Optimizing and Extending the Functionality of EXARL for Scalable Reinforcement Learning

Sai Chenna,* Katherine Cosburn,* Uchenna Ezeobi,* Maxim Moraru,* Hyun Lim*, Julien Loiseau*, Jamal Mohd-Yusof*, Robert Pavel*, Vinay Ramakrishnaiah*, Andrew Reisner*, Karen Tsai*
*Los Alamos National Laboratory

Introduction

REINFORCEMENT LEARNING (RL)

- A subset of machine learning wherein an agent interacts and learns from its environment according to some policy (mapping of states to actions) with the overall goal of achieving a maximum reward over time (Figure 1).
- Used in games, computer vision applications, and increasingly in scientific disciplines to solve control problems (specifically those that can be formulated as a Markov Decision Process).
- The actions an agent can take can be continuous (e.g. move according to some applied torque) or discrete (e.g. move left or right).
- In contrast to many RL examples, where the environment responses are easy to compute, the environments used in scientific RL studies are often complex and take substantial computation time to run, even in parallel.
- The ability to scale these computations to multiple nodes can aid in considerably reducing computation time, allowing for more robust testing and prototyping of ideas.

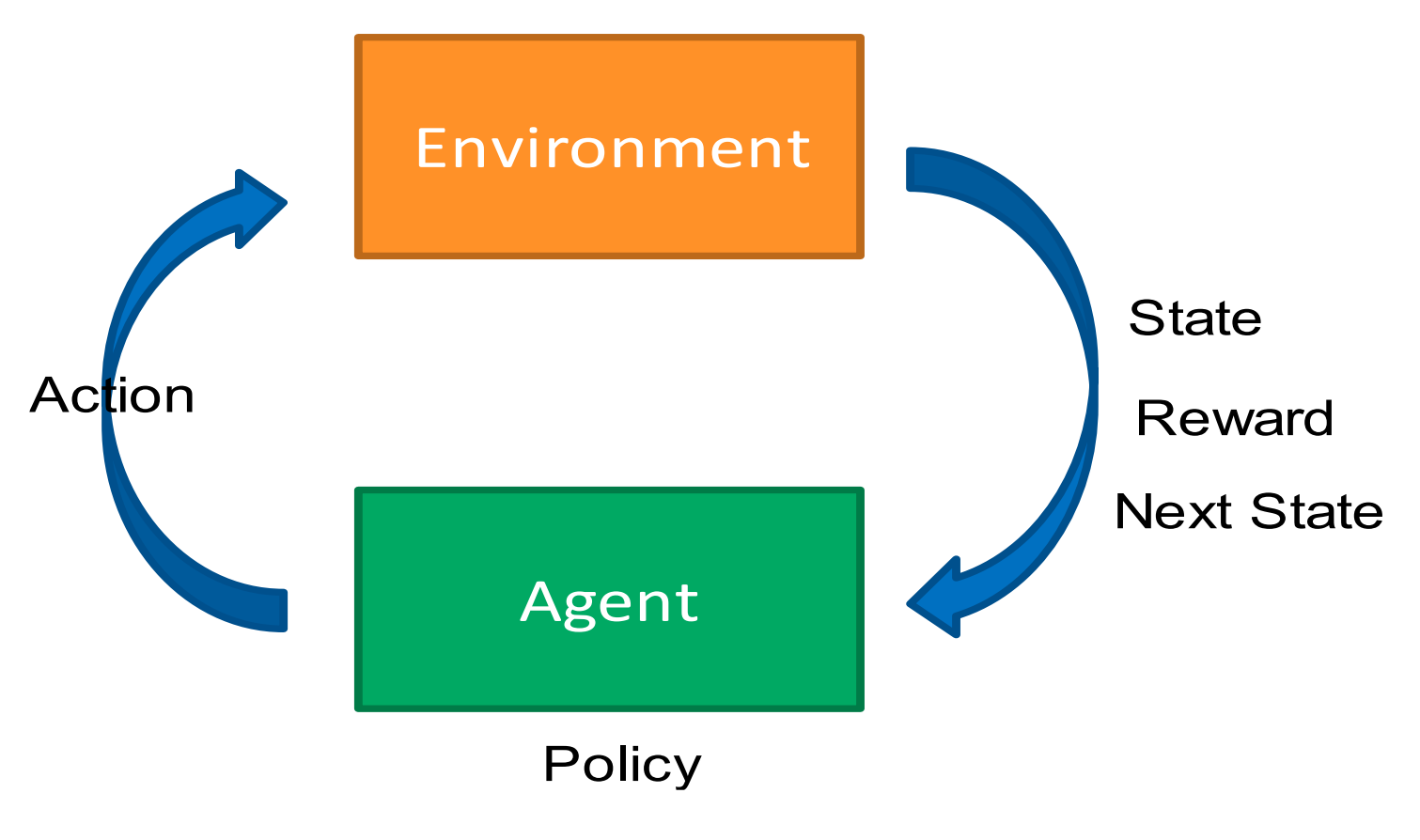

Figure 1. Basics of reinforcement learning.

EASILY EXTENDABLE ARCHITECTURE FOR REINFORCEMENT LEARNING (EXARL)

- A scalable RL framework for scientific applications.
- Provides customizable environments, agents, and workflows, which help improve performance and reduce execution time.
- The agent is divided into actors and learners.
- The actors are responsible for interacting with the environment and generating trajectories (training data).
- These trajectories are fed into the learners, which update the policy.
- The updated policy is shared with the actors, which perform the suitable to maximize the reward.
- Similar to IMPALA architecture, the actors and learners are decoupled in order to scale the reinforcement learning (Figure 2).

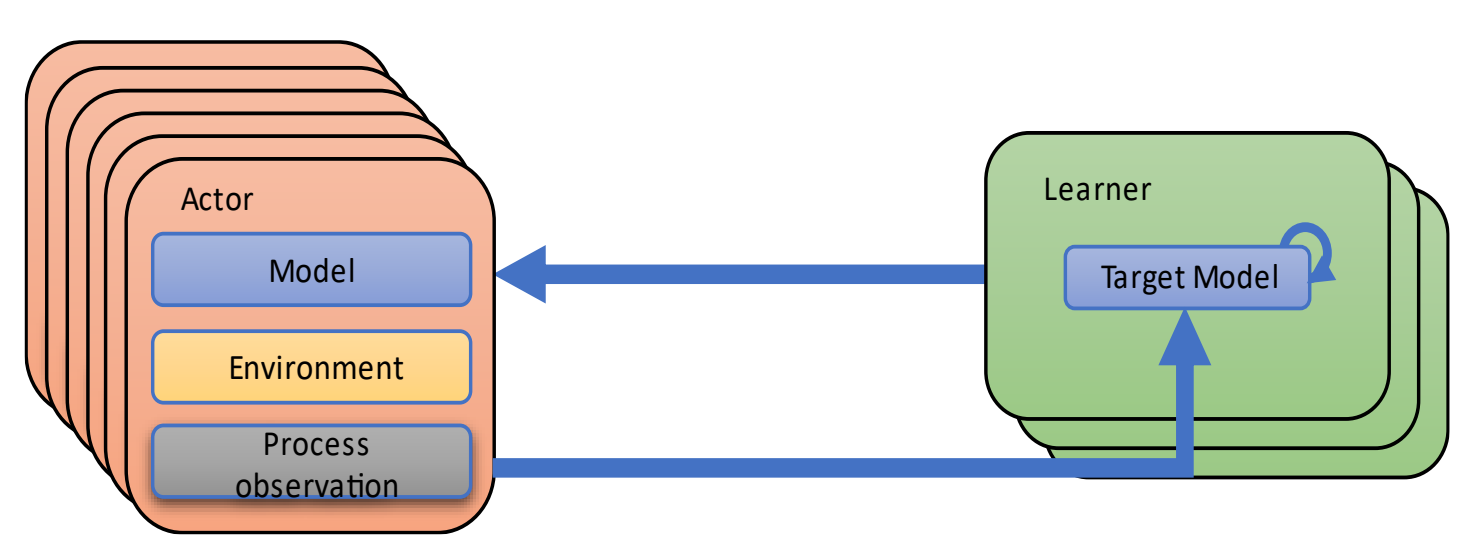

Figure 2. EXARL architecture.

EXABOOSTER ENVIRONMENT

- Control problem for FNAL particle accelerator at FermiLab (Figure 3).
- Reinforcement learning is used to control particle beam quality (i.e. reduce beam losses) in real time.
- Keeps the beam field from spreading (thus degrading the beam quality) by regulating the magnetic current of the booster.
- Discrete action environment: magnetic field can either be increased or decreased.
- Original work developed by PNNL, FNAL, University of California San Diego, Columbia University.

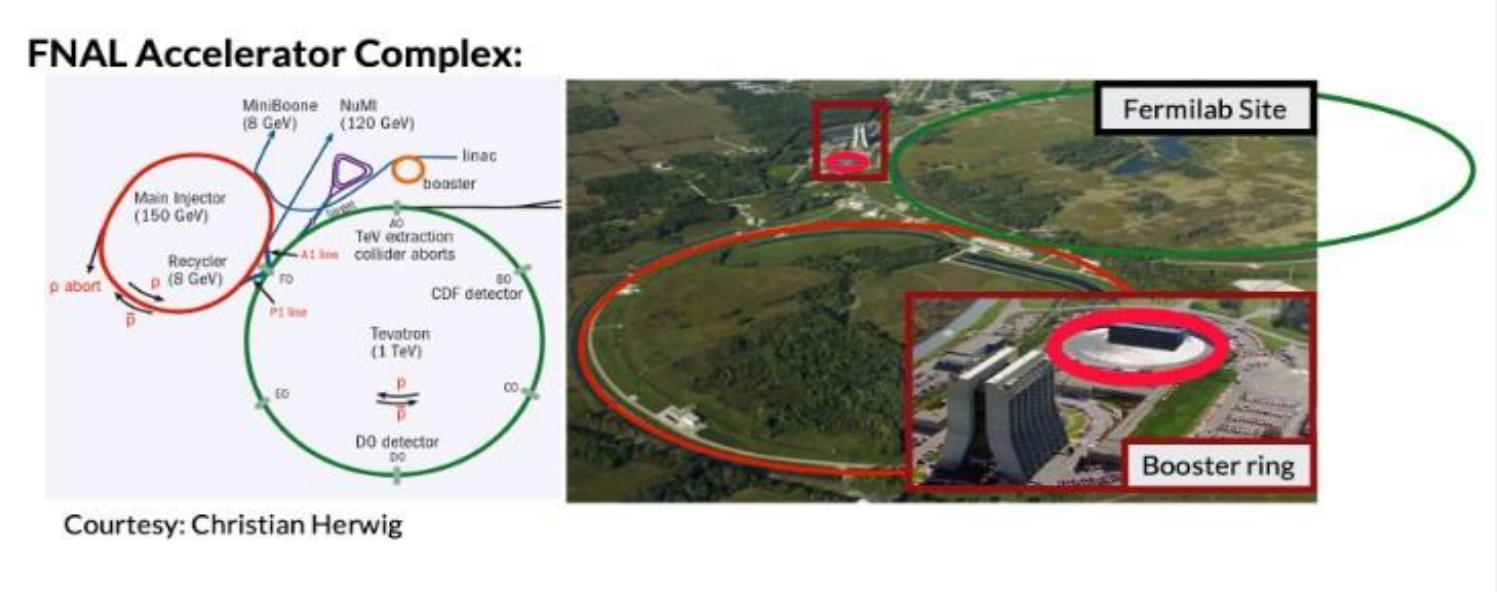

Figure 3. FNAL particle accelerator complex at FermiLab with particle beam booster ring shown.

