Distributed TD3 Training with MPI

CSE 633 – Parallel Computing

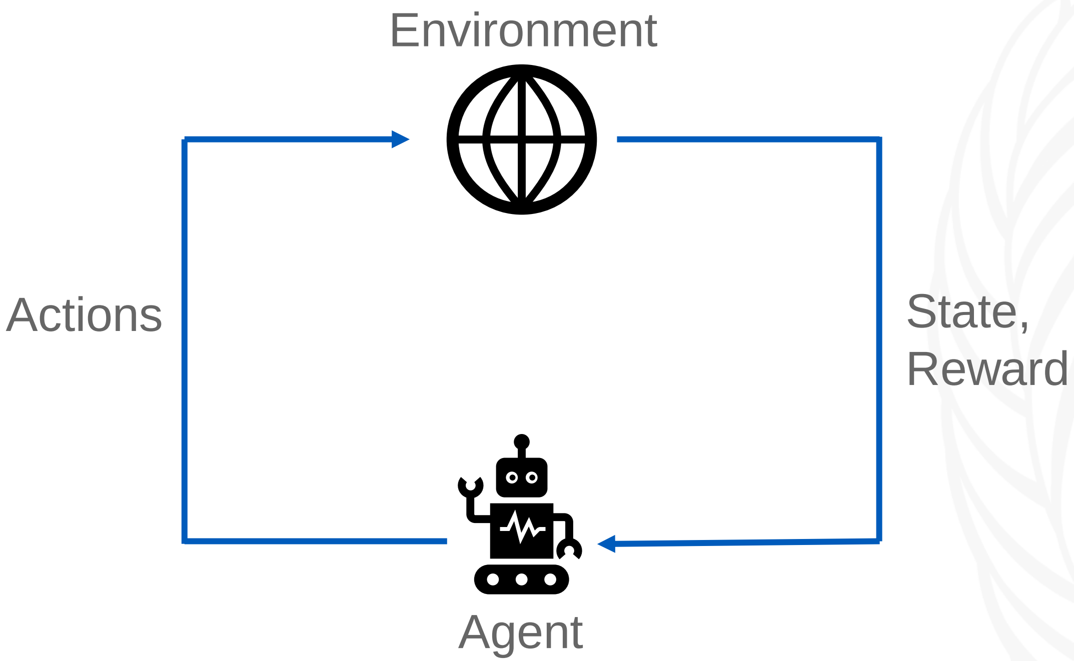
Instructor: Dr. Russ Miller

Presented by Elvis Rodrigues

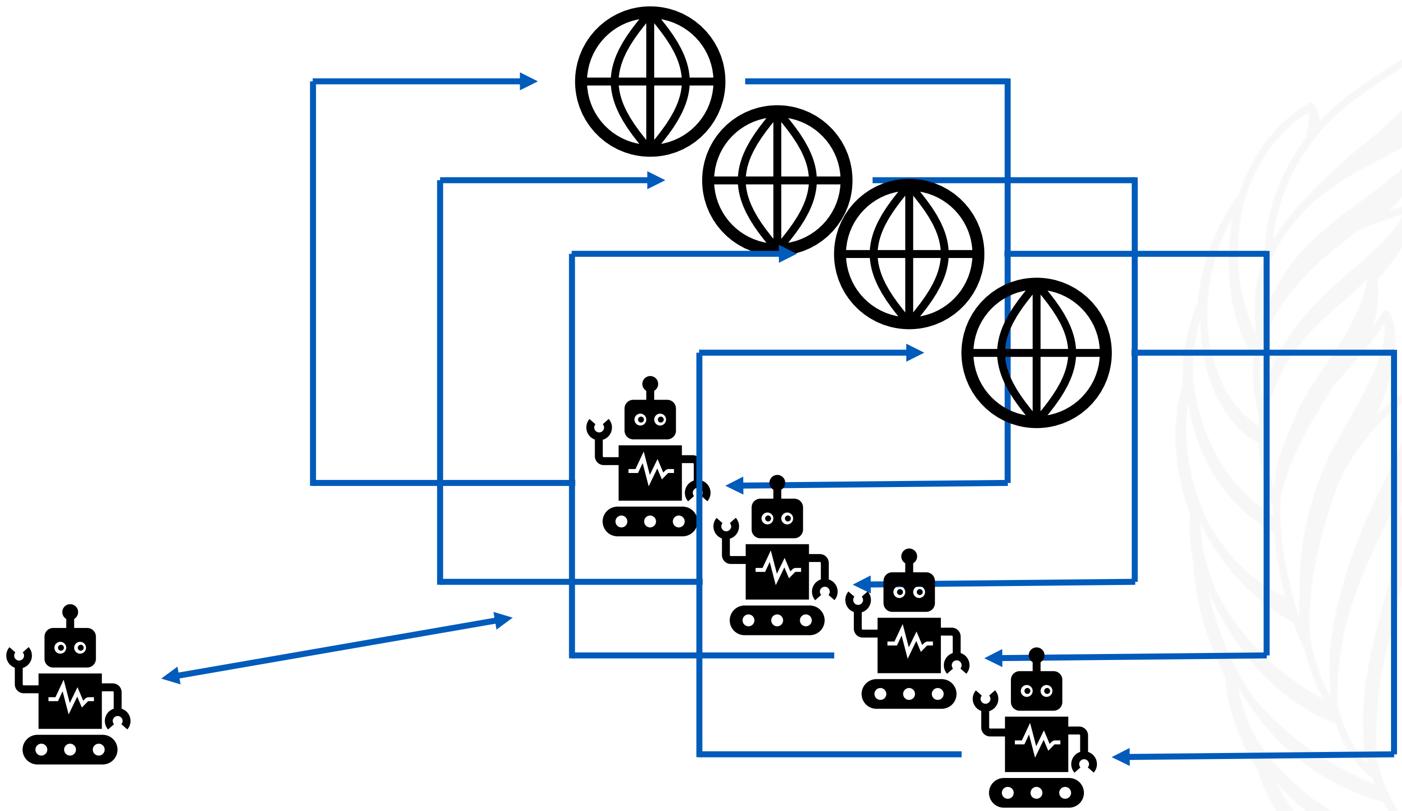
University at Buffalo The State University of New York

Reinforcement Learning

- Paradigm of machine learning algorithms with a focus on control problems.

