Journal of Machine Learning Research 23 (2022) 1-6 Submitted 2/22; Revised 8/22; Published 9/22

WarpDrive: Fast End-to-End Deep Multi-Agent Reinforcement Learning on a GPU

Tian Lan* TIAN.LAN@SALESFORCE.COM
Sunil Srinivasa* SSUNIL@GMAIL.COM
Huan Wang HUAN.WANG@SALESFORCE.COM
Stephan Zheng STEPHAN.ZHENG@SALESFORCE.COM
Salesforce Research
Palo Alto, CA 94301, USA

Editor: Joaquin Vanschoren

Abstract

WarpDrive is a flexible, lightweight, and easy-to-use open-source framework for end-to-end deep multi-agent reinforcement learning (MARL) on a Graphics Processing Unit (GPU), available at `https://github.com/salesforce/warp-drive`. It addresses key system bottlenecks when applying MARL to complex environments with high-dimensional state, observation, or action spaces. For example, WarpDrive eliminates data copying between the CPU and GPU and runs thousands of simulations and agents in parallel. It also enables distributed training on multiple GPUs and scales to millions of agents. In all, WarpDrive enables orders-of-magnitude faster MARL compared to common CPU-GPU implementations. For example, WarpDrive yields 2.9 million environment steps/second with 2000 environments and 1000 agents (at least 100× faster than a CPU version) in a 2d-Tag simulation. It is user-friendly: e.g., it provides a lightweight, extendable Python interface and flexible environment wrappers. It is also compatible with PyTorch. In all, WarpDrive offers a platform to significantly accelerate reinforcement learning research and development.

Keywords: Deep Reinforcement Learning, Multi-Agent Systems, GPU acceleration.

Introduction. Deep reinforcement learning (RL) is a powerful framework to train AI agents, e.g., in strategy games (OpenAI, 2018; Vinyals et al., 2019) and robotics (Gu et al., 2017). In particular, multi-agent[1] systems, especially those with many agents, are a frontier for RL research and applications; multi-agent RL (MARL) is relevant to economics (Zheng et al., 2022; Trott et al., 2021), dialogue agents (Li et al., 2016), and many other fields.

However, there are still many engineering and scientific challenges to the use of (multi-agent) RL. From an engineering perspective, RL implementations can be slow when simulations have many agents and high-dimensional state, observation, or action spaces, with experiments taking days or even weeks. For context, common distributed RL systems often use a set of *roll-out* and *trainer* workers. The roll-out workers run the environment to generate roll-outs, using the actions sampled from the policy models on the roll-out workers (Pretorius et al., 2021; Hoffman et al., 2020; Espeholt et al., 2018) or trainer workers (Espeholt et al., 2020). Roll-out workers typically use CPU machines, and sometimes,

∗. Tian Lan and Sunil Srinivasa contributed equally.

1. An *agent* is an actor in an environment. An *environment* is an instance of a simulation and may include many agents with complex interactions. An agent is neither an environment nor a policy model.

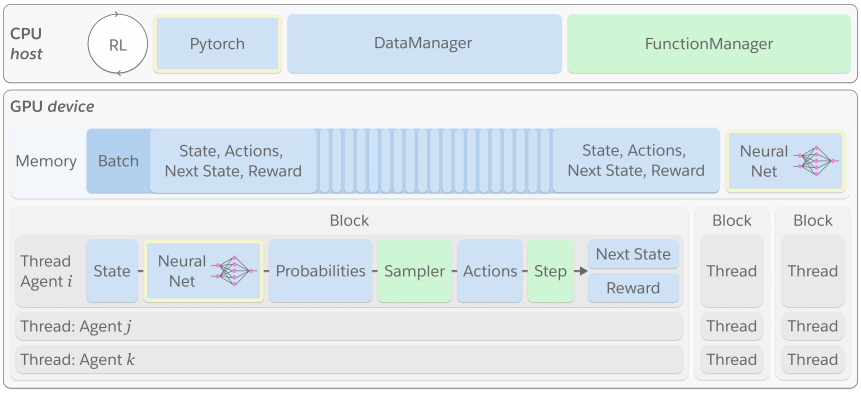

Figure 1: **WarpDrive's computational and data structures.** Computations are organized into GPU blocks, with multiple threads in each block. Block(s) run an environment; each thread runs an agent. Blocks can access the shared GPU memory that stores roll-out data and (deep) policy models. A `DataManager` and `FunctionManager` enable defining GPU-based MARL logic with Python APIs.

