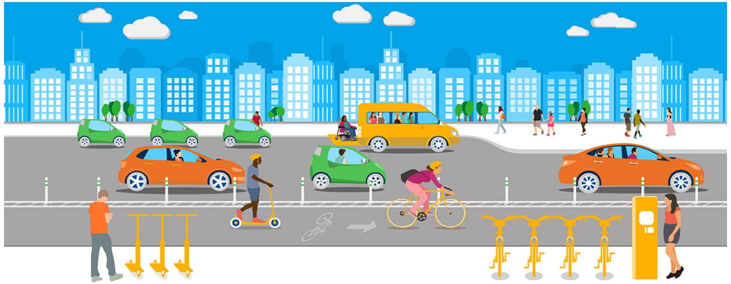
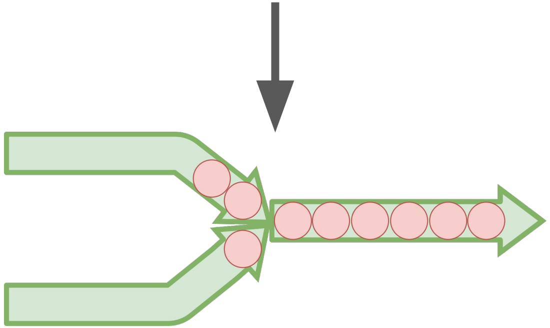
Swiss-Scale Multi-Mode Transport Simulation

Rodrigo Bruno, Michel Mueller, Gustavo Alonso, Torsten Hoefler
Systems Group, ETH Zurich

