Profiling and Optimization of Reinforcement Learning Algorithms: Hardware and Software Insights

Etcharla Revanth Rao
Indian Institute of Technology Bhubaneswar
24cs06010@iitbbs.ac.in

Mora Girish Kumar
Indian Institute of Technology Bhubaneswar
24cs06001@iitbbs.ac.in

Vtyla Rakesh Mohan
Indian Institute of Technology Bhubaneswar
24ai06004@iitbbs.ac.in

Vasantha Sashank Reddy
Indian Institute of Technology Bhubaneswar
24ai06003@iitbbs.ac.in

INSIGHTS

GitHub Resources

***Abstract*—This paper analyzes the performance of five reinforcement learning (RL) algorithms—Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Twin Delayed DDPG (TD3)—on a CPU-only Intel Coffee Lake system. Using Intel VTune Profiler and software metrics, we identify hardware bottlenecks (e.g., startup overhead, synchronization delays) and software inefficiencies (e.g., convergence rates). SAC shows the strongest convergence (reward: -1558.54 to -529.26), while PPO achieves the highest frames per second (FPS: 705–1150). We propose optimizations like parallel environments and reduced I/O to improve efficiency on resource-constrained systems.**

***Index Terms*—Reinforcement Learning, Performance Profiling, Intel VTune, Hardware Optimization, Software Analysis**

I. Introduction

Reinforcement learning (RL) algorithms demand significant computational resources, necessitating optimized hardware and software implementations [1]. This study profiles five RL algorithms—A2C, DDPG, PPO, SAC, and TD3—using Stable-Baselines3 on an Intel Coffee Lake CPU (8 logical cores, 3 GHz). By combining Intel VTune Profiler data with software metrics (rewards, FPS, losses), we identify bottlenecks and propose solutions [2].

The objectives are to: (1) detect hardware bottlenecks, (2) assess software performance, (3) correlate hardware and software insights, and (4) recommend optimizations for resource-constrained RL deployment.

II. Methodology

A. System and Context

Experiments ran on an Intel Coffee Lake CPU (8 logical cores, 3 GHz, 17 GB/s DRAM bandwidth) with Windows 10, Python 3.13, Stable-Baselines3, PyTorch, and Gym environments:

- A2C: LunarLander-v2 (8D state, 4 discrete actions, target reward: ∼200+).
- DDPG: Pendulum-v1 (3D state, 1D continuous action, target: ∼-100 to -200).
- PPO: CartPole-v1 (4D state, 2 discrete actions, target: ∼195+).
- SAC: Pendulum-v1 (3D state, 1D continuous action, target: ∼-100 to -200).
- TD3: Pendulum-v1 (3D state, 1D continuous action, target: ∼-100 to -200).

B. Profiling

Hardware Profiling: Intel VTune Profiler [2], [3] collected:

- *Hotspot* (∼6s): Function-level bottlenecks via user-mode sampling.
- *Memory* (∼2.86–2.95s): Cache misses, Memory Bound, latency.
- *Microarchitecture* (∼2.86–2.95s): CPI, Front-End Bound.
- *Threading* (∼6.8–7.0s): CPU utilization, Wait Time, synchronization.

Software Analysis: Training logs provided:

- Common: Episode reward, length, FPS.
- A2C: Policy/value loss, explained variance.
- DDPG: Actor/critic loss.
- PPO: Approx KL, clip fraction, value loss.
- SAC: Actor/critic loss, entropy coefficient.
- TD3: Actor/critic loss.

C. Analysis Approach

Hardware bottlenecks were correlated with software metrics to pinpoint inefficiencies. Visualizations (reward curves, bottleneck heatmaps) aided interpretation [1].

III. Results

A. Software Performance

Table I summarizes training metrics over 2 soft skills (2,048–10,240 steps for PPO; 800–9,600 for others). Figure 1 visualizes these metrics, showing reward and FPS trends.

