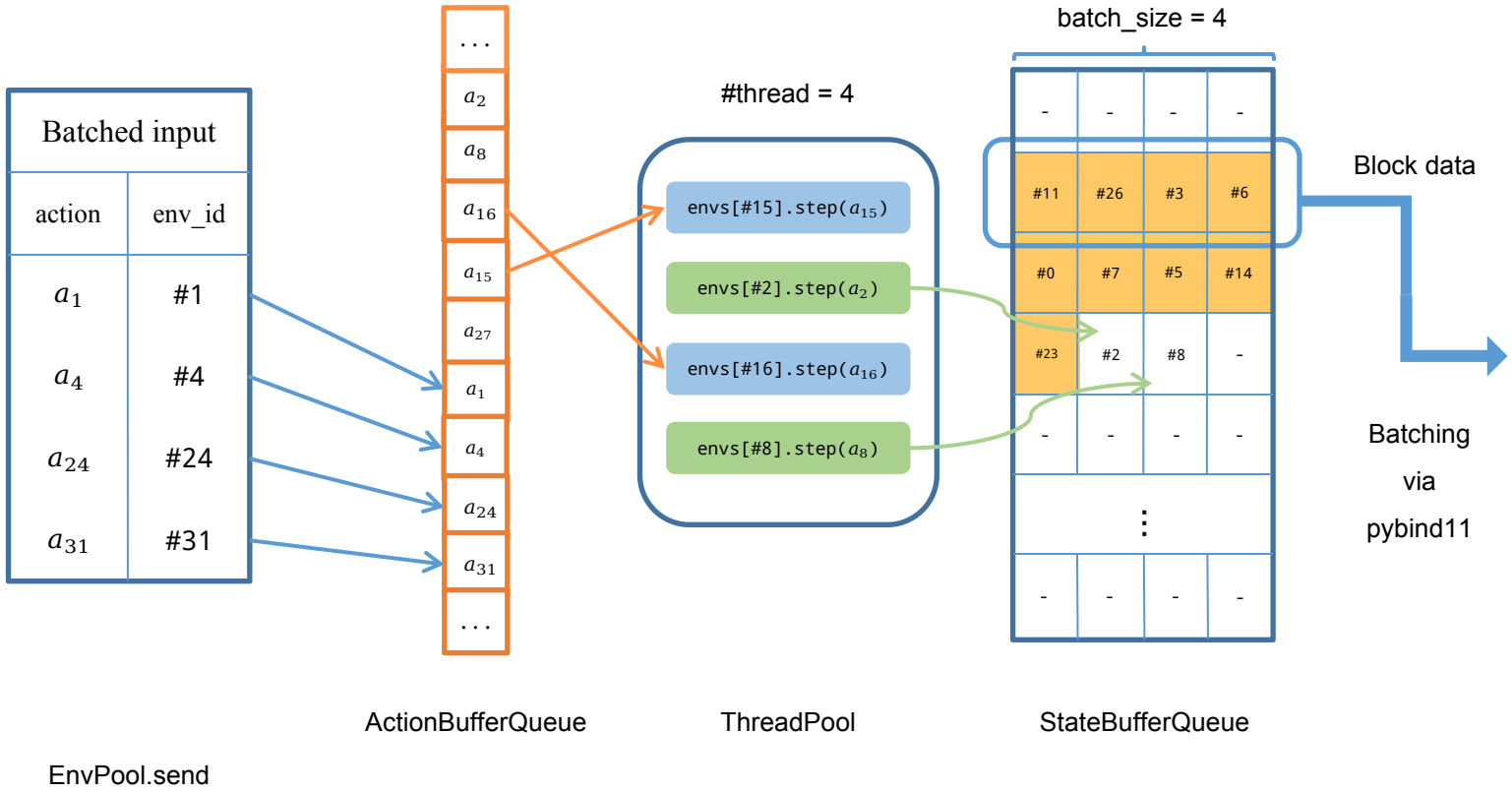
EnvPool.send



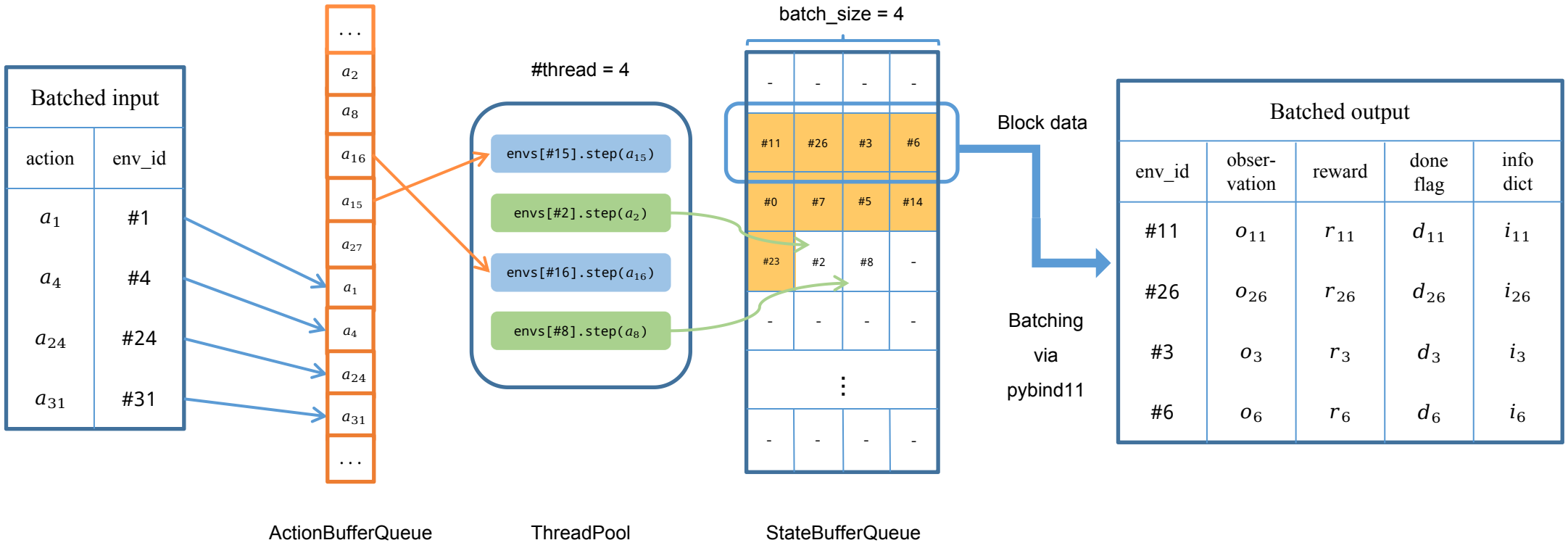
https://github.com/sail-sg/envpool



