Carnegie Mellon University

Learn To Race

Distributed RL and Optimization Thrust - Final Presentation

Shijie (Brandon) Bian

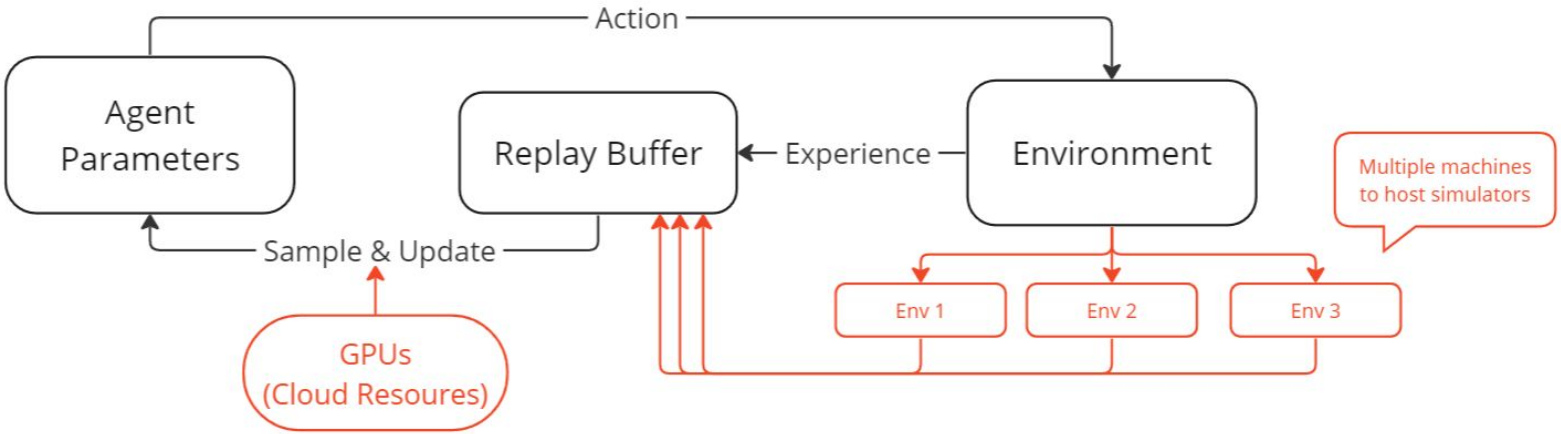
Advisors: Jon Francis, Eric Nyberg, Arav Agarwal

11-632 (Fall 2023)
Data Science Capstone

Introduction

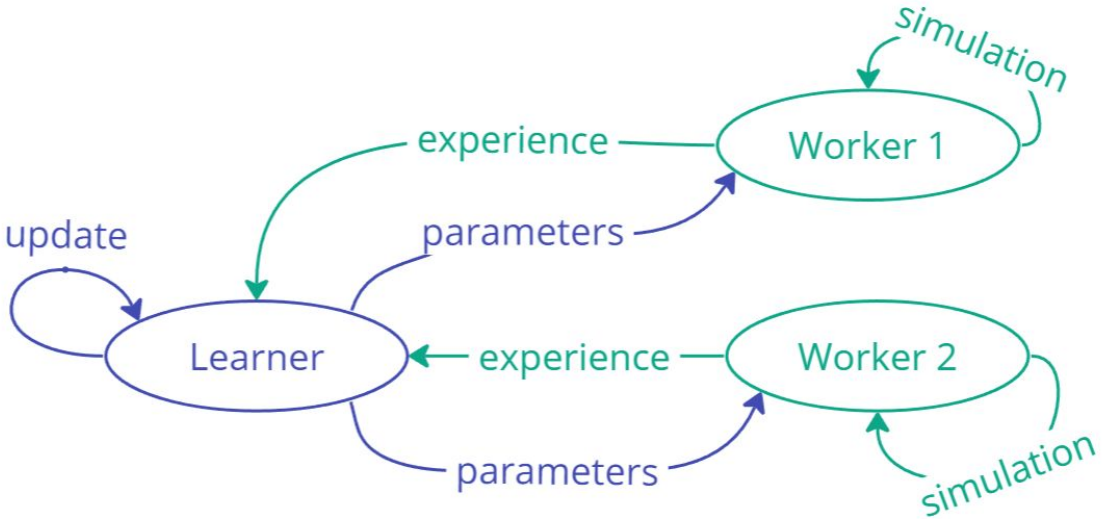
Introduction

- **Reinforcement Learning (RL)** may take days or weeks to train.
- Can we use **multiple GPUs** to accelerate training?
- Can we collect experience from **diverse environments** at the same time?



Relation to Prior Work

- **Previous iteration:** IMPALA architecture *[1]*
- **Learner**: performs parameter update
- **Worker (i.e., Actor)**: performs simulation and experience collection

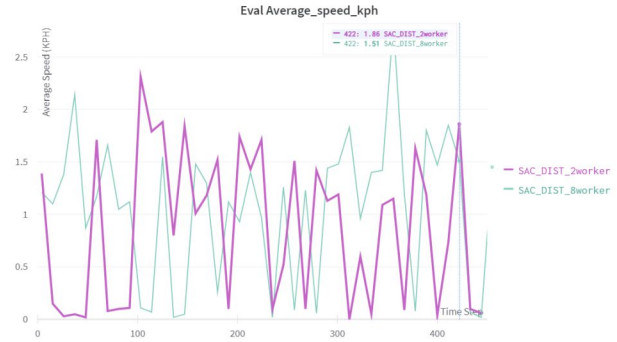


 [1] https://arxiv.org/abs/1802.01561

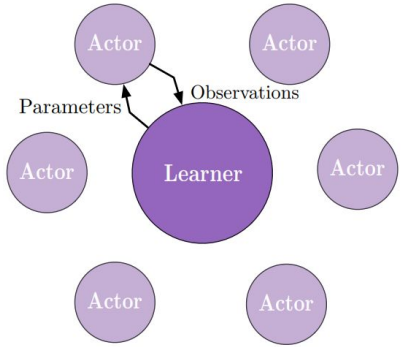
Problem Description

Problem Description

1. Performance does not scale



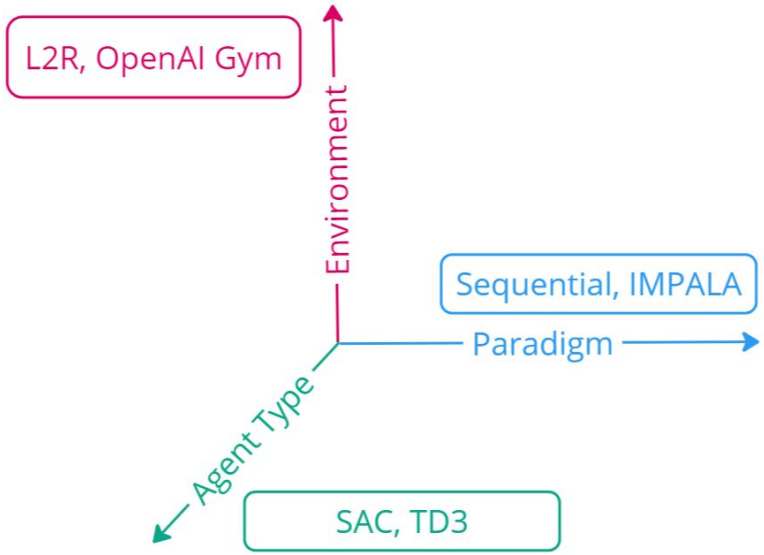
2. Learner overwhelmed



[1]

 [1] https://arxiv.org/abs/1802.01561

Problem Description



3. Convoluted control

- Overwhelming Kubernetes templates
- Manual creation / deletion of resource
- Branched code base
- Environment conflicts
- Difficult implementation of new modules

Proposed Solution

Solution **Distributed Training Processes & Resource Control Optimizations**	
Challenge	**Solution**
• Performance not scaling	• Utilizing OpenAI Gym environments and SAC implementations as baselines
• Learner node being overwhelmed	• Distribute training tasks to worker nodes
• Convoluted control	• Automated and unified resource control and deletion

Hypothesis

1. Training efficiency not scaling

→ Learner being a bottleneck

→ Agent and parameter configurations are not suitable for the task

2. OpenAI's environment and agent configuration

→ Serves as easier baselines to test the distributed system architecture

3. Training tasks can be allocated to workers

→ Alleviate workload on learner

4. Modularization of the paradigms

→ Helpful for trying out combinations of paradigms and RL environment

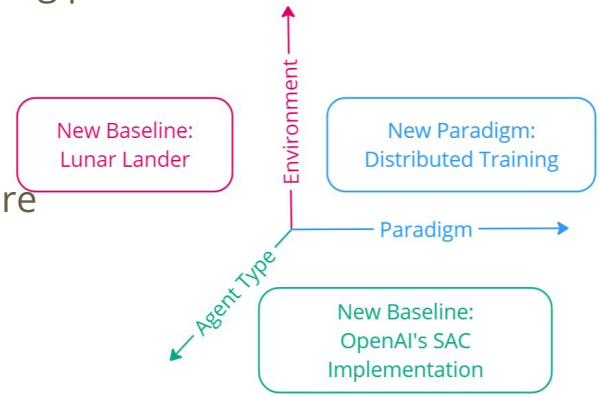
Features and Functionality

1. New distributed training paradigm

- Training tasks allocated to workers
- Learner responsible for controlling worker and distributing parameters

2. New baselines

- OpenAI Spinning-Up SAC agent and replay buffers
- OpenAI Lunar Lander environment
- Full compatibility with the distributed system architecture



3. Automated resource control

- Unified modules and Kubernetes templates
- Resource launching and shutdown based on user input
- Additional logging for both learner and worker across paradigms

4. Modularization

- Easier plug-in and implementation of new Paradigms / Agents / Environments

Resource Requirement

Data

- Arrival simulator: visual data input for encoding environment state
- Training and evaluation metrics: such as reward of agent, and training step time (simulation time)

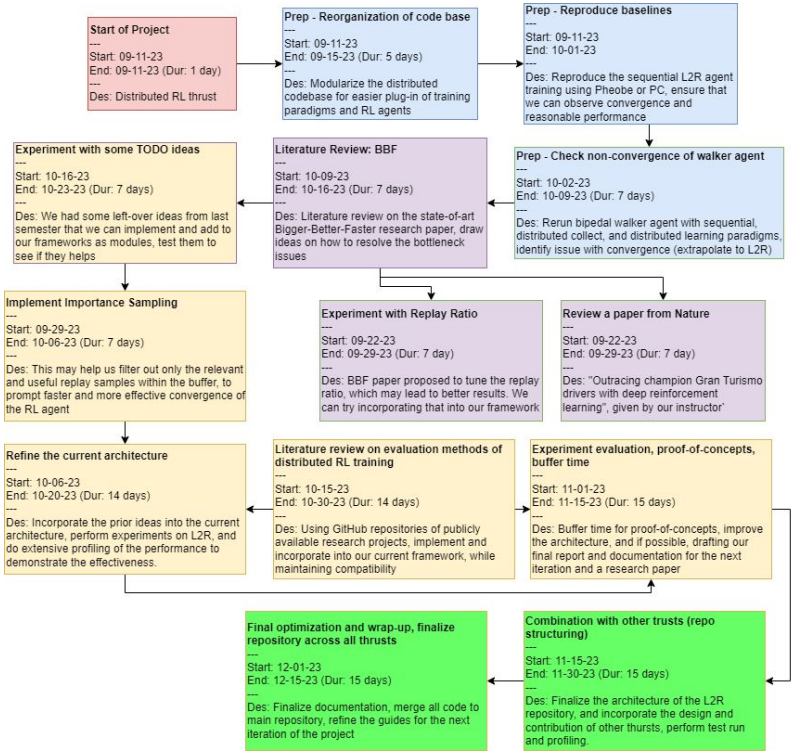
Hardware

- Phoebe clusters: Nvidia GPU (> 10 GB memory) and CPU (> 8 cores) support, allowing multi-pod allocation
- Cloud resource: AWS EC2 (g4dn.2xlarge) is used for system framework debugging purposes

