EpiRust

Building an ultra large-scale epidemic simulator using Rust language

Jayanta Kshirsagar & Sapana Kale

Engineering for Research (e4r™), Thoughtworks Technologies India.

/thoughtworks

From this talk

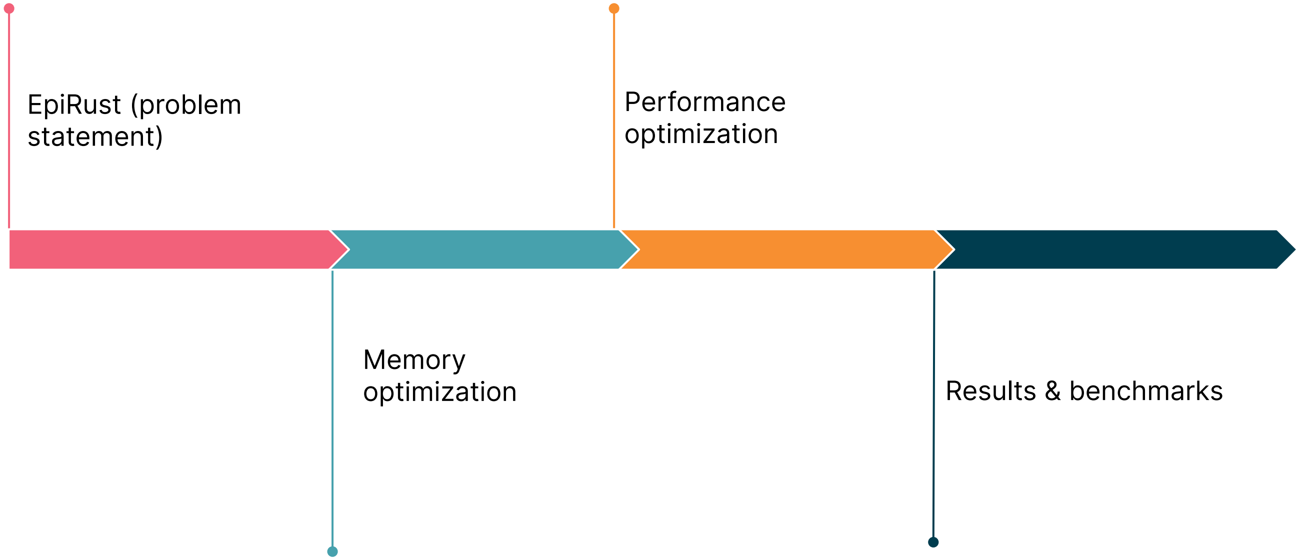
✓ What to expect?

- A case study in scientific computing
- Our journey of performance optimization to scale up and out
- What worked in Rust

✕ What is out of scope?

- In depth discussion about agent-based simulation models for Covid-19

Agenda



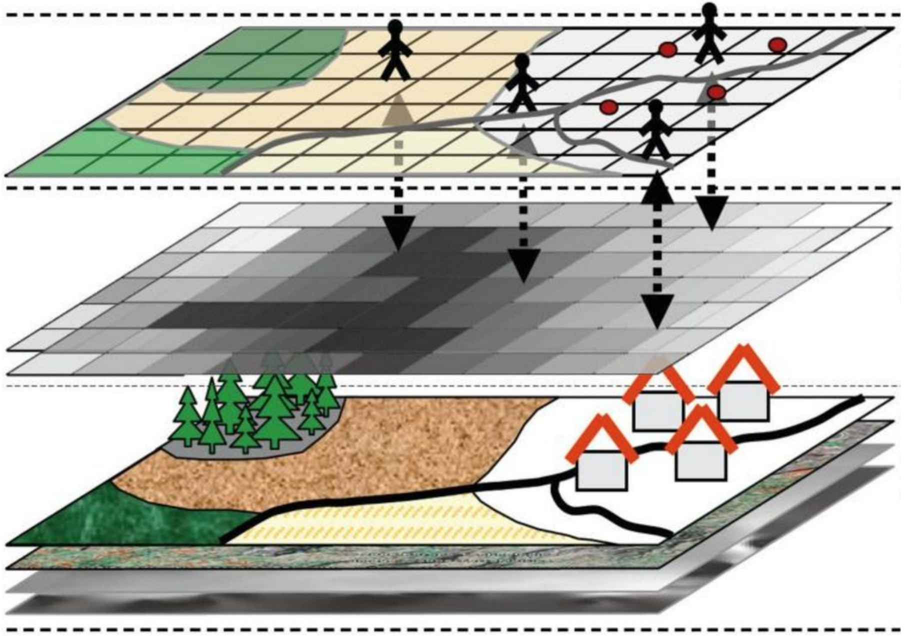
EpiRust (problem statement)