Methodology

MULTI-LEARNER ASYNCHRONOUS WORKFLOW

- All actors send training data (trajectories) to the master learner after interacting with the environment.
- The master learner then distributes batches of data to the remaining learners to perform distributed training.
- The updated policy from all learners is sent to all the actors.

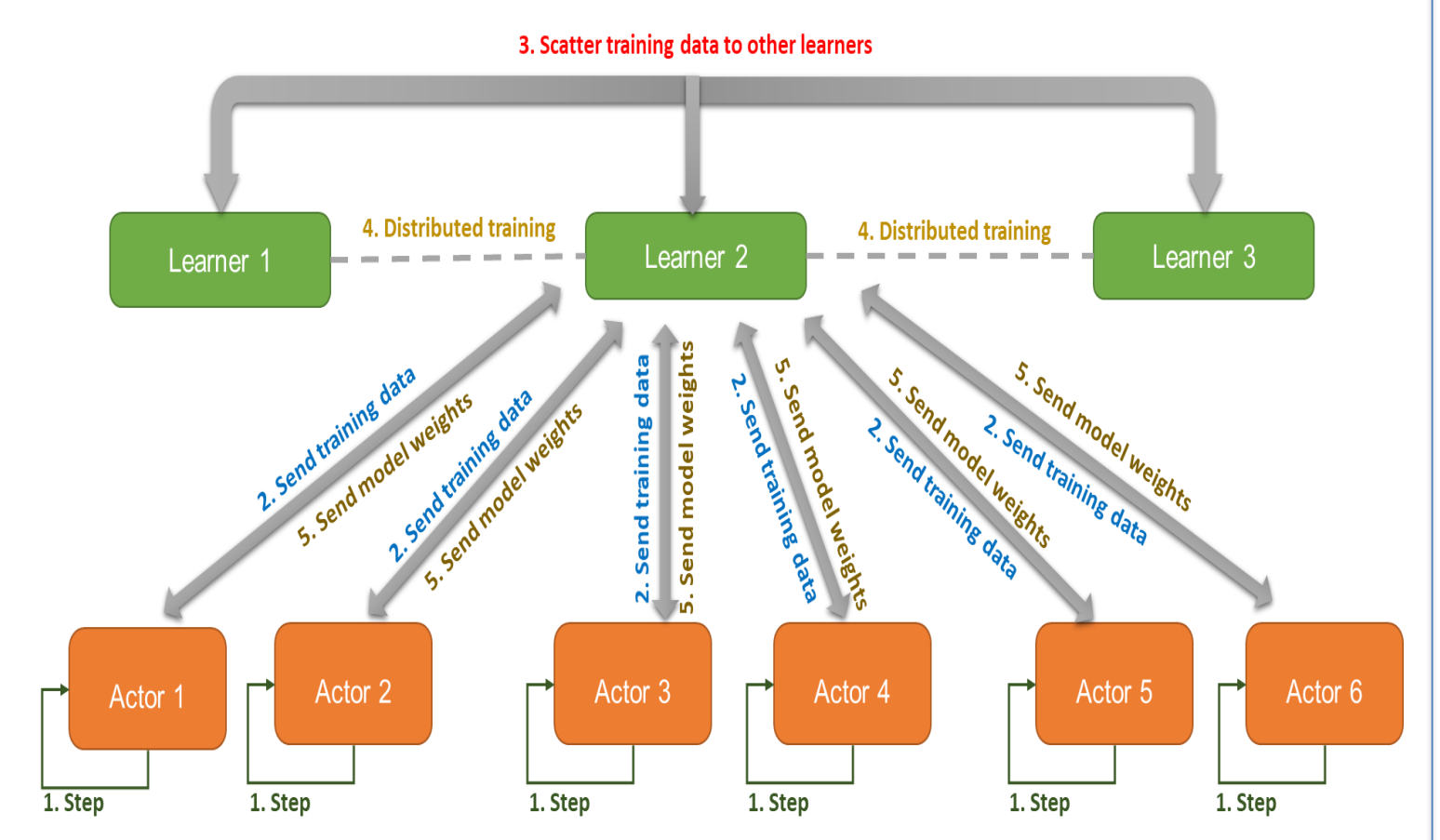

Figure 4. Multi-learner asynchronous workflow schematic.

RMA WINDOW SELECTION

- In Multi-learner RMA workflow, learners get training data from the actor's RMA window.
- **Current approach:** Each learner randomly selects one of the actor's RMA window. This can lead to slower convergence.
- **Proposed approach:** Allocate a set of actor RMA windows to each learner (Figure 5).
- **Advantage:** Guarantees no learner reads from the same actor's RMA window, thus optimizing training.

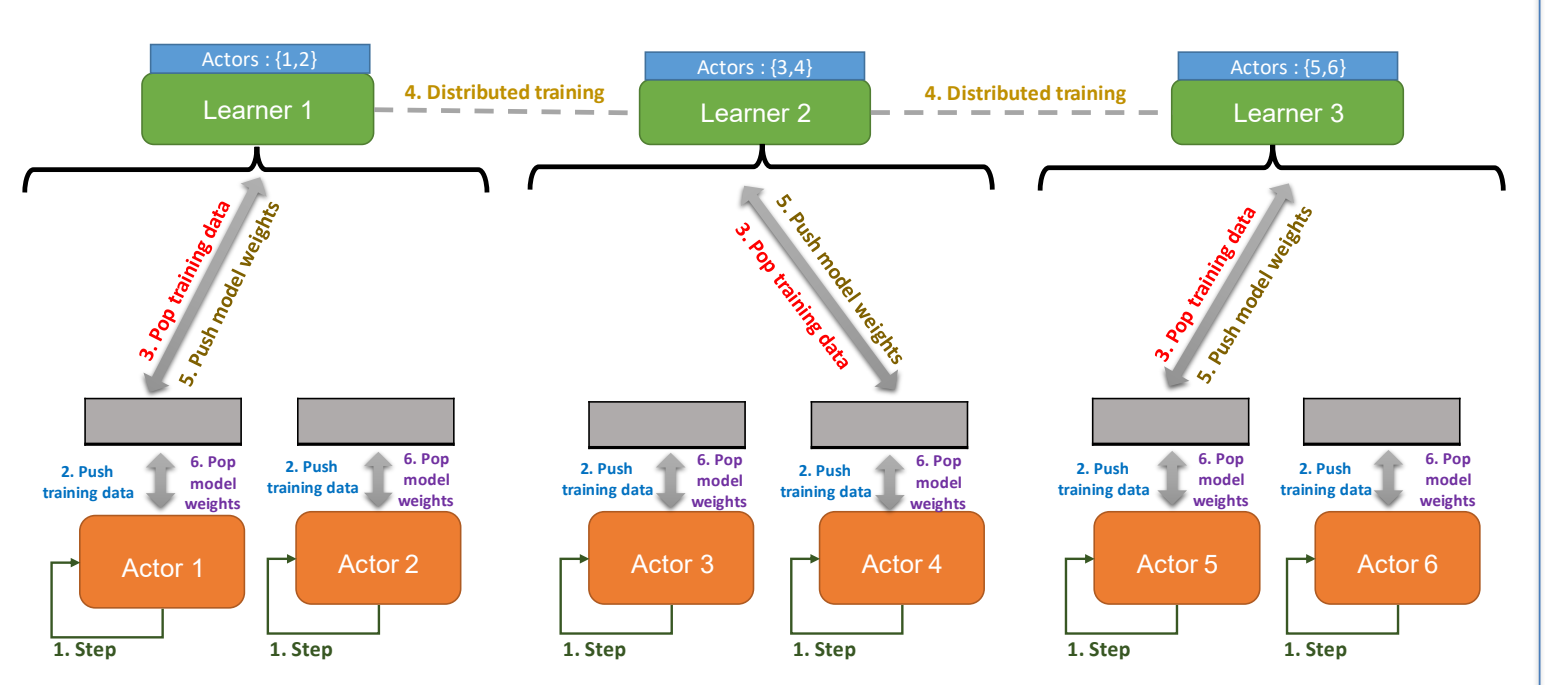

Figure 5. Multi-learner RMA workflow – range-based window selection.

RMA QUEUE WORKFLOW

- Each actor has a local queue data structure remotely accessible by all learners, implemented using RMA windows.
- Each group of actors is assigned to a specific learner (allowing to limit the number of simultaneous accesses to the same queue).
- Learners can retrieve training data from these queues and share the updated weights via a global RMA window.
- Actors and learners are synchronized using passive target RMA locks and the training part is performed in parallel using Horovod.