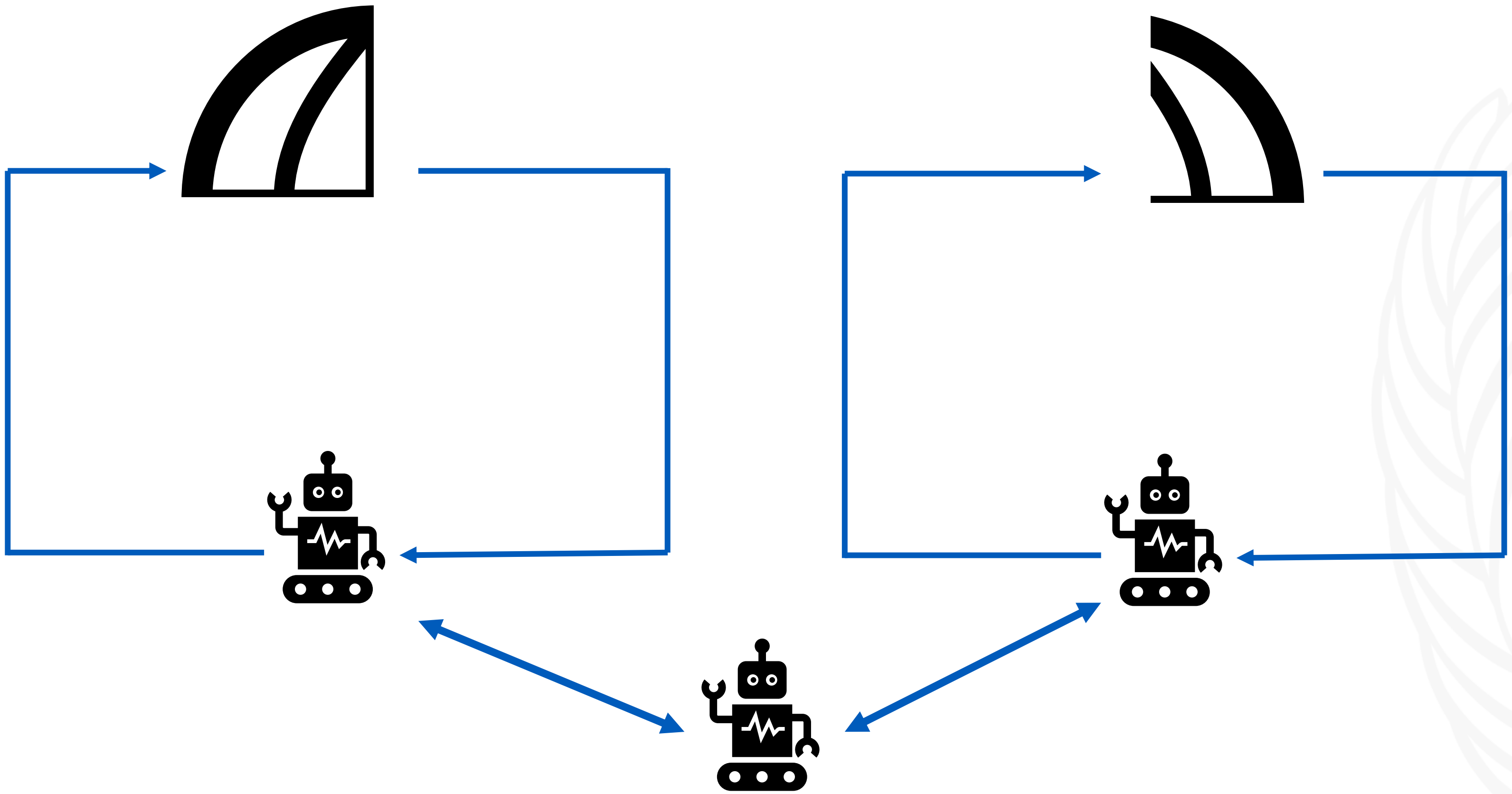


Distributed Reinforcement Learning



Akin to strong scaling

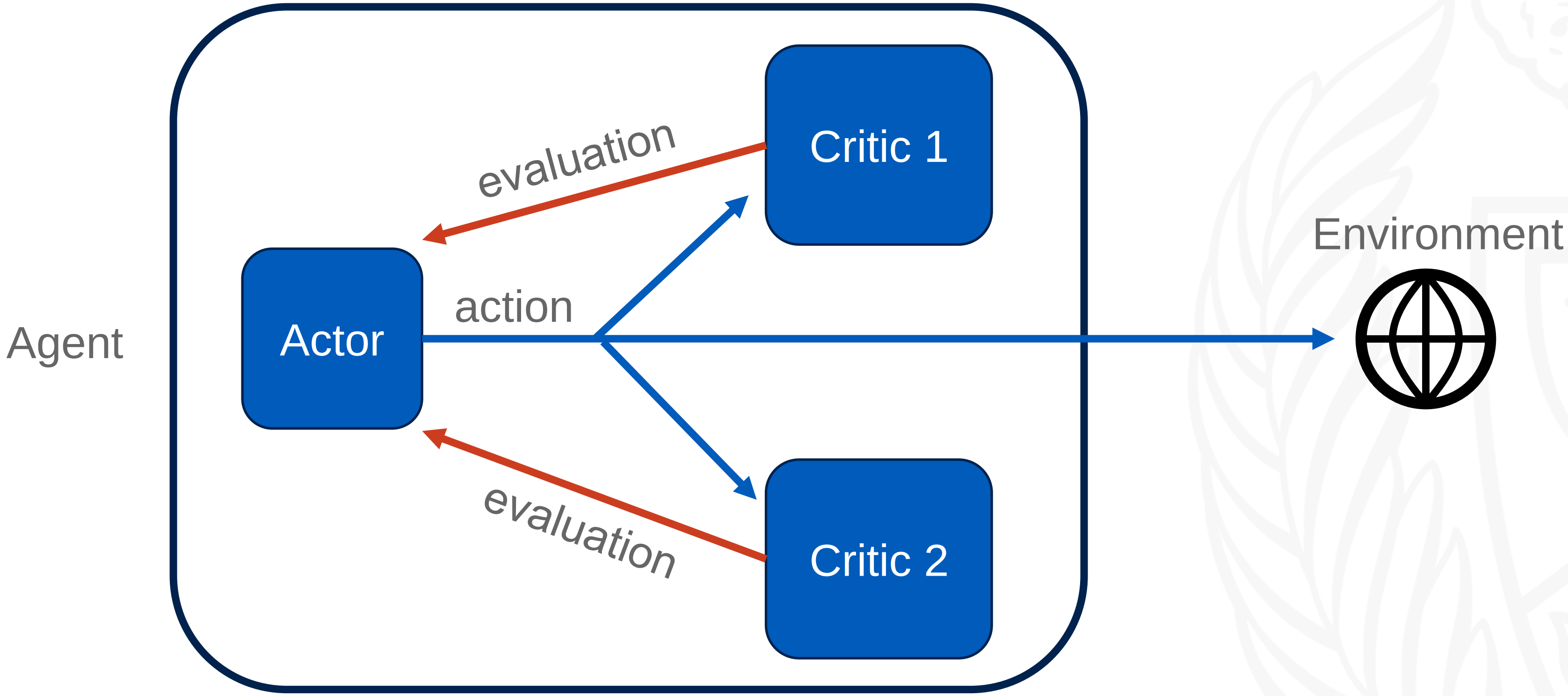
Sharded Environment



Akin to weak scaling

This was one of the goals, but unfortunately, I could not implement this in time.