Memory optimization

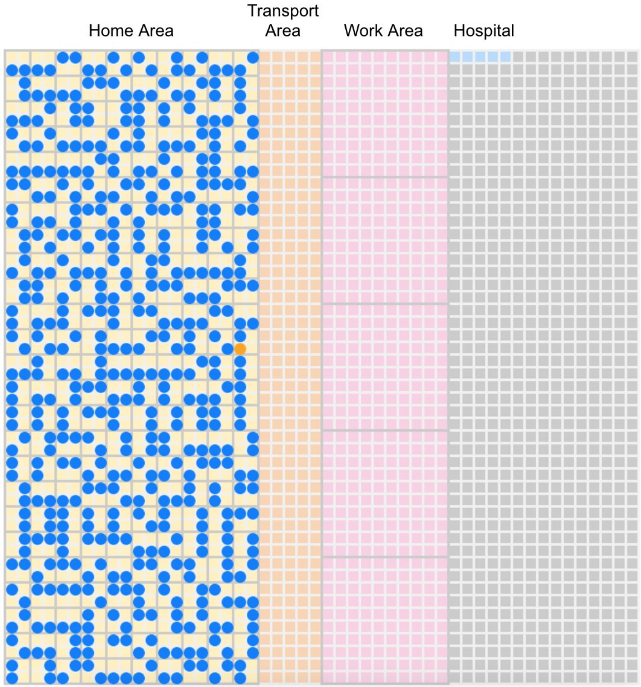
Performance optimization

Results & benchmarks

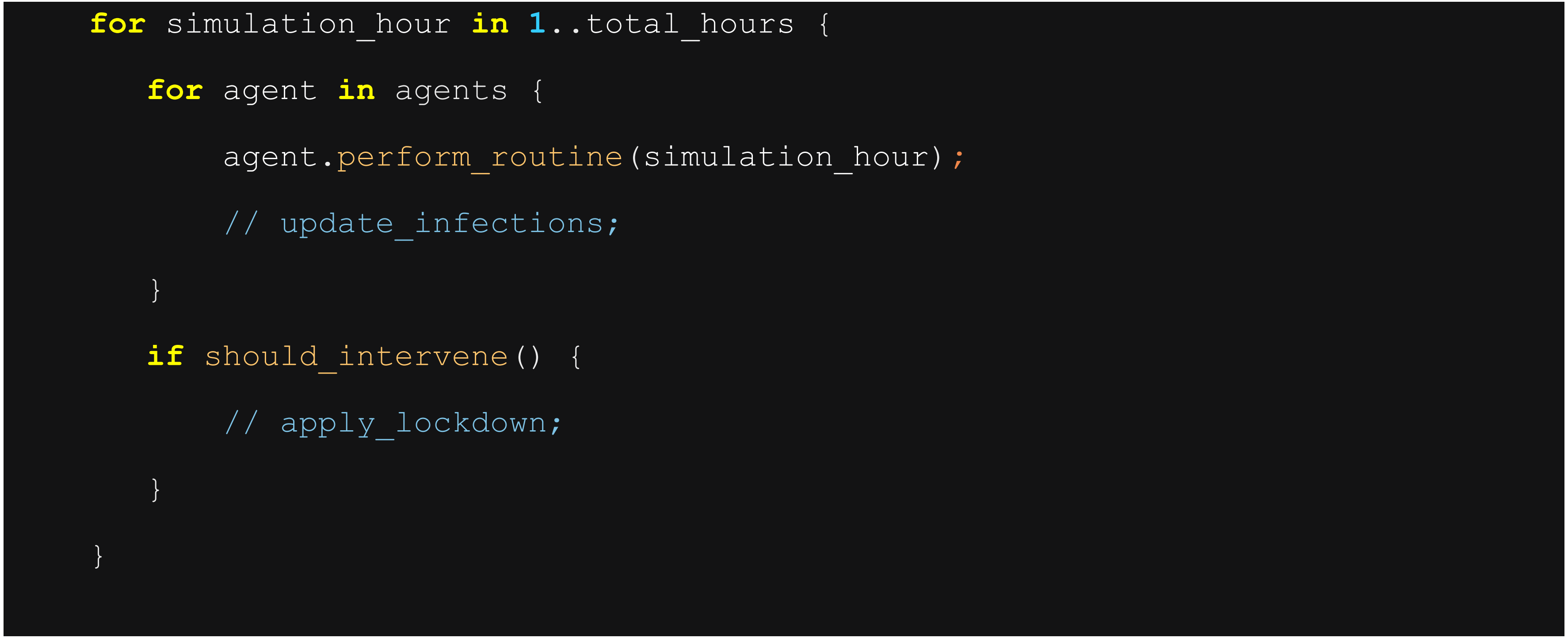
EpiRust - Agent-based, large-scale, open-source, epidemic simulator