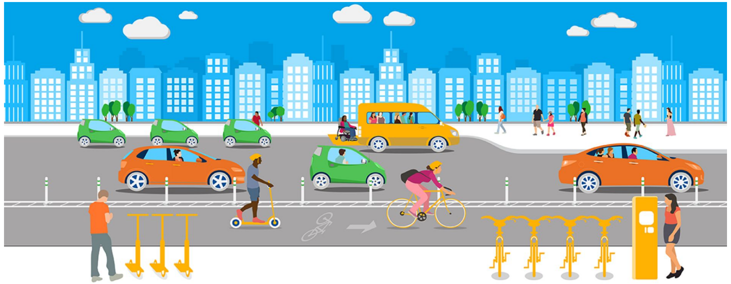
Systems@ETH zürich

Multi-Agent Transport Simulation

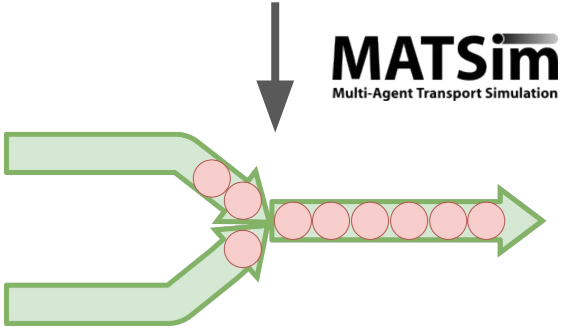




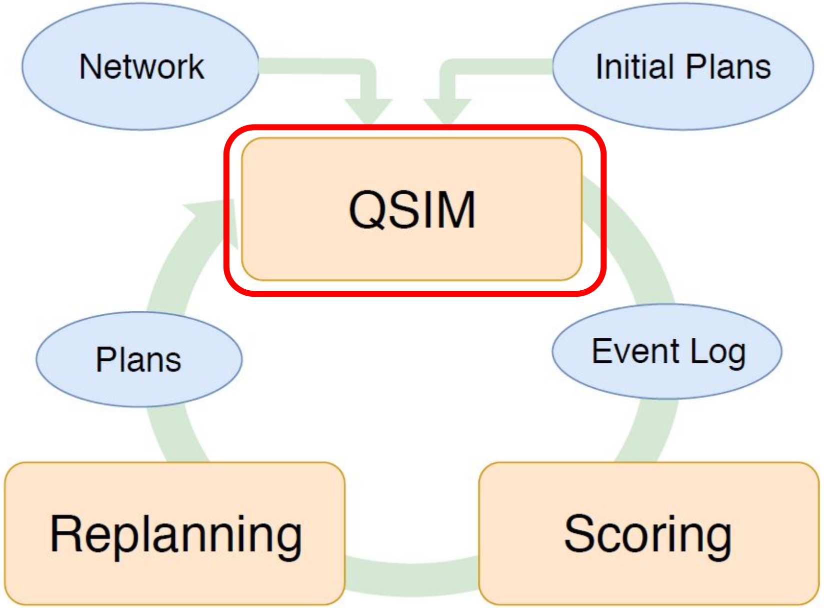
Multi-Agent Transport Simulation



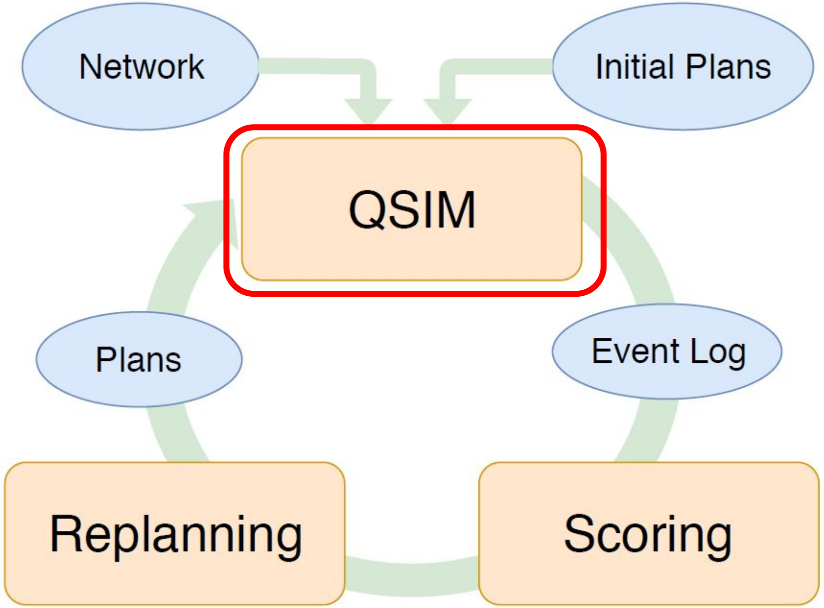
MATSim
Multi-Agent Transport Simulation

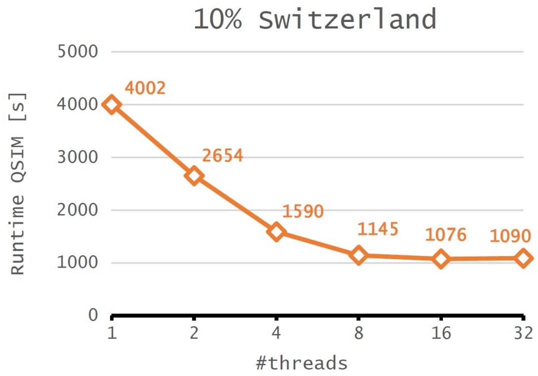


MATSim Recap

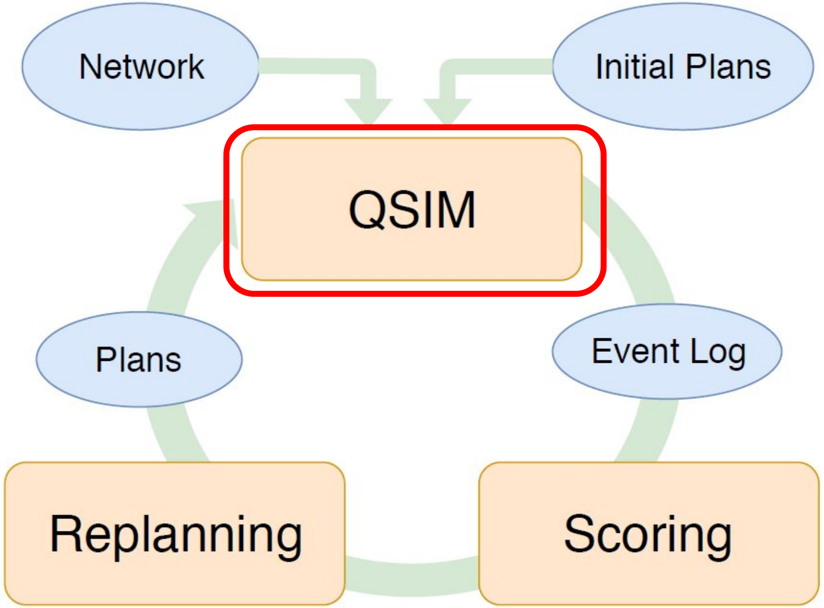


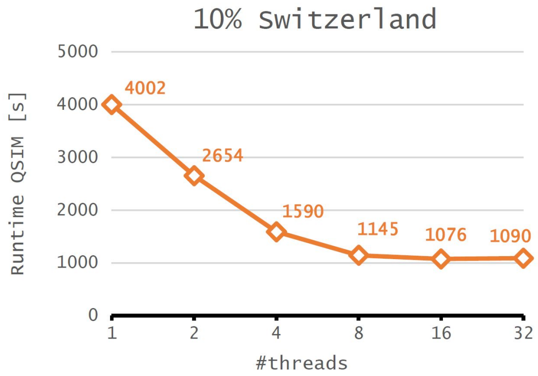
MATSim Recap





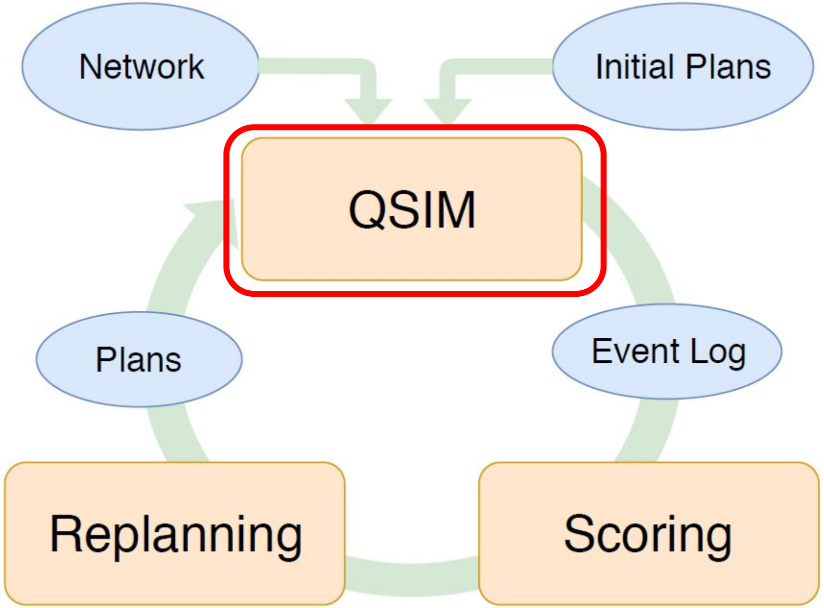
MATSim Recap

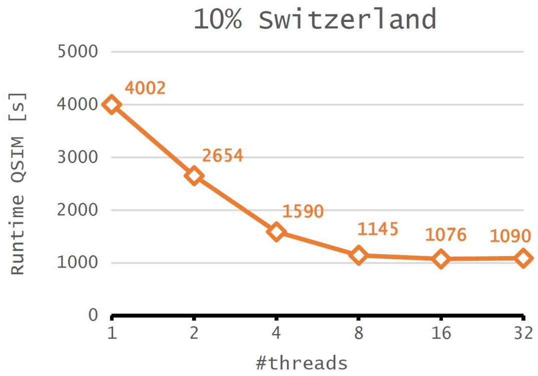




Can we scale to 100% CH?

MATSim Recap

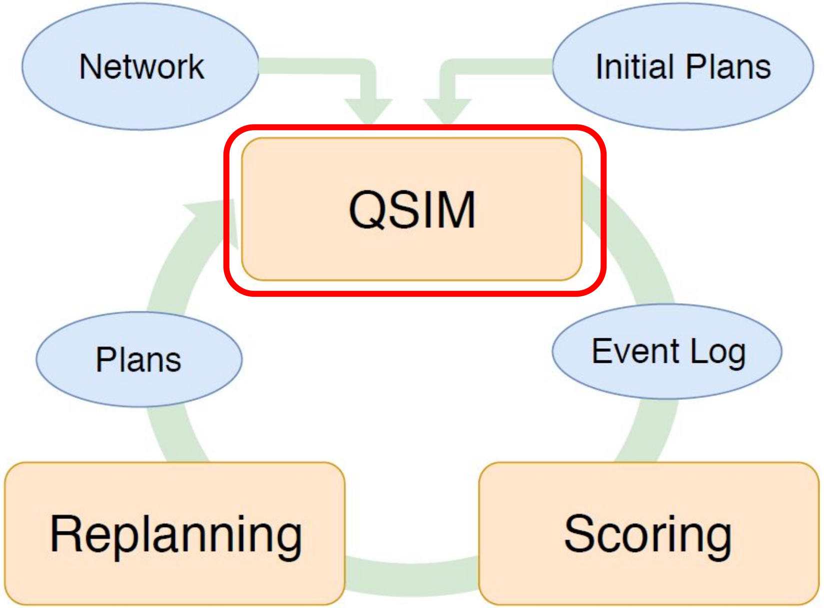


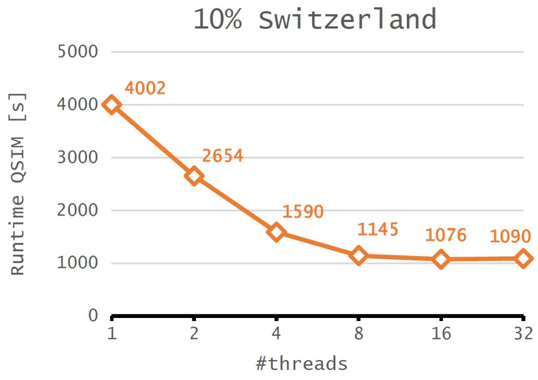


Can we scale to 100% CH?

