ParSwarm: A C++ Framework for Evaluating Distributed Algorithms for Robot Swarms

Zhi Wei Gan, Grace Cai, Noble Harasha, Nancy Lynch, Julian Shun

Massachusetts Institute of Technology

Robot Swarms

- Many agents collectively solve complex tasks
- Agents themselves have **simple** capabilities
- No central coordinator
- Simulators are important so we can explore the behavior of algorithms experimentally

Image source: Rubenstein et al. 2012

Background

Current state-of-the-art robot swarm simulators take into account complex physics simulations and are compatible with real-world robots, however they are slow.

Stage [Vaughan 2008]: 10^5 agents at 1/50 real-time speed

Why do we care about speed?

- Rapid algorithm prototyping
- Increasing the number of agents / size of experiment allows us to get insights about probabilistic bounds

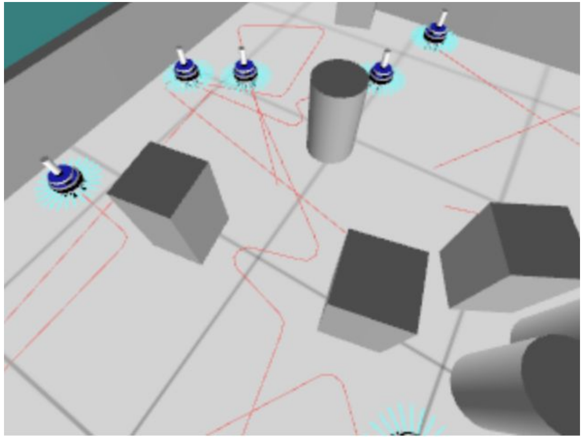
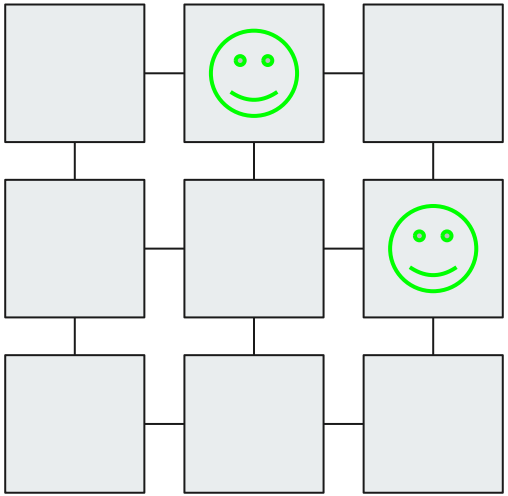


Image source: Pinciroli et al., 2012

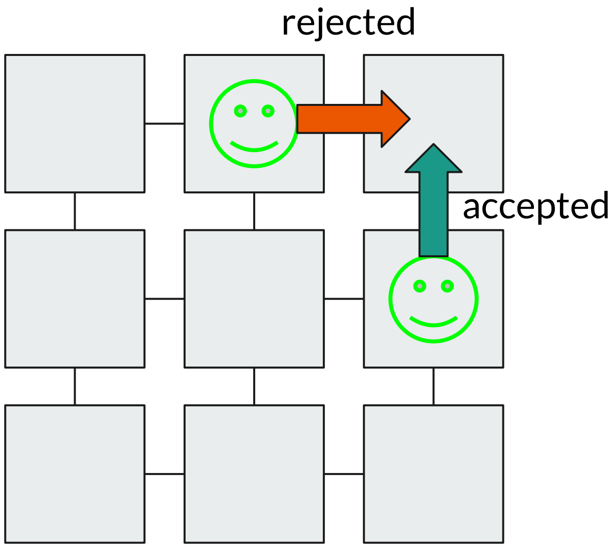
Mathematical Framework (Cai et al. 2023)



The model is a probabilistic, synchronous distributed system on a graph $G = (V, E)$

Vertices and agents have their own (arbitrary) states, e.g. color, position, size

Mathematical Framework (Cai et al. 2023)



On each round the agent proposes transitions based on the vertices and agents in a local area

Each transition is accepted or rejected based on some rule (agents on the same vertex)