Software

- Pytorch: for model training with GPU
- Kubernetes: for resource creation and control
- WandB: for metrics logging and visualization
- Mamba/Conda: for automated environment creation

Plan


Start of Project

Start: 09-11-23
End: 09-11-23 (Dur: 1 day)

Des: Distributed RL thrust
Prep - Reorganization of code base

Start: 09-11-23
End: 09-15-23 (Dur: 5 days)

Des: Modularize the distributed codebase for easier plug-in of training paradigms and RL agents
Prep - Reproduce baselines

Start: 09-11-23
End: 10-01-23

Des: Reproduce the sequential L2R agent training using Pheobe or PC, ensure that we can observe convergence and reasonable performance
Prep - Check non-convergence of walker agent

Start: 10-02-23
End: 10-09-23 (Dur: 7 days)

Des: Rerun bipedal walker agent with sequential, distributed collect, and distributed learning paradigms, identify issue with convergence (extrapolate to L2R)
Literature Review: BBF

Start: 10-09-23
End: 10-16-23 (Dur: 7 days)

Des: Literature review on the state-of-art Bigger-Better-Faster research paper, draw ideas on how to resolve the bottleneck issues
Experiment with some TODO ideas

Start: 10-16-23
End: 10-23-23 (Dur: 7 days)

Des: We had some left-over ideas from last semester that we can implement and add to our frameworks as modules, test them to see if they helps
Implement Importance Sampling

Start: 09-29-23
End: 10-06-23 (Dur: 7 days)

Des: This may help us filter out only the relevant and useful replay samples within the buffer, to prompt faster and more effective convergence of the RL agent
Experiment with Replay Ratio

Start: 09-22-23
End: 09-29-23 (Dur: 7 day)

Des: BBF paper proposed to tune the replay ratio, which may lead to better results. We can try incorporating that into our framework
Review a paper from Nature

Start: 09-22-23
End: 09-29-23 (Dur: 7 day)

Des: "Outracing champion Gran Turismo drivers with deep reinforcement learning", given by our instructor`
Refine the current architecture

Start: 10-06-23
End: 10-20-23 (Dur: 14 days)

Des: Incorporate the prior ideas into the current architecture, perform experiments on L2R, and do extensive profiling of the performance to demonstrate the effectiveness.
Literature review on evaluation methods of distributed RL training
---
Start: 10-15-23
End: 10-30-23 (Dur: 14 days)

Des: Using GitHub repositories of publicly available research projects, implement and incorporate into our current framework, while maintaining compatibility
Experiment evaluation, proof-of-concepts, buffer time

Start: 11-01-23
End: 11-15-23 (Dur: 15 days)

Des: Buffer time for proof-of-concepts, improve the architecture, and if possible, drafting our final report and documentation for the next iteration and a research paper
Final optimization and wrap-up, finalize repository across all thrusts
---
Start: 12-01-23
End: 12-15-23 (Dur: 15 days)

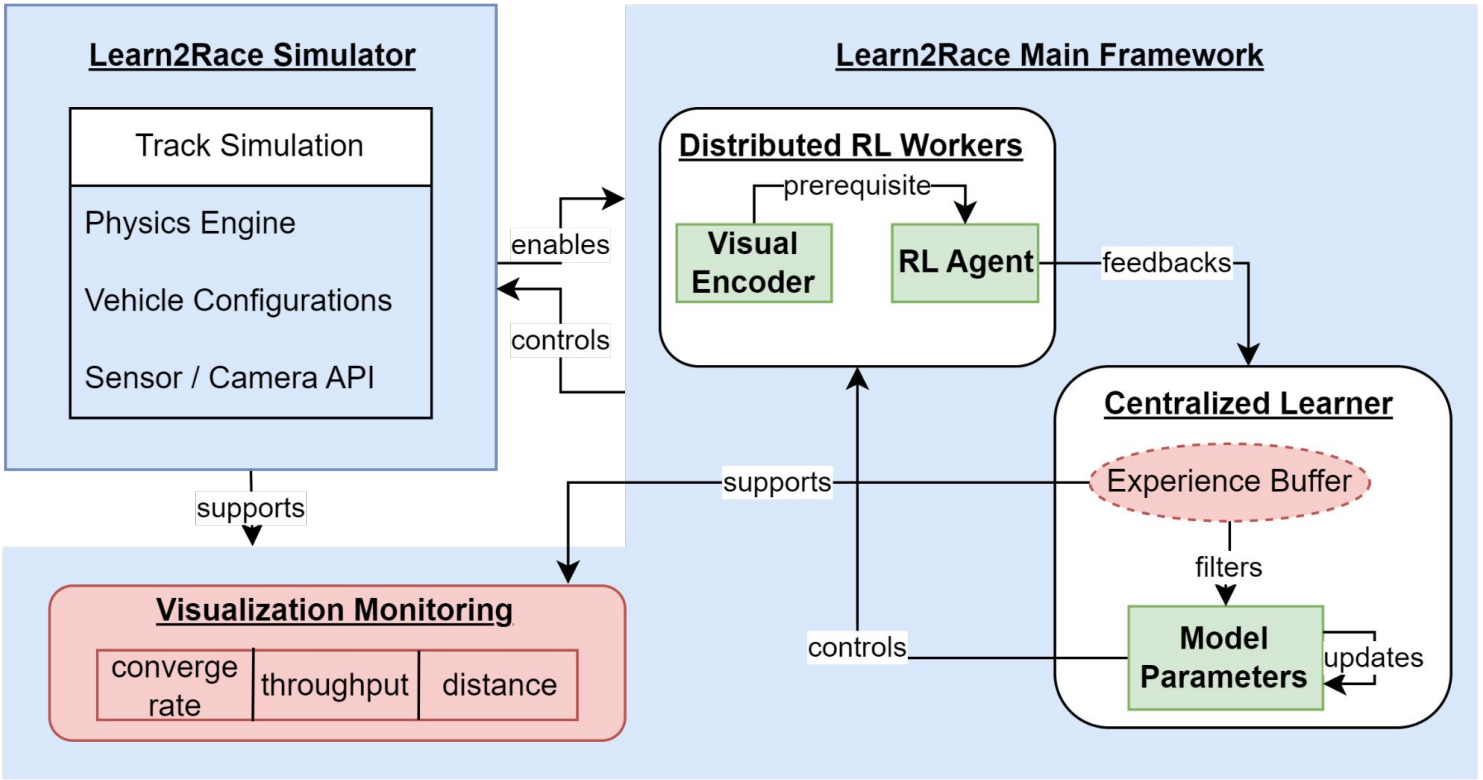
Des: Finalize documentation, merge all code to main repository, refine the guides for the next iteration of the project
Combination with other trusts (repo structuring)

Start: 11-15-23
End: 11-30-23 (Dur: 15 days)

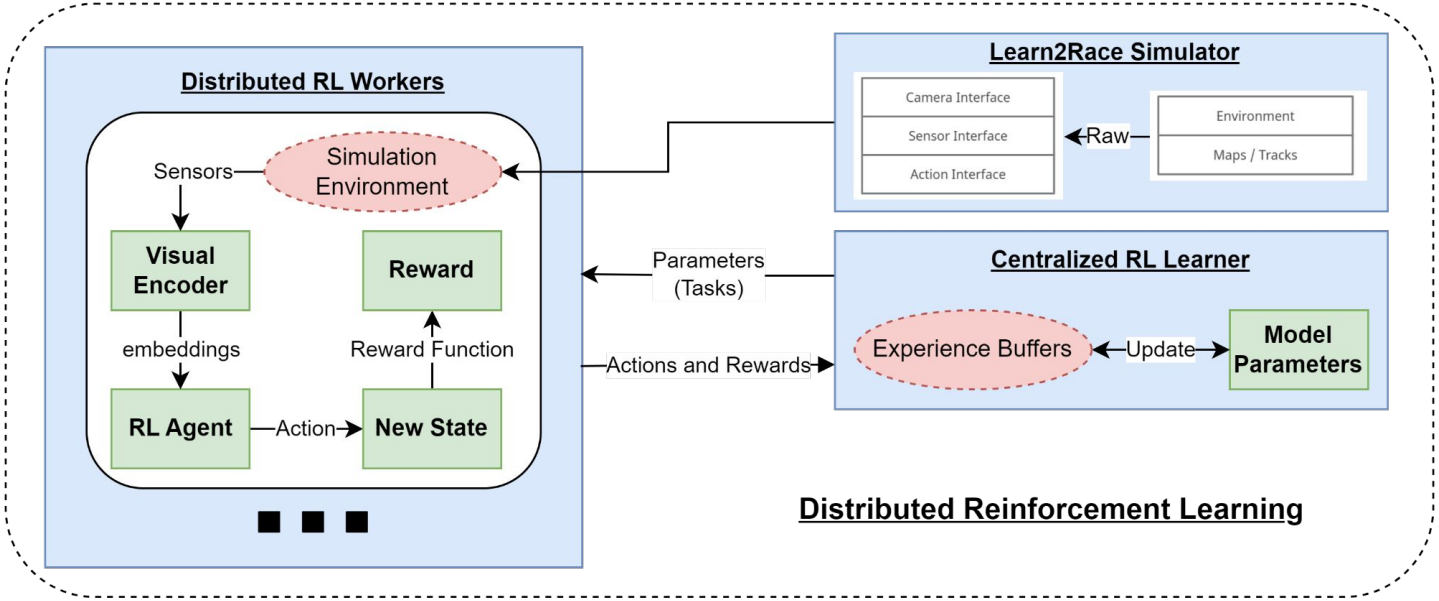
Des: Finalize the architecture of the L2R repository, and incorporate the design and contribution of other thursts, perform test run and profiling.