10 x more agents * 1,000 iterations / seconds in day

MATSim Recap



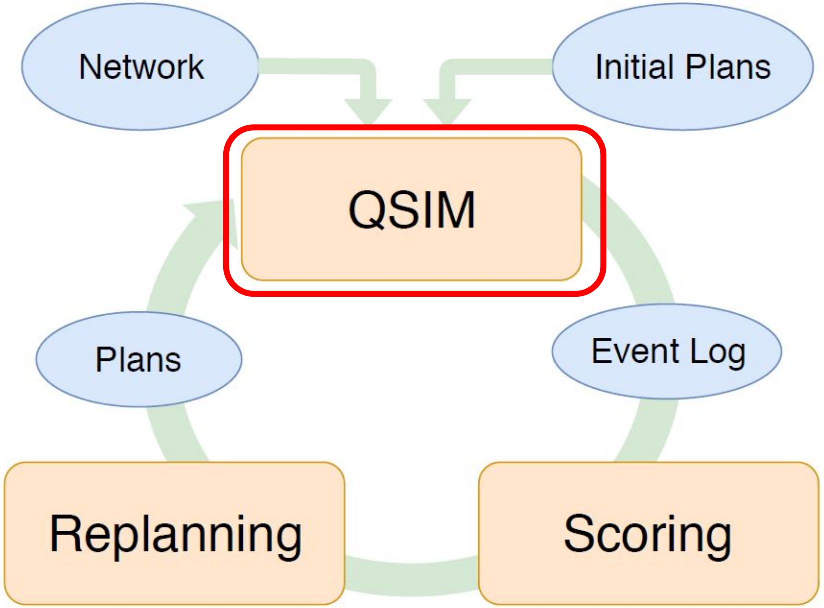


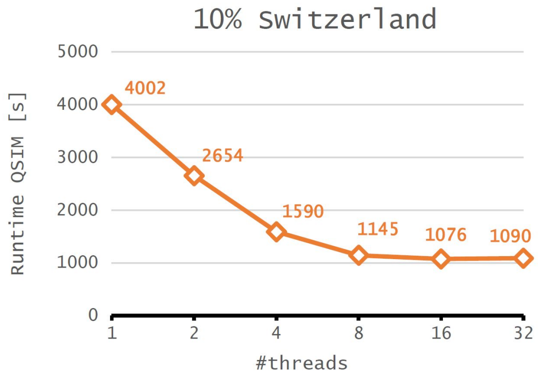
Can we scale to 100% CH?

10 x more agents * 1,000 iterations / seconds in day

~= 10 * 1,000 * 1,076 / 86,400

MATSim Recap





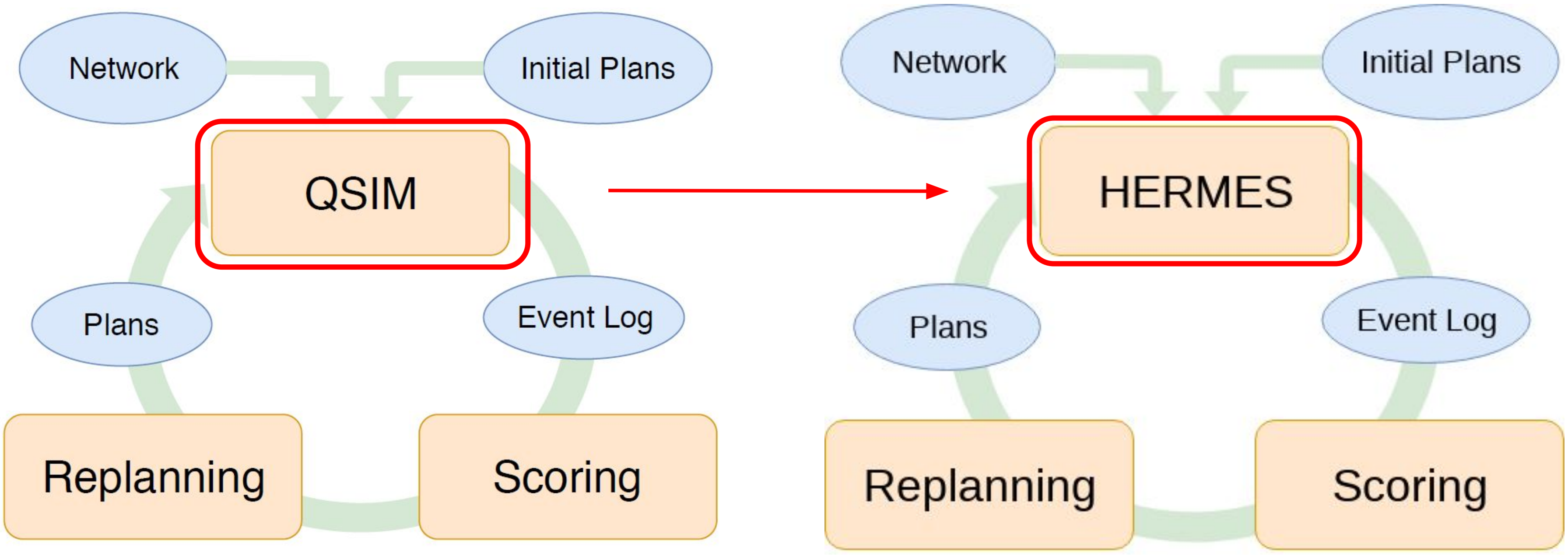
Can we scale to 100% CH?