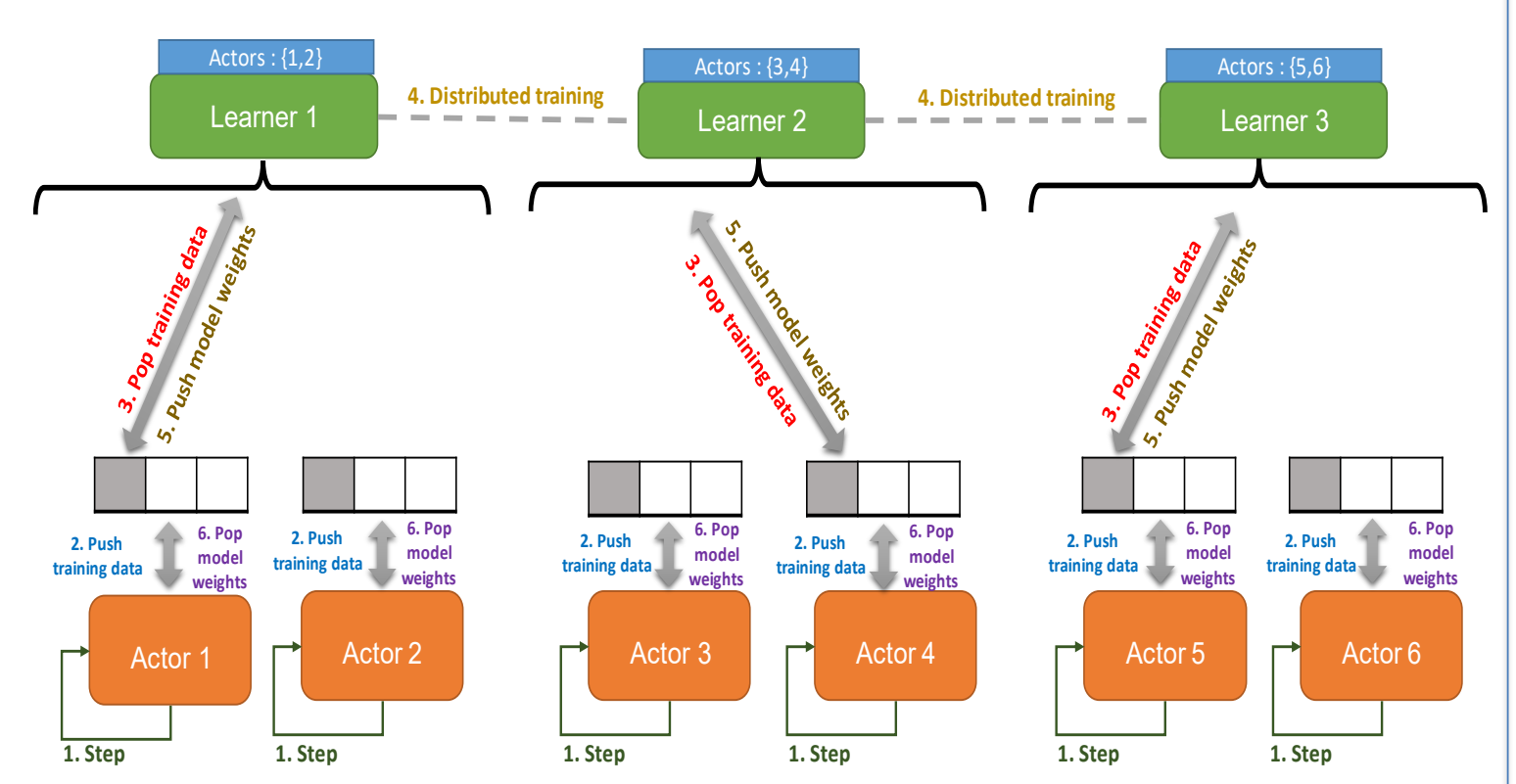

Figure 6. RMA queue workflow schematic.

- Learners that exhaust all 'active' actors (e.g. Learner 3 in Fig. 7) assist other learners by training on the batch data from the latter's actor queue.
- A "shared bitmap" indicates which actors are active preventing getting data from an empty queue.

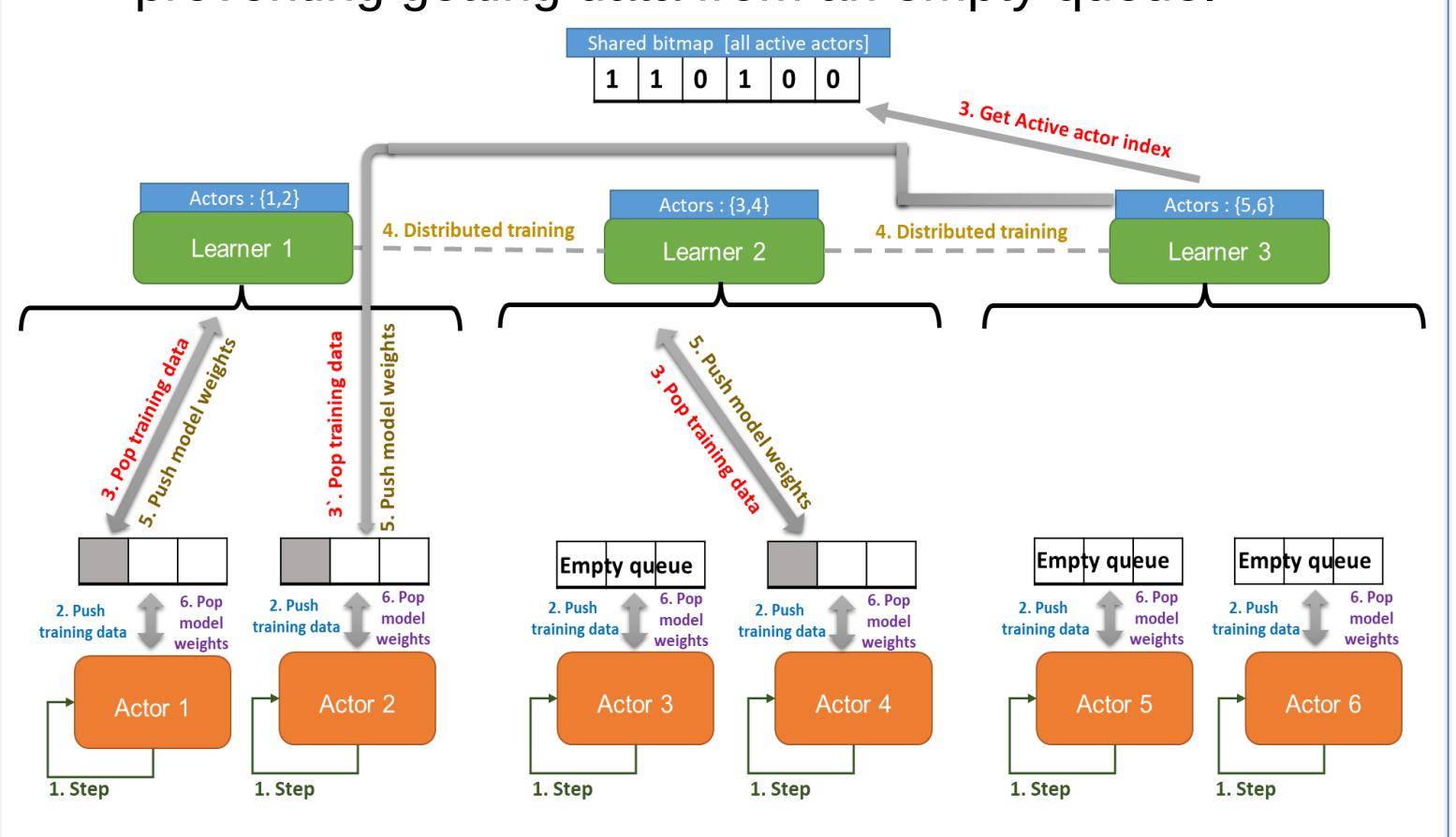

Figure 7. RMA queue workflow with shared bitmap implementation details.

OPTIMIZING DATA GENERATION PIPELINE

- Calculating the Bellman optimality equation on each experience is expensive (90% of computation time) while using Deep Q-Network (DQN) agent.
- **Optimization:** Offload data-generation on remaining environment processes (Figure 8).

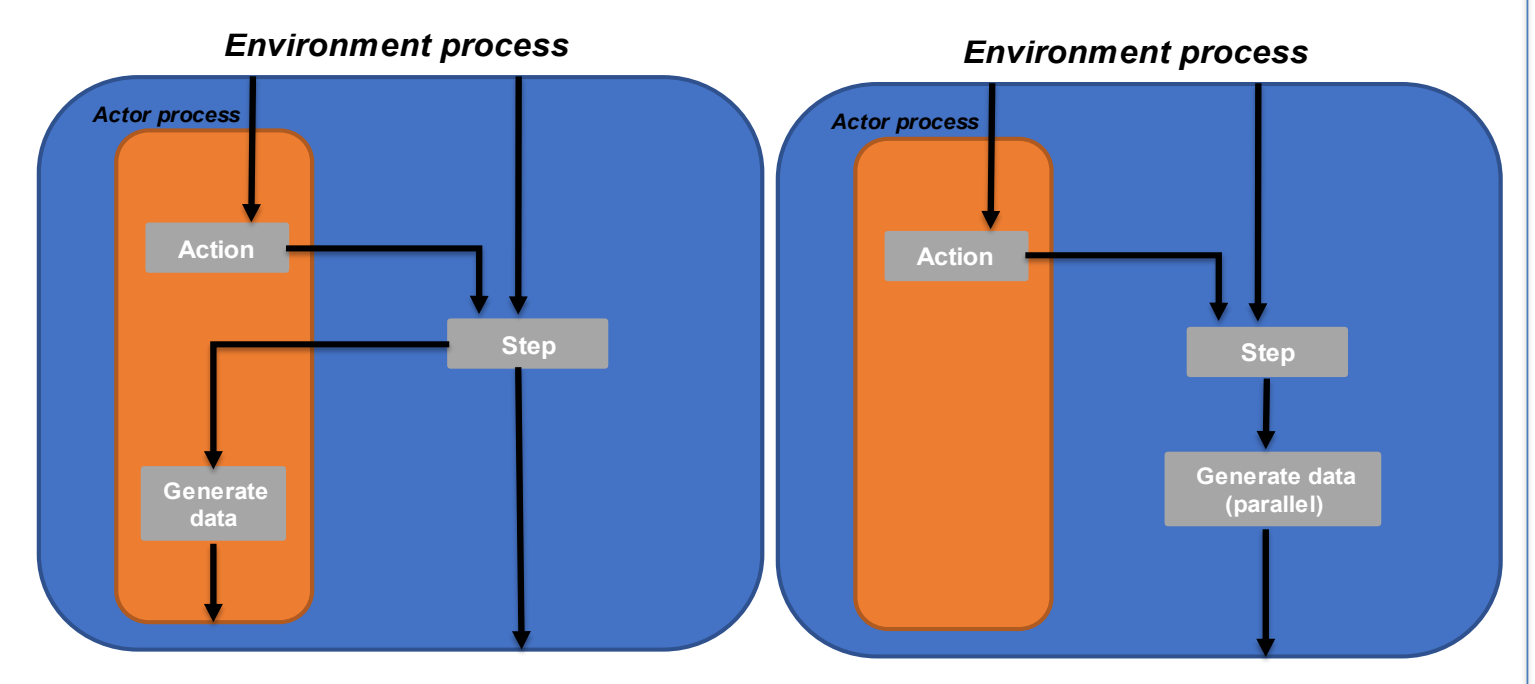

Figure 8. Data generation optimization process. Left shows previous implementation, right shows updated workflow.