GPU machines for richer environments. Trainer workers repeatedly gather the roll-out data (asynchronously) from the roll-out workers and optimize policies on CPU or GPU machines. While such a distributed design is highly scalable, worker communication can be expensive and individual machine utilization can be poor.

To improve performance, GPU and TPU-based RL frameworks exist (Tang et al., 2022; Hessel et al., 2021), but have focused on single-agent and domain-specific environments, e.g., for Atari (Dalton et al., 2020), or learning robotic control in 3-D rigid-body simulations (Freeman et al., 2021; Makoviychuk et al., 2021; Austin et al., 2019). Also, when running multi-agent simulations, generating roll-outs can become prohibitively slow when many agents use the same compute thread (Weng et al., 2022; Hu et al., 2021; Zheng et al., 2017). As such, it remains challenging to build efficient RL pipelines with simulations featuring complex interactions between multiple agents.

An End-to-End Solution. WarpDrive[2] is a framework for fast end-to-end (multi-agent) RL that addresses the aforementioned challenges and is available at `https://github.com/salesforce/warp-drive`. It is domain agnostic: WarpDrive-compatible simulations have been used in, e.g., economics (Trott et al., 2021) and climate modeling (Zhang et al., 2022).

WarpDrive runs the entire RL workflow end-to-end on a single GPU, using a single store of data for simulation roll-outs, action inference, and training. This minimizes CPU-GPU data communication and significantly reduces simulation and training time. Also, WarpDrive runs multiple multi-agent simulations in tandem, capitalizing on the parallelization capabilities of GPUs. In particular, WarpDrive runs each agent (within each simulation) on a unique GPU thread, with granular control over the agents' interactions across threads. It can manipulate the state updates and action sampling across agents in parallel. These

2. The name *WarpDrive* is inspired by the science fiction concept of a superluminal spacecraft propulsion system. Moreover, a *warp* is a group of 32 threads that execute at the same time in (certain) GPUs.

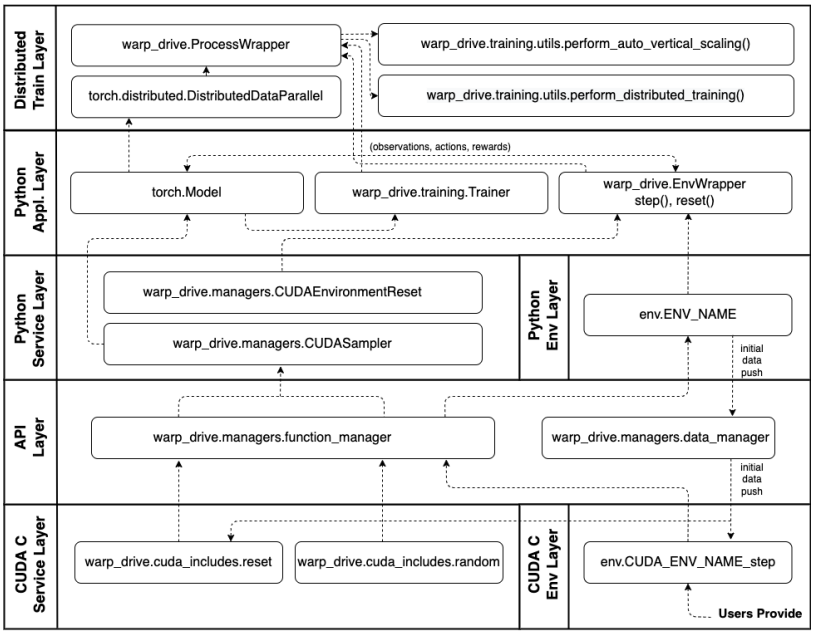

Figure 2: **Code structure.** All software layers and components are modular, generic, and incrementally executable, so creating and extending fast MARL pipelines is easy.

design choices enable running thousands of concurrent simulations, or millions of agents in parallel on a single GPU, and training on extremely large batches of experience. WarpDrive can also train across multiple GPUs to linearly scale up the throughput further.