10 x more agents * 1,000 iterations / seconds in day

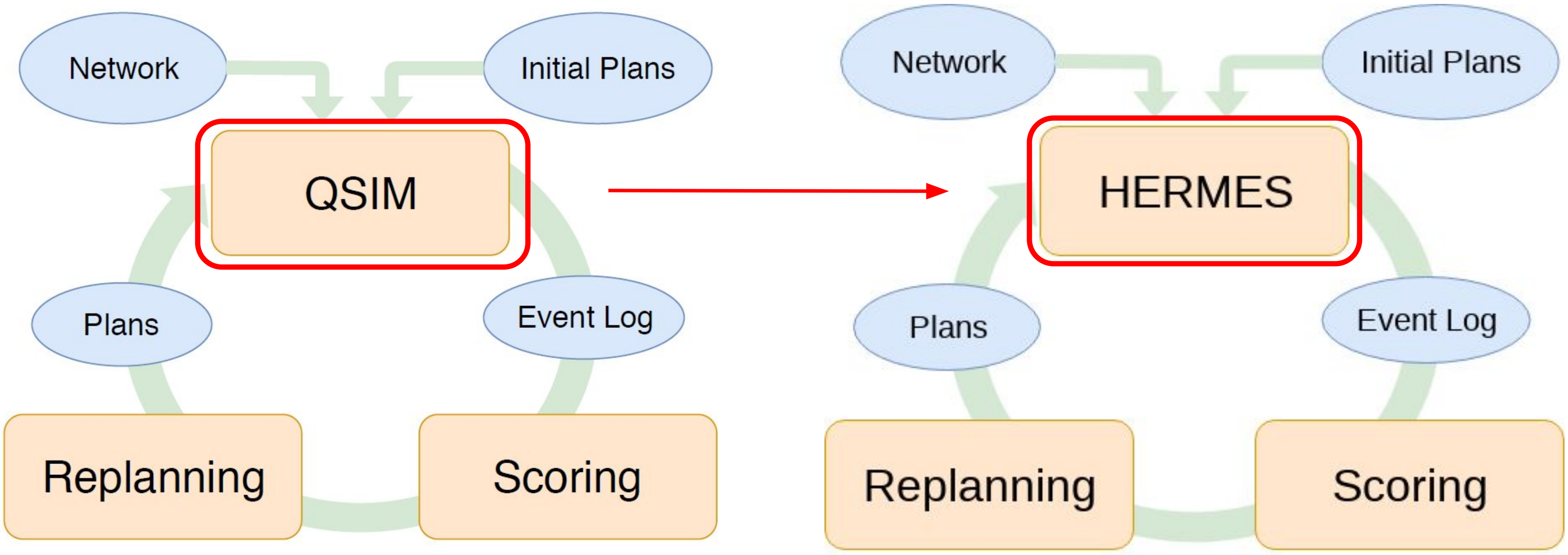
~= 10 * 1,000 * 1,076 / 86,400

~= **116 days**

High pERformance Multi-mode transport nEtwork Simulation



High pERformance Multi-mode transport nEtwork Simulation



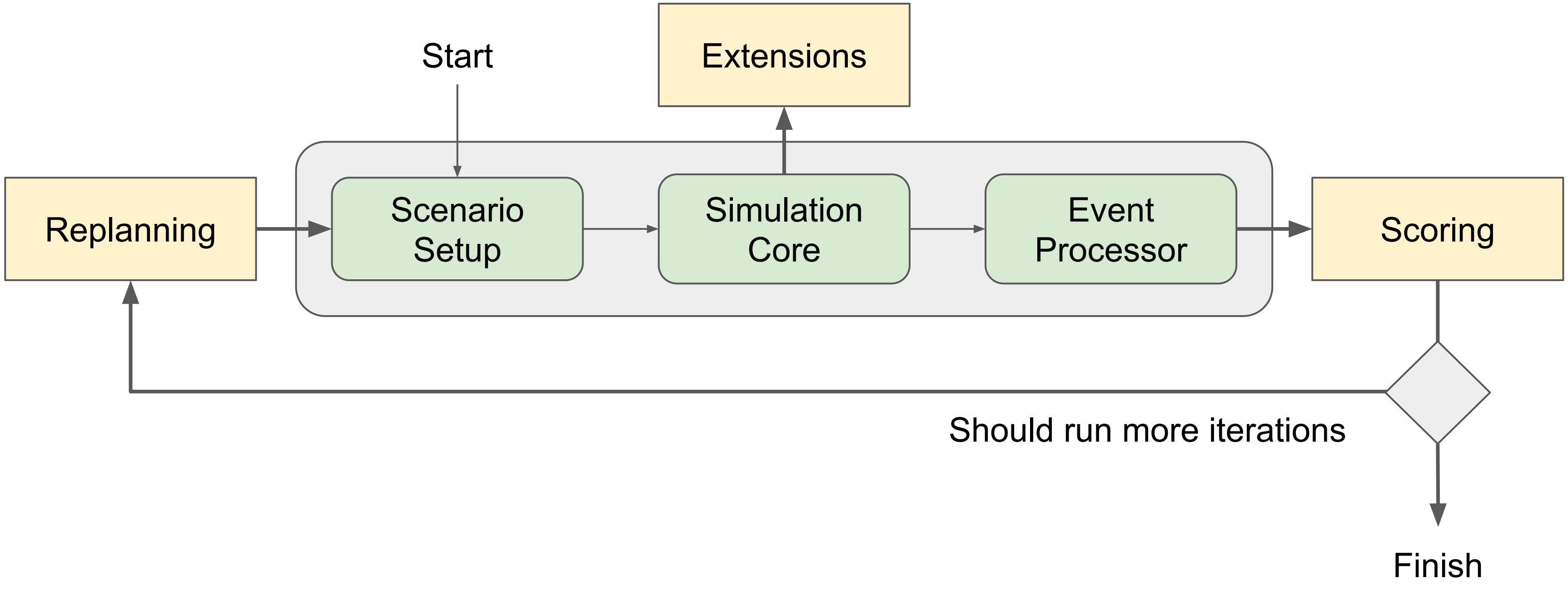
Goal: Faster end-to-end simulation for large scale scenarios!

HERMES - Design Principles

- **Event driven simulation**
  - simulation effort is proportional to the number of triggered events
  - HERMES reacts when something “happens” in the network (similar to Charypar et al.)

- **Optimize for the Common Case**
  - Very optimized fast-path for the common/standard simulation features
  - Non-standard features execute outside the optimized core
    - Easy to extend

HERMES - Architecture



Scenario Setup (MATSim to HERMES)

- **Data is compressed as much as possible**
  - No String identifiers -> only 32 bit integers
  - Plan is an array of 64 bit values that encode a network interaction

- **Data is stored to avoid multiple hops in memory**
  - No use of lists and limited use of maps
  - Algorithms are designed to use only identifiers and other pre-computed values

- **Data structures are only built once (first iteration)**
  - Further iterations only update the plans

Core Simulation Algorithm (HERMES)

Algorithm 1 Hermes Simulation Algorithm

```
procedure SIMULATE
    for step in iteration do
        for agent in delayed_agents(step) do
            Process_Agent(agent)
        for link in delayed_links(step) do
            Process_Link(link)
procedure PROCESS_AGENT(agent)
    switch agent.plan.top do
        case leg
            // handle agent leg, push to link
        case activity
            // handle agent activity, add to delayed_agents
        case pt
            // handle public transport interaction
        case default
            // call extension code to process plan entry
procedure PROCESS_LINK(link)
    while can_process_agent(link.top) do
        Process_Agent(link.top)
        link.pop()
    delayed_links[link.top.finish].append(link)
```

Core Simulation Algorithm (HERMES)

Algorithm 1 Hermes Simulation Algorithm

```
procedure SIMULATE
    for step in iteration do
        for agent in delayed_agents(step) do
            Process_Agent(agent)
        for link in delayed_links(step) do
            Process_Link(link)
procedure PROCESS_AGENT(agent)
    switch agent.plan.top do
        case leg
            // handle agent leg, push to link
        case activity
            // handle agent activity, add to delayed_agents
        case pt
            // handle public transport interaction
        case default
            // call extension code to process plan entry
procedure PROCESS_LINK(link)
    while can_process_agent(link.top) do
        Process_Agent(link.top)
        link.pop()
    delayed_links[link.top.finish].append(link)
```

