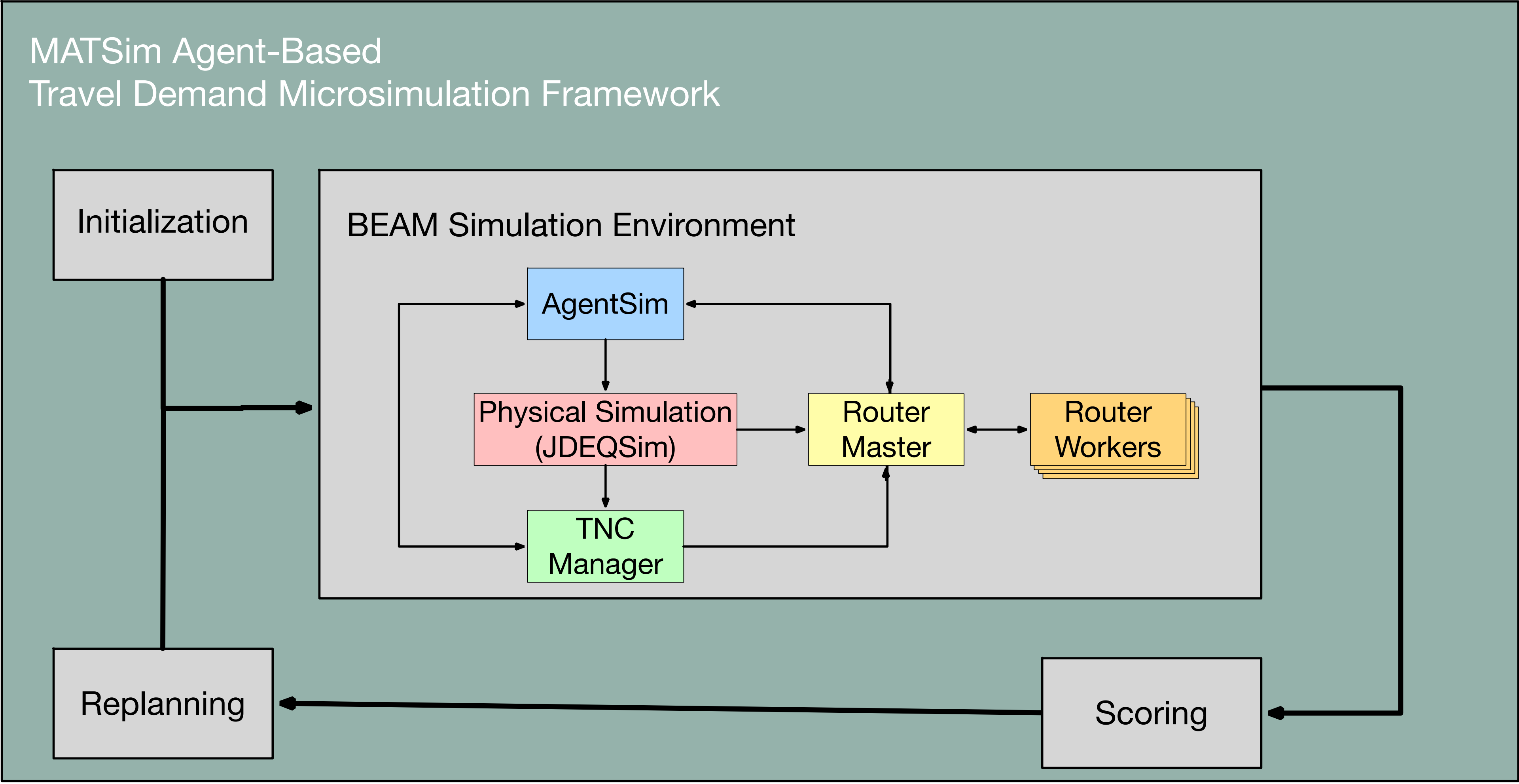
Achieving massively parallel agent-based microsimulations through the actor model of computation

Presenter: Sid Feygin, *UC Berkeley*

Development Team:

Colin Sheppard, Rashid Waraich, Andrew Campbell, UC *Berkeley*
Michael Zilske, Anand Gopal, *Lawrence Berkeley National Lab*
Art Balayan, Zeeshan Bilal, Justin Pihony, *Seven Summits*

7th Innovations in Travel Modelling Conference
June 25, 2018 -- Atlanta, GA


MATSim Agent-Based
Travel Demand Microsimulation Framework
Initialization
BEAM Simulation Environment
AgentSim
Physical Simulation
(JDEQSim)
Router
Master
Router
Workers
TNC
Manager
Replanning
Scoring

Concurrency Challenges in BEAM

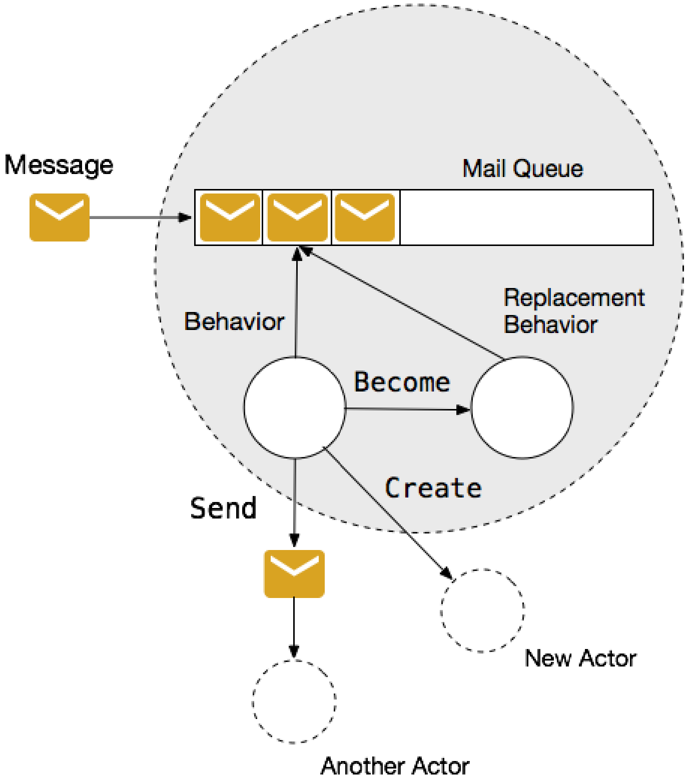
BEAM Goals:

- Simulate within-day mobility behavior for hundreds to millions of synthetic agents.
- Behavioral mode choice model based on random utility maximization
  - Compute optimal route for feasible mode alternatives between activities.
- Maintain compatibility with existing MATSim codebase (to extent possible)

Challenges:

- Router to asynchronously handle multimodal requests for each agent multiple times per day.
- Take best advantage of AWS cloud infrastructure to scale to size of simulation while minimizing execution time.
- Multi-threaded inter-agent communication required for within-day planning