Design - Overview

Learn-to-Race Context

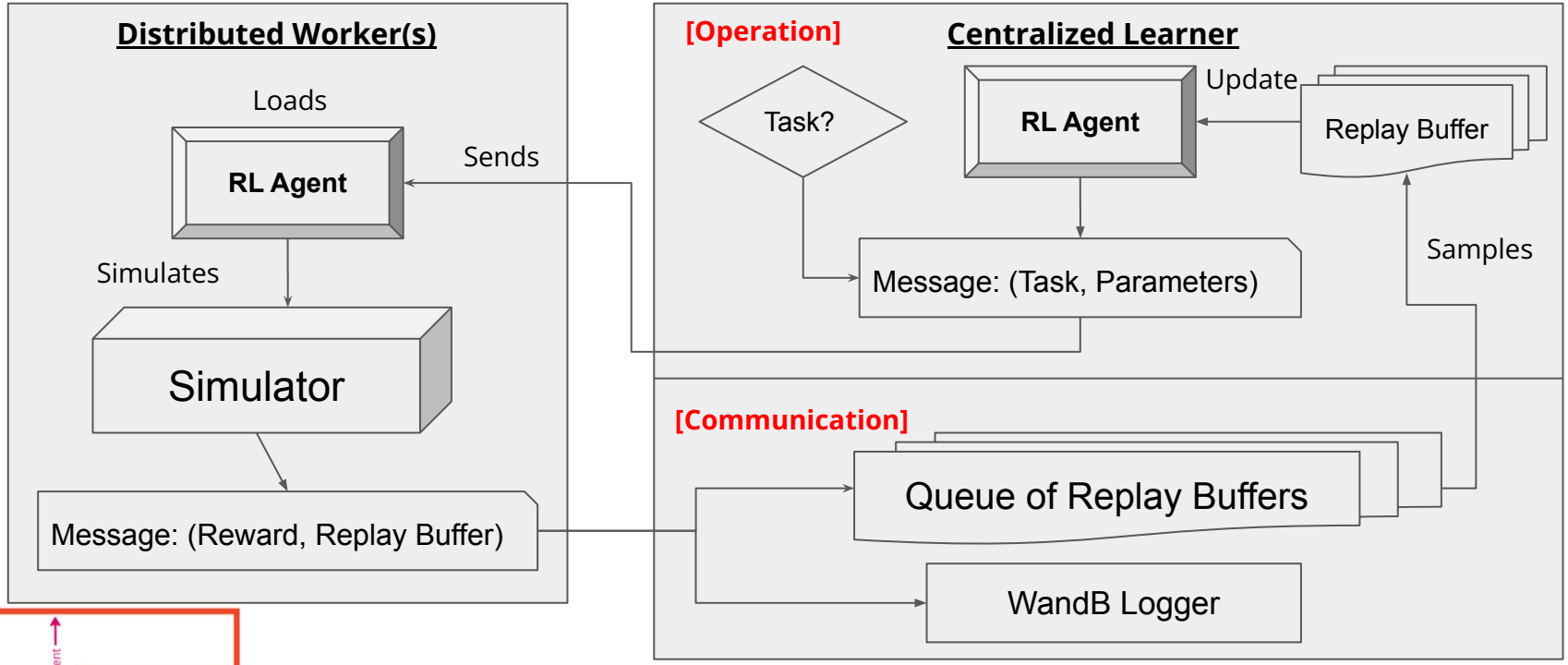


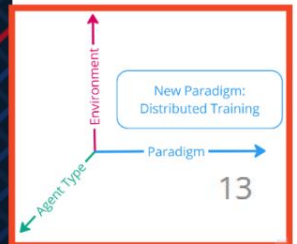
Paradigm - High-level Design



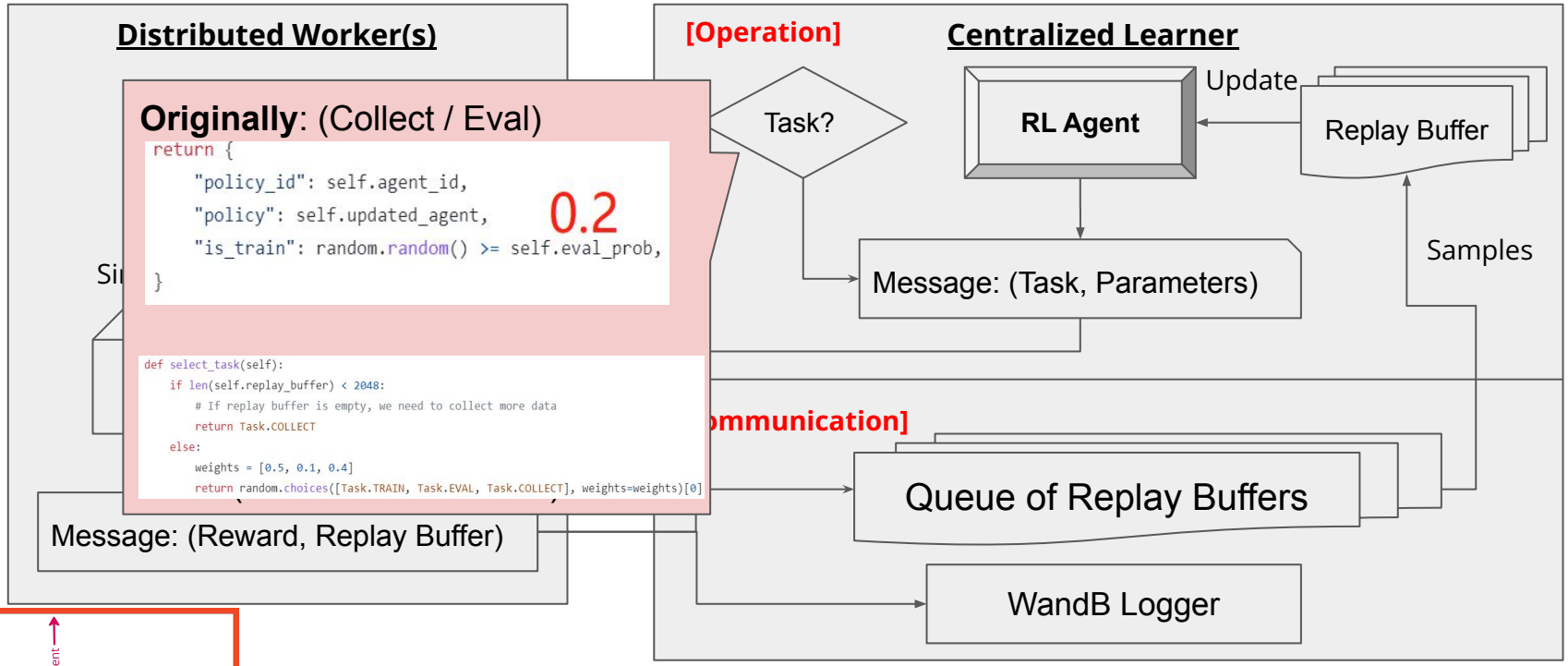
Design - Training Paradigms

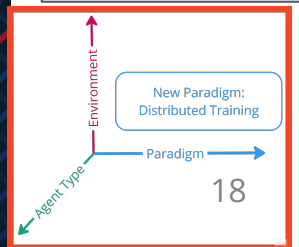
Paradigm - Distributed Training / Update





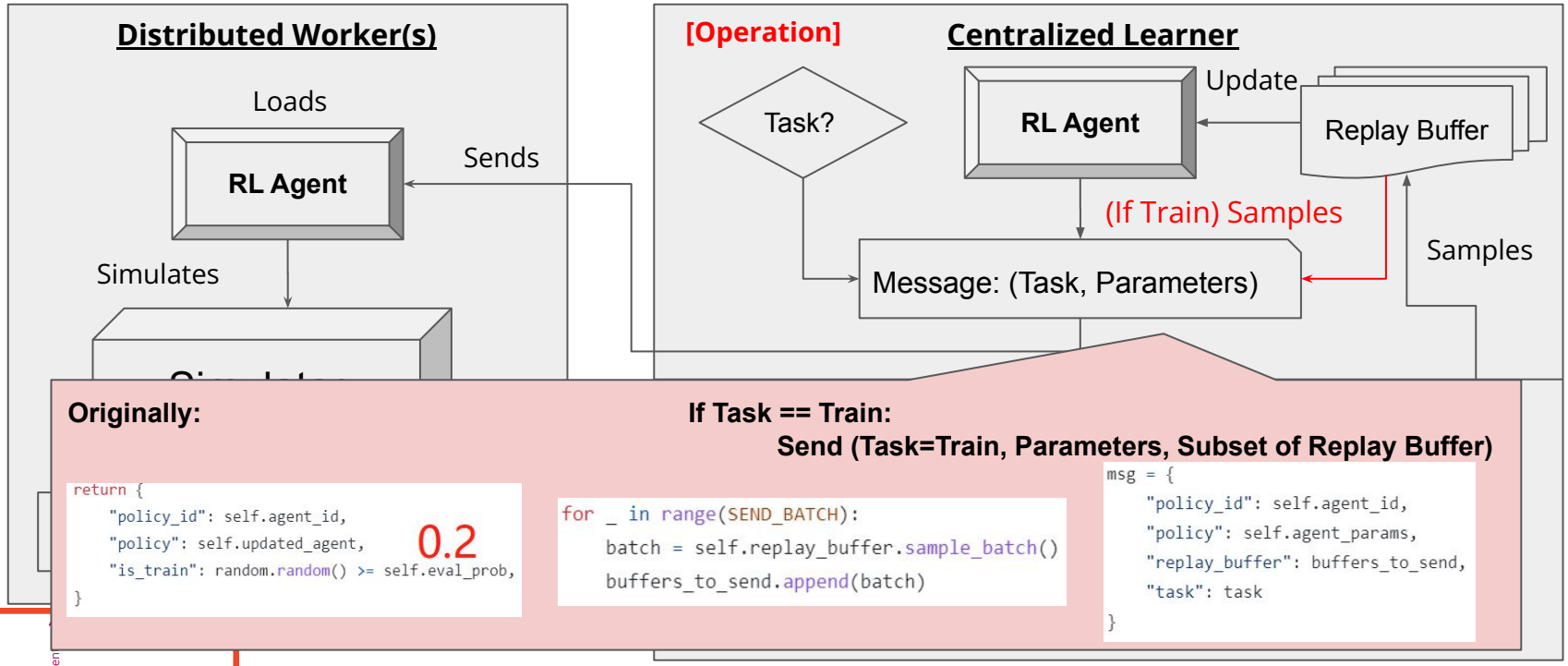
Paradigm - Distributed Training / Update

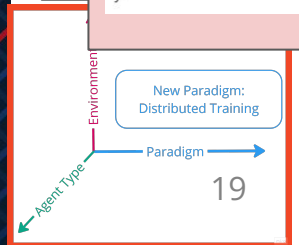




(1) Allocating training task to workers

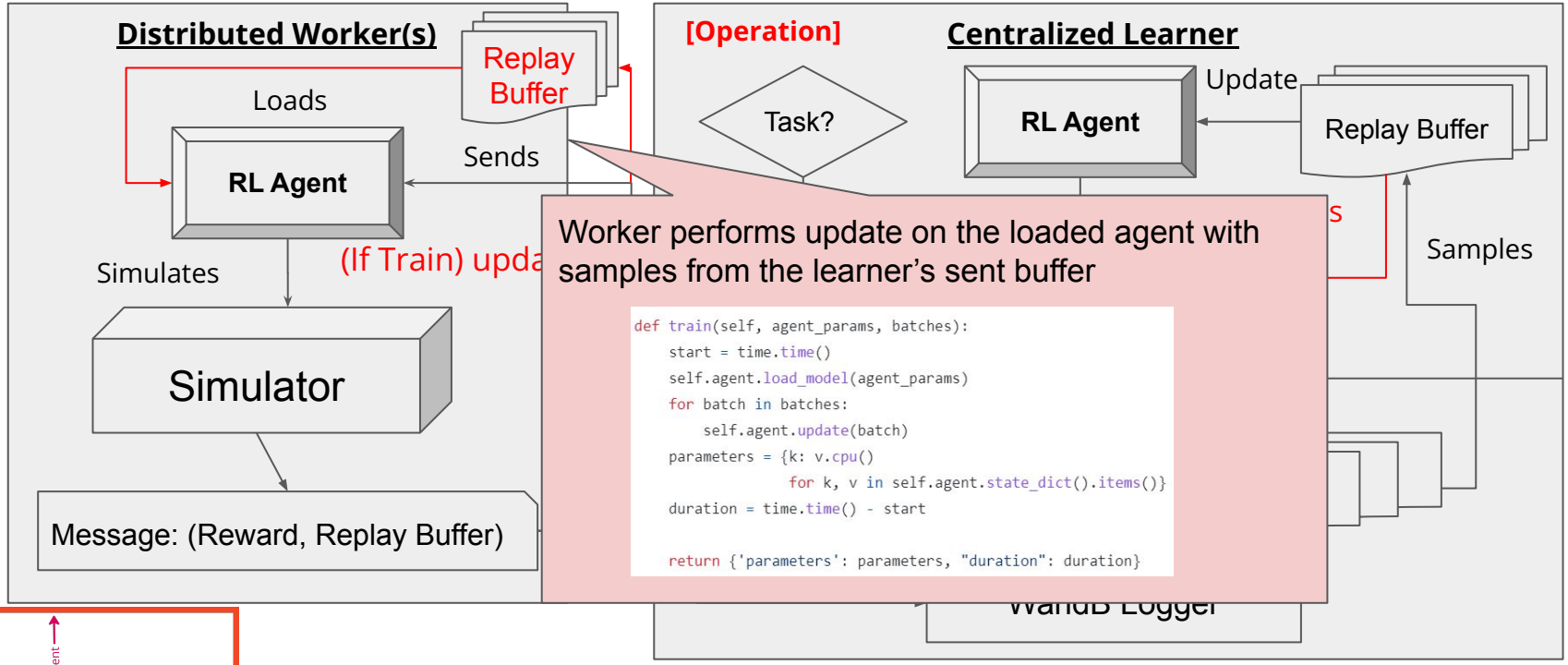
Paradigm - Distributed Training / Update