Event-driven

Core Simulation Algorithm (HERMES)

Algorithm 1 Hermes Simulation Algorithm

```
procedure SIMULATE
    for step in iteration do
        for agent in delayed_agents(step) do
            Process_Agent(agent)
        for link in delayed_links(step) do
            Process_Link(link)
procedure PROCESS_AGENT(agent)
    switch agent.plan.top do
        case leg
            // handle agent leg, push to link
        case activity
            // handle agent activity, add to delayed_agents
        case pt
            // handle public transport interaction
        case default
            // call extension code to process plan entry
procedure PROCESS_LINK(link)
    while can_process_agent(link.top) do
        Process_Agent(link.top)
        link.pop()
    delayed_links[link.top.finish].append(link)
```

Event-driven

Known plan entry types are fast

Core Simulation Algorithm (HERMES)

Algorithm 1 Hermes Simulation Algorithm

```
procedure SIMULATE
    for step in iteration do
        for agent in delayed_agents(step) do
            Process_Agent(agent)
        for link in delayed_links(step) do
            Process_Link(link)
procedure PROCESS_AGENT(agent)
    switch agent.plan.top do
        case leg
            // handle agent leg, push to link
        case activity
            // handle agent activity, add to delayed_agents
        case pt
            // handle public transport interaction
        case default
            // call extension code to process plan entry
procedure PROCESS_LINK(link)
    while can_process_agent(link.top) do
        Process_Agent(link.top)
        link.pop()
    delayed_links[link.top.finish].append(link)
```

Event-driven

Known plan entry types are fast

Unknown plan entry triggers callback

Core Simulation Algorithm (HERMES)

Algorithm 1 Hermes Simulation Algorithm

```
procedure SIMULATE
    for step in iteration do
        for agent in delayed_agents(step) do
            Process_Agent(agent)
        for link in delayed_links(step) do
            Process_Link(link)
procedure PROCESS_AGENT(agent)
    switch agent.plan.top do
        case leg
            // handle agent leg, push to link
        case activity
            // handle agent activity, add to delayed_agents
        case pt
            // handle public transport interaction
        case default
            // call extension code to process plan entry
procedure PROCESS_LINK(link)
    while can_process_agent(link.top) do
        Process_Agent(link.top)
        link.pop()
    delayed_links[link.top.finish].append(link)
```

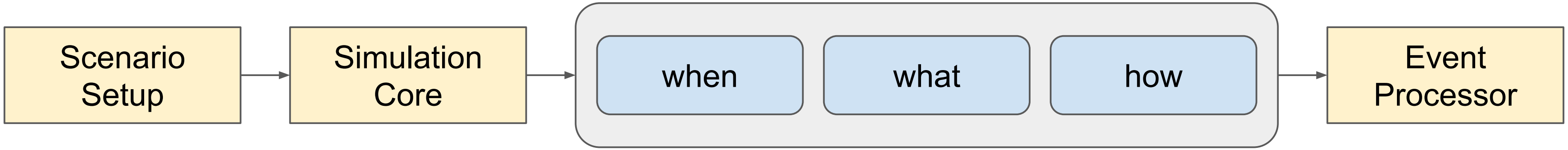
Event-driven

Known plan entry types are fast

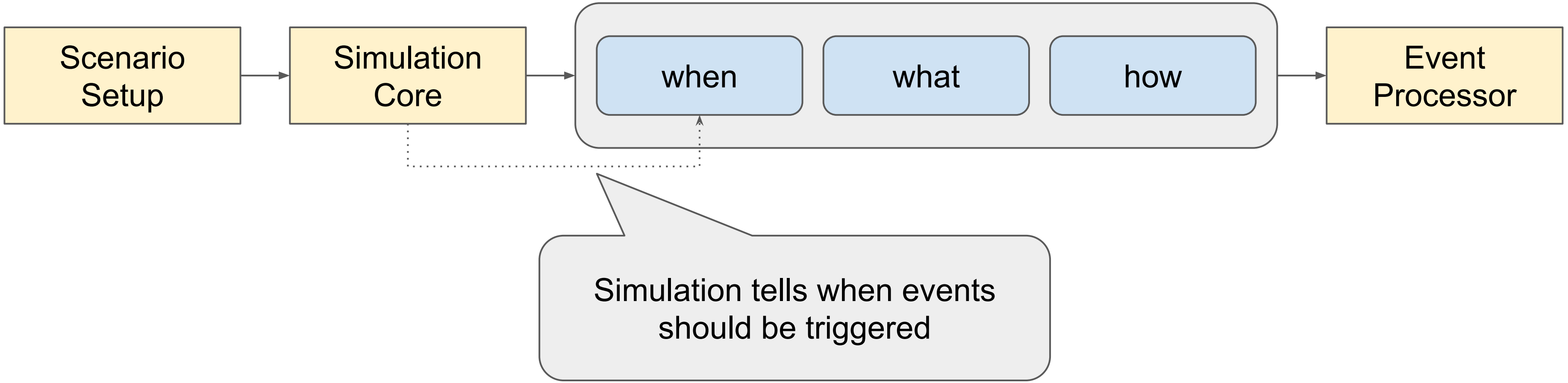
Unknown plan entry triggers callback

Time of the next activation is calculated for links and agents

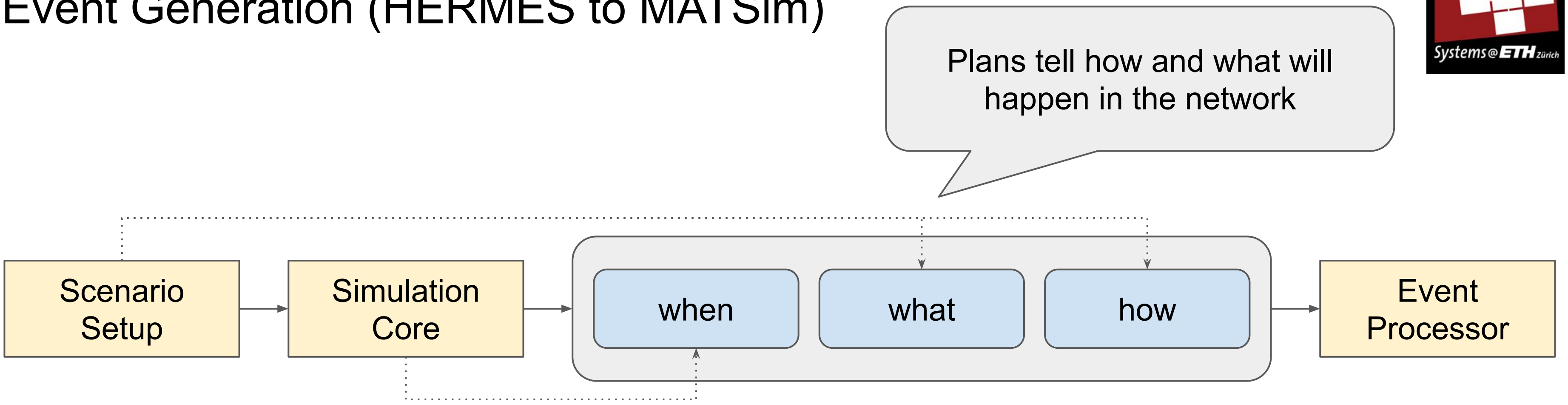
Event Generation (HERMES to MATSim)