Figure 1 shows the structure of WarpDrive. The `DataManager` and `FunctionManager` are two key Python classes (residing on the CPU) to facilitate all the CPU-GPU (also referred to as *host* and *device*) communication and interactions that are relevant to RL. They connect to the CUDA back-end and provide simple APIs to build high-level Python applications. Figure 2 provides an overview of modules and their relationships in WarpDrive. For more details on WarpDrive and its APIs, please see the documentation in Lan et al. (2021).

Tooling. WarpDrive builds on CUDA (Compute Unified Device Architecture), a platform and programming model that allows users to run programs (called *kernels*) on (CUDA-enabled) GPU hardware. Given any *gym*-style (Brockman et al., 2016) multi-agent environment, the first version of WarpDrive provides utility functions to facilitate re-writing the environment in CUDA C, in order to run the simulation on the GPU. WarpDrive also comprises quality-of-life tools to run end-to-end MARL training using just a few lines of code, and is compatible with PyTorch. As such, WarpDrive is flexible and accommodates environments with flexible multi-agent interactions, models, and learning algorithms, and allows creating and extending RL pipelines that maximize the utility of GPUs.

Benchmarks. We benchmark WarpDrive in three environments: discrete and continuous versions of the game of Tag (similar to Predator-Prey (Lowe et al., 2017) and Pursuit (Zheng et al., 2017)) and a more complex COVID-19 economic simulation (Trott et al., 2021). All

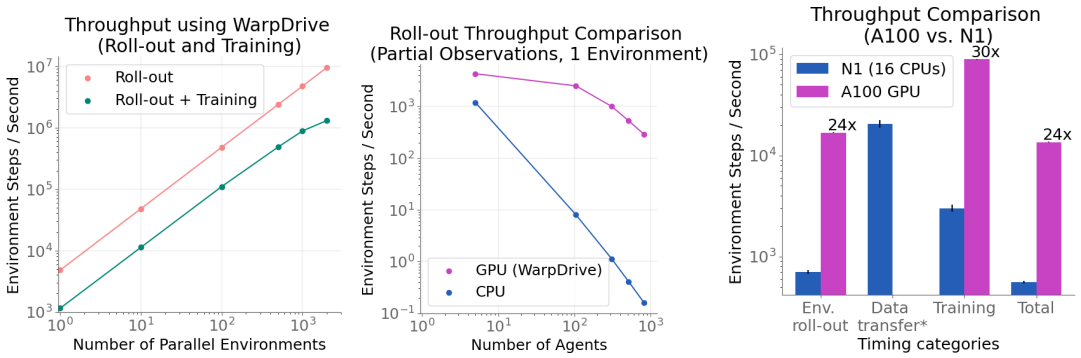

Figure 3: **WarpDrive Performance.** *Left:* Roll-out and training throughput in discrete Tag versus the number of parallel environments (log-log scale) with 5 agents: the throughput scales (almost) linearly. *Middle:* Roll-out throughput in continuous Tag versus the number of agents (log-log scale): the throughput (environment steps per second) drops linearly on the CPU, but much slower on the GPU (with WarpDrive). *Right:* In the COVID-19 economic simulation (Trott et al., 2021) with 60 environments: WarpDrive achieves a 24× (total) throughput boost.

experiments were run on the Google Cloud Platform. We see that WarpDrive running on a single GPU machine, *a2-highgpu-1g* (`https://cloud.google.com/compute/docs/gpus#a100-gpus`) scales almost linearly to thousands of environments and agents, and yields orders of magnitude faster MARL compared to a CPU implementation on the *n1-standard-16* (`https://cloud.google.com/compute/docs/general-purpose-machines#n1_machines`).

Figure 3 (left) shows how WarpDrive's performance scales in discrete Tag: it scales linearly to over thousands of environments (with 5 agents) and yields *almost perfect parallelism*. For example, during a roll-out, WarpDrive runs at 4300 environment steps per second with 1 environment and 8.3M environment steps per second with 2000 environments.