An accepted transition changes the position/state of the agent or vertex

Parallel Framework Motivation

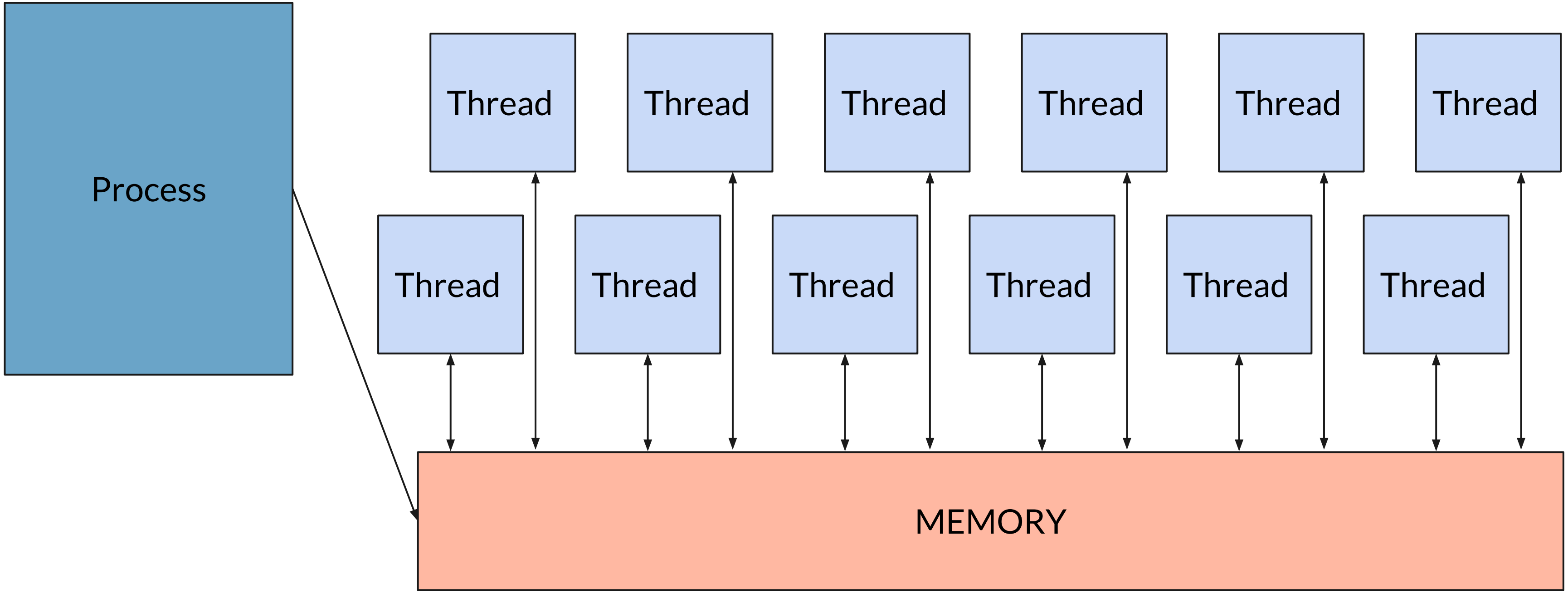
Current sequential implementation of the model is slow.

For **task allocation**: 50 x 50 grid with 100 agents and 16 tasks took 10 seconds to run ~500 iterations

Possible Approaches:

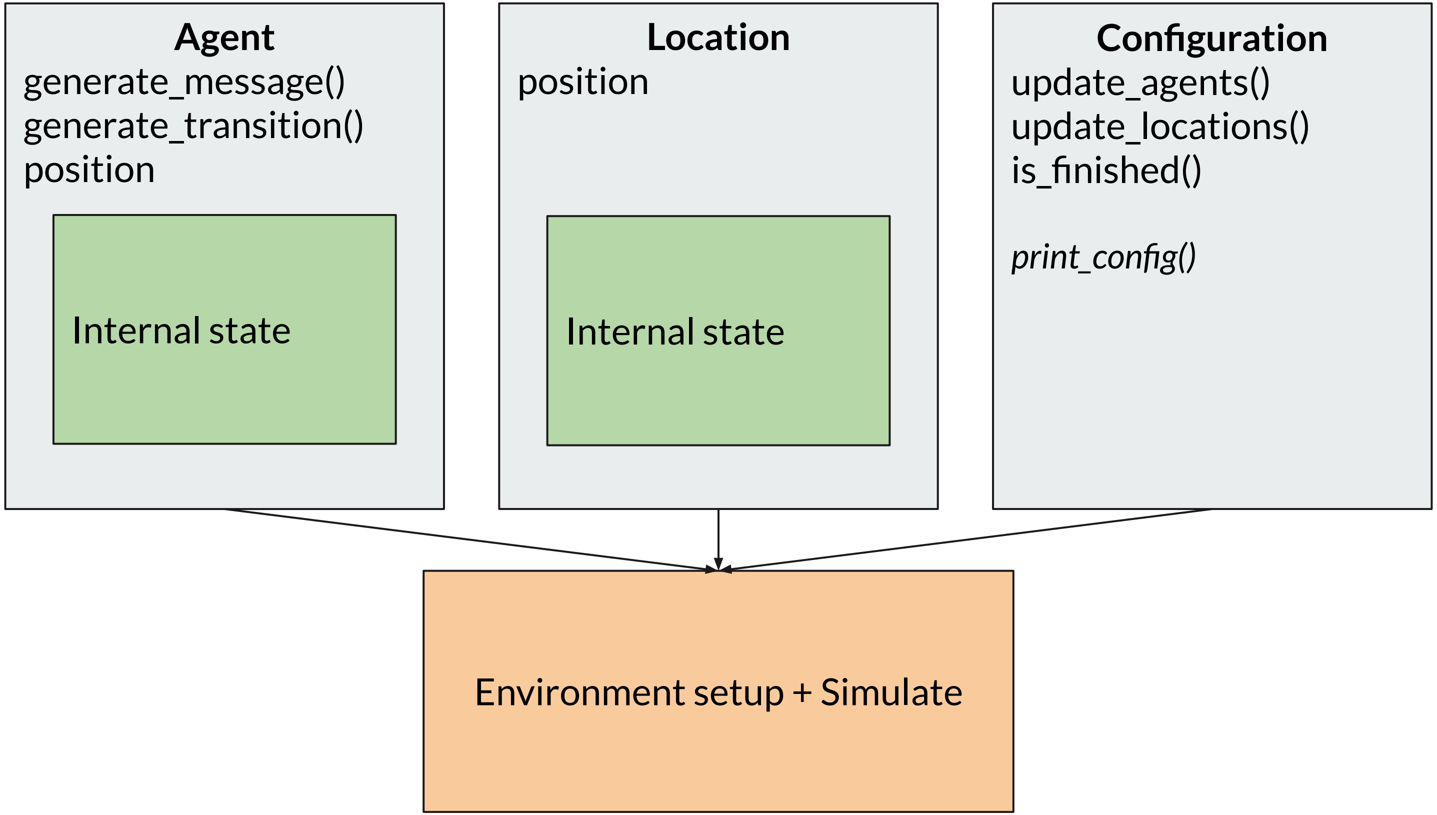
- Speed up the sequential implementation (not scalable)
- **Shared-Memory Parallel Programming**

