

The City College of New York

CSC 59867-E: Senior Project II

AI Agents for Decision Making in the Real World

Scaling Multi-Agent AI problems: Modern Distributed Learning and JAX

Saptarashmi Bandyopadhyay

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

October 26, 2026 CSC 59867 - E

Scaling Up: Distributed Processing for Multi-Agent AI

Saptarashmi Bandyopadhyay

Architecture 2: Ring All-Reduce

- **Characteristics:** Decentralized, Peer-to-peer
- ***All-Reduce*:** A collective communication primitive - all nodes participate, all nodes receive the final aggregated result
- In the **Ring All-Reduce** algorithm, each node only communicates with its two immediate neighbors, forming a logical ring
- Two Phases: ***Reduce-Scatter*** and ***All-Gather***

Saptarashmi Bandyopadhyay

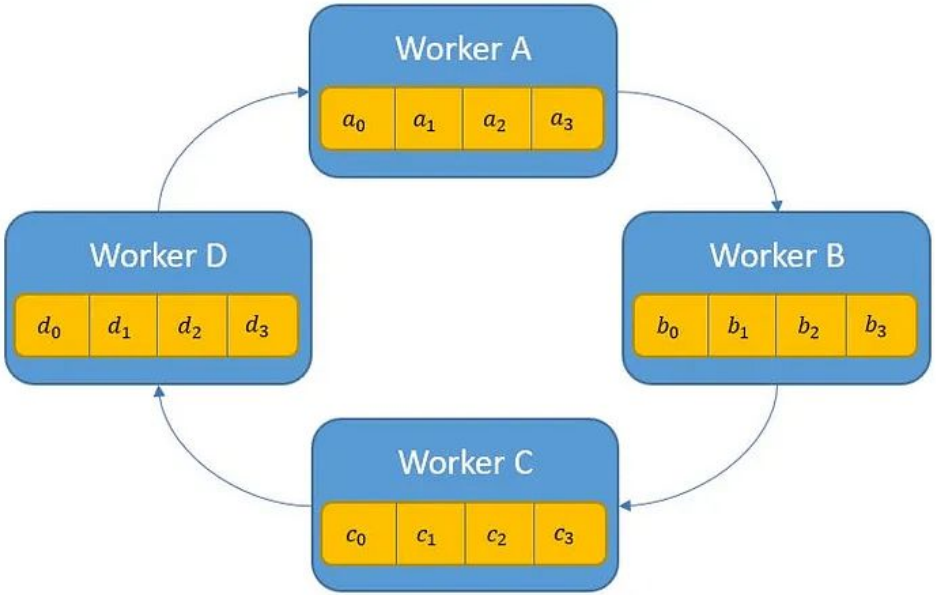
Ring All-Reduce

- **Phase 1: Reduce-Scatter** (or Share-Reduce)
  - In $N-1$ steps each node sends a chunk to its right neighbor and receives a chunk from the left neighbor
  - It adds the incoming chunk to its corresponding chunk
  - Result: Each node holds one chunk of the final, fully-summed vector
- **Phase 2: All-Gather** (or Share-Only)
  - In $N-1$ nodes circulate their summed chunks
  - Each node now has a copy of the final result

Saptarashmi Bandyopadhyay

Example: Ring All-Reduce

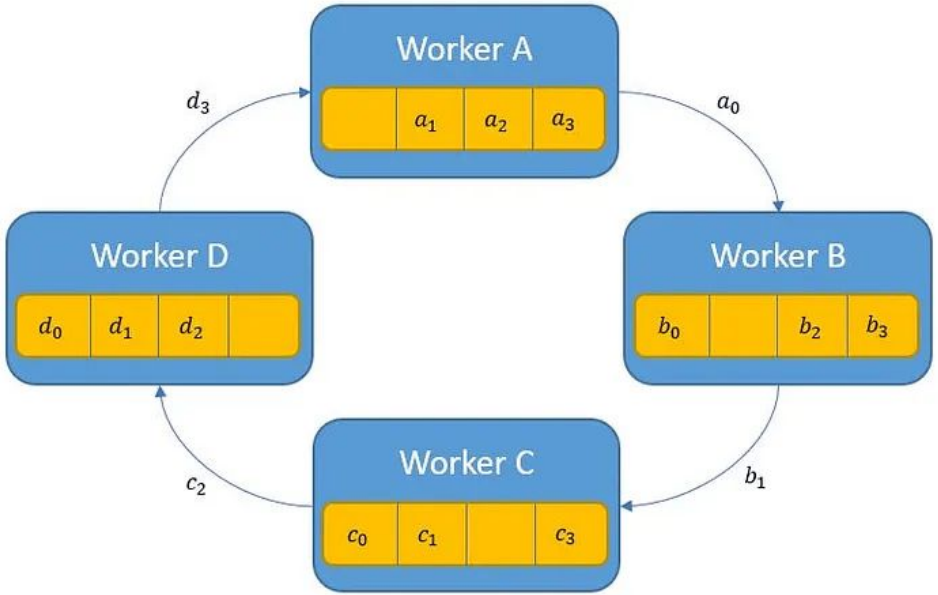
- Start with a ring of nodes. Each node has a “piece” of the final result
- Our goal is to have each node hold the sum of the pieces