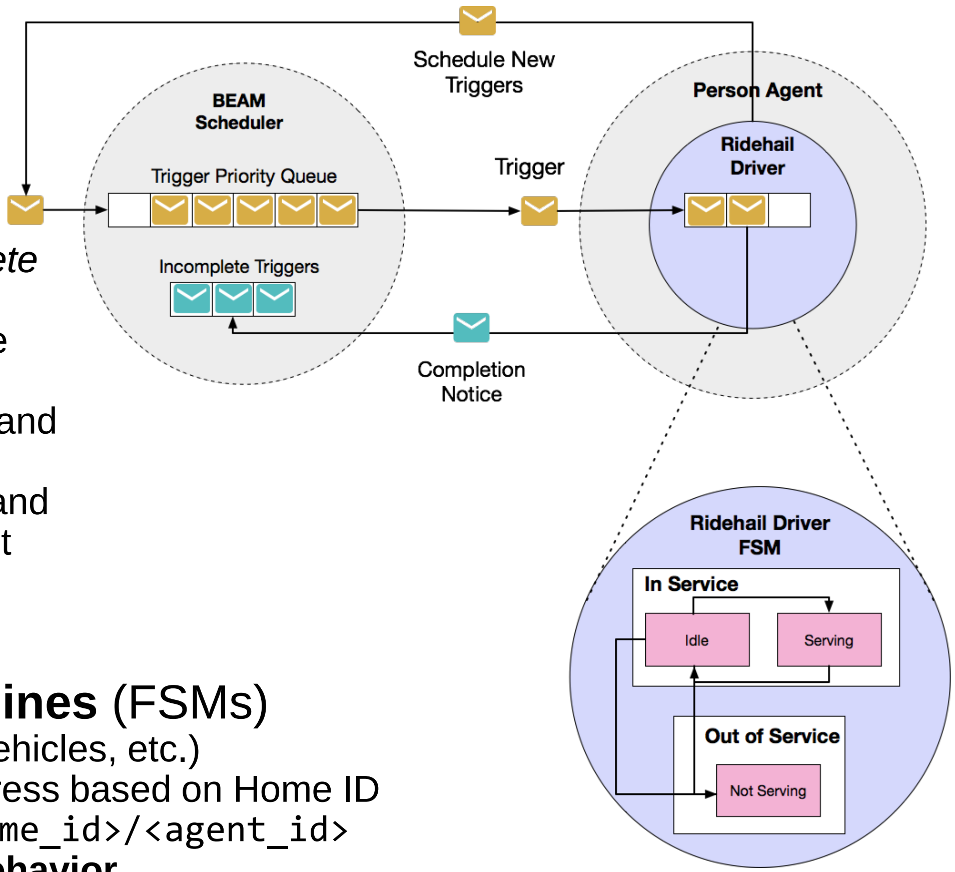
Actor Model of Computation - Brief Overview



- Actors:
  - Lightweight addressable thread encapsulating **state** and **behaviors** as reaction to **messages**
  - Actors can only message other actors they “know”
  - Specialized for concurrency
    - No shared memory (state) between actors
    - Lock-free, non-blocking
- Actor System:
  - Hierarchical
  - Fault tolerant (“Let It Crash”)
- Implementation (Akka)
  - Runs on Java Virtual Machine (**Scala**/Java)
  - Built-in networking (remoting) and cluster capabilities
  - Lightweight actors and messages