SEED WORKFLOW

- Inspired from SEED architecture and based on MPI P2P routines.
- Lower bandwidth requirements relative to base model.
- Only one copy of the model → there is no issue of copies going out of sync.

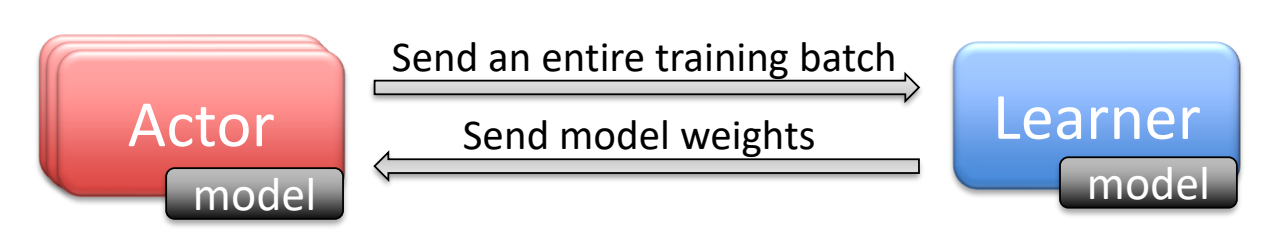

Figure 9. Current EXARL architecture (inspired from IMPALA)

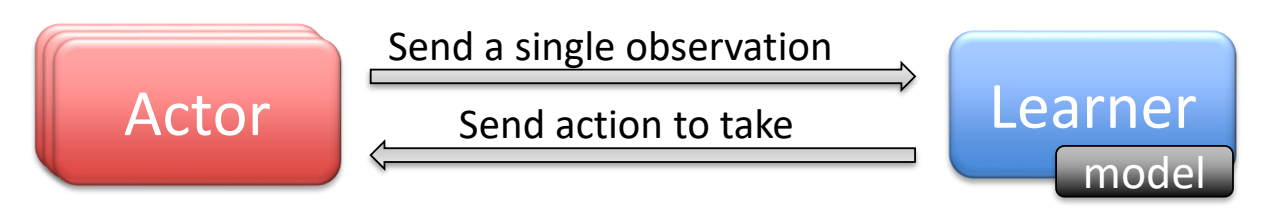

Figure 10. Architecture inspired from SEED (implemented using MPI)

(ASYNCHRONOUS) ADVANTAGE ACTOR CRITIC (A2C/A3C)

- An actor-critic network that acts on discrete or continuous action spaces. Current implementation is for discrete action space (e.g. ExaBooster environment).
- A2C/A3C is faster to train & has more diverse data than DQN.
- It is an on-policy agent, i.e. the policy an actor acts with should be the same as the policy a learner learns with. EXARL framework can't guarantee that behavior.
- To correct for that, we add the "v-trace" algorithm to the loss functions. This correction assumes the ratio between the two policies is always equal to one.

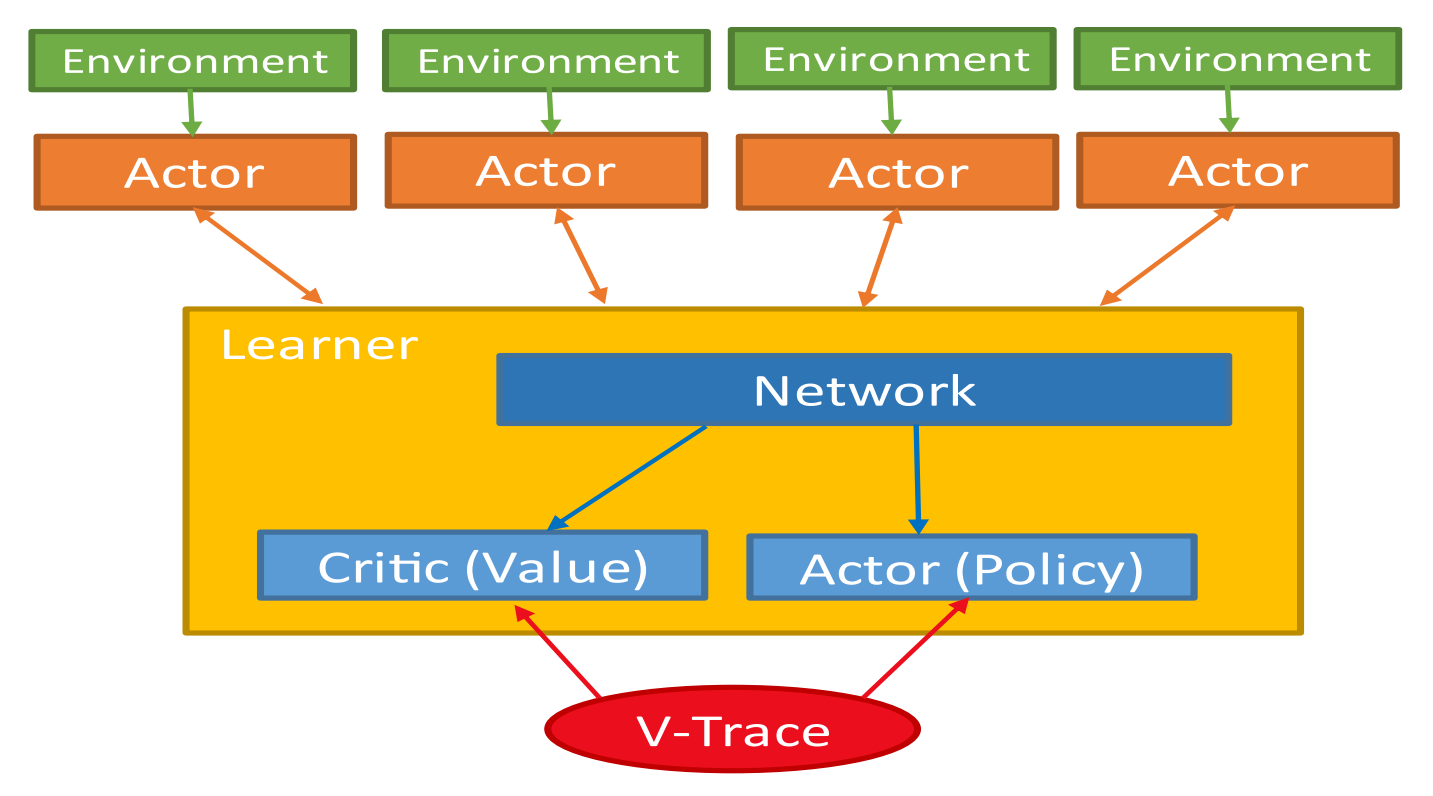

Figure 11. (Asynchronous) Advantage Actor Critic agent schematic. The v-trace algorithm is added to the loss functions of the critic and actor networks.

TWIN DELAYED DEEP DETERMINISTIC POLICY GRADIENT (TD3)

- Actor-critic network for continuous action-space environments meant to correct for shortcomings in the Deep Deterministic Policy Gradient (DDPG) agent.
- It is an off-policy agent, meaning the policy an actor acts is independent of the policy a learner learns.
- Address the overestimation of the Q-value in DDPG by using 3 tricks: (1) clipped Double Q-learning, (2) delayed policy agent, and (3) target policy smoothing.

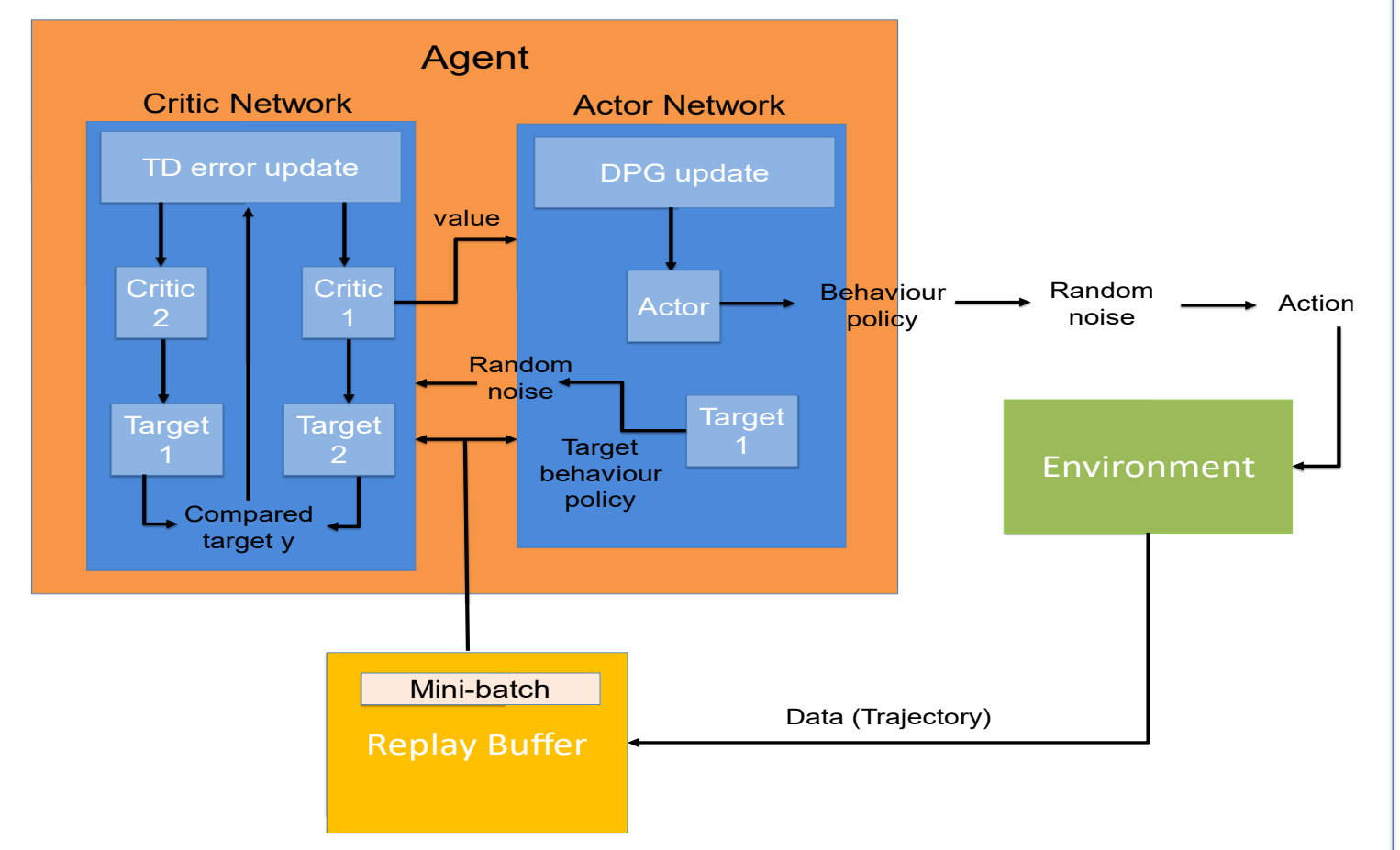

Figure 12. Twin Delayed Deep Deterministic Policy Gradient agent architecture.