Twin Delayed Deep Deterministic Policy Gradient (TD3)



MPI-TD3 Critic Psuedocode - Worker

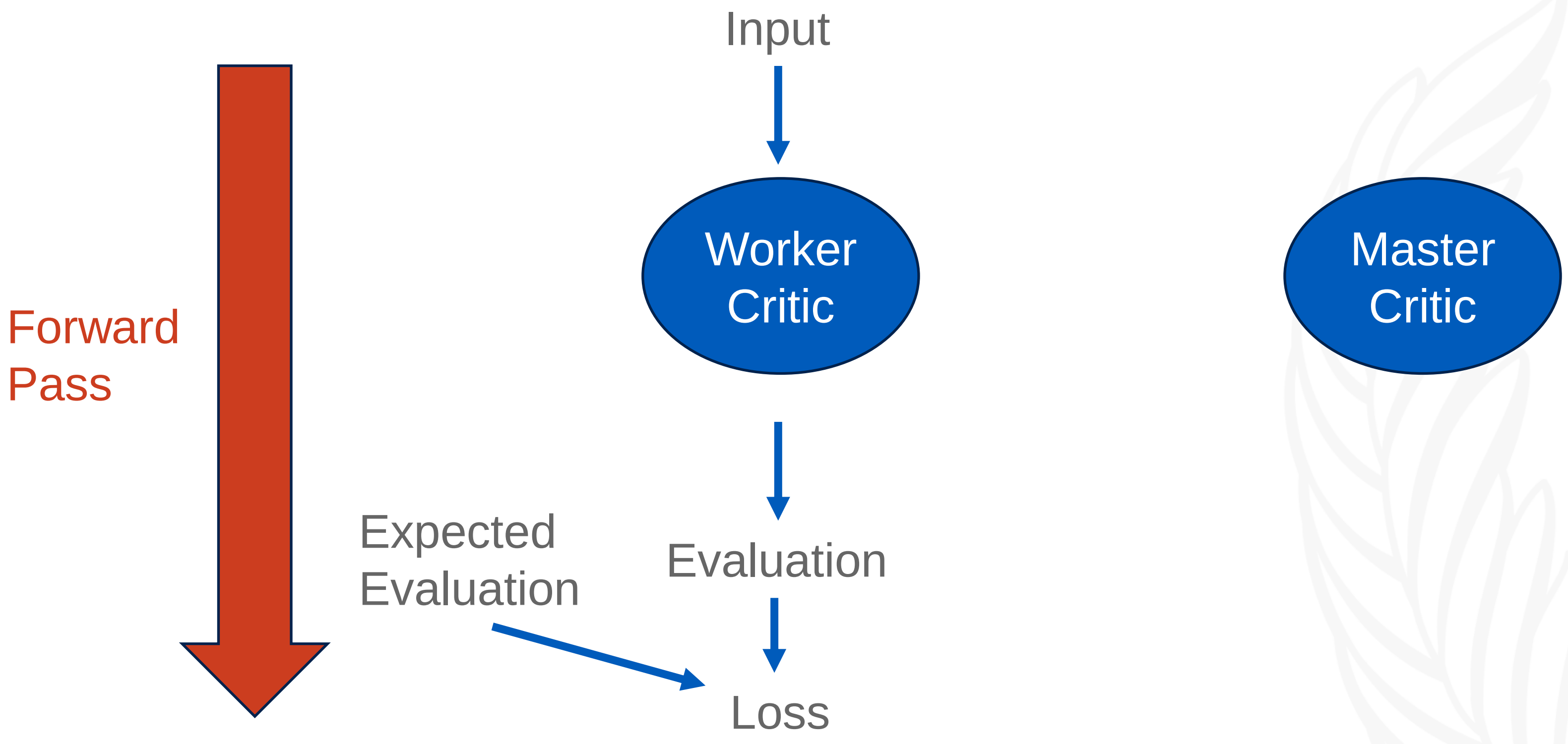
```
// Called at each step
function train():
        samples <- replay_buffer.sample()
        next_actions <- Actor(next_states)
        target_q1, target_q2 <- Critic(next_states, next_actions)
        current_q1, current_q2 <- Critic(states, actions)
        critic_loss <- L1_loss(current_q1, target_q1) +
        L1_loss(current_q2, target_q2)
        critic_loss.backward() // computes gradients
        // MPI calls below are simplified; done in PyTorch
        MPI_gather(critic.grad)
        MPI_broadcast(critic.params, 0) // receive master's parameters
```

Actor pseudocode is omitted, but its implementation is similar

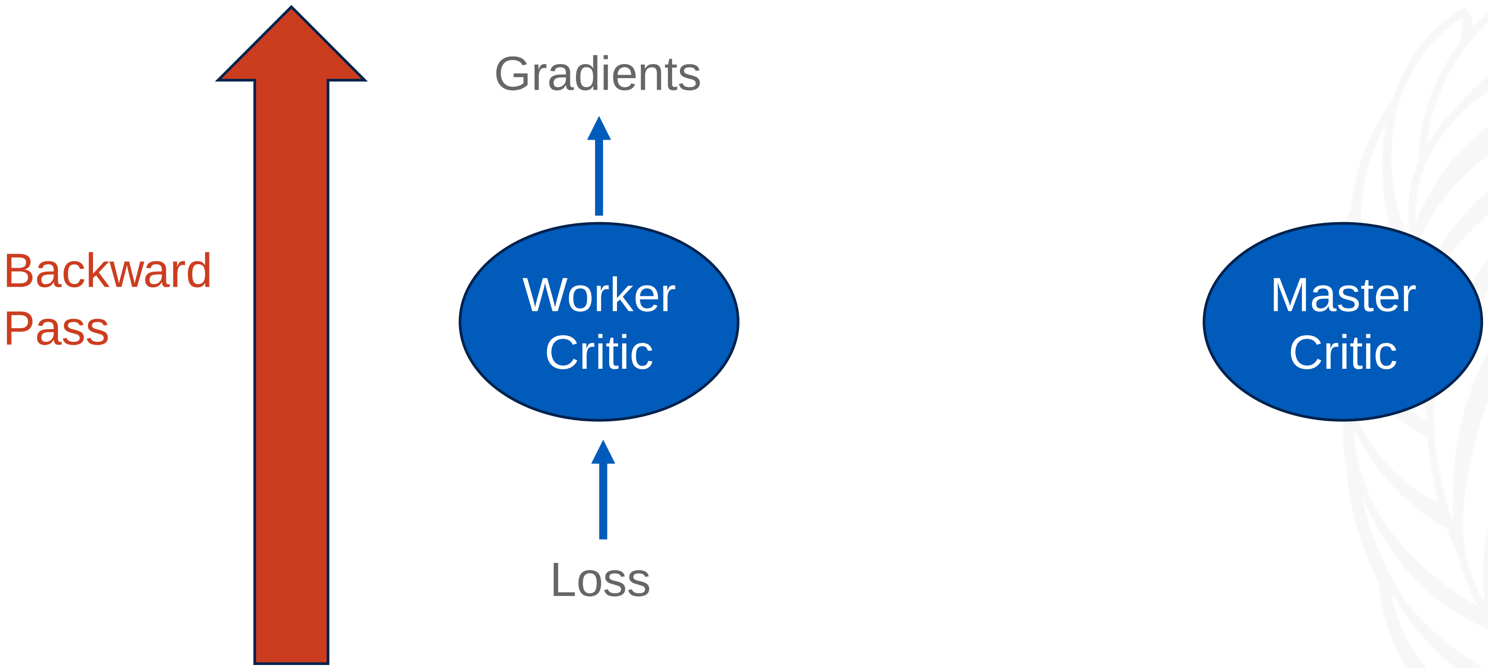
MPI-TD3 Critic Psuedocode - Master

```
// Called at each step by Rank 0
function do_update():
        // MPI calls below are simplified; done in PyTorch
        MPI_gather(critic.grad, 0) // receive gradients from workers
        for each worker:
                copy gradients to critic network
                update network parameters by stepping
        MPI_broadcast(critic.params) // broadcast master’s parameters
```