source: https://www.mdpi.com/2413-8851/2/2/36

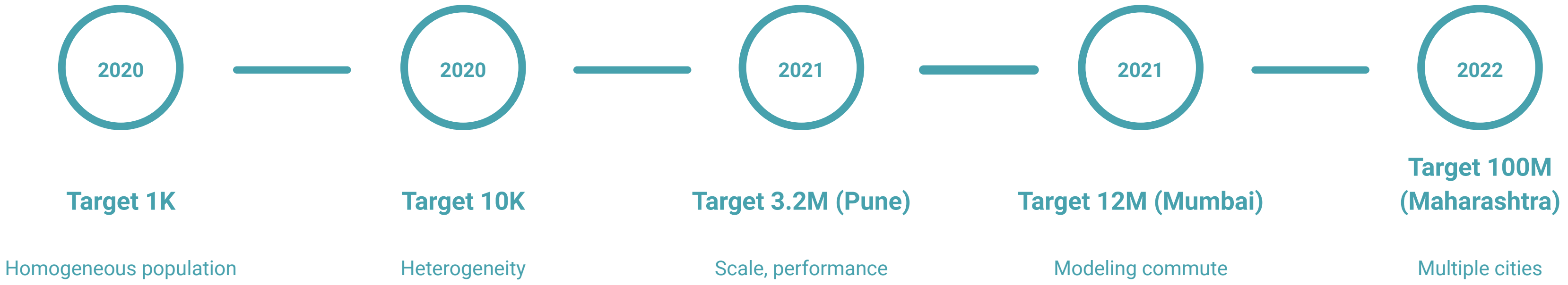


EpiRust Simplified



```
for simulation_hour in 1..total_hours {
    for agent in agents {
        agent.perform_routine(simulation_hour);
        // update_infections;
    }
    if should_intervene() {
        // apply_lockdown;
    }
}
```

A bird’s eye view of the journey



2020

Target 1K

Homogeneous population

2020

Target 10K

Heterogeneity

2021

Target 3.2M (Pune)

Scale, performance

2021

Target 12M (Mumbai)

Modeling commute

2022

Target 100M (Maharashtra)

Multiple cities

EpiRust Complexity

- Compute intensive (number of behaviours for 1080 simulated hours)
    - 1k population ~ 7 million
    - 1m population ~ 7 billion
    - 100m population ~ 700 billion
- Scale
    - Sparseness of the problem
    - Memory footprint
- Domain complexity (Disease dynamics, Interventions)
- Order of agent execution
    - The agents being executed one after another
    - 2D Buffering algorithm

How did we start?