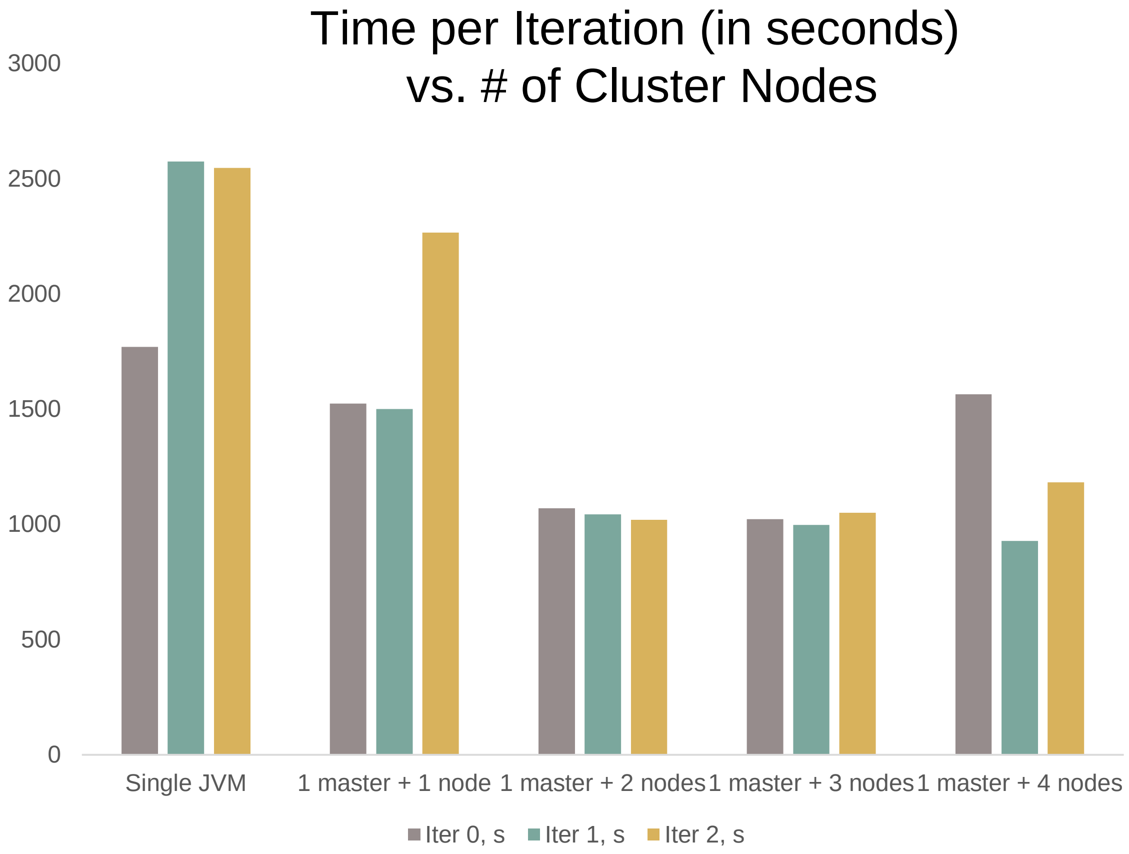
Agents as Actors

- BEAM Simulation Environment
    - **BEAM Scheduler** controls *parallel discrete event simulation* using **Triggers**
    - Any **BEAMAgent** can add Triggers to the **Trigger Priority Queue**
    - Triggers are dispatched *asynchronously* and over a defined **window**
    - Scheduler holds all dispatched Triggers and waits for a **completion notice** from agent before moving window forward.



- BEAM Agents are **finite state machines** (FSMs)
    - Abstraction over agent types (persons, vehicles, etc.)
    - Each agent-actor has unique logical address based on Home ID
        - *Example*: `../sys/user/beam/<home_id>/<agent_id>`
    - **Event** message may activate different **behavior**
        - *New behavior*: react differently or to different set of triggers

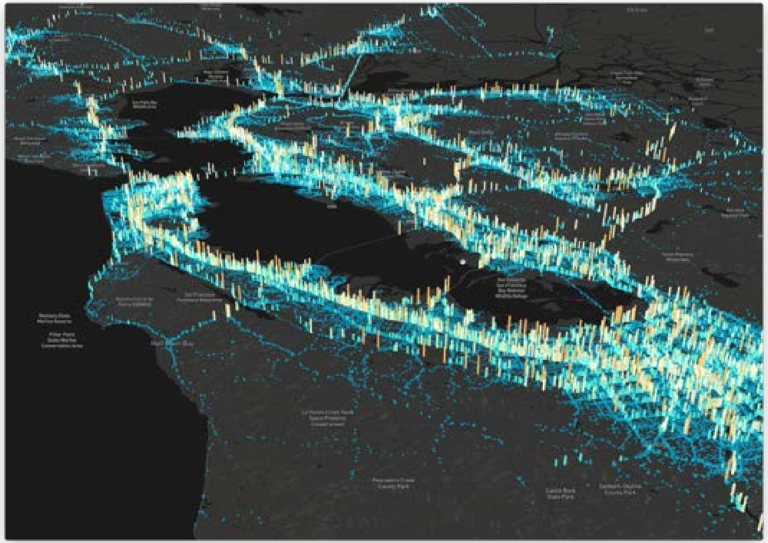
Results: Router Clustering



Conclusions

Benefits of Actor Model in BEAM:

- Focus on developing asynchronous within-day behavior rather than debugging low level concurrency complexities.
- Akka is well-supported on JVM, simplifying compatibility with MATSim.
- Scale routing horizontally across several nodes.



Ongoing Parallelization Work:

- Fixing processing lags due to garbage collection
- So far, only router uses clustering capability– considering parallelization of full model.

QUESTIONS?