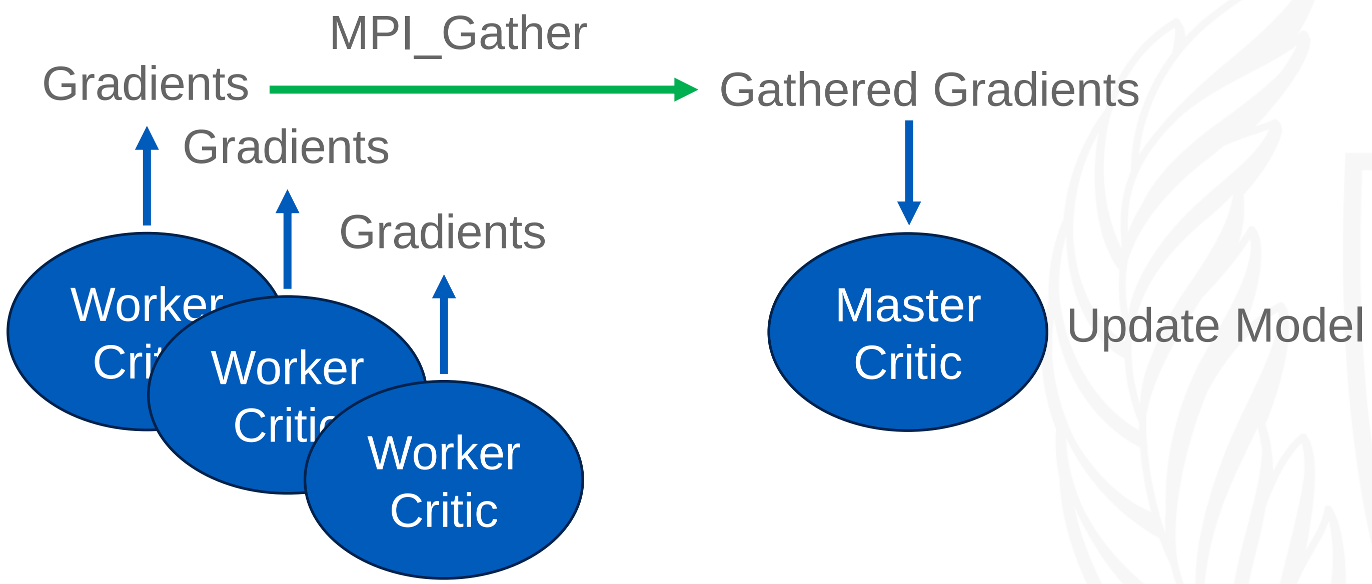
Distributed TD3 – Forward Pass



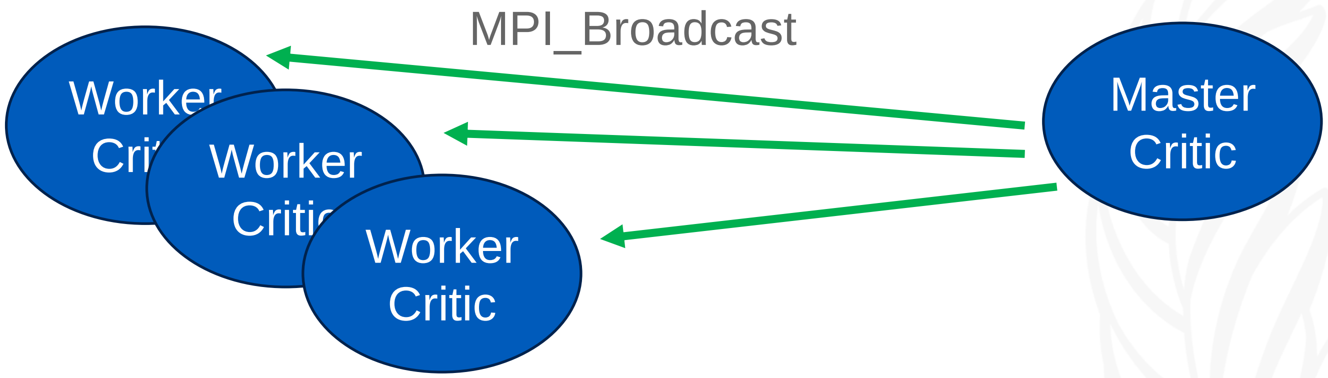
Distributed TD3 – Backward Pass



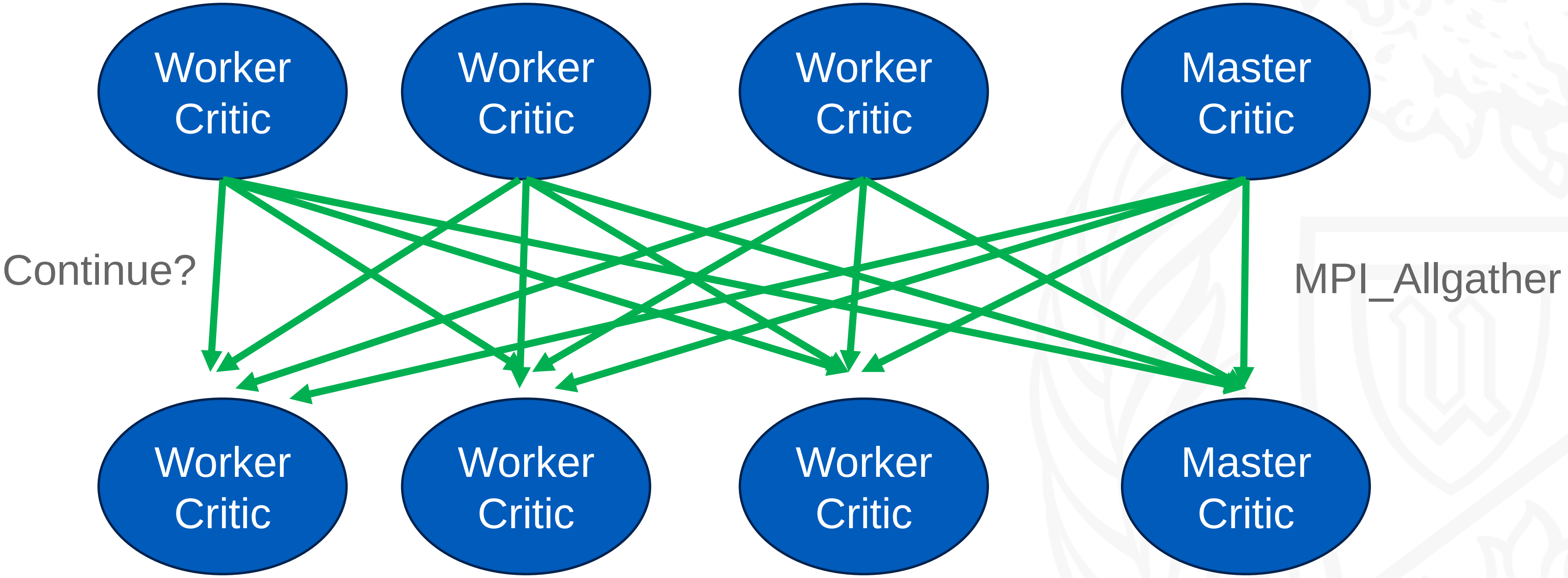
Distributed TD3 – Model Update