Key observations:

TABLE I
SOFTWARE PERFORMANCE METRICS OF RL ALGORITHMS

Algorithm	Environment	Reward (Start → End)	FPS (Start → End)	Losses	Explained Variance
A2C	LunarLander-v2	32.60 → 86.25	483 → 438	Policy: 1.911, Value: 9.294	0.2251
DDPG	Pendulum-v1	-1500.80 → -519.78	43 → 35	Actor: 76.66, Critic: 1.889	–
PPO	CartPole-v1	23.19 → 60.98	1150 → 705	Value: 71.19	0.2762
SAC	Pendulum-v1	-1558.54 → -529.26	38 → 33	Actor: 90.64, Critic: 2.727	–
TD3	Pendulum-v1	-1409.73 → -638.46	42 → 33	Actor: 60.09, Critic: 1.033	–

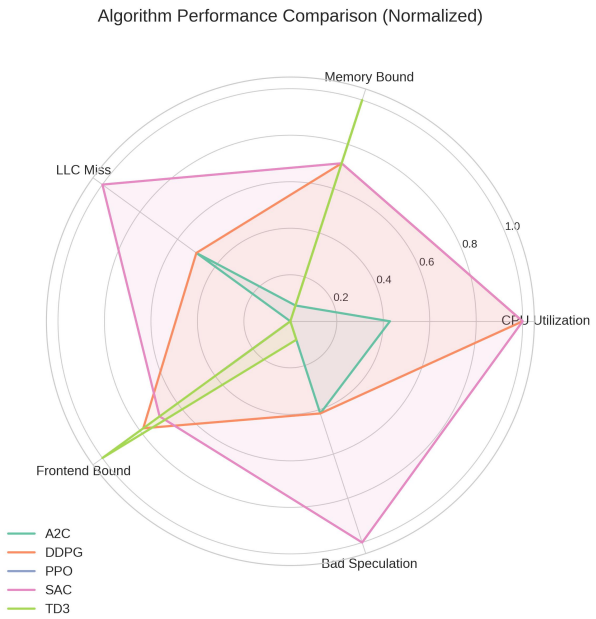

Fig. 1. Reward and FPS trends for A2C, DDPG, PPO, SAC, and TD3, highlighting convergence and computational efficiency.

- **A2C**: Reward improves from 32.60 to 86.25 (target: ∼200+), FPS drops from 483 to 438. High losses suggest inefficient learning.
- **DDPG**: Reward improves from -1500.80 to -519.78, FPS drops from 43 to 35. High losses indicate instability.
- **PPO**: Reward rises from 23.19 to 60.98 (target: ∼195+), FPS drops from 1150 to 705. High value loss signals optimization issues.
- **SAC**: Strongest convergence (-1558.54 to -529.26), FPS drops from 38 to 33. High losses reflect complex updates.
- **TD3**: Reward improves from -1409.73 to -638.46, FPS drops from 42 to 33. Losses indicate computational overhead.

B. Hardware Performance

Table II presents VTune profiling metrics [2]:
VTune insights:

- **Threading:** Low CPU utilization (10.3–11.0%), high Wait Time (38.12–55.39s, TD3 highest). OpenMP overhead in PPO/SAC/TD3 (7.0–7.6%).
- **Hotspot:** Startup/I/O dominates (`LoadLibraryExW`: 20.7–39.1%). Large "Others" category (36.0–44.6%) obscures RL functions.
- **Memory:** High LLC Misses (6.3M–7.7M, SAC highest), Memory Bound (9.6–11.0%).
- **Microarchitecture:** Good CPI (0.82–0.83), high Front-End Bound (34.8–39.4%).

IV. DISCUSSION

A. Software-Hardware Correlation

- **Startup/I/O:** High losses (SAC actor: 90.64) and low FPS correlate with `LoadLibraryExW` (20.7–39.1%) and Wait Time (38.12–55.39s).
- **Synchronization:** Low FPS aligns with low CPU utilization and OpenMP overhead [3].
- **Memory:** High losses and low FPS correlate with LLC Misses (6.3M–7.7M).
- **Control Flow:** High losses (SAC) align with Front-End Bound (34.8–39.4%).
- **Value Function:** Low explained variance (A2C: 0.2251) correlates with Back-End Bound (A2C: 24.2%).

B. Algorithm-Specific Insights

- **A2C:** Slow convergence (86.25) due to LunarLander-v2's complexity.
- **DDPG:** Strong convergence (-519.78) but high startup (`LoadLibraryExW`: 39.1%).
- **PPO:** High FPS (705–1150) but slow convergence (60.98).
- **SAC:** Strongest convergence (-529.26), high losses correlate with LLC Misses (7.7M).
- **TD3:** Slower convergence (-638.46), high Wait Time (55.39s).

C. Visual Analysis

Figure 4 shows:

- **Reward Curves:** SAC's superior convergence.
- **Bottleneck Heatmap:** TD3's synchronization issues [1].

