Distributed TD3 Training with MPI

CSE 633 – Parallel Computing
Instructor: Dr. Russ Miller
Presented by Elvis Rodrigues
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 Pseudocode - 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

- Forward Critic
- Input
- Worker
- Evaluation
- Expected Evaluation
- Master Critic
- Loss
- Evaluation
- Forward Pass
Distributed TD3 – Backward Pass

Backward Pass

Gradients

Worker Critic

Loss

Master Critic
Distributed TD3 – Model Update

Worker Critic → Gradients → MPI_Gather → Gathered Gradients → Update Model
Worker Critic → Gradients → Worker Critic
Worker Critic → Gradients → Worker Critic
Master Critic
Distributed TD3 – Parameter Broadcast
Distributed TD3 – Health Check

Worker Critic → Worker Critic → Worker Critic → Master Critic → Master Critic

Worker Critic → Worker Critic → Worker Critic

Continue? → MPI_Allgather
Environment

Pendulum-v1 from Gymnasium
Continuous state and action space
Episode reward cutoff is -200
Results – Time Elapsed
Results – Steps Elapsed

Number of Steps to Convergence

Number of Steps (Zoomed In)
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

![Speedup (w/ reference to Step)](chart1)

![Step Speedup (Zoomed In)](chart2)
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