Figure 3 (middle) shows roll-out performance in continuous Tag. For a single environment, as we scale up from 5 agents to 820 agents, i.e., by a factor of 164, the roll-out throughput drops by a factor of 7500 on a single CPU, but only a factor of 15 on the GPU (with WarpDrive). CPUs are not as effective as GPUs at thread-level parallelism; in contrast, GPUs can run agents on separate threads and maintain high throughput: e.g., with 820 agents, the roll-out throughput on the GPU is *2000 times faster than on the CPU*.

Such gains hold in more complex environments: Figure 3 (right) shows that in an economic simulation of COVID-19, WarpDrive achieves 24× *more steps per second* with 60 environments, compared to a 16-CPU node. Importantly, because there are no repeated CPU-GPU data transfers, both roll-out and training are an order of magnitude faster.

Future Directions. We hope WarpDrive encourages using and creating (tools for) GPU simulations and hope to extend and integrate WarpDrive with other tools for quickly building simulations and machine learning workflows on GPUs and other accelerators. In all, we hope that WarpDrive contributes to the democratization of high-performance RL systems and advances in (multi-agent) machine learning and artificial intelligence.

References

Jacob Austin, Rafael Corrales-Fatou, Sofia Wyetzner, and Hod Lipson. Titan: A parallel asynchronous library for multi-agent and soft-body robotics using nvidia cuda, 2019. URL https://github.com/jacobaustin123/Titan.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://github.com/openai/gym.

Steven Dalton, Iuri Frosio, and Michael Garland. Accelerating reinforcement learning through gpu atari emulation, 2020. URL https://github.com/NVlabs/cule.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski. Seed rl: Scalable and efficient deep-rl with accelerated central inference, 2020. URL https://github.com/google-research/seed_rl.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation, 2021. URL https://github.com/google/brax.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

Matteo Hessel, Manuel Kroiss, Aidan Clark, Iurii Kemaev, John Quan, Thomas Keck, Fabio Viola, and Hado van Hasselt. Podracer architectures for scalable reinforcement learning, 2021.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning, 2020. URL https://github.com/deepmind/acme.

Hengyuan Hu, Adam Lerer, Brandon Cui, David Wu, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning, 2021. URL https://github.com/facebookresearch/hanabi_SAD.

Tian Lan, Sunil Srinivasa, Huan Wang, and Stephan Zheng. Warpdrive: Extremely fast end-to-end deep multi-agent reinforcement learning on a gpu, 2021. URL https://arxiv.org/abs/2108.13976.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. URL `https://github.com/NVIDIA-Omniverse/IsaacGymEnvs`.

OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

Arnu Pretorius, Kale-ab Tessera, Andries P. Smit, Claude Formanek, St John Grimbly, Kevin Eloff, Siphelele Danisa, Lawrence Francis, Jonathan Shock, Herman Kamper, Willie Brink, Herman Engelbrecht, Alexandre Laterre, and Karim Beguir. Mava: a research framework for distributed multi-agent reinforcement learning, 2021. URL `https://github.com/instadeepai/Mava`.

Yujin Tang, Yingtao Tian, and David Ha. Evojax: Hardware-accelerated neuroevolution, 2022. URL `https://github.com/google/evojax`.

Alexander Trott, Sunil Srinivasa, Douwe van der Wal, Sebastien Haneuse, and Stephan Zheng. Building a foundation for data-driven, interpretable, and robust policy design using the ai economist, 2021. URL `https://github.com/salesforce/ai-economist`.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. Envpool: A highly parallel reinforcement learning environment execution engine, 2022. URL `https://github.com/sail-sg/envpool`.

Tianyu Zhang, Andrew Williams, Soham Phade, Sunil Srinivasa, Yang Zhang, Prateek Gupta, Yoshua Bengio, and Stephan Zheng. Ai for global climate cooperation: Modeling global climate negotiations, agreements, and long-term cooperation in rice-n, 2022. URL `https://github.com/mila-iqia/climate-cooperation-competition`.

Lianmin Zheng, Jiacheng Yang, Han Cai, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence, 2017. URL `https://github.com/geek-ai/MAgent`.

Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C. Parkes, and Richard Socher. The ai economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advances*, 8(18):eabk2607, 2022. doi: 10.1126/sciadv.abk2607. URL `https://www.science.org/doi/abs/10.1126/sciadv.abk2607`.