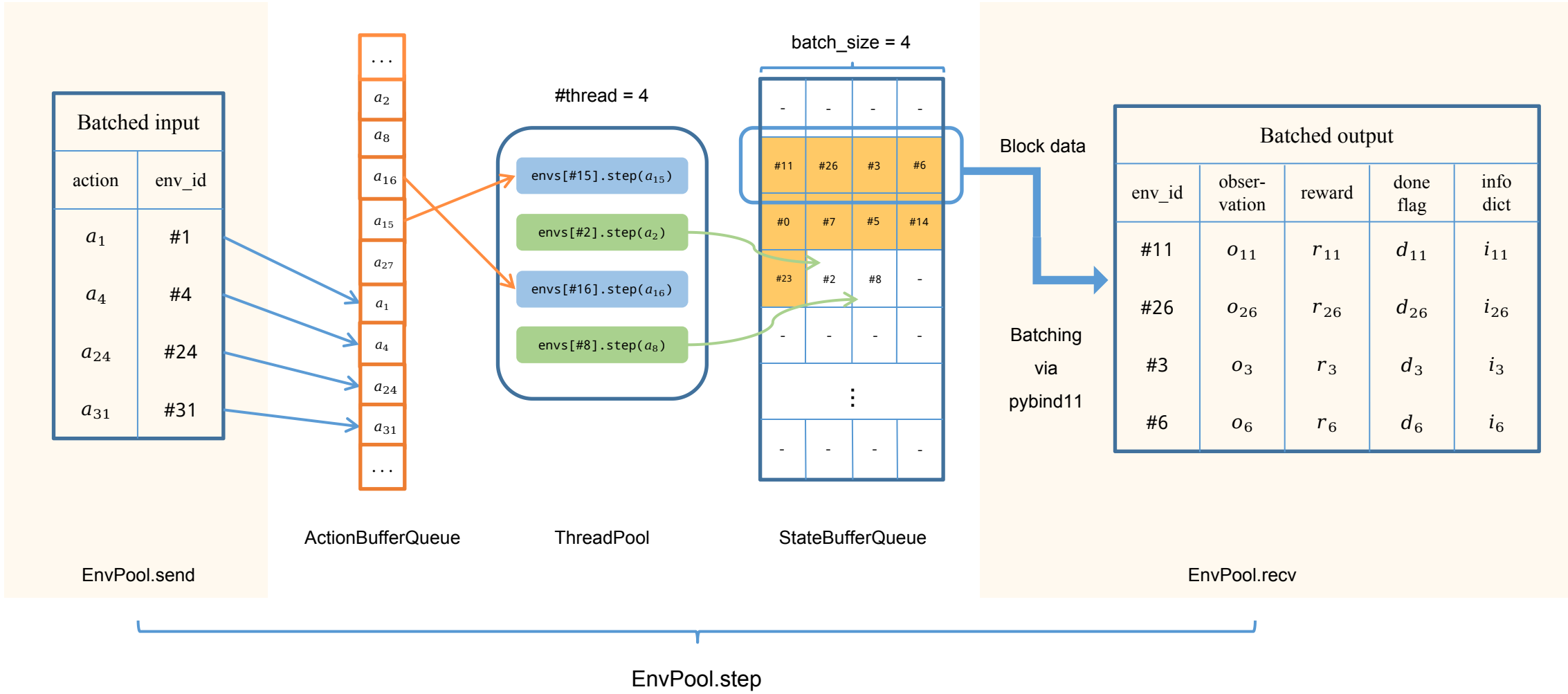
https://github.com/sail-sg/envpool



https://github.com/sail-sg/envpool