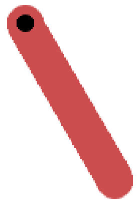
Distributed TD3 – Parameter Broadcast



Distributed TD3 – Health Check



Environment

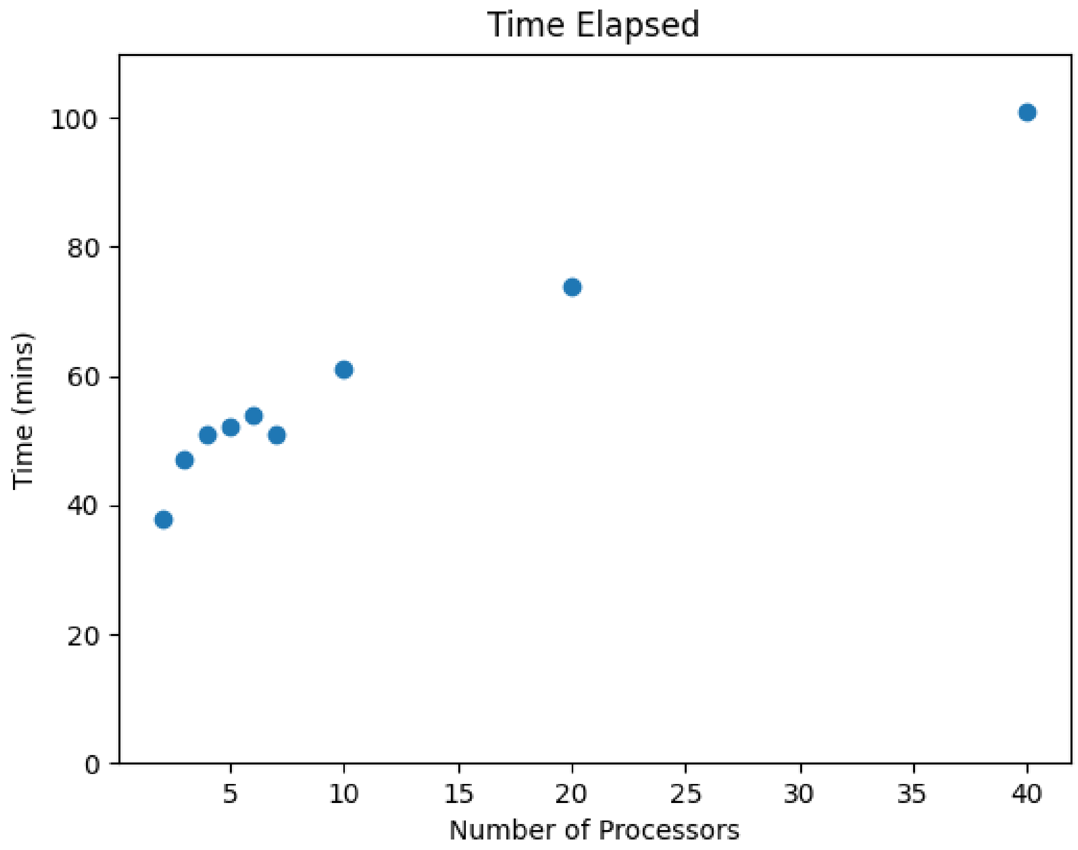


Pendulum-v1 from Gymnasium

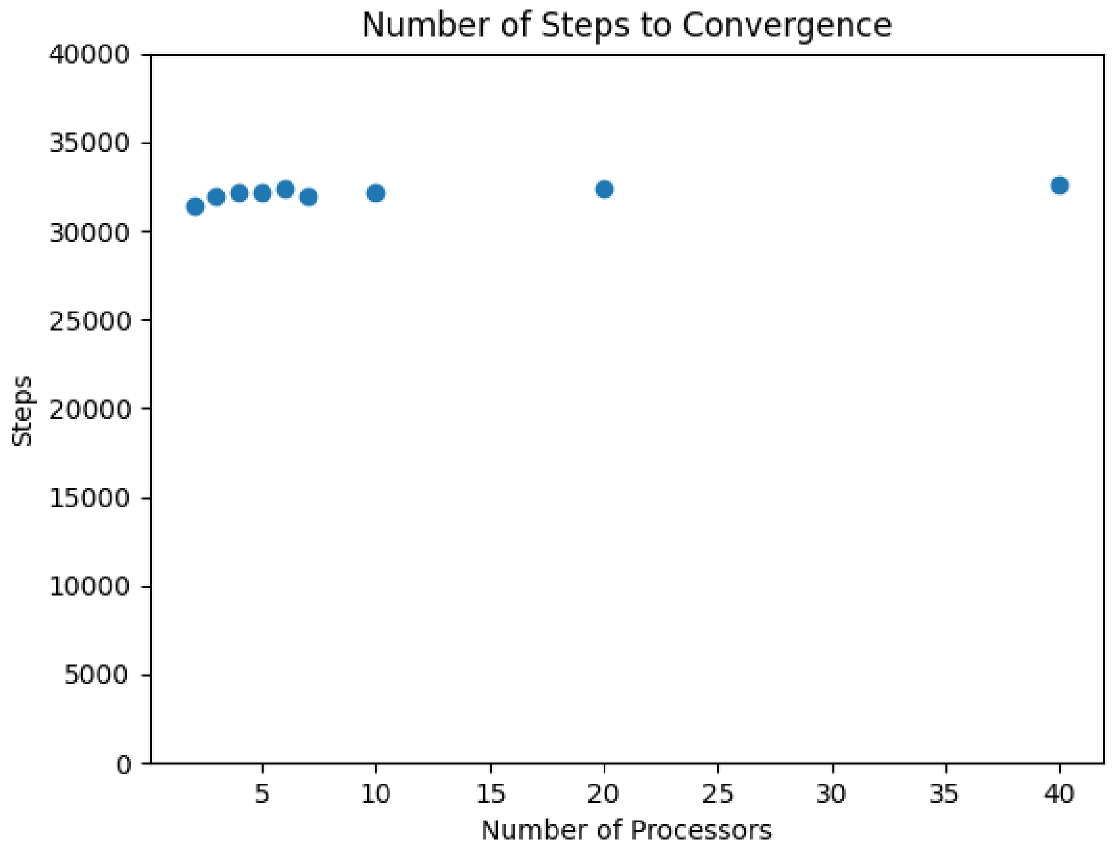