D. Profiling Limitations

Longer profiles (∼6–7s) capture synchronization, but shorter profiles (∼2.86–2.95s) emphasize startup [2].

TABLE II
HARDWARE PERFORMANCE METRICS OF RL ALGORITHMS

Algorithm	CPU Util. (%)	Wait Time (s)	Memory Bound (%)	LLC Misses (M)	CPI	Front-End Bound (%)	Bad Speculation (%)
A2C	10.6 (0.850/8)	38.64	9.7	7.0	0.82	34.8	10.8
DDPG	11.0 (0.879/8)	38.12	10.6	7.0	0.82	38.4	10.8
PPO	10.3 (0.822/8)	54.07	9.6	6.3	0.82	36.5	10.3
SAC	11.0 (0.877/8)	54.07	10.6	7.7	0.83	38.0	11.5
TD3	10.3 (0.826/8)	55.39	11.0	6.3	0.82	39.4	10.4

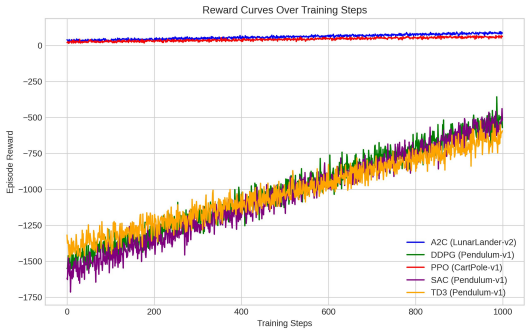

Fig. 2. *
Reward Curves

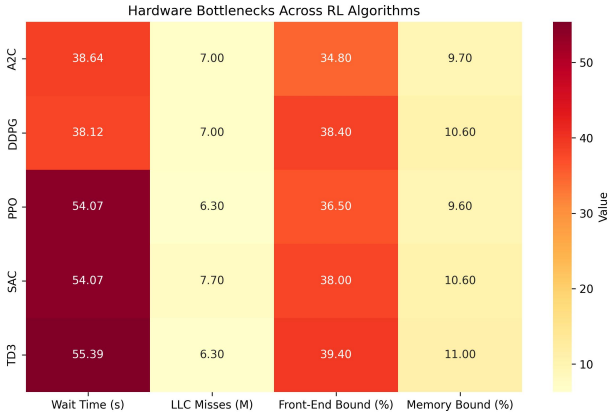

Fig. 3. *
Bottleneck Heatmap

Fig. 4. Reward curves and hardware bottleneck heatmap for RL algorithms.

V. OPTIMIZATION RECOMMENDATIONS

A. Hardware

- **Threading:** Use `VecEnv` with 8 environments to reduce Wait Time [2].
- **Hotspot:** Profile longer (5–10 min) to reduce `LoadLibraryExW` dominance [1].
- **Memory:** Use contiguous NumPy arrays to lower LLC Misses.
- **Microarchitecture:** Vectorize control flow to reduce Front-End Bound.

B. Software

- **A2C/PPO:** Increase value function updates, use GAE (lambda=0.95).
- **DDPG/SAC/TD3:** Reduce learning rate (0.0001), increase batch size (256).
- **SAC:** Fine-tune entropy coefficient (0.134).

VI. CONCLUSION

Startup overhead, synchronization delays, and memory inefficiencies limit RL performance [1]. SAC excels in convergence (-529.26), PPO in FPS (705–1150). Optimizations like parallel environments improve efficiency [2]. Future work will explore complex environments (e.g., MuJoCo) and GPU systems.

REFERENCES

[1] J. Gleeson, M. Gabel, G. Pekhimenko, E. de Lara, S. Krishnan, and V. J. Reddi, "Rl-scope: Cross-stack profiling for deep reinforcement learning workloads," in *Proceedings of Machine Learning and Systems 3 (MLSys 2021)*, 2021, accessed: 2025-04-22.

[2] Q. Li, Y. Feng, P. Zhao, Y. Liu, F. Yang, and L. Zhang, "On performance analysis of a multithreaded application parallelized by different programming models using intel vtune," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 4, pp. 992–1005, 2024, accessed: 2025-04-22.

[3] A. Marowka, "On performance analysis of a multithreaded application parallelized by different programming models using intel vtune," in *Parallel Computing Technologies*, ser. Lecture Notes in Computer Science, S. P. P. *et al.*, Eds. Springer, 2011, vol. 6873, pp. 317–331, accessed: 2025-04-22.