Performance Improvements

SCALING MULTI-LEARNER WORKFLOWS

- **Results:** Preliminary experiments demonstrated good scalability up to 1024 processes (32 processes per node) on ExaBooster environment (as shown in Figure 12).
- Efficient overlap of simultaneous computations of Bellman equation across actors could be a potential cause for superlinear speedup observed beyond 4 nodes (with single node as baseline).
- Multi-learner RMA workflow performed better than asynchronous workflow – which uses a master learner to distribute training data to remaining learners.
- Owing to this centralized approach (Master-worker), multi-learner asynchronous workflow has relatively poor scalability compared to the RMA variants.
- Multi-learner asynchronous workflow is suitable for on-policy agents (like A2C) where the training is done towards the end of episode.

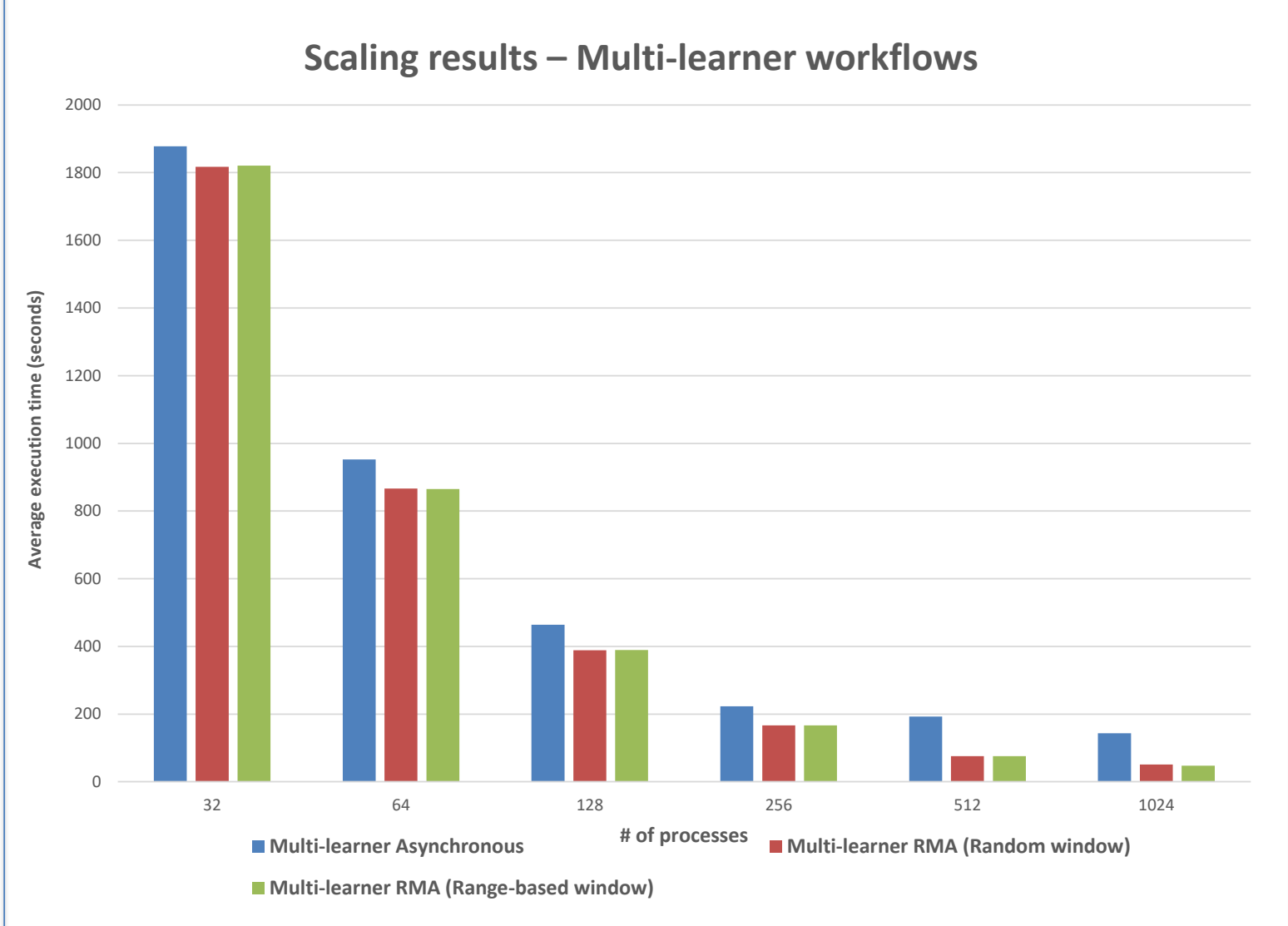

Figure 13. Scaling results of Multi-learner workflows on Darwin Testbed Cluster.

RMA WINDOW SELECTION

- **Results:** In random window selection approach, multiple learners may select a same actor RMA window, thereby using same data in a distributing training setting –**poor convergence**.
- Range-based window selection guarantees no two learners train on the same data.
- As shown in Figure 14, Range-based window selection (orange line) had faster convergence than random window selection policy (blue line).

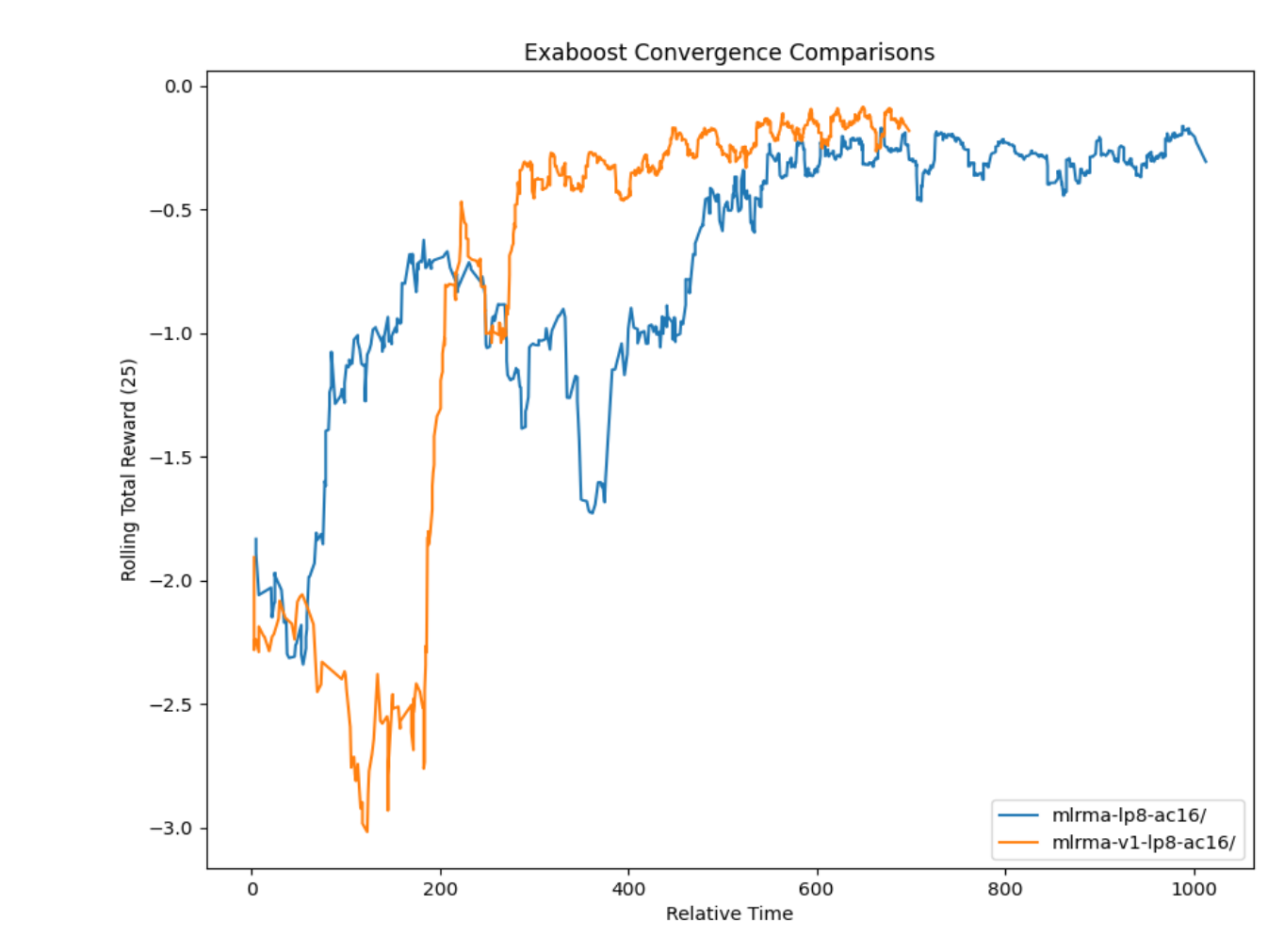

Figure 14. Convergence comparison of two window selection policies of Multi-learner RMA workflow used on ExaBooster environment.

RMA QUEUE WORKFLOW

- **Results:** Total execution time decreased by 77% while using 20 learners and 120 actors on 4 nodes, and by 84% while using 200 actors and 40 learners (8 nodes).