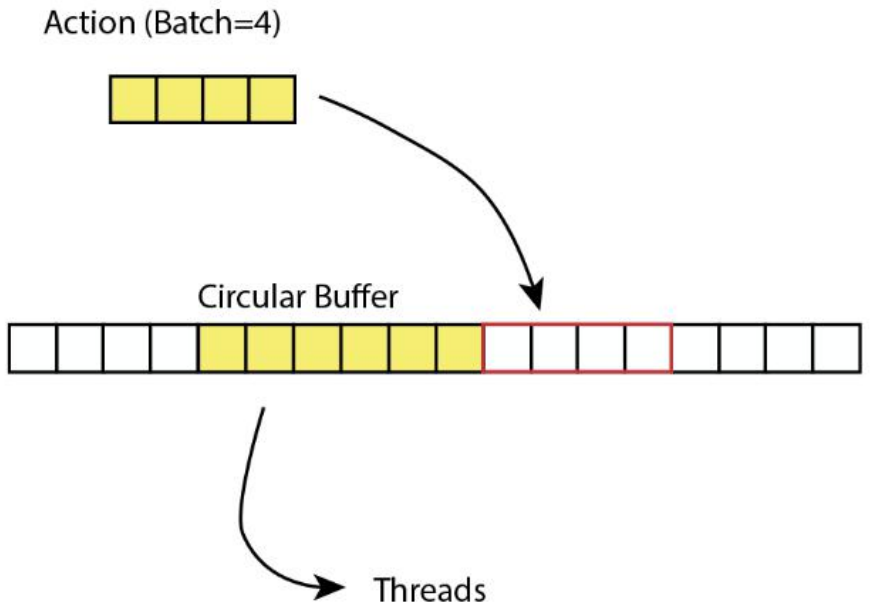
https://github.com/sail-sg/envpool



https://github.com/sail-sg/envpool

ActionBufferQueue (event queue)