Continuous state and action space

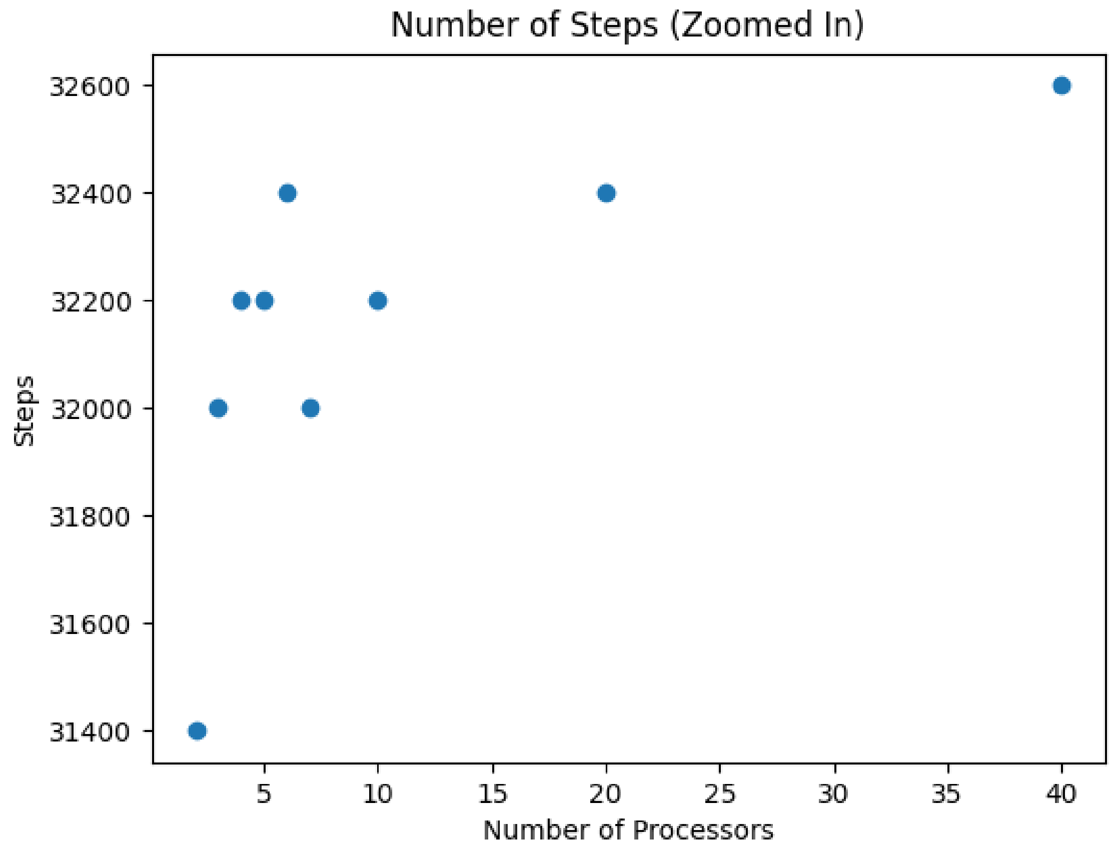
Episode reward cutoff is -200

Results – Time Elapsed

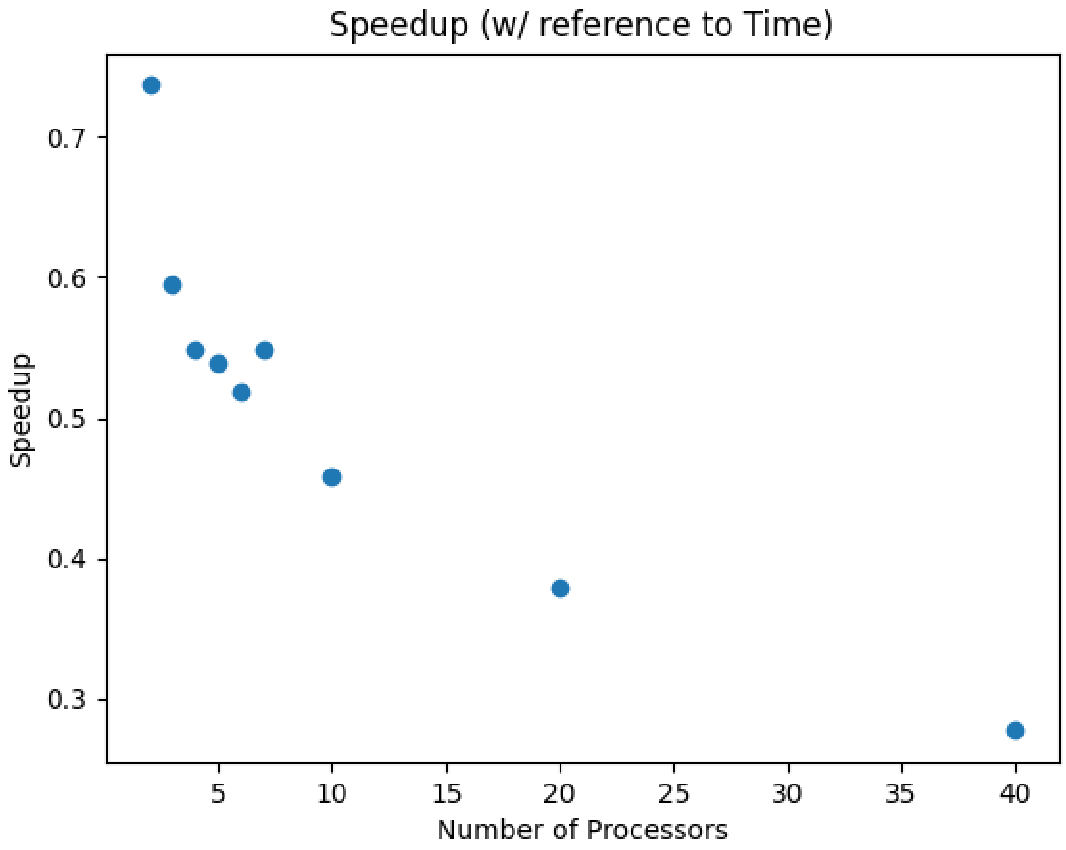


Results – Steps Elapsed





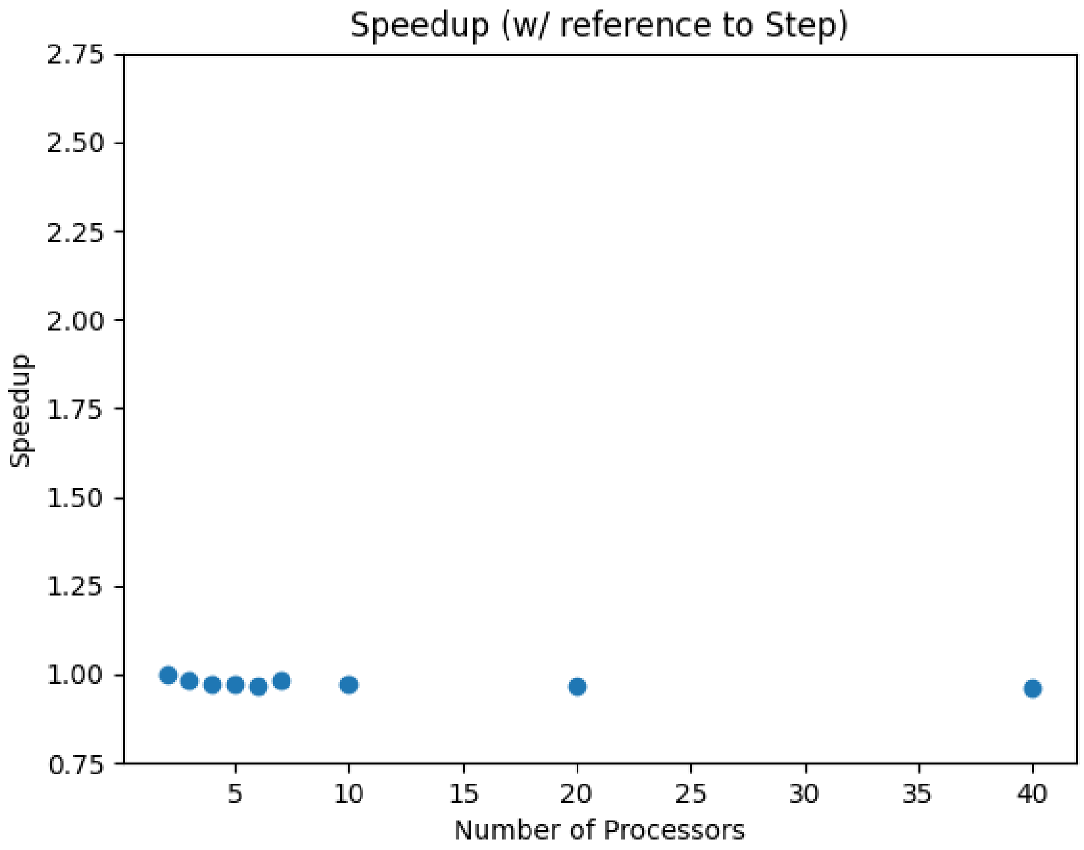
Results – Time Speedup