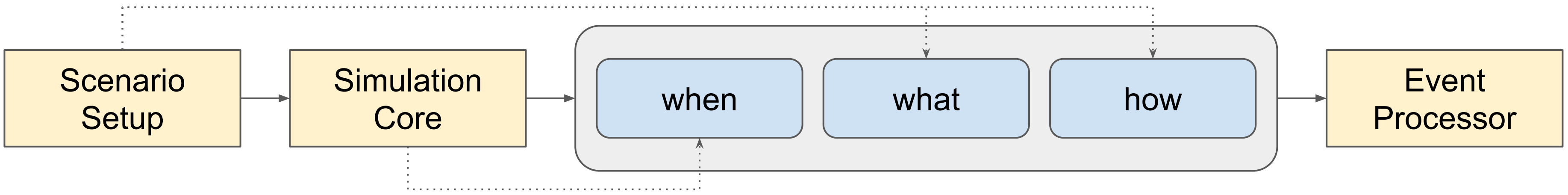
Event Generation (HERMES to MATSim)



Event Generation (HERMES to MATSim)

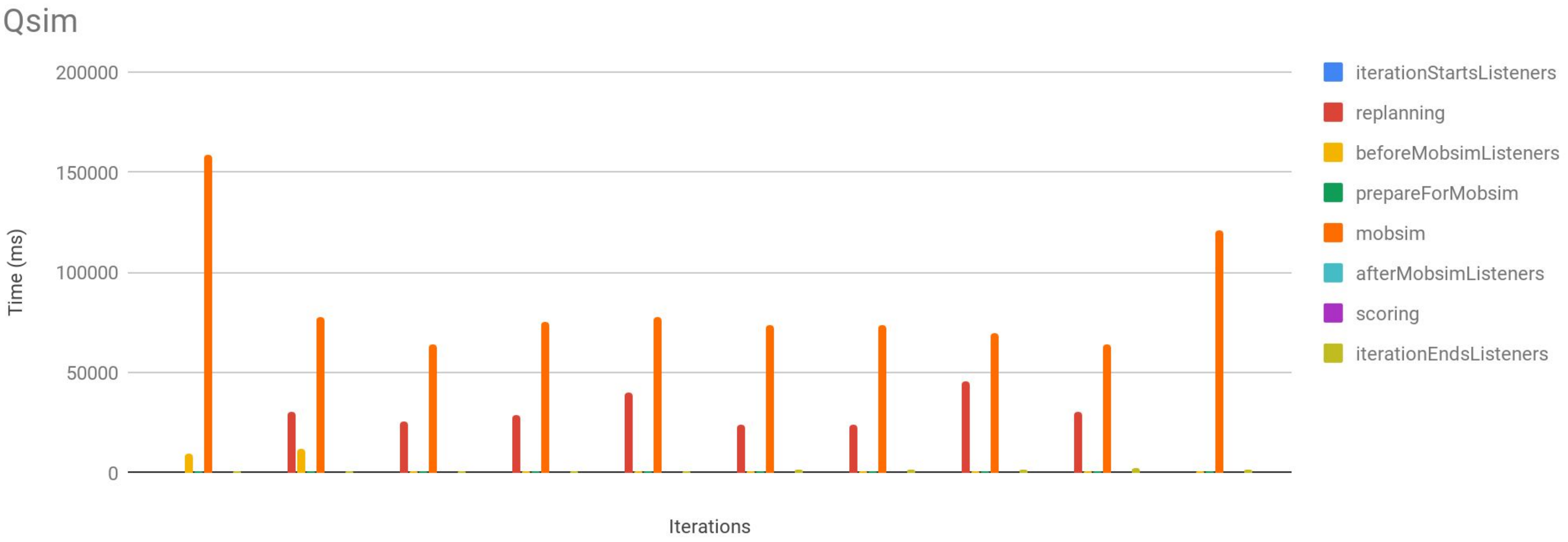


Event Generation (HERMES to MATSim)

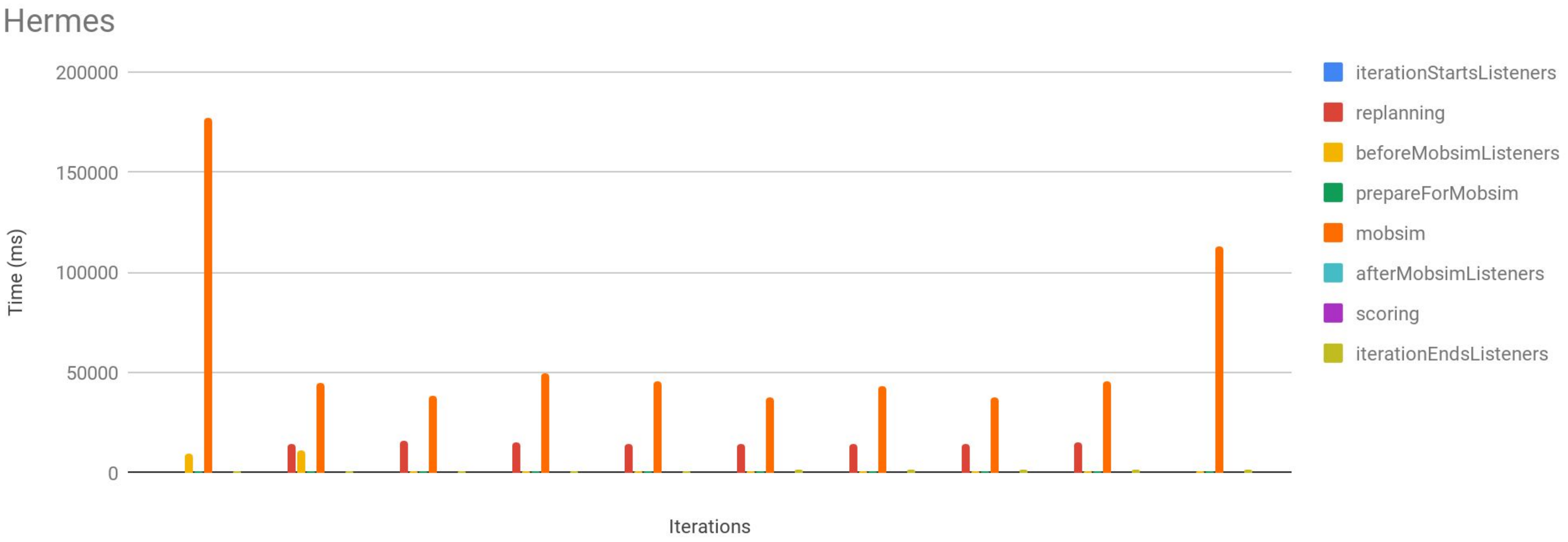


Events are pre-generated at Scenario Setup. Simulation only tags the time.

