Asynchronous Reinforcement Learning on Multicore CPUs

Daniel Gardin

Deep reinforcement learning (DRL) has achieved remarkable progress in sequential decision-making tasks through the use of **function approximation**, which enables machine learning methods to address discrete optimization problems. While most modern implementations rely on GPU acceleration for neural network training [1], **Reinforcement Learning remains predominantly CPU-bound** in many practical settings [2]. This limitation arises because policy optimization depends not only on gradient computation but also on continuous interaction with simulators that generate experience. These interactions may involve complex control logic, stochastic transitions, and reward computations that are difficult to parallelize efficiently on GPUs.

This project investigates the **implementation and performance characteristics of asynchronous reinforcement learning (RL) algorithms** on multicore CPU architectures. Each environment step involves state transitions, reward evaluation, and termination checks, which are often sequential and computationally demanding. As a result, the rate of experience generation, rather than the speed of gradient computation, frequently becomes the dominant bottleneck, making the parallelization of data collection a direction for improving algorithmic efficiency. Thus, by leveraging multicore CPUs to run multiple simulators concurrently, agents can collect data asynchronously and sustain high throughput without specialized hardware.

The central objective is to design and evaluate a **modular C implementation** of asynchronous actor–critic and policy gradient algorithms [3], including **A3C**, **A2C**, and **REINFORCE**. Each learning thread interacts with a local environment replica, computes gradients independently, and periodically synchronizes updates to a shared global parameter vector using **MPI**. The analysis focuses on **convergence behavior**, **communication overhead**, and **scalability** as the number of threads increases.

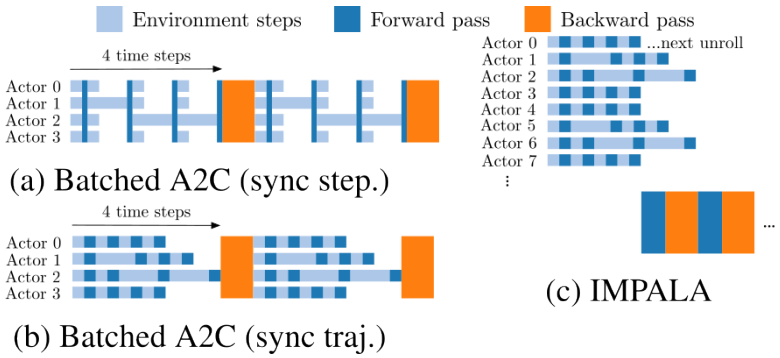

Figure 1: Comparison of parallelism in A2C and IMPALA algorithms. In A2C, simulation pauses during gradient updates, while IMPALA decouples simulation and learning by performing updates asynchronously (adapted from Espeholt et al. [4]).

To evaluate performance, experiments are conducted on **small discrete-control and grid-based navigation tasks**, chosen for reproducibility and controlled stochasticity [5]. Evaluation metrics include **wall-clock convergence time**, **policy stability**, and **sample efficiency** as functions of available CPU cores. Comparisons with sequential baselines quantify computational gains and reveal how asynchronous updates affect both **learning variance** and **overall scalability**.

References

[1] Jacky Liang, Viktor Makoviychuk, Ankur Handa, Nuttapong Chentanez, Miles Macklin, and Dieter Fox. Gpu-accelerated robotic simulation for distributed reinforcement learning. In *Conference on Robot Learning*, pages 270–282. PMLR, 2018.

[2] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[3] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

[4] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[5] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.