Model: Shared-Memory Parallel Programming

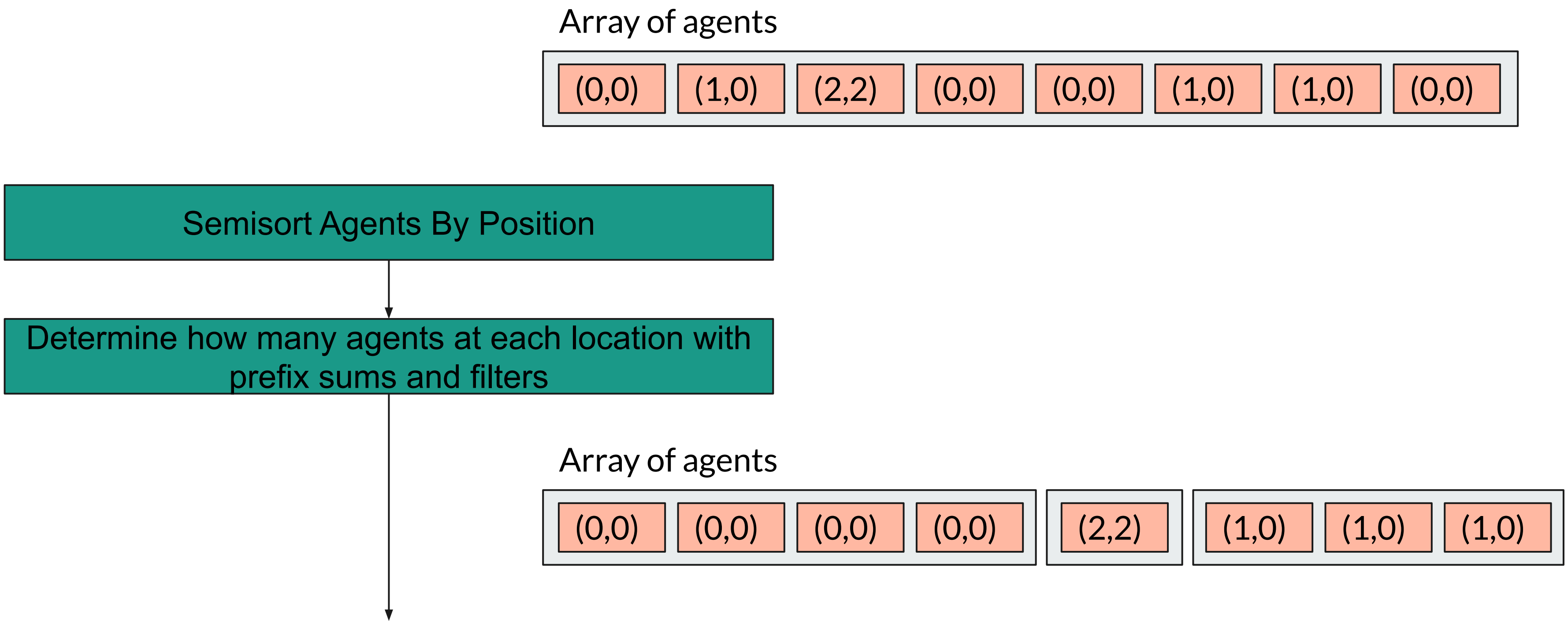


May run into race conditions, framework is used to abstract most of the details

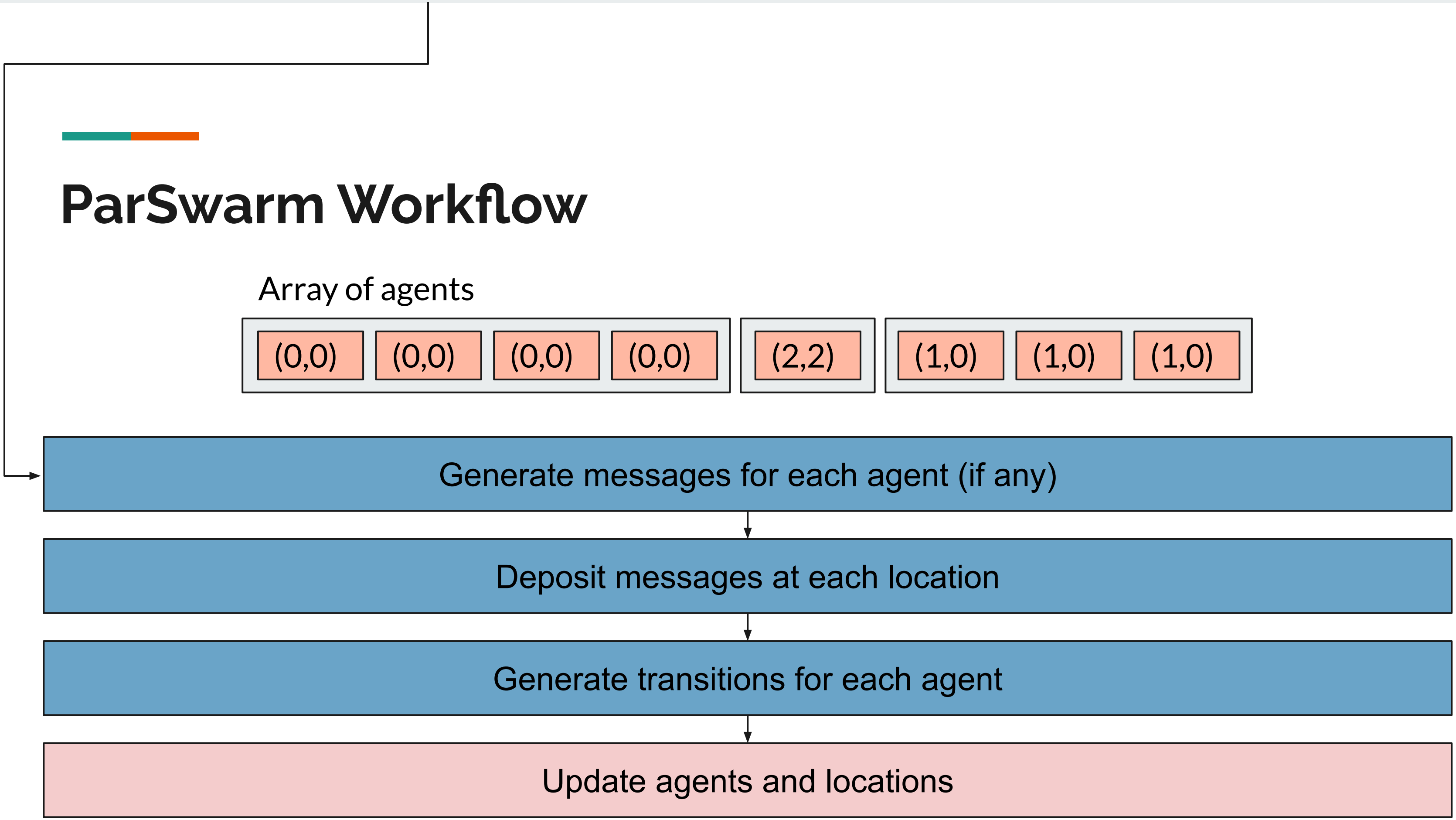
ParSwarm User-Defined Functions and Classes