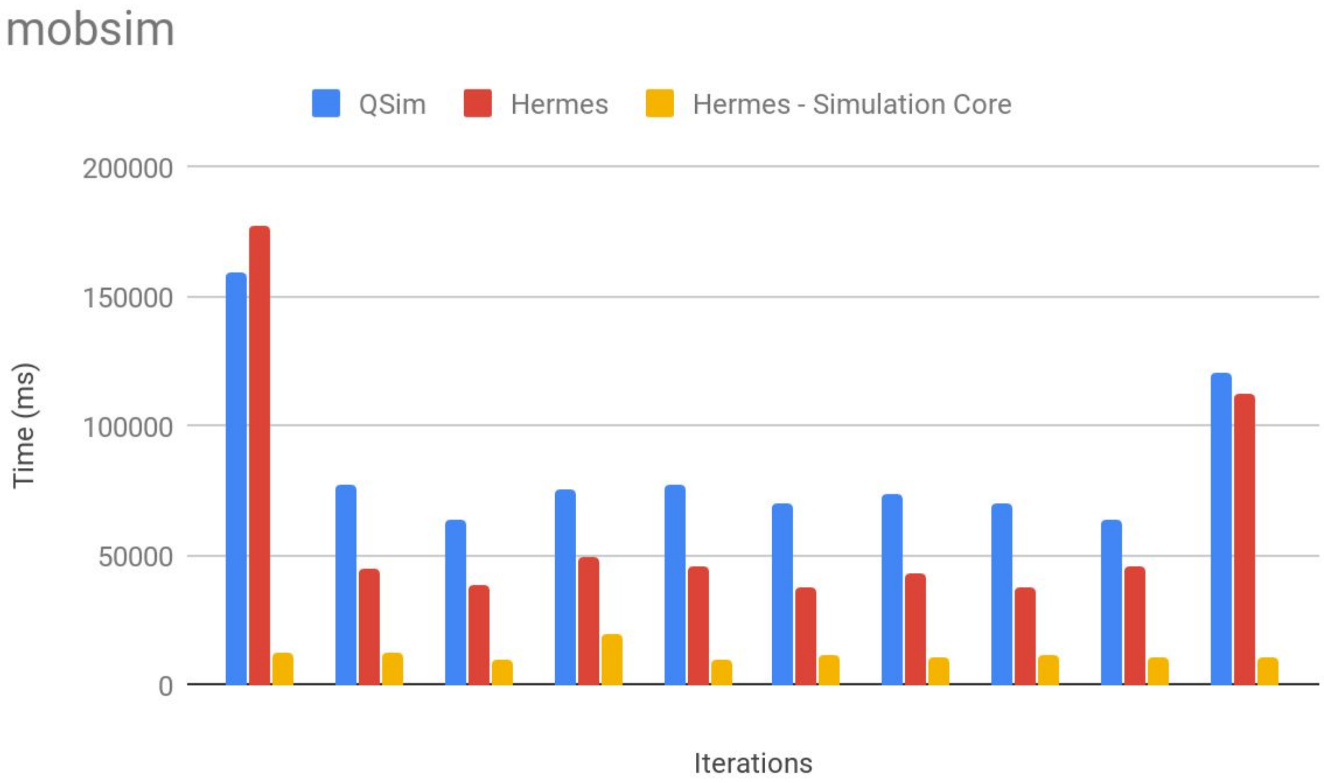
Performance Analysis (Berlin 1%)



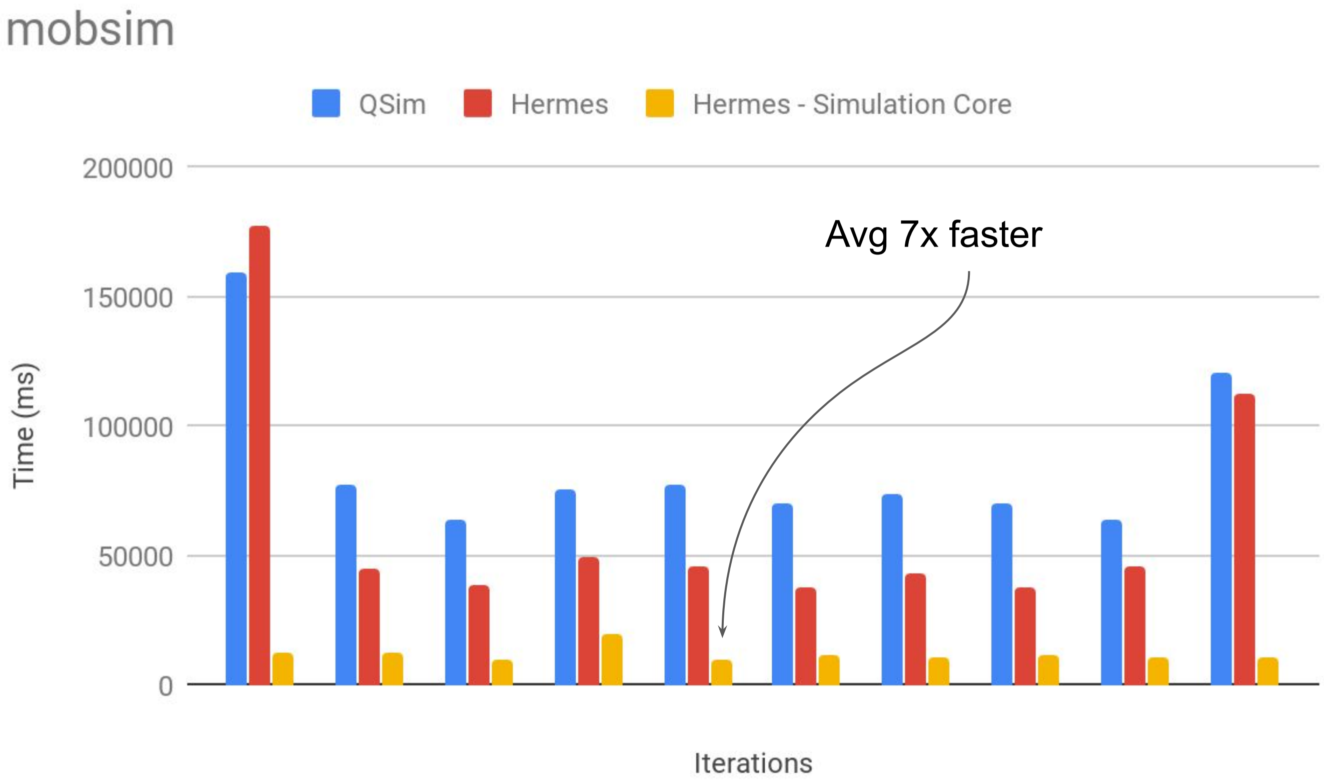
Performance Analysis (Berlin 1%)