Simulating Pune city

- Serial, grid based implementation
- Population ~ 3.2 Million
- 5660x5660 = 32,035,600 cells
- Memory consumption: Approx. 5-10GB

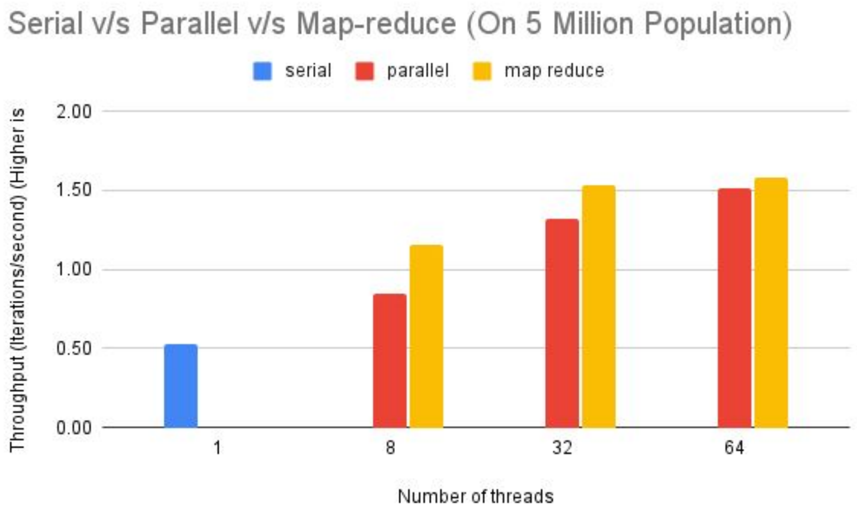
Photo by Sopan Shewale on Unsplash

Optimization for memory

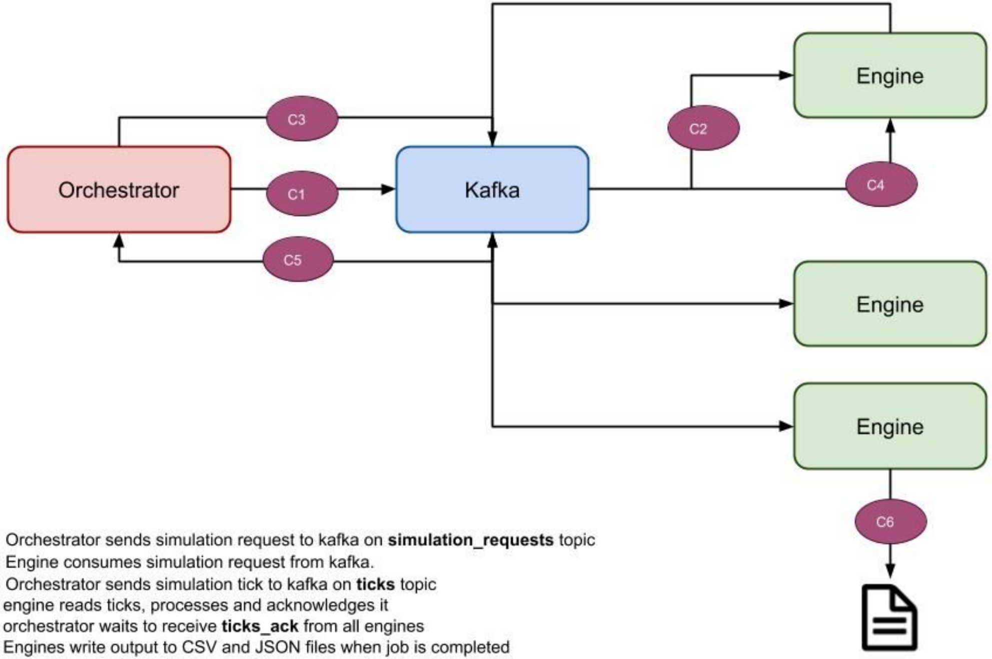
- Representing grid As a **HashMap:**
  - Map<Point, Citizen>
  - Number of agents = number of entries
  - O(1) operations
  - Memory: few 100 mb
- Choosing optimal hashing algorithm
  - HashBrown with AHash
  - Comparing FXHash, **FNV**, and many others