Saptarashmi P adhyay

Example: Ring All-Reduce

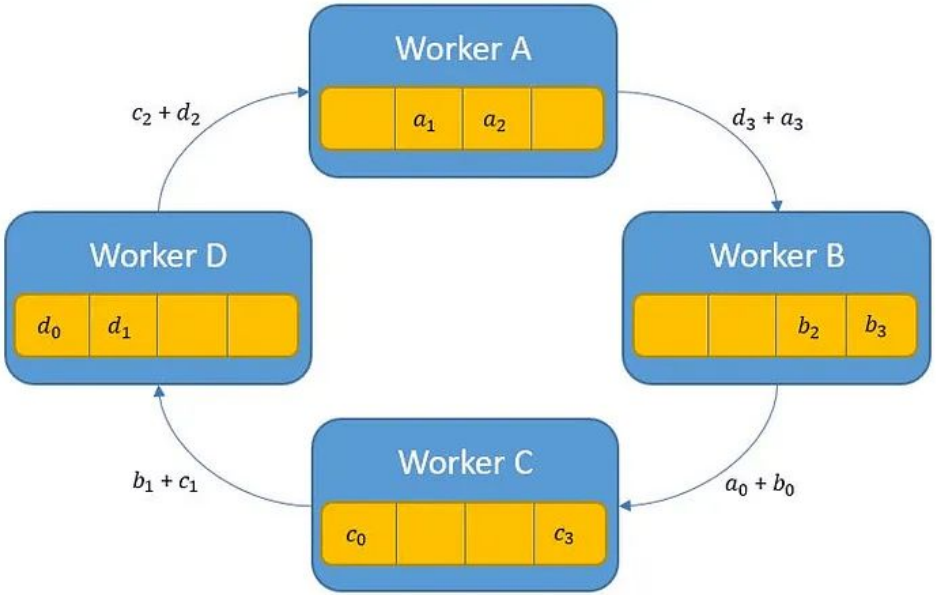
- **Reduce-Scatter:** Each node sends a chunk to its next neighbor and receives a chunk from its previous neighbor



Saptarashmi P adhyay

Example: Ring All-Reduce

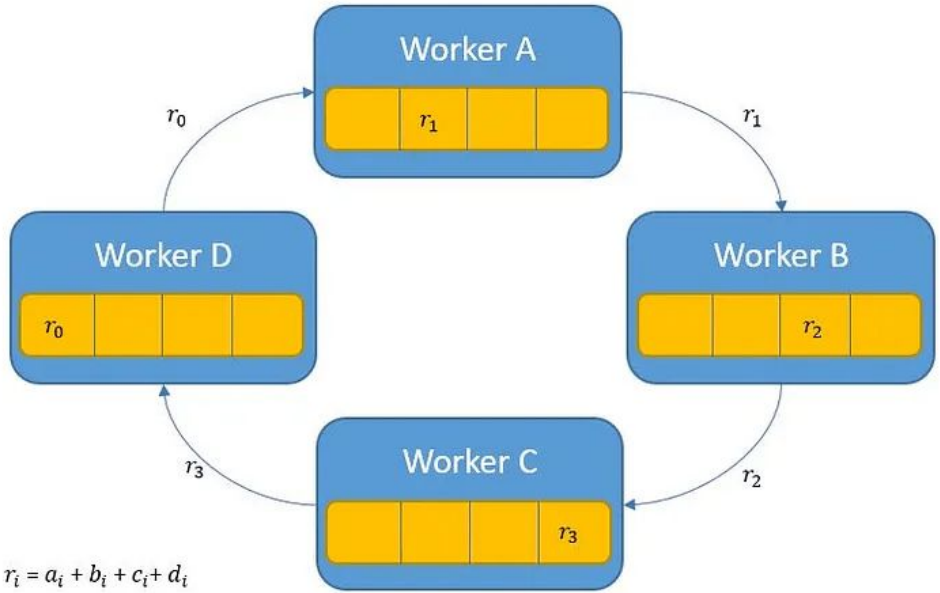
- **Reduce-Scatter:** As the reduce continues, each node builds up a single piece of the final, aggregated result