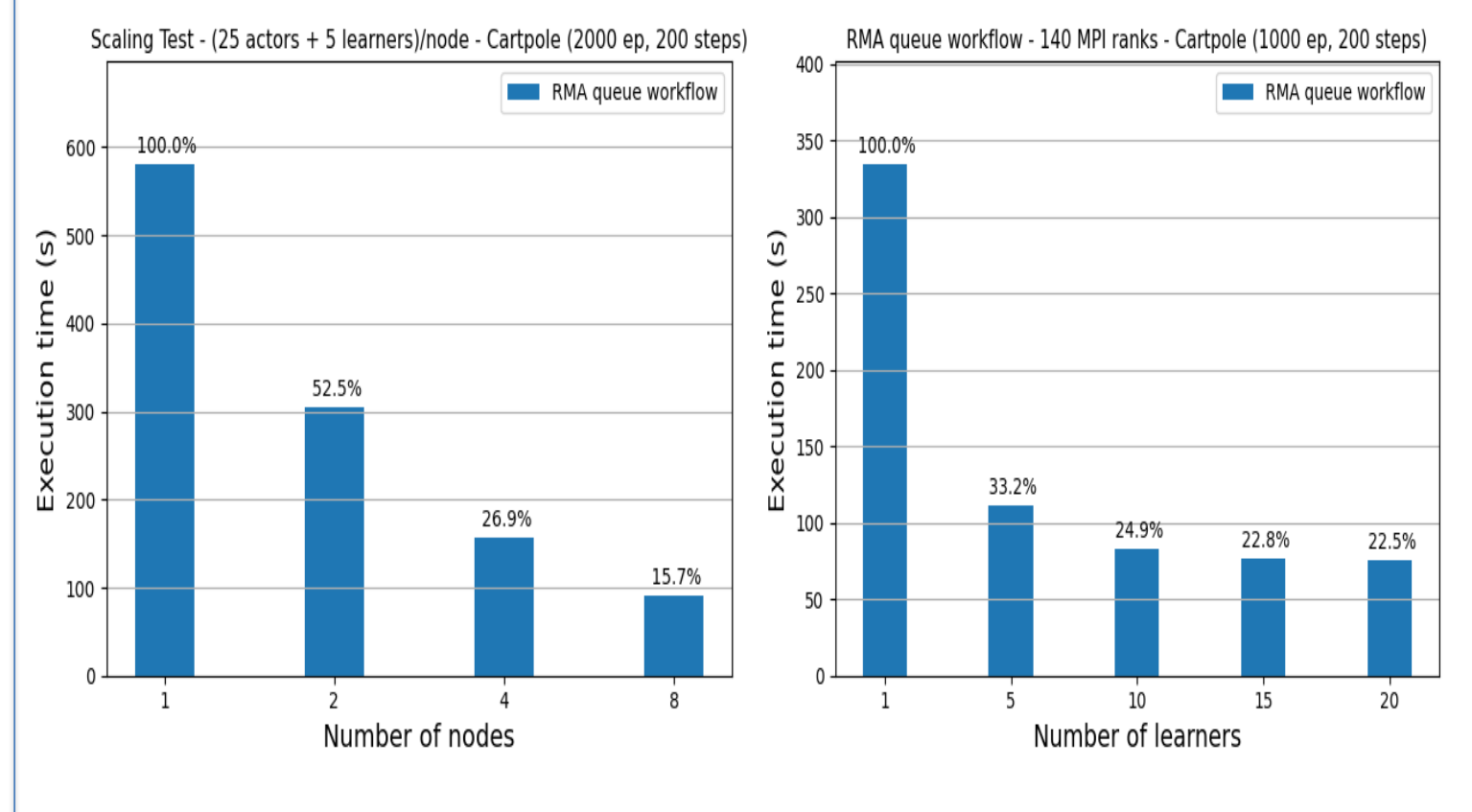

Figure 15. Results for RMA queue scaling

OPTIMIZING DATA GENERATION PIPELINE

- **Results:** Average speedup of 3.3x upon scaling the workload to 4 processes.
- Observed linear speedup when scaling the workload onto multiple processes.

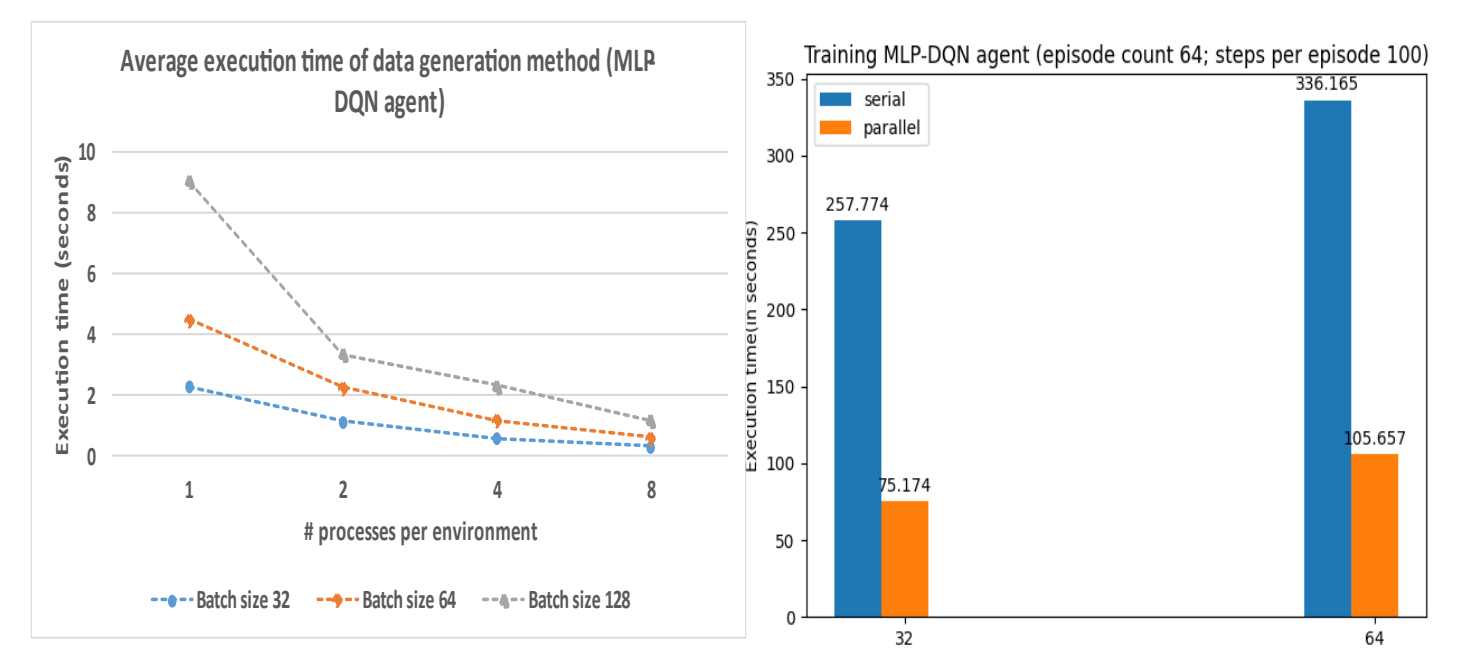

Figure 16. Results for data generation pipeline improvements.

SEED WORKFLOW

- Our SEED implementation obtained similar results to current IMPALA implementation (Figure 16).
- The neural network model used for these experiments was relatively small. Therefore, the amount of transferred data during an iteration is similar in both implementations.

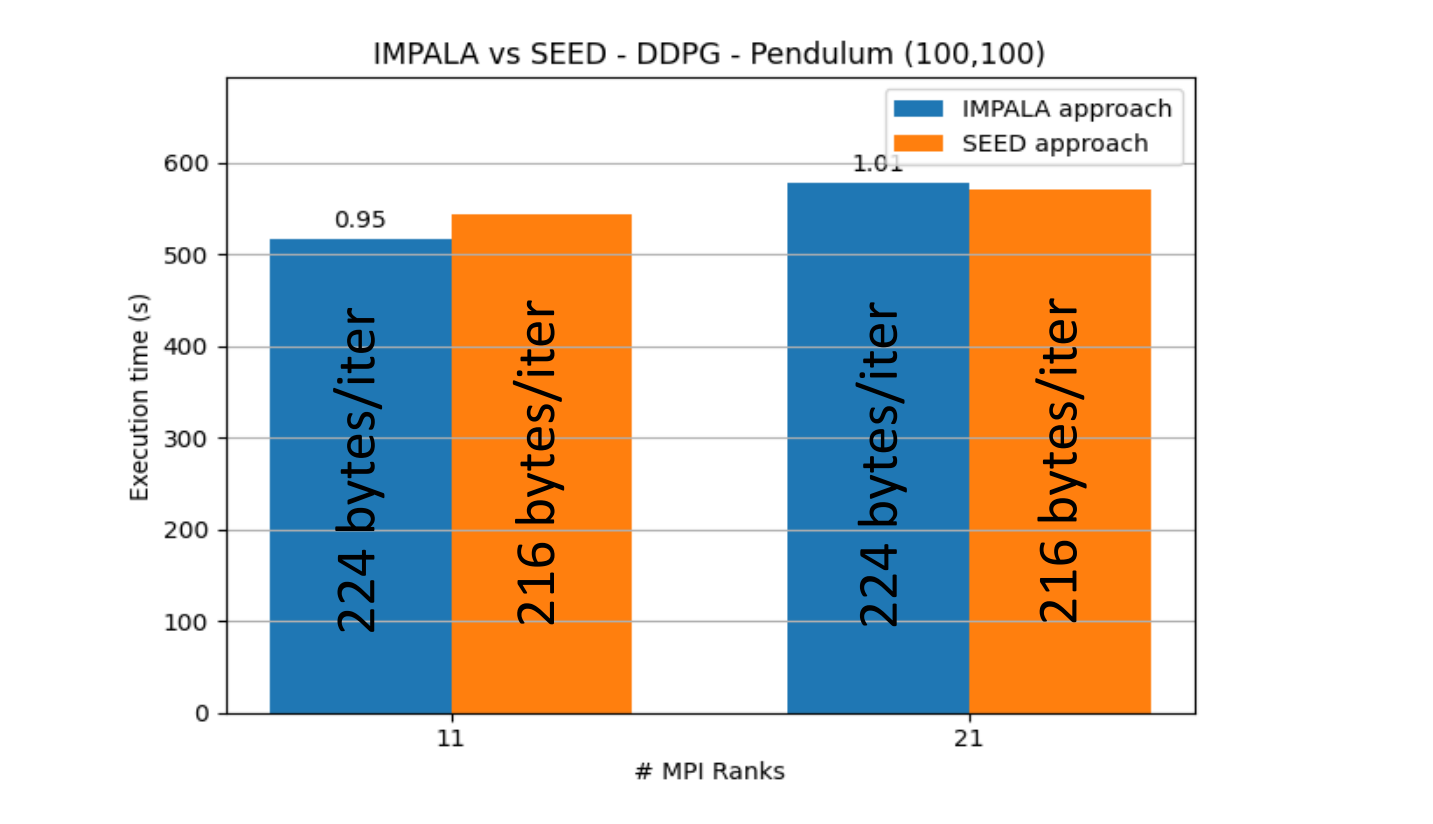

Figure 17. SEED vs IMPALA architecture (MPI implementations)

Additional Agents

ADVANTAGE ACTOR CRITIC WITH V-TRACE

- **Results:** (Asynchronous) Advantage Actor Critic (A2C/A3C) performs poorly without v-trace on the OpenAI gym CartPole environment but outperforms Deep Q-Network (DQN) when v-trace is added. Note that "convergence" in the CartPole environment is 200.
- A2C/A3C also converges faster in ExaBooster environment than DQN.
- These results are consistent with the literature, ie. A2C/A3C is expected to be the state-of-the-art after DQN.