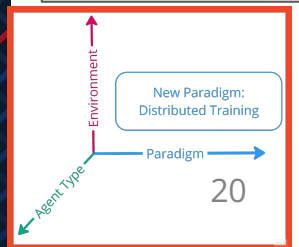




(2) For training tasks, send samples from learner's buffer

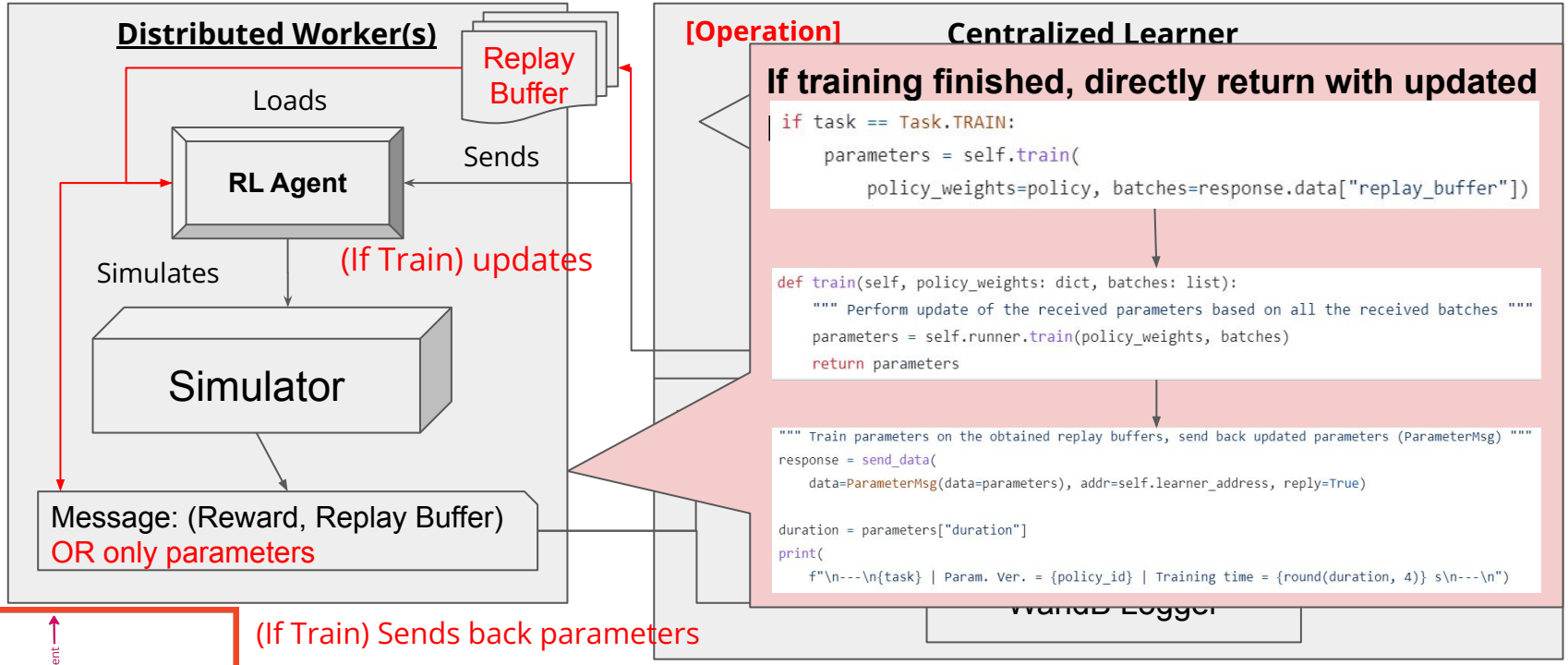
Paradigm - Distributed Training / Update

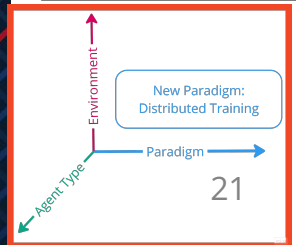




(3) Worker keeps local replay buffer for training local agent

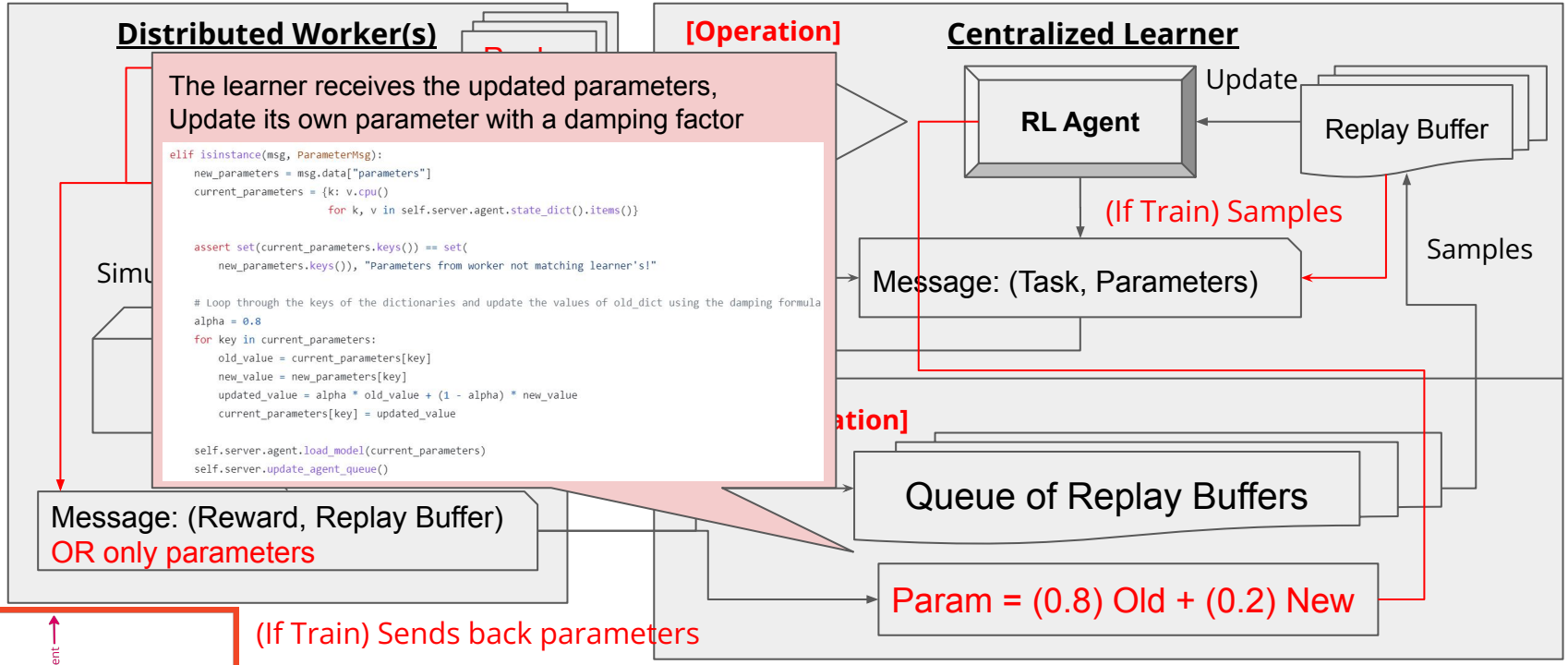
Paradigm - Distributed Training / Update

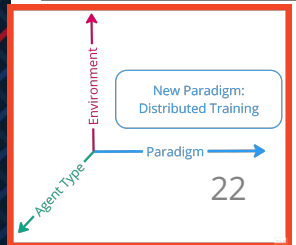




(4) Worker sends back updated parameters without simulation

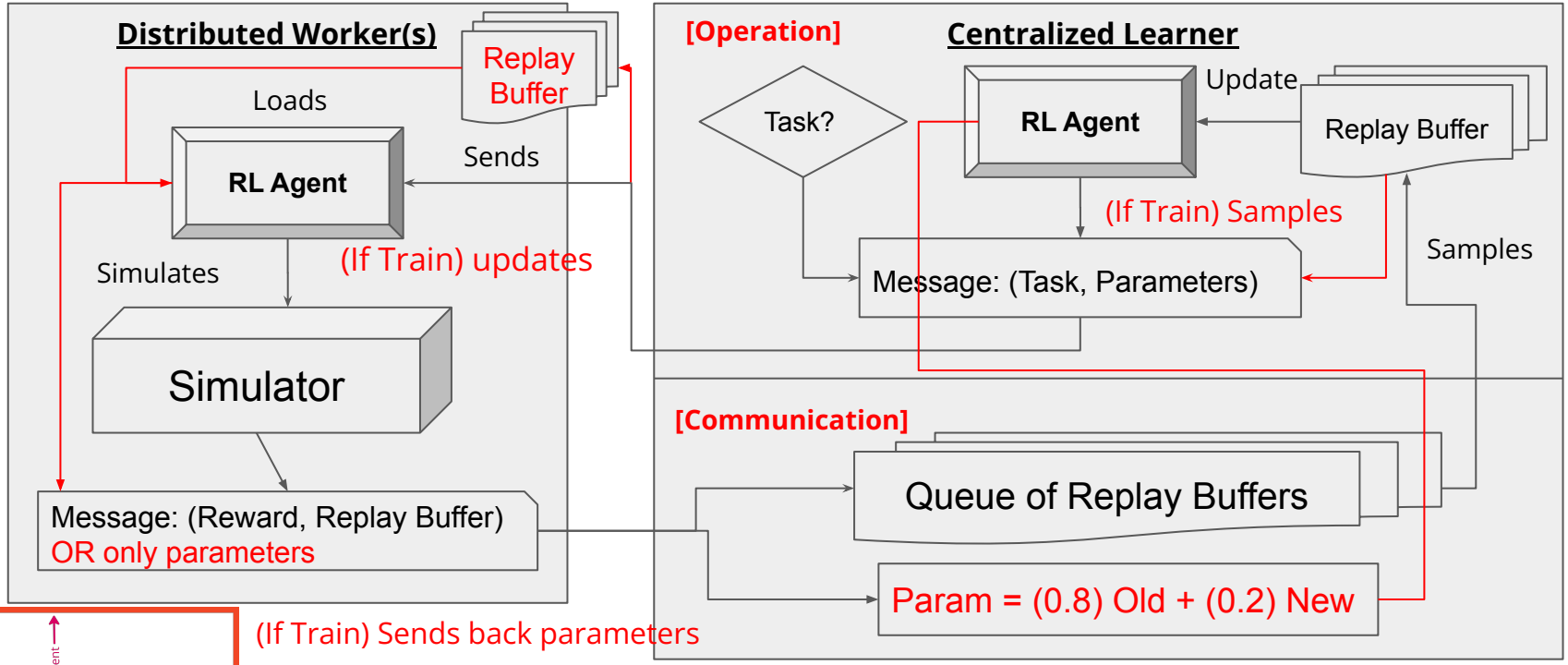
Paradigm - Distributed Training / Update