Saptarashmi P adhyay

Example: Ring All-Reduce

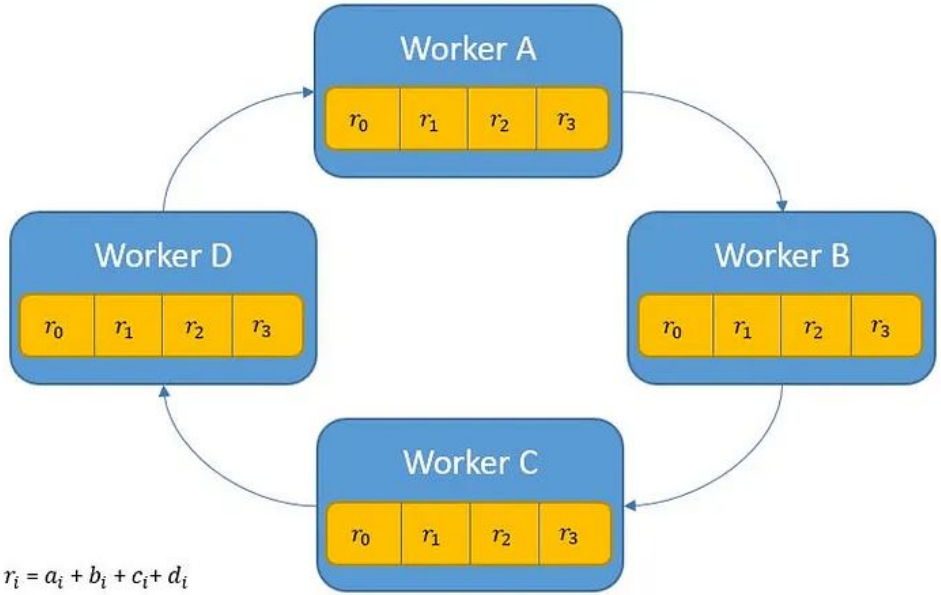
- **Reduce-Scatter:** At the end of the first phase, each node holds a chunk of the final result



Saptarashmi P adhyay

Example: Ring All-Gather

- **All-Gather:** In Phase 2, we simply pass along the chunks of the final result in a similar way to how we passed chunks in All-Reduce
- This time, we only store (*share*), we do not add (*reduce*) the data



Saptarashmi P adhyay

Ring All-Reduce Bandwidth Optimality

- In the Parameter Server, our communication load was $O(N \cdot M)$, where N is the number of worker nodes and M is the model size
- In Ring All-Reduce, each node sends and receives a total of $2 \times \frac{N-1}{N} \times M$ bytes of data
- As N grows, this approaches $2M$, so the communication cost per node becomes *independent* of N
- This is why we call All-Reduce *bandwidth-optimal*