Optimization for throughput

- Parallelization
  - Map-reduce
  - Parallel iterators



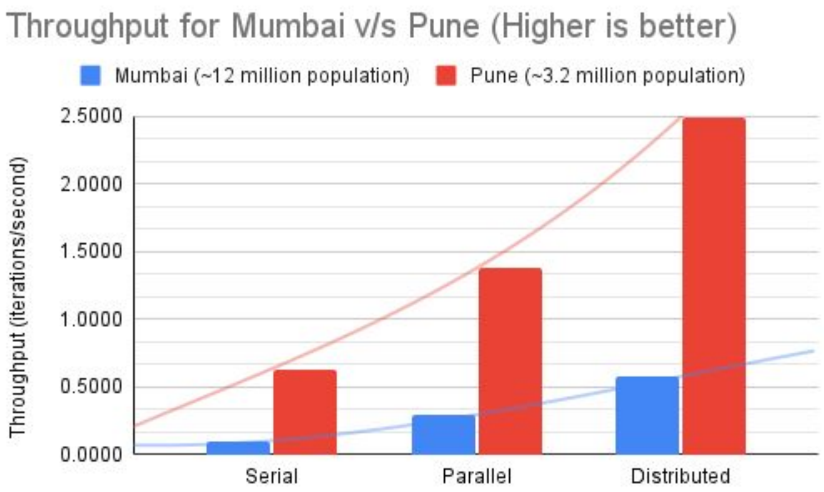
Scaling out for modeling multiple/ larger cities



C1 Orchestrator sends simulation request to kafka on **simulation_requests** topic
C2 Engine consumes simulation request from kafka.
C3 Orchestrator sends simulation tick to kafka on **ticks** topic
C4 engine reads ticks, processes and acknowledges it
C5 orchestrator waits to receive **ticks_ack** from all engines
C6 Engines write output to CSV and JSON files when job is completed
Additionally
Engine consumes incoming travelers, and sends outgoing travelers back to kafka, orchestrator updates travel matrix when necessary

Optimization for throughput

- Distributed setup
    - Mumbai v/s Pune
    - Mumbai (0.5M * 24) v/s 100K * 100



No. of agents	No. of engines	Total Population	Throughput
0.5M	24	12M	0.57
100K	100	10M	3.03

Cloud migration for further scale out

- Epirust containerisation
- Using Kubernetes (k8s) for managing containers at scale
- Helm chart to package the application
- ELK Stack, Prometheus & Grafana for logging and monitoring purpose

Rust features

- Closer to metal performance
  - No runtime, no garbage collection
- Memory management
  - Comparison with C, CPP, etc.
- Fearless concurrency
  - Compile error rather than exceptions
- Productivity
  - Ecosystem

Team

Tarun Abichandani

Shabbir Bawaji

Shubham Chauhan

Akshay Dewan

Meenakshi Dhanani

Harshal Hayatnagarkar

Sapana Kale

Swapnil Khandekar

Dhananjay Khaparkhuntikar

Jayanta Kshirsagar

Saurabh Mookherjee

Bhawna Sharma

Vatsala Verma

Chhaya Yadav

Dr. Mihir Arjunwadkar - SP Pune University (Collaborator)

Dr. Gautam Menon - Ashoka University (Collaborator)

Thank You!

More about EpiRust can be found here.

Sapana Kale
kasapana@thoughtworks.com

Jayanta Kshirsagar
jayantak@thoughtworks.com

Engineering for research (e4r)
e4r@thoughtworks.com

/thoughtworks