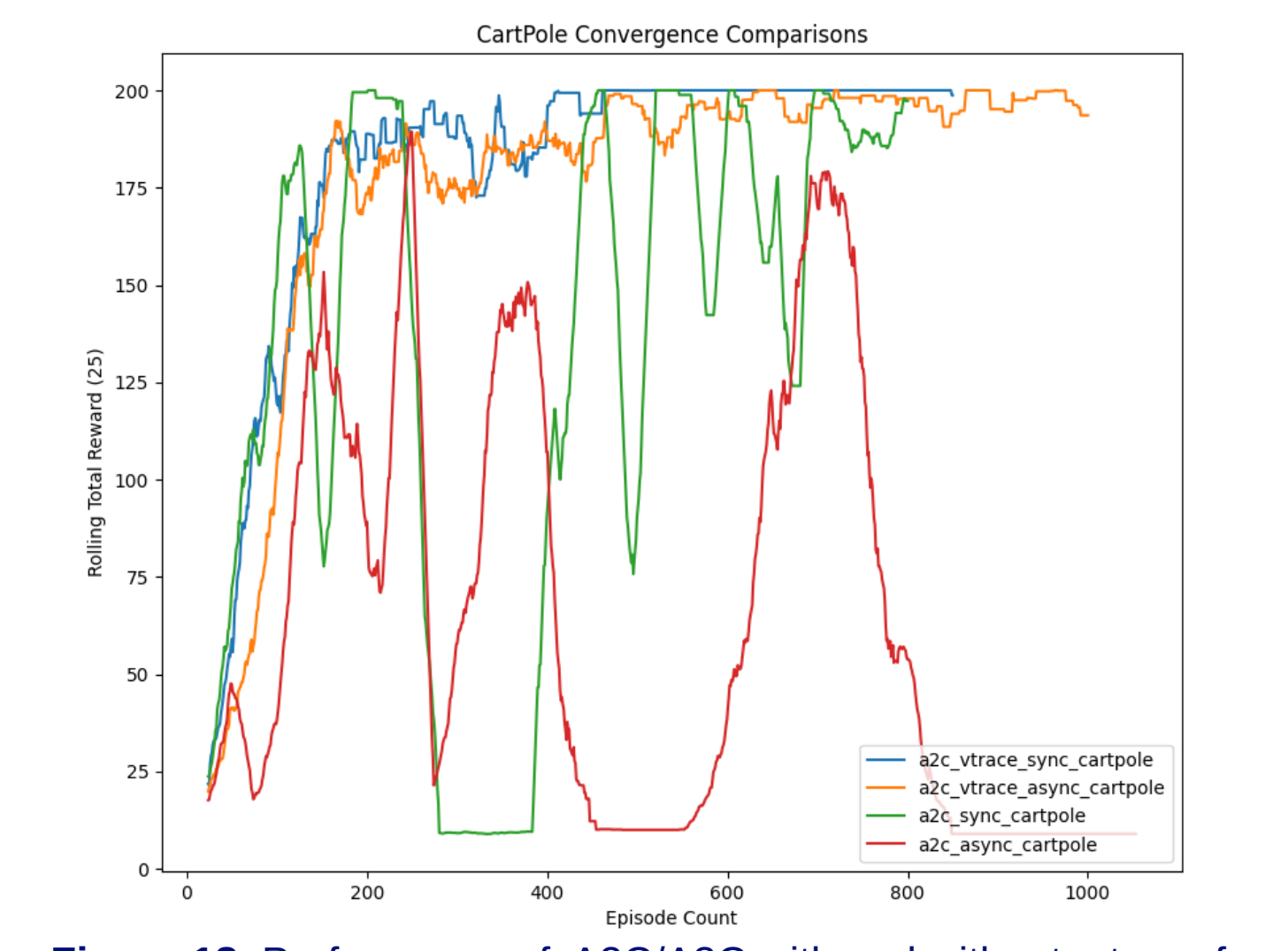

Figure 18. Performance of A2C/A3C with and without v-trace for CartPole environment. Note that A2C/A3C without v-trace does not converge, whilst A2C/A3C with v-trace does.

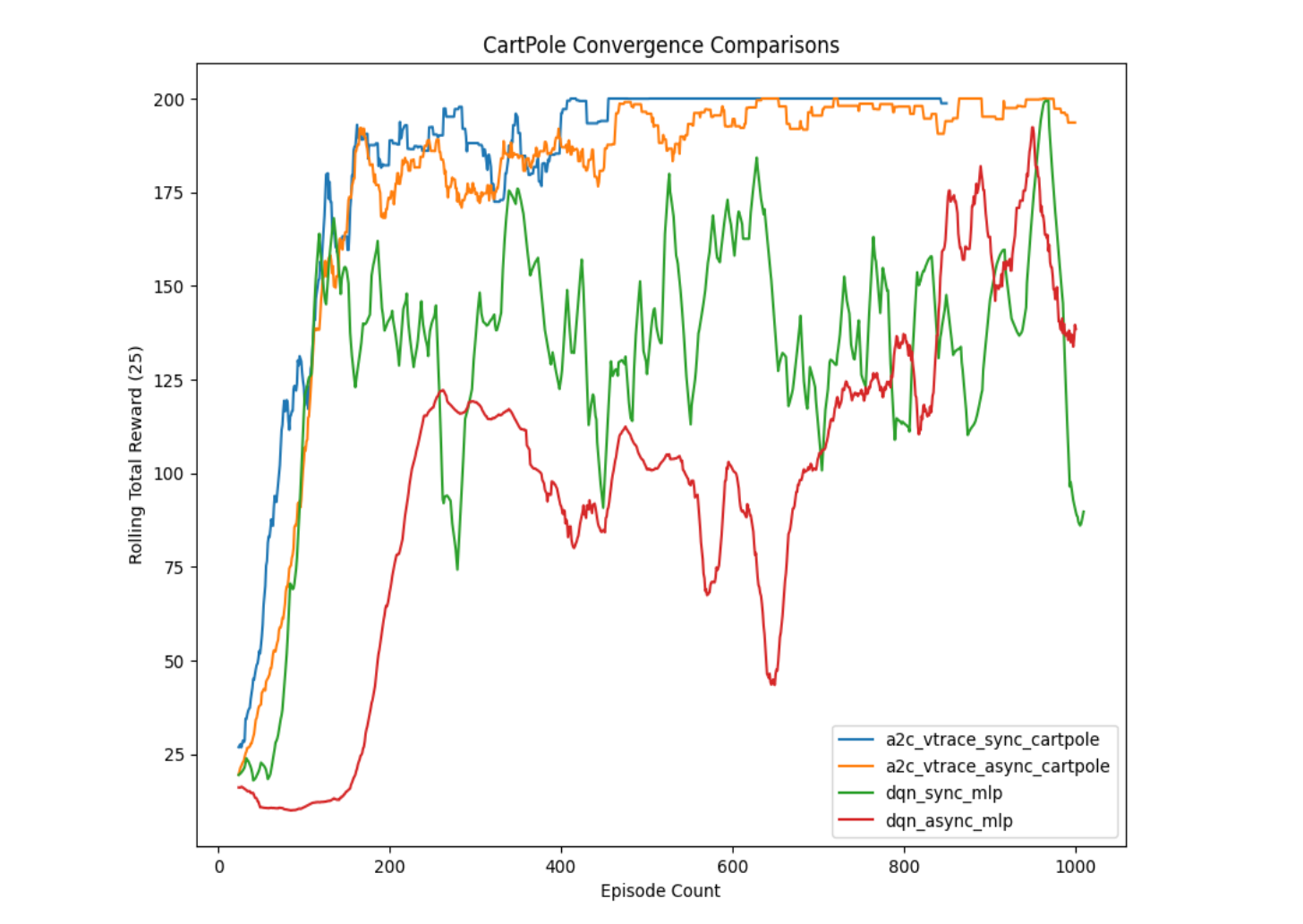

Figure 19. Performance of A2C/A3C versus DQN for CartPole environment. Note that DQN does not converge, whilst A2C/A3C with v-trace does.