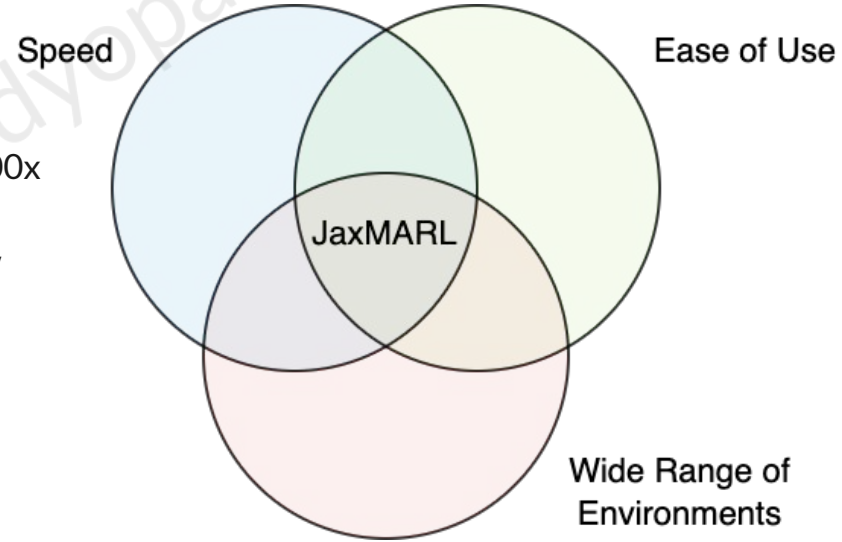
Saptarashmi Bandyopadhyay

MARL in Practice: JaxMARL

Saptarashmi Bandyopadhyay

Introduction

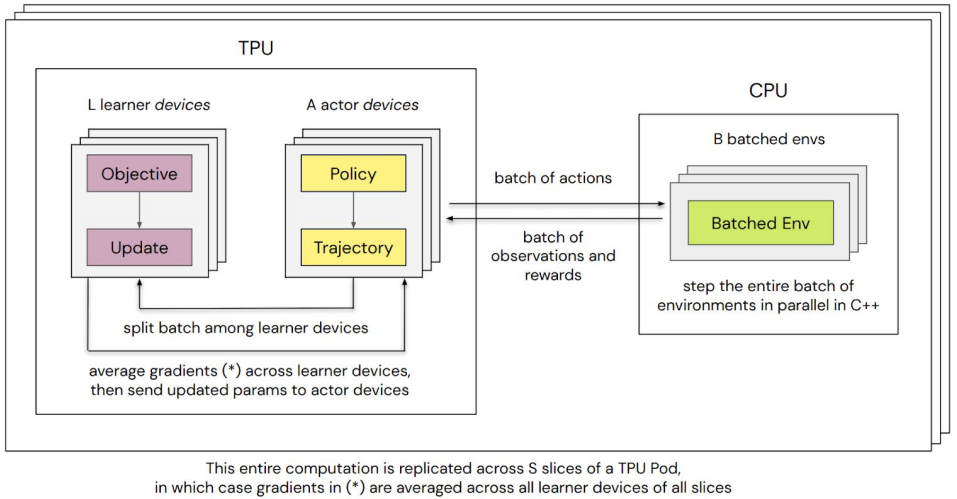
- Multi-Agent RL can be pretty slow
- JAX-enabled hardware acceleration can make it 12500x faster
- This means that experiments that once took days now take hours or minutes



Saptarashmi Bandyopadhyay

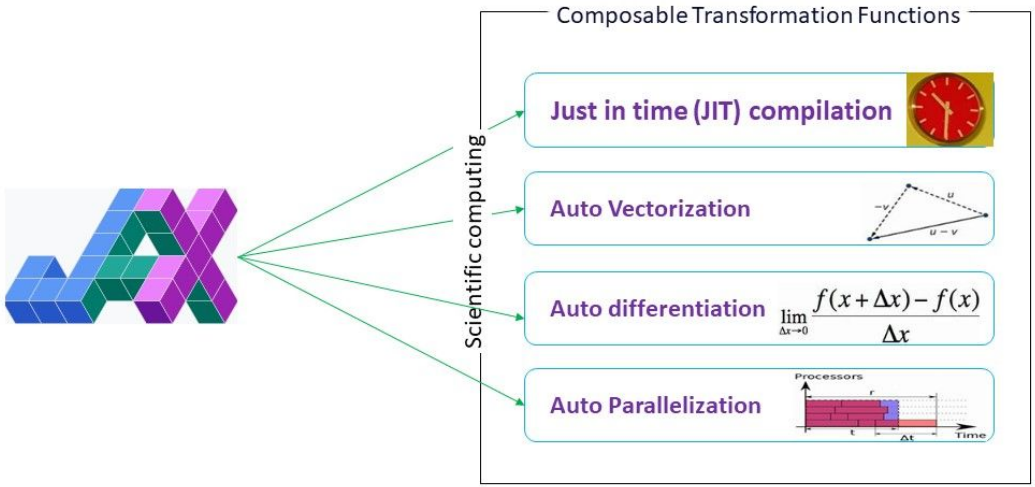
Recap: Non-JAXMARL Training Approaches

- Current RL environments before JAXMARL are normally run on CPU
- RL training and inference on the environments are done on the hardware accelerator
  - GPU or TPU
- Research Question: Why not run many environment threads in parallel?
  - Can help to increase training speed?!