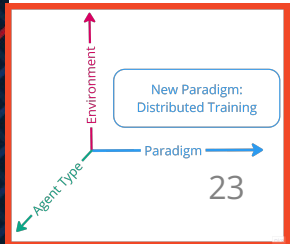




(5) Weighted update of the learner's agent weights

Paradigm - Distributed Training / Update





Design - Agent and Environments

RL Agent Design - OpenAI's SAC

- Motivated by non-convergence of bipedal walker from existing approach
- Adopted OpenAI Spinning-Up *[1]* implementation of SAC agent and replay buffer

```
class MLPActorCritic():
    # Input: Encoded observation and action embedding
    # Output: Recommended action, Log prob. of action w.r.t policy distribution
    self.pi = (
        net_layer = MLP((obs_dim + act_dim) -> 256 -> 256),
        mu_layer = Linear(256 -> act_dim),
        log_std_layer = Linear(256 -> act_dim),
        pi_distribution = Normal(mu, log),
        pi_action = pi_distribution.sample(),
        logp_pi = pi_distribution.log_prob(pi_action)
    )

    # Input: Encoded observation and action embedding
    # Output: Scalar Q-value (quality of action in current state)
    self.q1 = self.q2 = MLP((obs_dim + act_dim) -> 256 -> 256 -> 1)
```

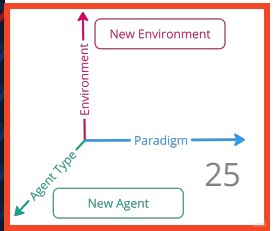
Policy-Network: learns to output the probability distribution of actions over current state, for action selection

Q-value Network: learns to output the quality of taking certain actions based on current state

miro

(Pseudocode)

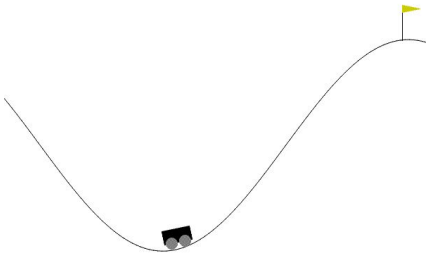
Two Q-networks for stability and robustness
(target Q-net periodically synced using weighted update of parameters)



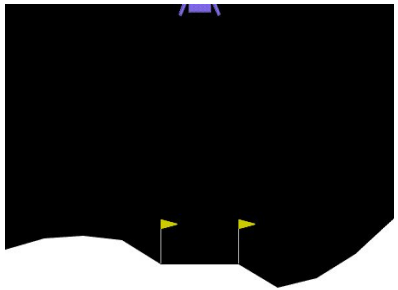
[1] https://github.com/openai/spinningup

Environment Design - OpenAI's Gym

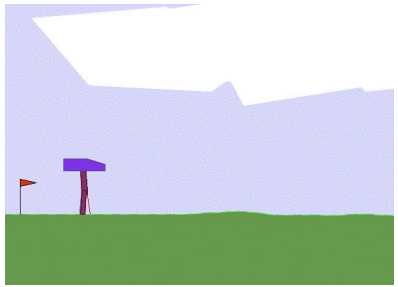
- Compatibility with OpenAI's Box-2d Continuous environments
- Newly added Lunar Lander environment
- Baselines for evaluating distributed paradigm



Mountain Car

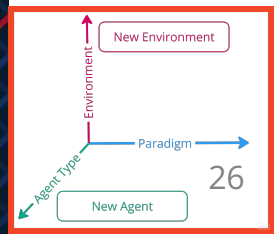


Lunar Lander



Bipedal Walker

Increasing Difficulty



Parameters Design - Hugging Face

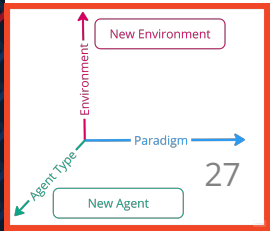
- Reinforcement Learning is prone to **hyperparameter initialization**
- For OpenAI's Gym environment's hyperparameter design
  - **Stable-Baselines3** *[1]*
  - **Spinning-Up** *[2]*

Environment Id	Observation Space	Action Space	Reward Range	tStepL	Trials	rThresh
MountainCar-v0	Box(2,)	Discrete(3)	(-inf, inf)	200	100	-110.0
MountainCarContinuous-v0	Box(2,)	Box(1,)	(-inf, inf)	999	100	90.0
Pendulum-v0	Box(3,)	Box(1,)	(-inf, inf)	200	100	None
CartPole-v0	Box(4,)	Discrete(2)	(-inf, inf)	200	100	195.0

sb3/sac-MountainCarContinuous-v0
Reinforcement Learning • Updated Oct 11, 2022 • 9

sb3/sac-BipedalWalker-v3
Reinforcement Learning • Updated Oct 11, 2022 •

sb3/sac-LunarLanderContinuous-v2
Reinforcement Learning • Updated Oct 11, 2022 •



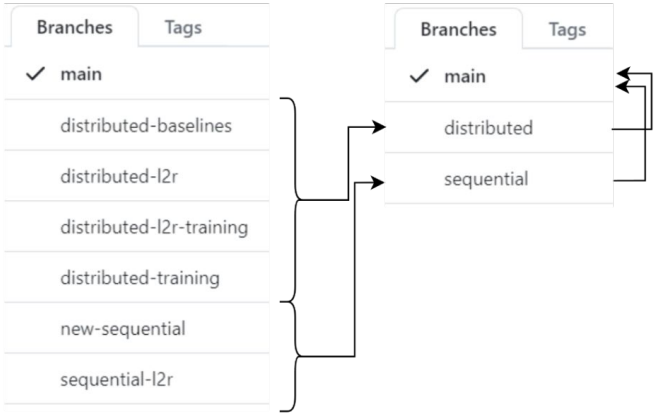
[1] https://github.com/DLR-RM/stable-baselines3

[2] https://github.com/openai/spinningup

Design - Optimizations

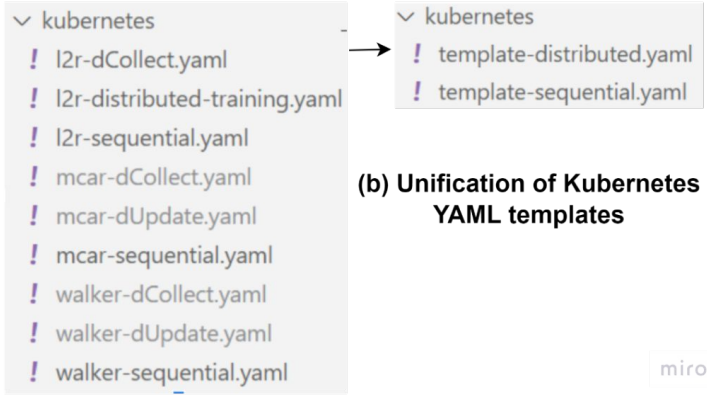
Optimization - Unifications / Encapsulation / Modularity

- One branch for all possible paradigm-environment configurations

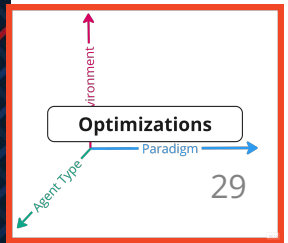


(a) Unification of branches

- Automatic generation of Kubernetes YAML files
- Changes made to template can propagate to configurations

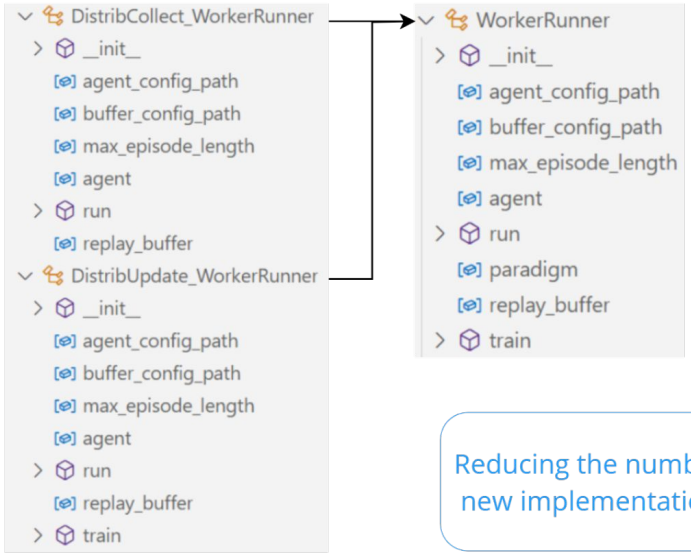


(b) Unification of Kubernetes YAML templates

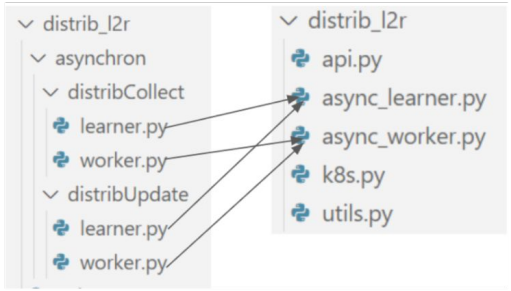


Optimization - Unifications / Encapsulation / Modularity

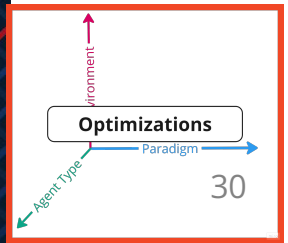
(c) Unification of Worker Runner class



(d) Unification of learner and worker definitions

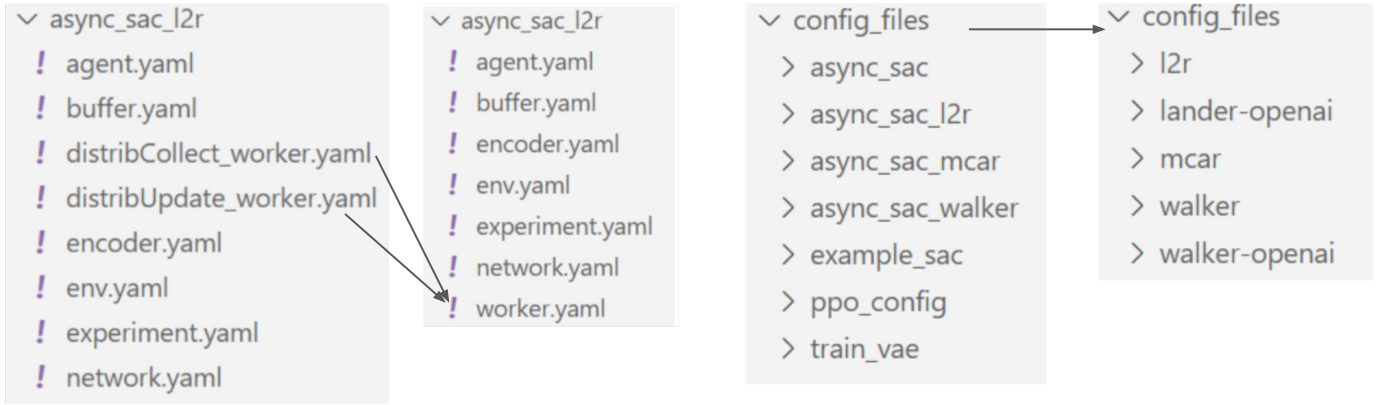


Reducing the number of classes and files, allowing users to add new implementations conforming to the base class definitions



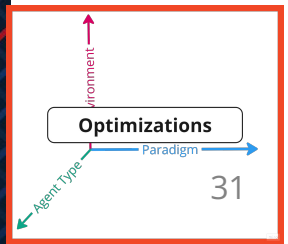
Optimization - Unifications / Encapsulation / Modularity

(e) Unification of worker configuration YAML



One config file per environment, instead of per environment-agent-paradigm configuration

miro



Optimization - Automatic Resource Control

- **Resource launching**: automatically create, store, and launch Kubernetes YAML files based on user's input configurations. Naming of resources descriptive and unique to avoid conflict.