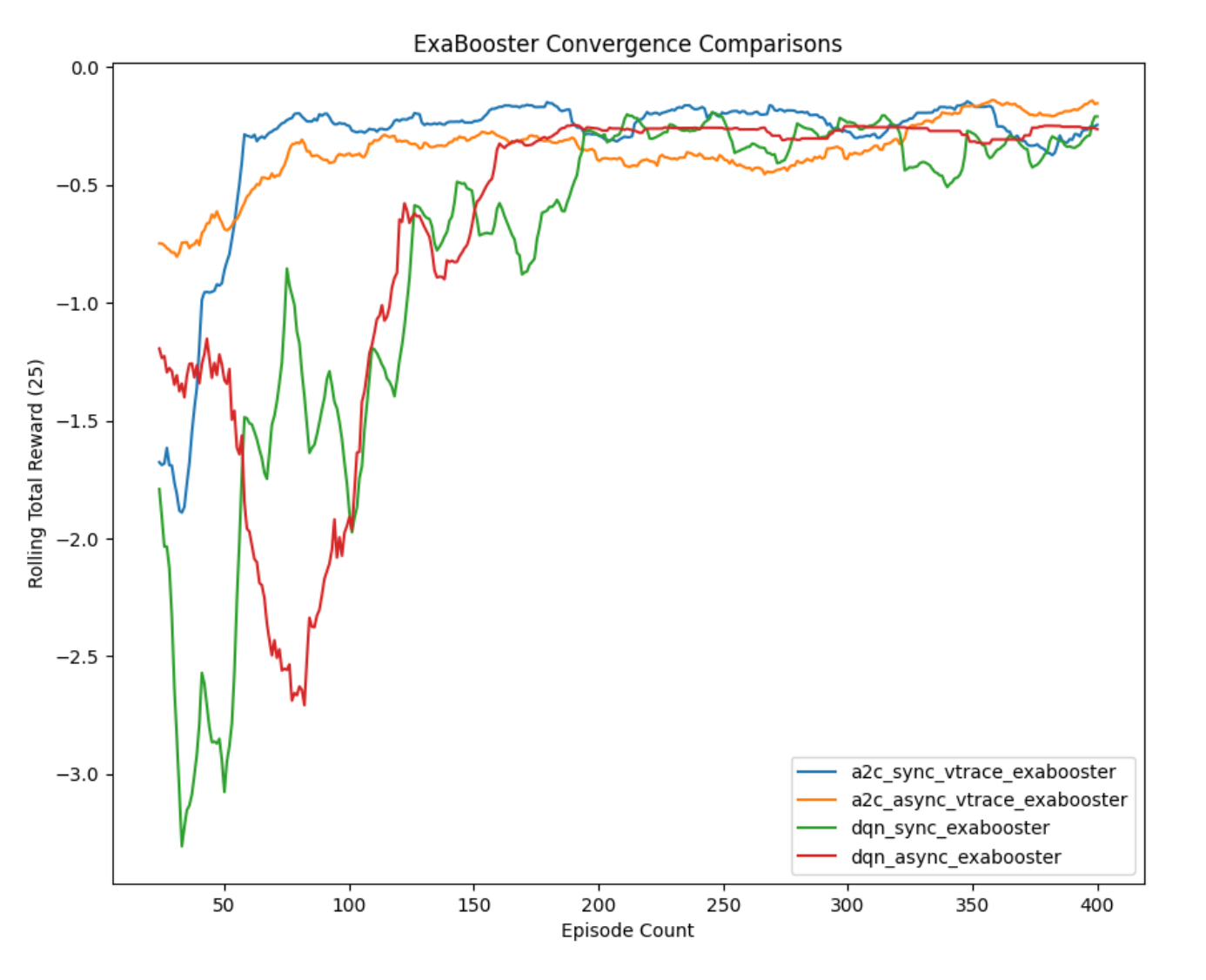

Figure 20. Performance of A2C/A3C (blue, orange) against DQN (green, red) for ExaBooster environment. Here A2C/A3C converges faster than DQN.

TWIN DELAYED DEEP DETERMINISTIC POLICY GRADIENT

- **Results:** Twin Delayed Deep Deterministic Policy Gradient Agent (TD3) has comparable performance to Deep Deterministic Policy Gradient (DDPG).

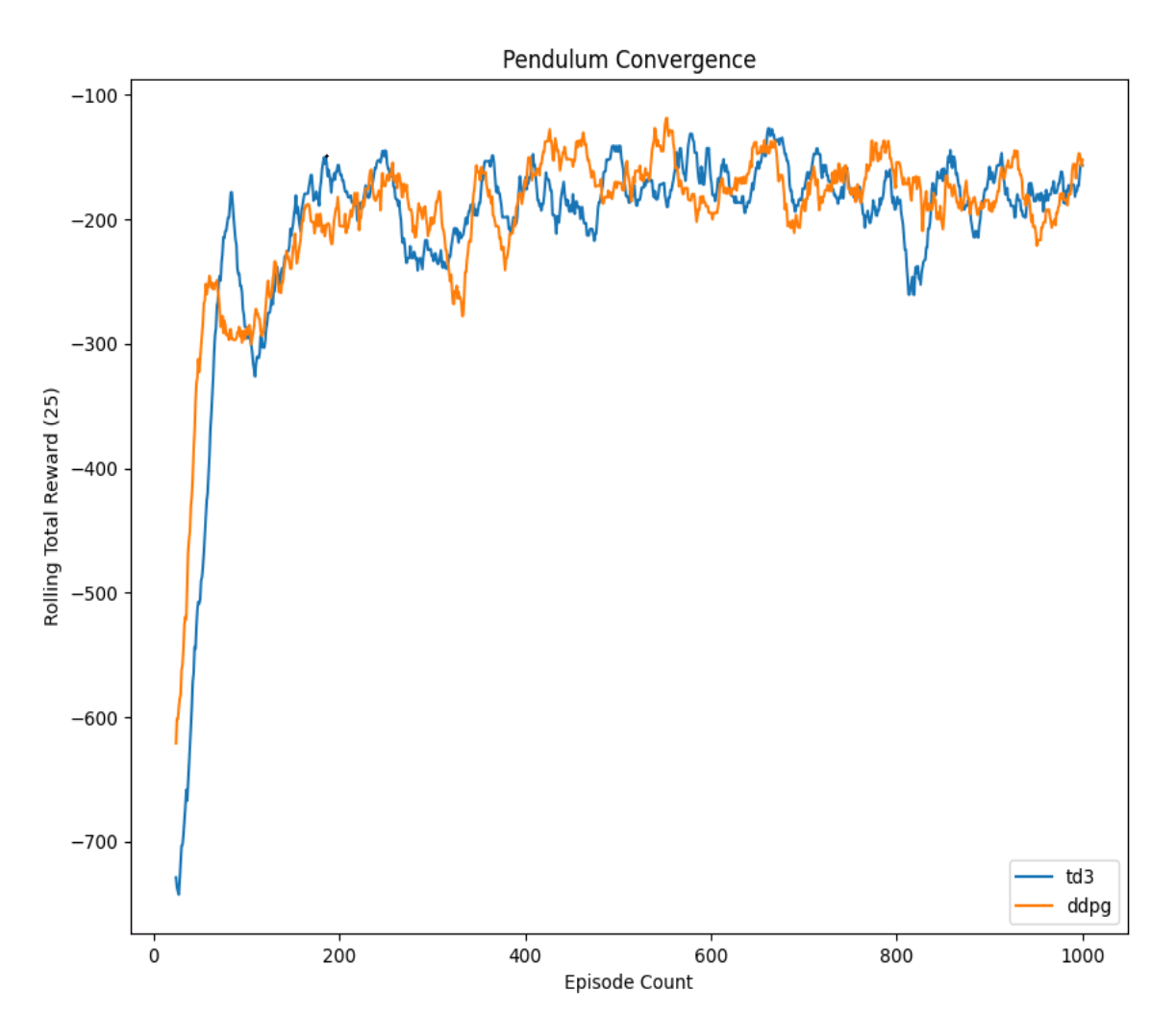

Figure 21. Performance of DDPG versus TD3.

DEEP DETERMINISTIC POLICY GRADIENT & PRIORITIZED EXPERIENCE REPLAY

- **Results:** Deep Deterministic Policy Gradient (DDPG) with Prioritized Experience Replay (PER) converges faster than without, which we would expect from the literature.

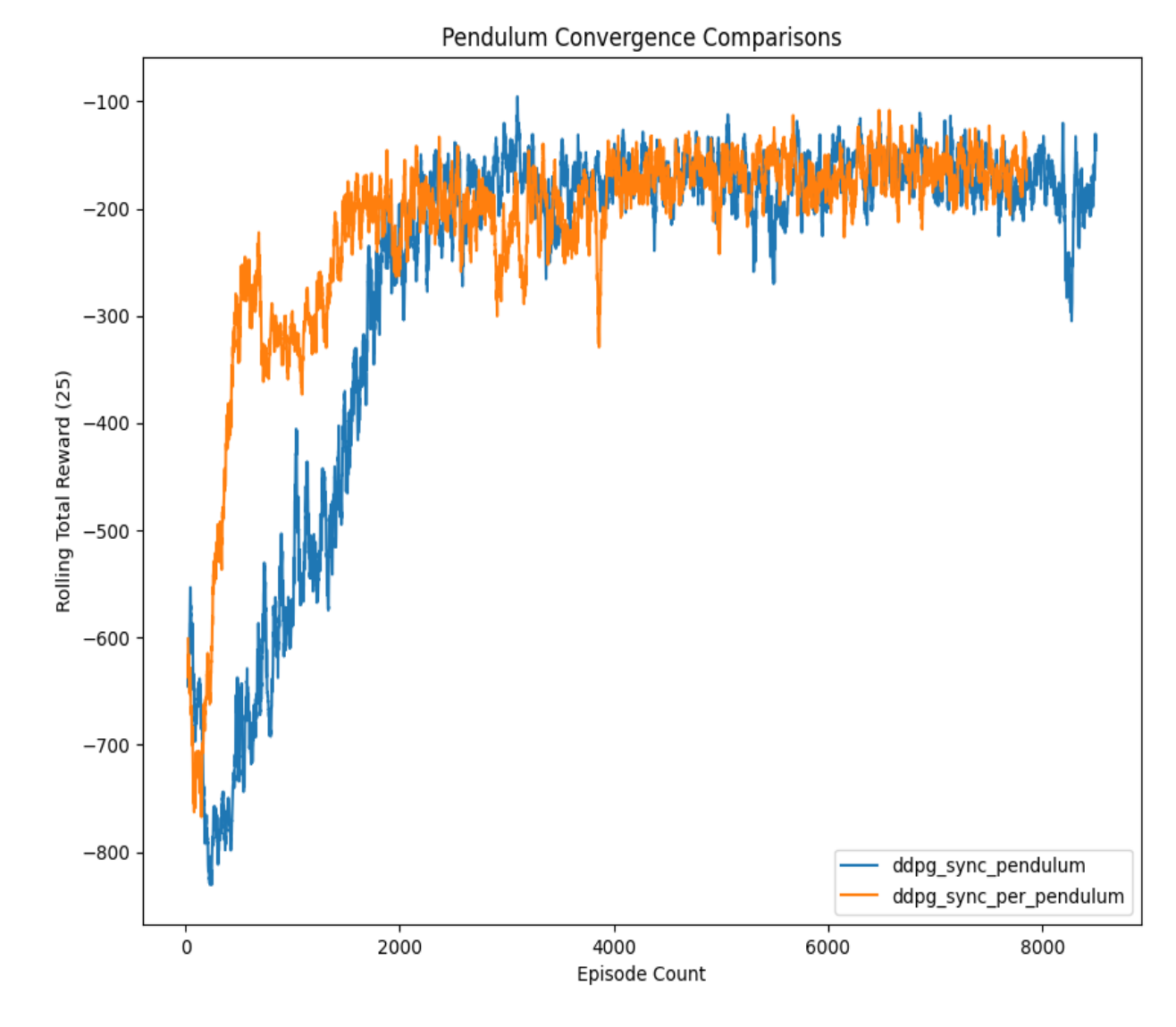

Figure 22. Performance of DDPG with and without PER.

* **All Tests run on Darwin cluster with Intel Broadwell (36 cores) + IB ConnectX-4**

Conclusions & Future Work

- EXARL is a scalable reinforcement learning framework for complex scientific environments.
- Improved upon the existing framework by accelerating the data generation pipeline for faster convergence.
- Demonstrated improved scalability performance using efficient RMA communication patterns.
- Expanded the capability of EXARL by including additional agents like A2C/A3C and TD3
- Explored algorithmic improvements such as v-trace and prioritized experience replay.
- **Future work:** Evaluate our improvements on complex scientific environments and scale our framework on large-scale systems.