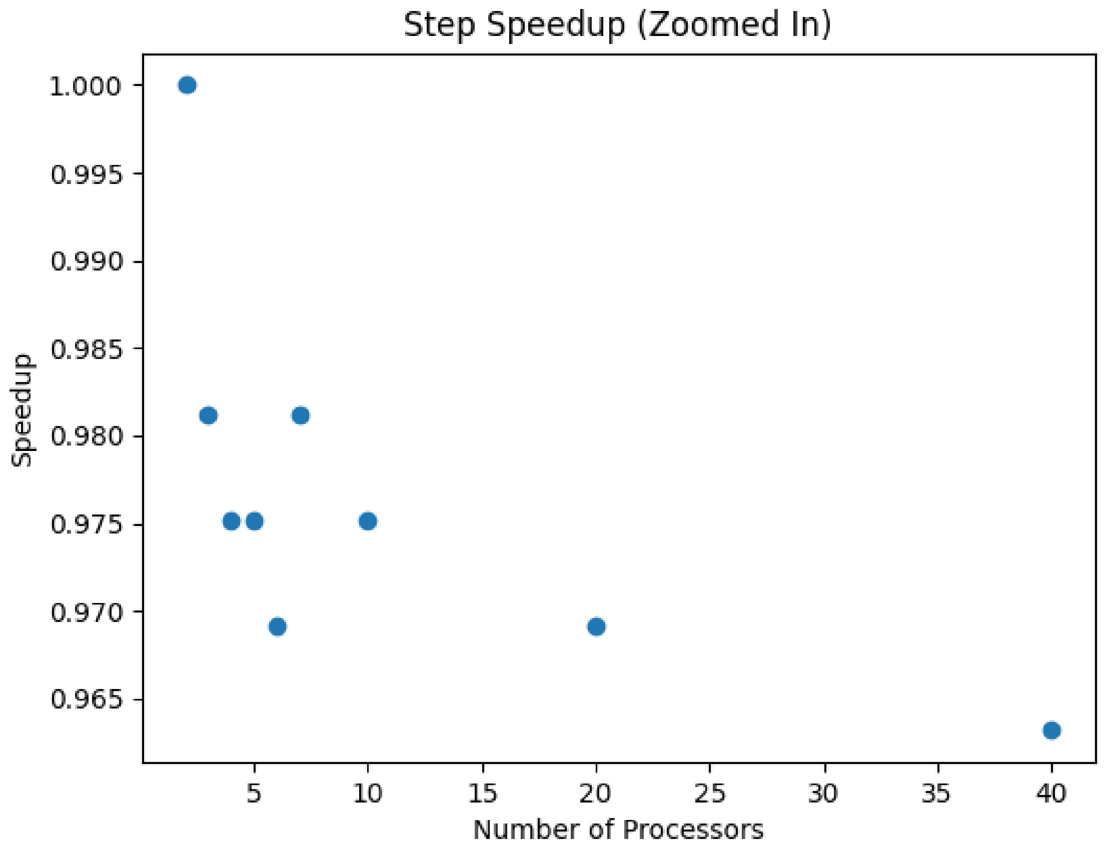


Cost of sending gradients and receiving weights potentially outweighs any benefits from distributed training.

This is likely because of the significant overhead of encoding Python objects.