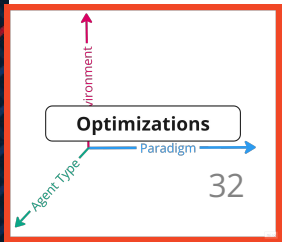
```
sbian@phoebe-login:~/distributed$ python3 launch.py
Select RL environment (l2r/mcar/walker/walker-openai/lander-openai): lander-openai
Select training paradigm (sequential/dCollect/dUpdate): dUpdate
Input number of distributed workers: 3
Input WandB experiment name: lander-openai-dupdate-3workers
----------
RL Environment = [lander-openai] | Training Paradigm = [dUpdate] | Number of Workers = [3] | Experiment Name = [lander-openai-dupdate-3workers] ---
----------
[Warning] Port name exceeded 15 characters, renaming: lander-openai-dupdate -> lander-open-z9x
replicaset.apps/lander-openai-dupdate-workers created
pod/lander-openai-dupdate-learner created
service/lander-openai-dupdate-learner created
```

Resource Launching Script

- **Resource shutdown**: automatically locate and destroy the resources according to user input configuration.

```
sbian@phoebe-login:~/distributed$ python3 shutdown.py
Select RL environment (l2r/mcar/walker/walker-openai/lander-openai): lander-openai
Select training paradigm (sequential/dCollect/dUpdate): dUpdate
---
--- Shutting down [lander-openai] with training paradigm [dUpdate] ---
---
replicaset.apps "lander-openai-dupdate-workers" deleted
service "lander-openai-dupdate-learner" deleted
pod "lander-openai-dupdate-learner" deleted
```

Resource Shutdown



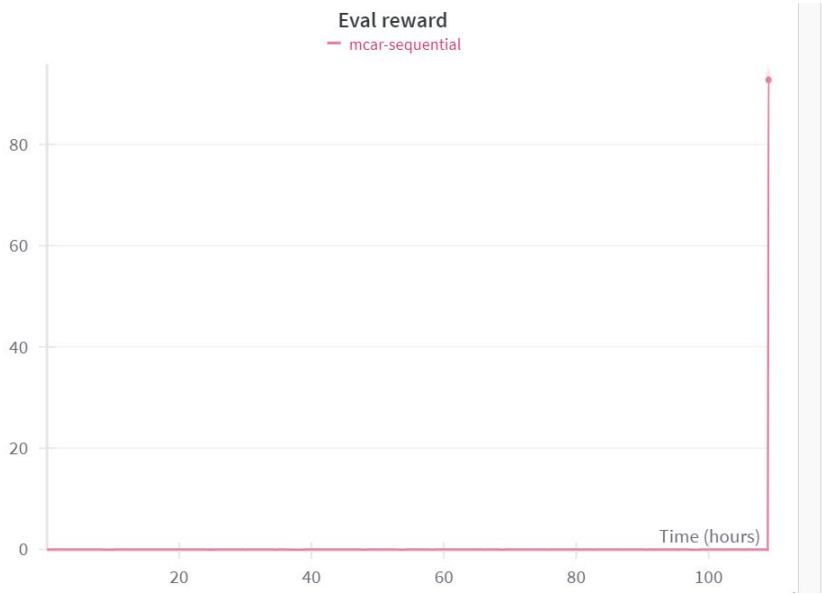
Results

Results - Overview

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car			
Bipedal Walker			
Bipedal Walker (OpenAI)			
Lunar Lander (OpenAI)			
Learn-to-Race			

Mountain Car + Sequential

- Convergence, after prolonged period of time



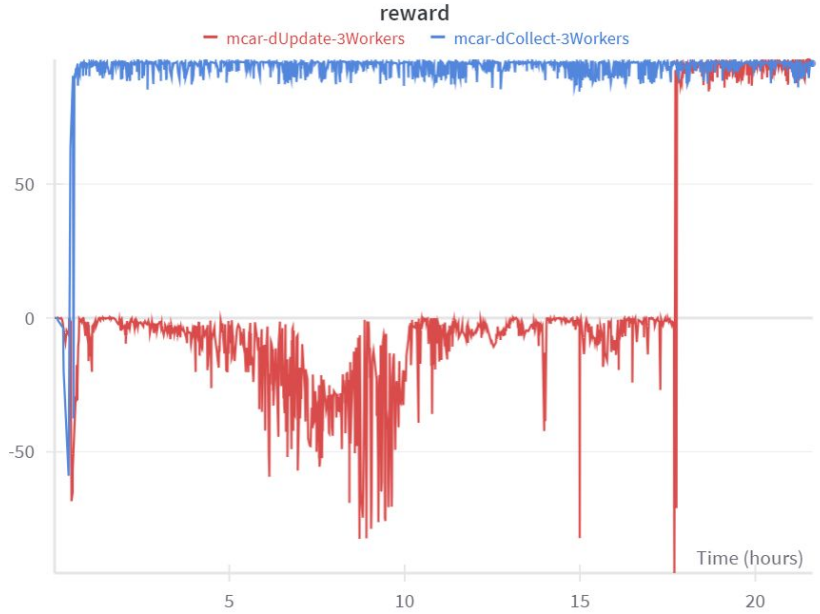
Results - Overview

- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (very slow, ~100 h)		
Bipedal Walker			
Bipedal Walker (OpenAI)			
Lunar Lander (OpenAI)			
Learn-to-Race			

Mountain Car + Distributed Paradigms

- Faster convergence, distributed update paradigm works better with less workers (right)



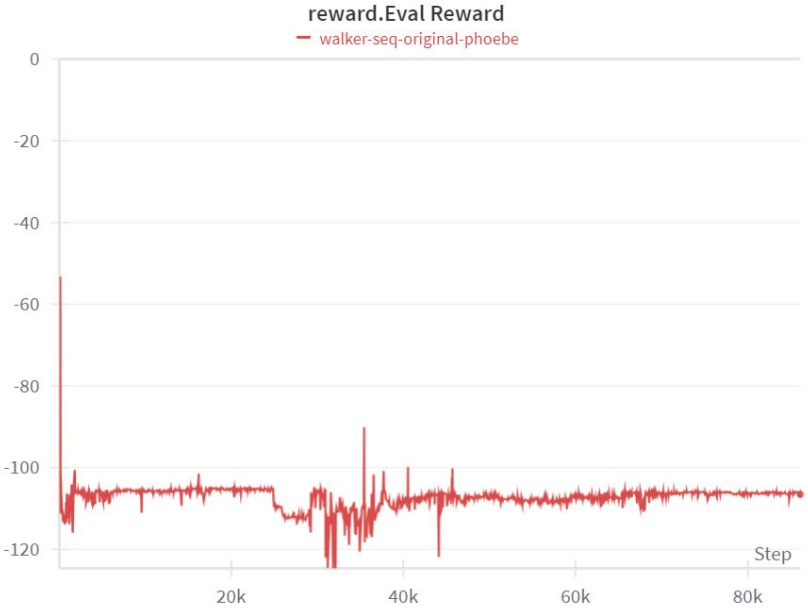
Results - Overview

- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker			
Bipedal Walker (OpenAI)			
Lunar Lander (OpenAI)			
Learn-to-Race			

Bipedal Walker + Sequential

- Non convergence (motivation to try out OpenAI's implementation)



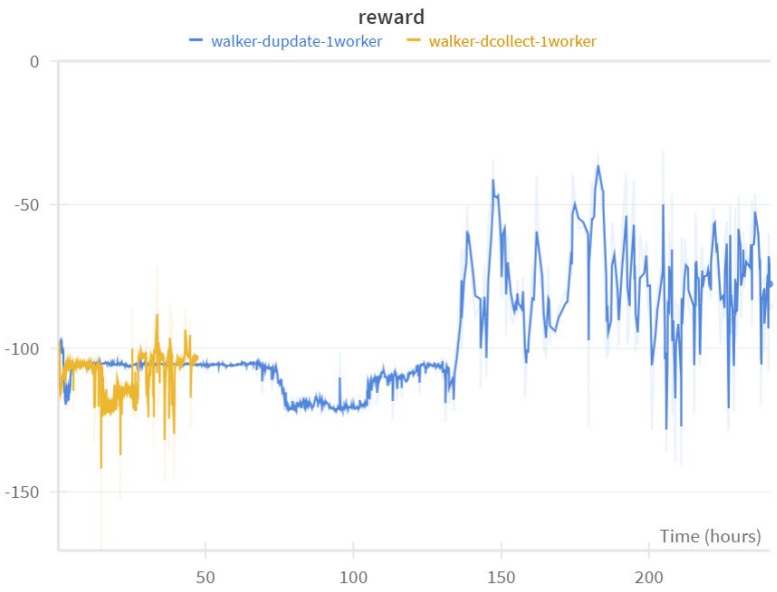
Results - Overview

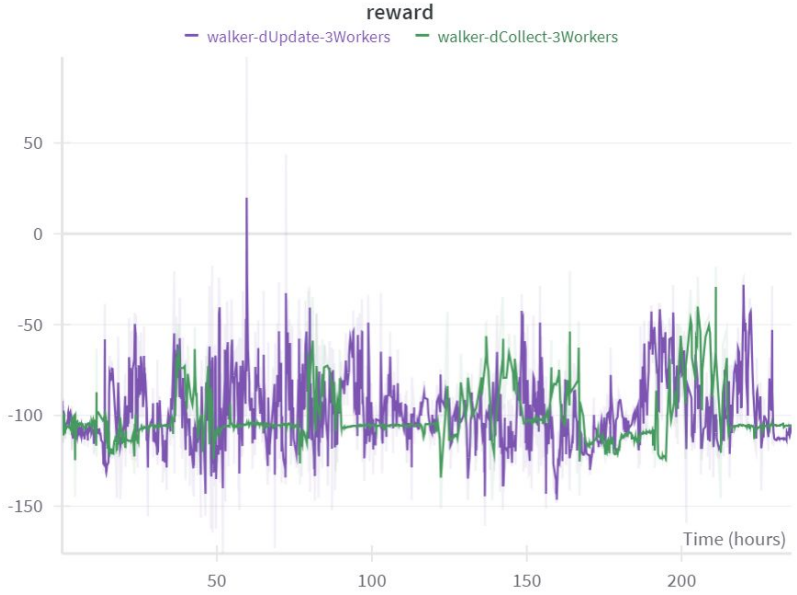
- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)		
Bipedal Walker (OpenAI)			
Lunar Lander (OpenAI)			
Learn-to-Race			