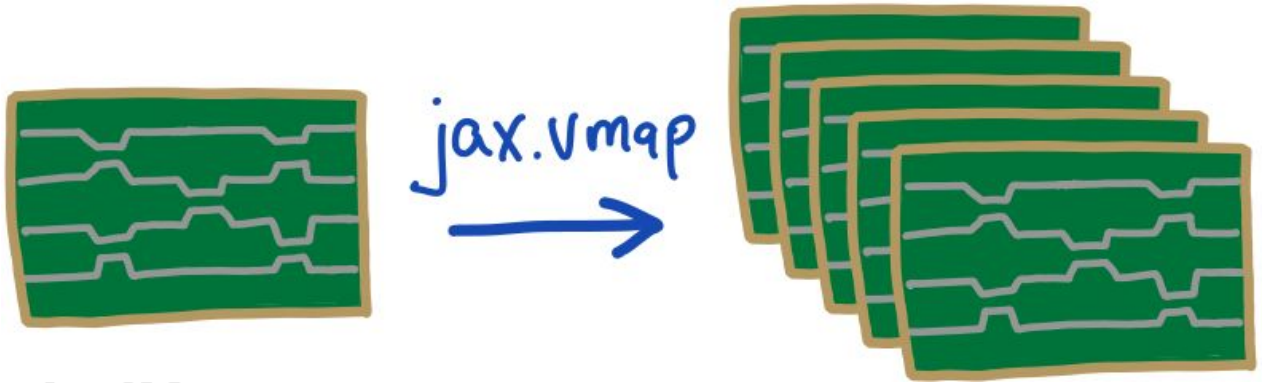


This entire computation is replicated across S slices of a TPU Pod,
in which case gradients in (*) are averaged across all learner devices of all slices

Use JAX for Multi-Agent Training



Vectorizing Environments with JAX vmap



Using JAX with JIT

- Vmap many environments in parallel for faster runtime!
- Avoid slow CPU-GPU data transfer
- Avoid Python command overhead
- Allow for Jax JIT optimisation (operator fusion)
- Avoid messy and complicated multi-processing setups!

Saptarashmi Bandyopadhyay

Multi-Agent Environments

- We implement 8 popular MARL environments
- We provide Q-learning and PPO

```python
import jax
from jaxmarl import make

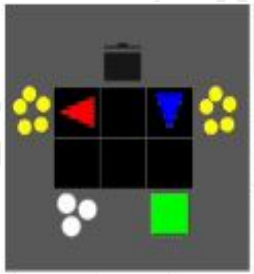
key = jax.random.PRNGKey(0)
key, key_reset, key_act, key_step = jax.random.split(key, 4)

# Initialise and reset the environment.
env = make('MPE_simple_world_comm_v3')
obs, state = env.reset(key_reset)

# Sample random actions.
key_act = jax.random.split(key_act, env.num_agents)
actions = {agent: env.action_space(agent).sample(key_act[i]) \
           for i, agent in enumerate(env.agents)}

# Perform the step transition.
obs, state, reward, done, infos = env.step(key_step, state, actions)
```

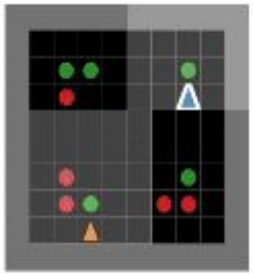
Overcooked



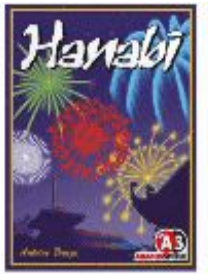
MABrax

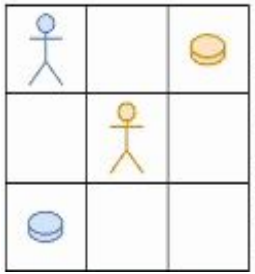


STORM

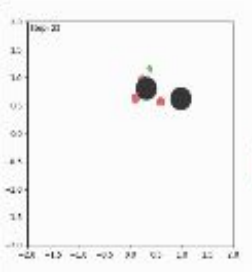


Hanabi

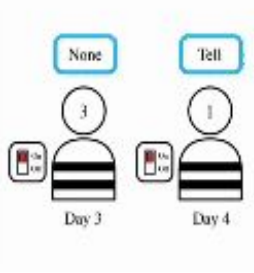




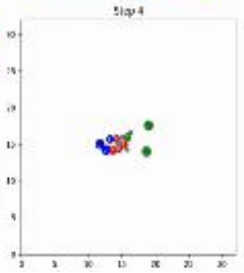
Coin Game



Multi-Particle Tag



Switch riddle



Starcraft (SMAX)

Questions?

Saptarashmi Bandyopadhyay