- lock-free circular buffer
- 使用semaphore实现无锁



https://github.com/sail-sg/envpool

ThreadPool

- 做到无context switching

  num_threads == num_cores

  开启thread affinity绑定每个线程至固定CPU核上

- 提高ThreadPool利用率

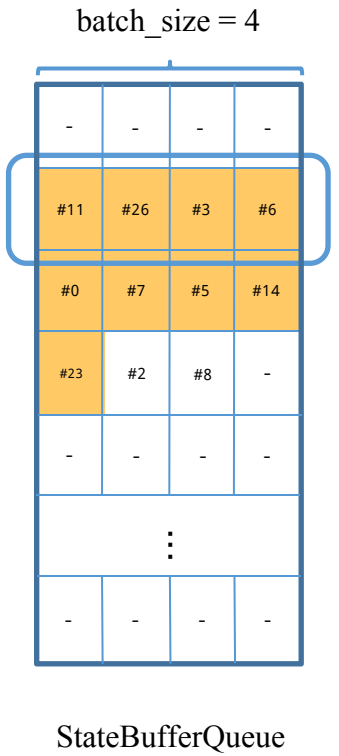
  num_envs > num_threads

  batch_size < num_envs （异步模式）

StateBufferQueue (result queue)

- lock-free circular queue with pre-allocated buffer
- 使用std::atomic与semaphore达到lock-free
- pre-allocated buffer能够不拷贝内存就能打batch传出去

- 每个slot代表一个env（multi-agent env也是如此）
- 一行是一个block，长度=batch_size，填满就能取走
- 使用atomic pointer来进行slot的分配与标记



https://github.com/sail-sg/envpool

如何本地编译？

- https://envpool.readthedocs.io/en/latest/pages/build.html

- EnvPool使用bazel进行管理
- bazel vs cmake

  bazel可以不用把源代码复制过来到repo底下，cmake比较麻烦

  第三方依赖多的时候cmake写起来比较麻烦

  跑单元测试的时候，bazel能够只跑与修改相关的test来节约时间

  ……

如何本地编译？

- https://envpool.readthedocs.io/en/latest/pages/build.html

- make bazel-build 会生成 .whl 文件在 dist/ 文件夹中，可以直接pip install
- make bazel-test 会跑测试

- make docker-dev 会自动创建一个ubuntu 20.04的docker开发环境，可以开箱即用

如何集成新环境？

- https://envpool.readthedocs.io/en/latest/pages/env.html

- 1. 继承 envpool/core/env.h 的接口，把环境接上

  推荐写一下test
- 2. 写两行 C++ template 把 env 包装成 envpool
- 3. 修改一下文件夹底下的 BUILD 文件使得 bazel 能够解析编译（类似CMakeLists.txt）

  如果有第三方依赖则需要改动 third_party/ 和 envpool/workspace0.bzl
- 4. python 注册环境，使得 envpool.make("...") 能够生成对应环境
- 5. 编译出 .whl 文件，使用 pip install 进行安装

如何使用 env.h 的接口？

- https://github.com/sail-sg/envpool/blob/master/envpool/dummy/dummy_envpool.h

- 定义Spec：Config, StateSpec, ActionSpec
- -1在SinglePlayer环境中没用，MultiPlayer表示活着的人数

- State/Action是个类似于python dict的东西，使用 state[“obs”_] 即可读写

  没看错，C++版的dict
- State/Action字典中的每个value都是Array，可以直接转换成NumPy array
- Reset() 和 Step(action) 中，使用state = Allocate(num_players) 即可申请内存写state，默认num_players=1

THANK YOU

https://github.com/sail-sg/envpool

sea connecting the dots | AI Lab