Bipedal Walker + Distributed Paradigms

- Non convergence, patterns of periodical surges, with distributed update having more fluctuations and crossing 0





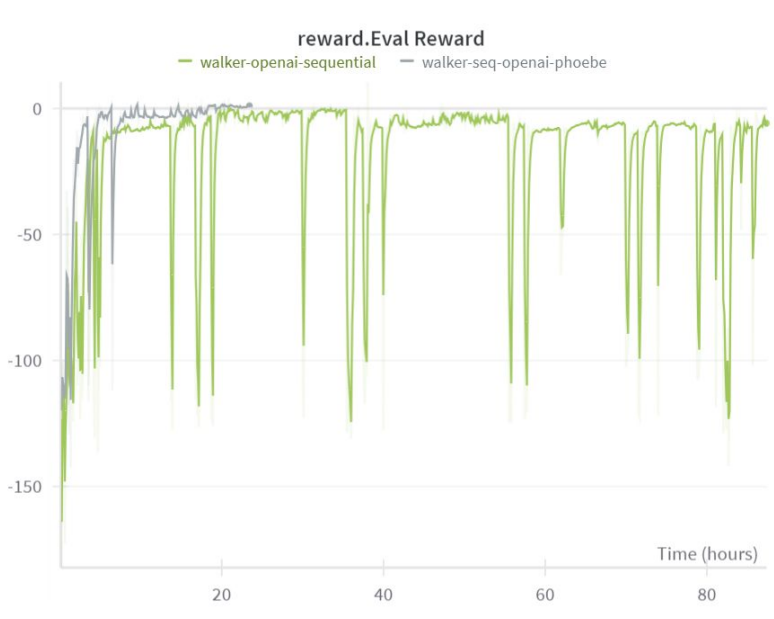
Results - Overview

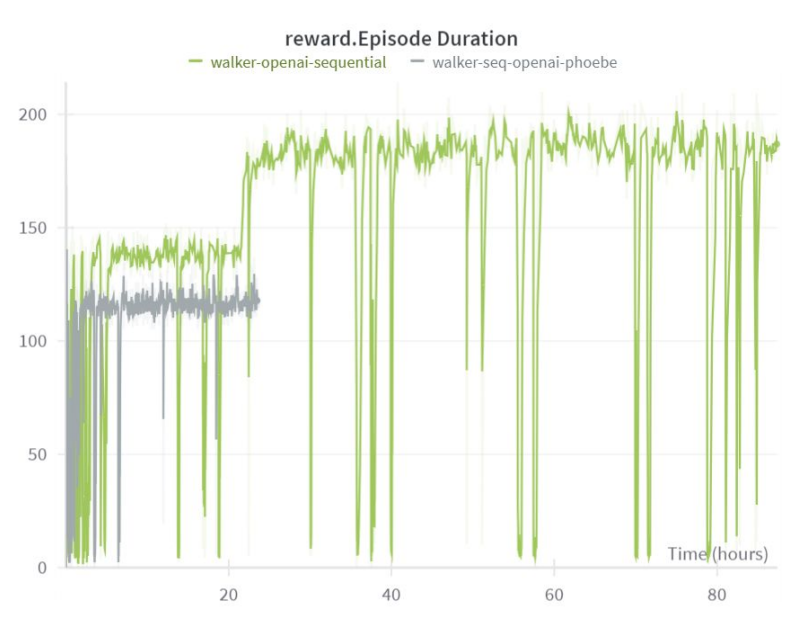
- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)			
Lunar Lander (OpenAI)			
Learn-to-Race			

Bipedal Walker (OpenAI) + Sequential

- Rough convergence to 0, but not exceeding
- Episode duration has a positive correlation





Results - Overview

- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)	~ (rough converge to reward near 0)		
Lunar Lander (OpenAI)			
Learn-to-Race			

Bipedal Walker (OpenAI) + Distributed Paradigms

- No clear convergence

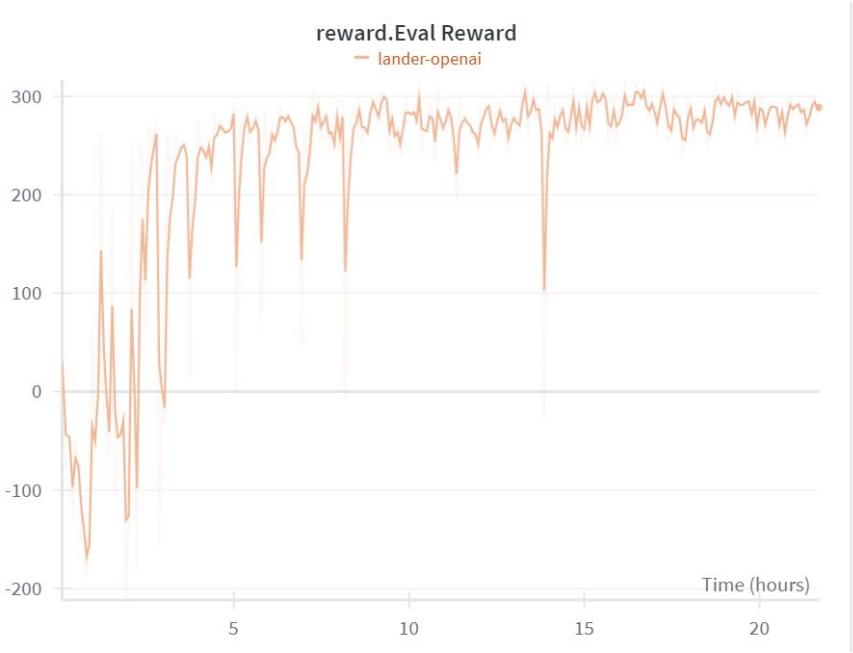
Results - Overview

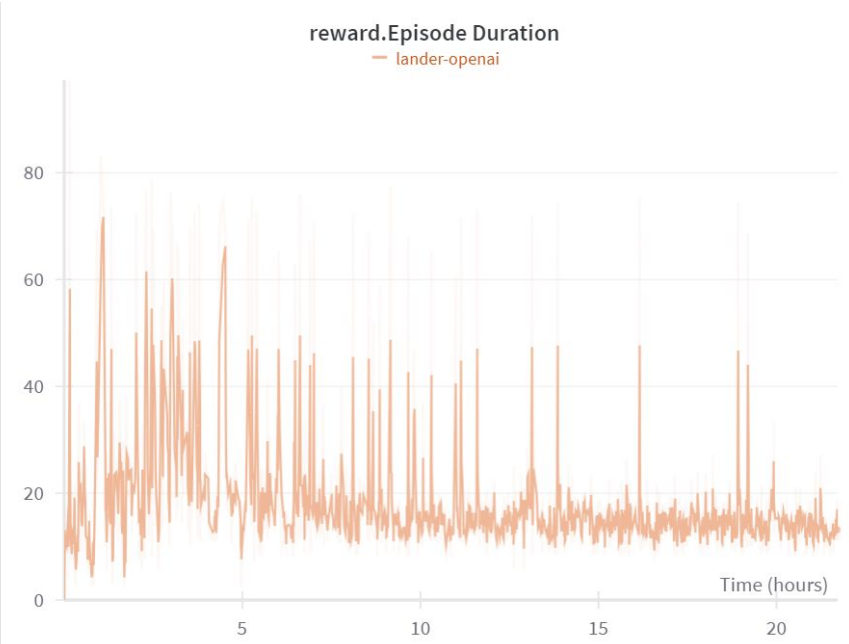
- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)	~ (rough converge to reward near 0)	~ (no convergence)	~ (no convergence)
Lunar Lander (OpenAI)			
Learn-to-Race			

Lunar Lander (OpenAI) + Sequential

- Convergence, no sign of forgetting, so episode duration stays low





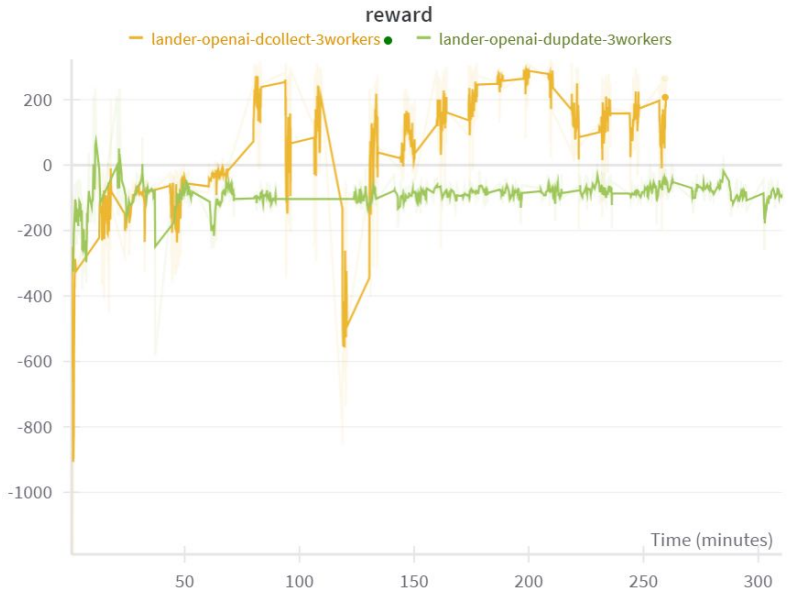
Results - Overview

- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)	~ (rough converge to reward near 0)	~ (no convergence)	~ (no convergence)
Lunar Lander (OpenAI)	✔ (~10 h, no sign of forgetting)		
Learn-to-Race			

Lunar Lander (OpenAI) + Distributed Paradigms

- Able to reach high reward fast (diverse environment?), but then degradation (forgetting?) - may benefit from checkpointing