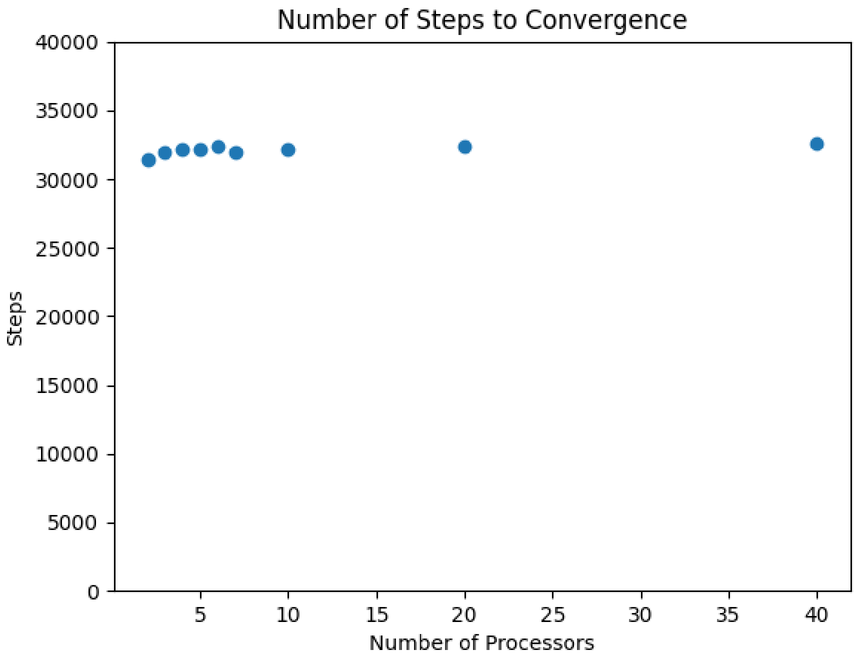
Results – Step Speedup





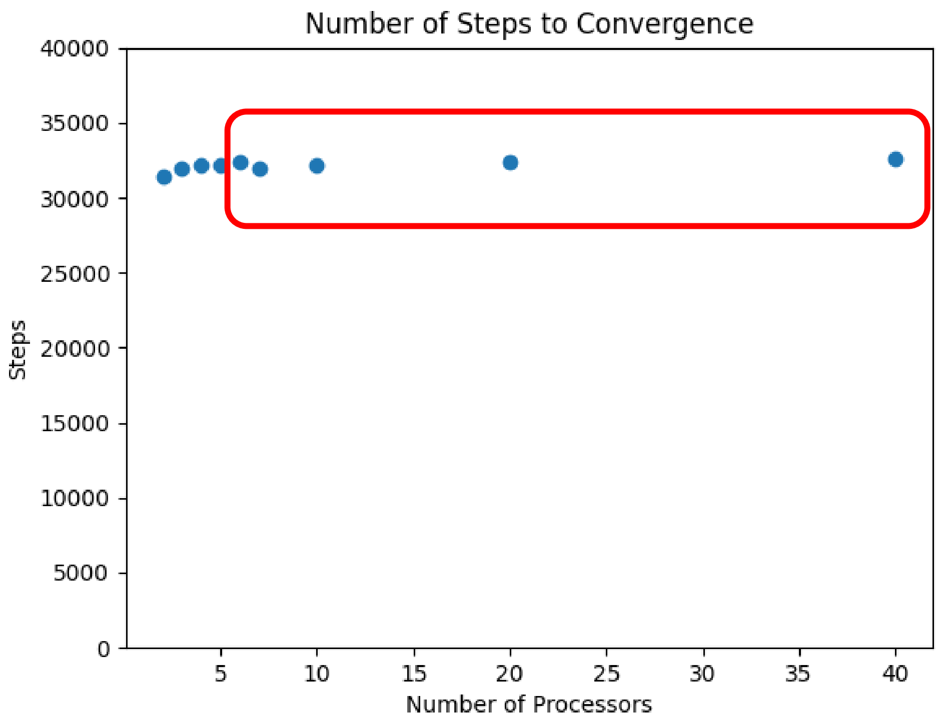
Potential Issues: Buggy Implementation

- Master agent does learn, but most workers are idle.



On some runs, CPU utilization in some nodes was near 0%

Potential Issues: Bad Environment Choice

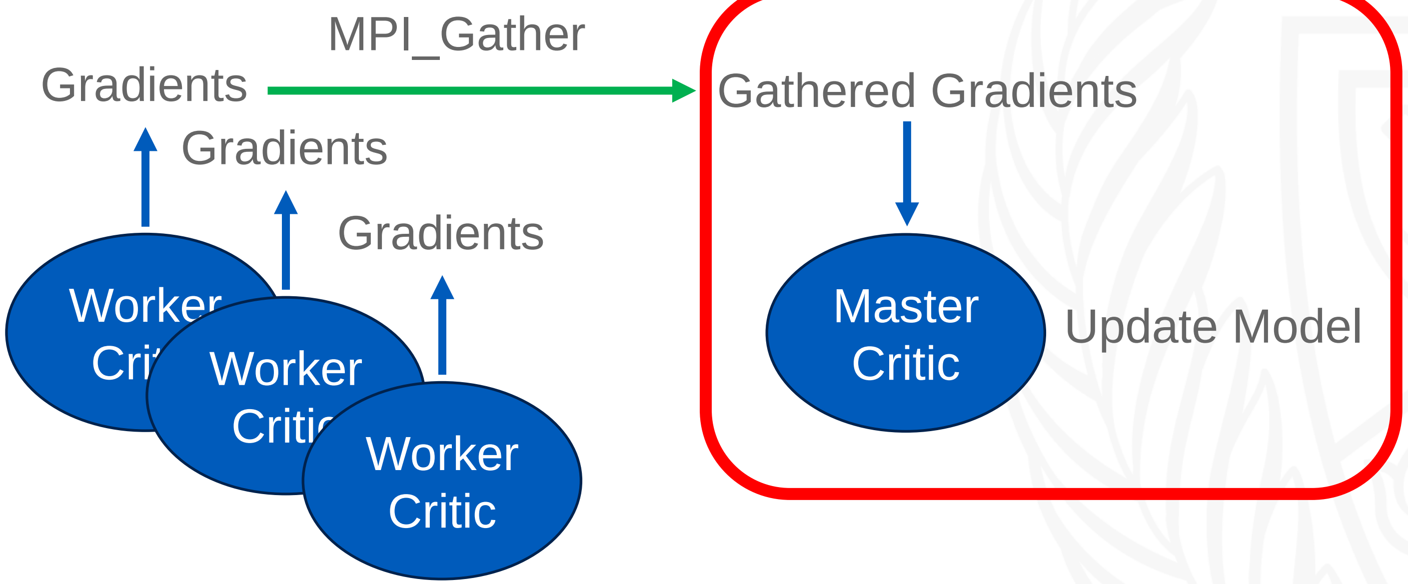


Environment may be too ‘simple’.

This can be confirmed with more complex environments.

Potential Issues: Serial Work

If the gradients here are not ‘diverse’, then most of the work in the bottlenecked part could be equivalent to serial work.



Potential Future Work

- True asynchronous training without MPI_gather and MPI_broadcast
- Decentralized version that fetches gradients with MPI_Allgather
- Environment sharding for intractable environments

References

- Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1587–1596. URL: https: //proceedings.mlr.press/v80/fujimoto18a.html.
- Stephen Dankwa and Wenfeng Zheng. "Twin-Delayed DDPG: A Deep Reinforcement Learning Technique to Model a Continuous Movement of an Intelligent Robot Agent". In: *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*. ICVISP 2019. Vancouver, BC, Canada: Association for Computing Machinery, 2020. ISBN: 9781450376259. DOI: 10.1145/3387168.3387199. URL: https://doi.org/10.1145/3387168.3387199.
- Jiaolv Wu et al. "A-TD3: An Adaptive Asynchronous Twin Delayed Deep Deterministic for Continuous Action Spaces". In: *IEEE Access* 10 (2022), pp. 128077–128089. DOI: 10.1109/ACCESS.2022.3226446.