ParSwarm Workflow



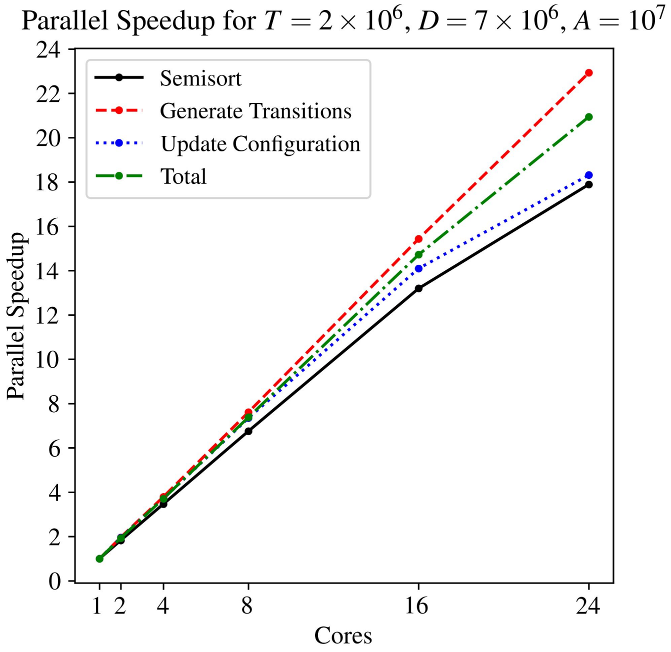
ParSwarm Workflow



Task Allocation: 1000x1000 Grid, 1,000,000 Agents

24 cores: ~1.32 s / iter
> 18x parallelism

For the 50x50 experiment, we get a 500x speedup over the Python implementation



Density Estimation (Musco et al. 2016)

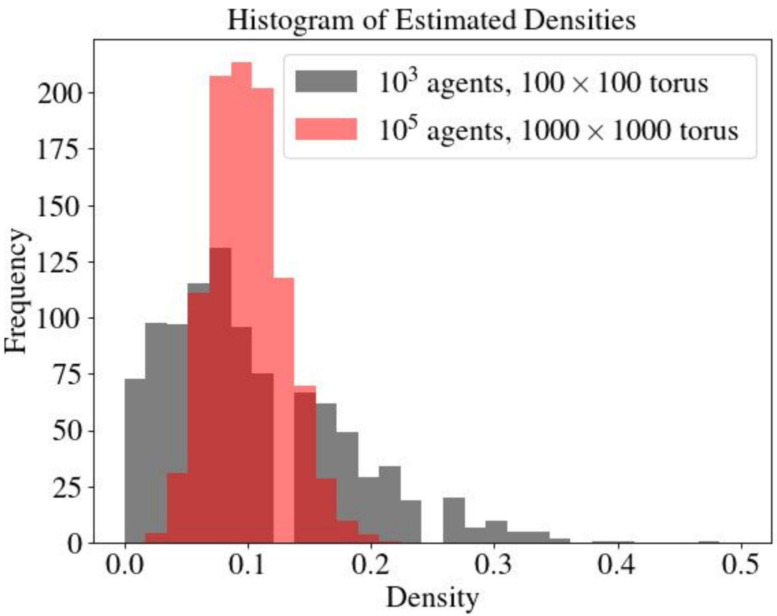
Setup:

- $N \times M$ torus grid with α agents
- Agent density is defined as: $\frac{\alpha}{N \times M}$

Algorithm:

- Agents random walk, counting the number of agents C they run into for t rounds.
- Musco et al. show: $\frac{C}{t} = \frac{\alpha}{N \times M}$ w.h.p.

Density Estimation Experiments



True density is 0.1

Framework allows for larger tests to be run to exhibit high probability behavior

The large experiment took 4 seconds to run on 24 cores

Future Work

Prototyping more **complex algorithms**

Increase **user-friendliness**, add more helper functions (in-progress)

Add better support for experiments with **many types of agents**

Thank you! Questions?