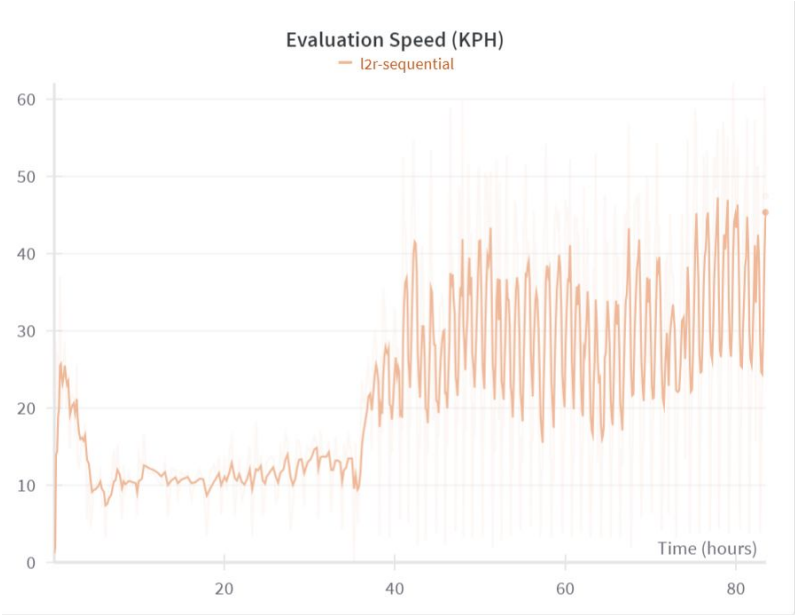
Results - Overview

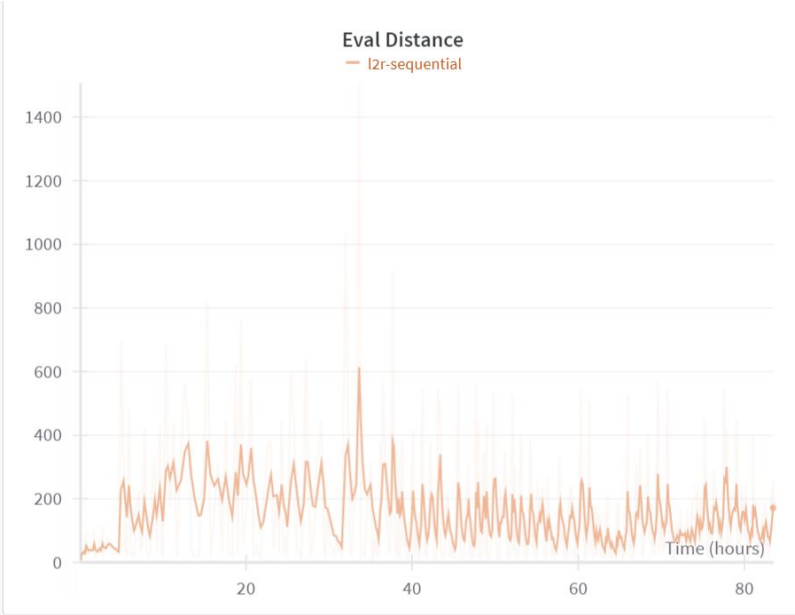
- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)	~ (rough converge to reward near 0)	~ (no convergence)	~ (no convergence)
Lunar Lander (OpenAI)	✔ (~10 h, no sign of degradation)	✔ (reach high reward <2h, with *slight degradation*)	~ (reach high reward <1h, but then degradation)
Learn-to-Race			

Learn-to-Race + Sequential

- Using a revised base Docker image
- Slow convergence *[1]* of car traveling speed





 [1] Or stable improvement

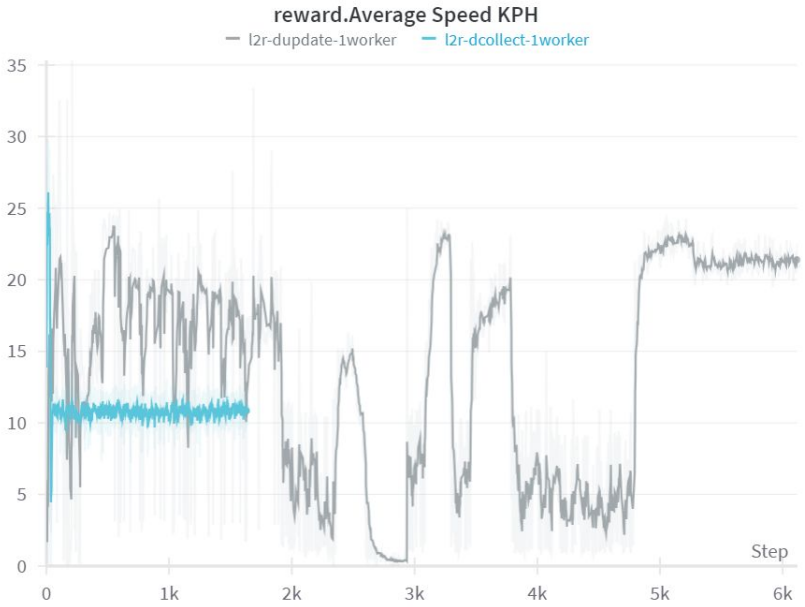
Results - Overview

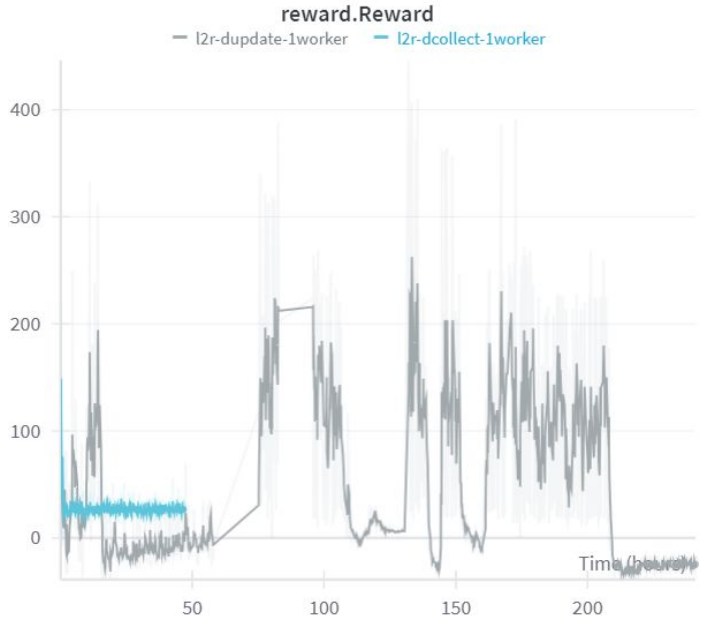
- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)	~ (rough converge to reward near 0)	~ (no convergence)	~ (no convergence)
Lunar Lander (OpenAI)	✔ (~10 h, no sign of degradation)	✔ (reach high reward <2h, with *slight degradation*)	~ (reach high reward <1h, but then degradation)
Learn-to-Race	✔ (speed KPH)		

Learn-to-Race + Distributed Paradigms

- Distributed collection paradigm started high but converged to low
- Distributed update paradigm periodically achieves better performance





Results - Overview

- Legend: ✔ means convergence, ~ means non-convergence

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)	~ (rough converge to reward near 0)	~ (no convergence)	~ (no convergence)
Lunar Lander (OpenAI)	✔ (~10 h, no sign of degradation)	✔ (reach high reward <2h, with *slight degradation*)	~ (reach high reward <1h, but then degradation)
Learn-to-Race	✔ (speed KPH)	~ (stay at low performance)	~ (periodic improvement)

Analysis

Analysis

- Let's divide-and-conquer this table

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)
Bipedal Walker	~ (reward < 0)	~ (periodical surges)	~ (can cross 0, fluctuations)
Bipedal Walker (OpenAI)	~ (rough converge to reward near 0)	~ (no convergence)	~ (no convergence)
Lunar Lander (OpenAI)	✔ (~10 h, no sign of degradation)	✔ (reach high reward <2h, with *slight degradation*)	~ (reach high reward <1h, but then degradation)
Learn-to-Race	✔ (speed KPH)	~ (stay at low performance)	~ (periodic improvement)

Analysis

RL Environments	Sequential	Distributed Collection	Distributed Update
Mountain Car Ref	✔ (slow, ~100 h)	✔ (faster with more workers)	✔ (faster with less workers)

- **Simple environment**: all able to converge, no forgetting/degradation
- **Distributed paradigms**: achieves ~10x faster convergence
- **Distributed collection faster with more workers**
  - Fast training, learner not bottleneck, workers collect diverse experience
- **Distributed update slower with more workers**
  - Network, sharing of parameters, coordinations all become overhead

Analysis

RL Environments	Sequential	Distributed Collection	Distributed Update
Bipedal Walker (OpenAI) Ref	~ (rough converge to reward near 0)	~ (no convergence)	~ (no convergence)
Lunar Lander (OpenAI) Ref	✔ (~10 h, no sign of degradation)	~ (reach high reward <2h, but then degradation)	~ (reach high reward <1h, but then degradation)

- **OpenAI Vanilla baselines:** sequential version outperforms original gym baselines, but distributed baseline behavior is not stable
- **Distributed Collection (dCollect) v.s. Distributed Update (dUpdate)**
  - Both performs similarly, with unsatisfactory results for bipedal walker, and the typical “learn-fast, forget-fast” issue for lunar lander

Analysis

RL Environments	Distributed Collection	Distributed Update
Bipedal Walker Ref	~ (periodical surges)	~ (can cross 0, fluctuations)
Learn-to-Race Ref	~ (converge to low performance)	~ (periodic improvement)

- **Harder environments**: fluctuations due to diverse environments, potential to reach high performance faster, but suffering from degradation (i.e., learning fast but forgetting fast)
- **Distributed Collection (dCollect) v.s. Distributed Update (dUpdate)**
  - dUpdate induces more fluctuations due to distributed update of parameters (damping factor)
  - dUpdate (due to high fluctuations) generally has more potential to reach better rewards
  - dUpdate can benefit from checkpointing (greedily saving best checkpoint, if environment is static)

Conclusion

Summary of Analysis

RL Environments	Sequential Paradigm	Distributed Paradigms
Easy Gym (mountain car)	✔ stable convergence × very slow	✔ stable convergence, much faster × communication overhead
Hard Gym (bipedal walker)	✔ if convergence, should stay stable × often stuck at worse reward	✔reach high reward fast (learn fast) due to diverse simulations × suffer from degradation (forget fast)
Vanilla SAC	✔ better convergence performance than non-Vanilla	× generally poor behavior (need to investigate further)
Learn-to-Race	✔ observed stable improvement × often stuck at worse reward	✔ observed periodical improvement, faster in comparison × fluctuations, unstable

Conclusion

- **Contribution**
  - New training paradigm that distribute training tasks to workers
  - Codebase and resource control optimizations
  - Experiments and analysis of diverse configuration combinations
- **Observations**
  - *Handling RL and distributed system design is challenging*
  - For easy tasks like mountain car, distributed paradigms tend to converge faster
  - For medium tasks like lunar lander and bipedal walker, distributed paradigms with diverse environments fluctuate and can thus reach high reward pretty quickly, but suffers from forgetting and degradation → may be mitigated through checkpointing
  - For Learn-to-Race task, distributed Update paradigm has periodical improvements
- **Future Work**
  - Torch device handling (store replay buffer on CPU, Gym action must be numpy)
  - Importance sampling and dynamically adjusted task allocations
  - Should take other thrusts into consideration

Appendix

Terminology

- **Reinforcement learning (RL)**: a machine learning training procedure in which an agent learns how to take actions within an environment based on reward and penalty feedback.
- **Agents**: an intelligent artificial entity can perceive its environment and respond reasonably.
- **State**: the parameters that define the status of the agent.
- **Action**: the action taken by the agent based on its environment and state to maximize the reward function.
- **Environment**: the environmental feedback that the agent is within.
- **Reward function**: a formulated representation indicating the gain of the agent at a certain state and environment through taking a certain action.

References

1. Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015.
2. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017
3. Weights and Biases: https://wandb.ai/
4. Wang, Limin, et al. "Tdn: Temporal difference networks for efficient action recognition." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.
5. T. Weiss and M. Behl, "DeepRacing: A Framework for Autonomous Racing," 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 1163-1168, doi: 10.23919/DATE48585.2020.9116486.

Reflection

- **Documentations**
  - **Vision**: define the project scope, goals, and objectives
  - **Requirement**: identify and prioritize project requirements within our scope
  - **Design**: detailed understanding of modularized dependencies and risks
  - **Plan**: project timelines, milestones, and syncing between thrusts
- **Change of perception**
  - Inter-thrust communication
  - Communication with teaching staff
- **Working as a group**
  - Fixed meeting schedules
  - Fixed allocation of work and responsibilities

Changes To Previous Deliverables

- We have changed to use the new docker image jingyua to substitute l2r2022 since it seemed to provide better performance for the learn-to-race environment. The results, reasoning, and analysis are included in the previous slides.
- We have made significant changes to the previous github repositories to reduce the number of branches to two (sequential and distributed paradigm), which was discussed in the previous slides.