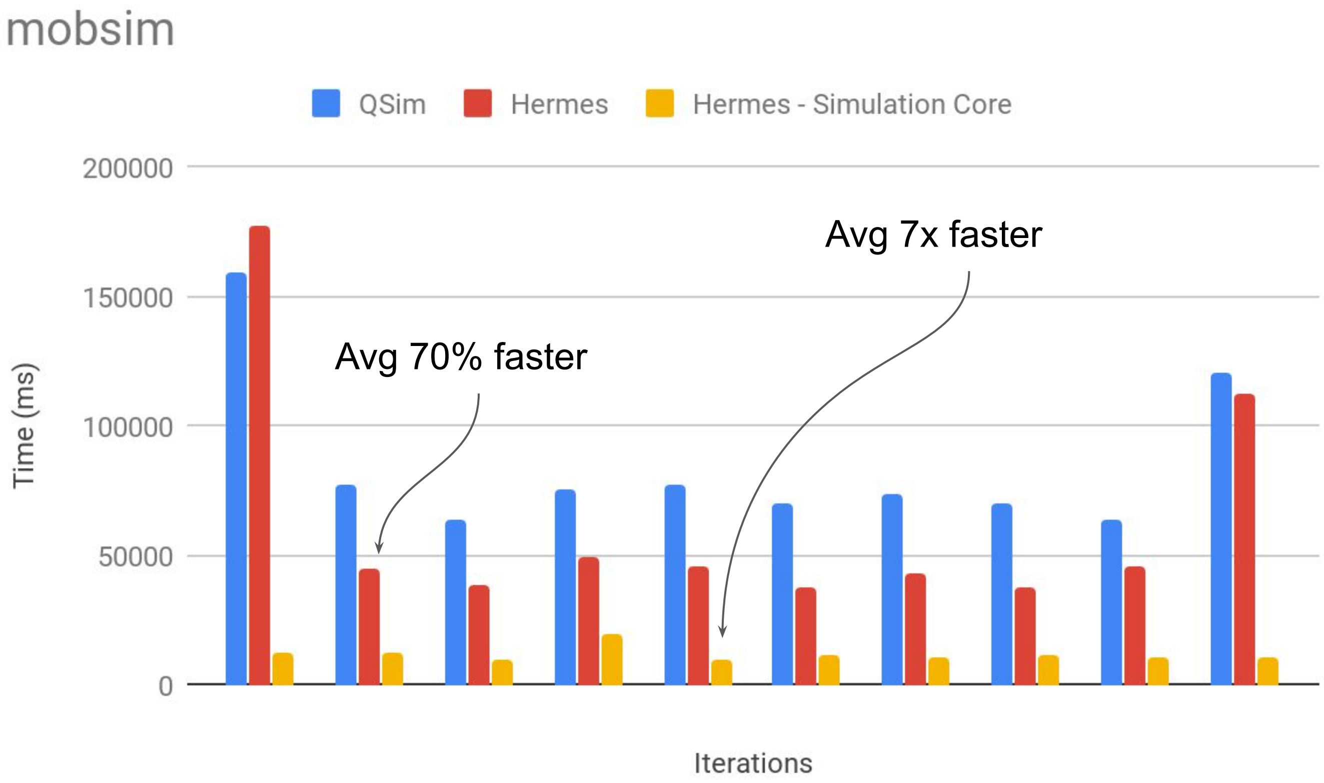
Performance Analysis (Berlin 1%)



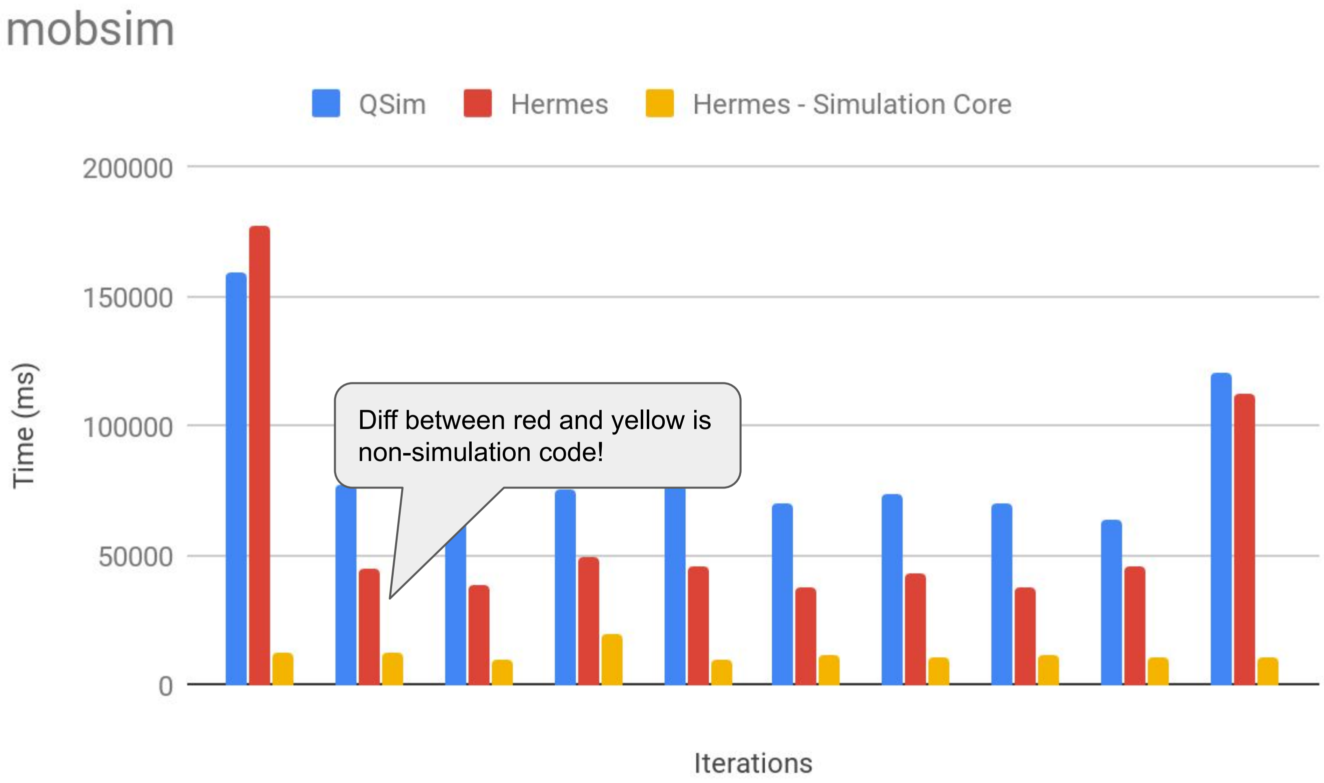
Performance Analysis (Berlin 1%)



Performance Analysis (Berlin 1%)



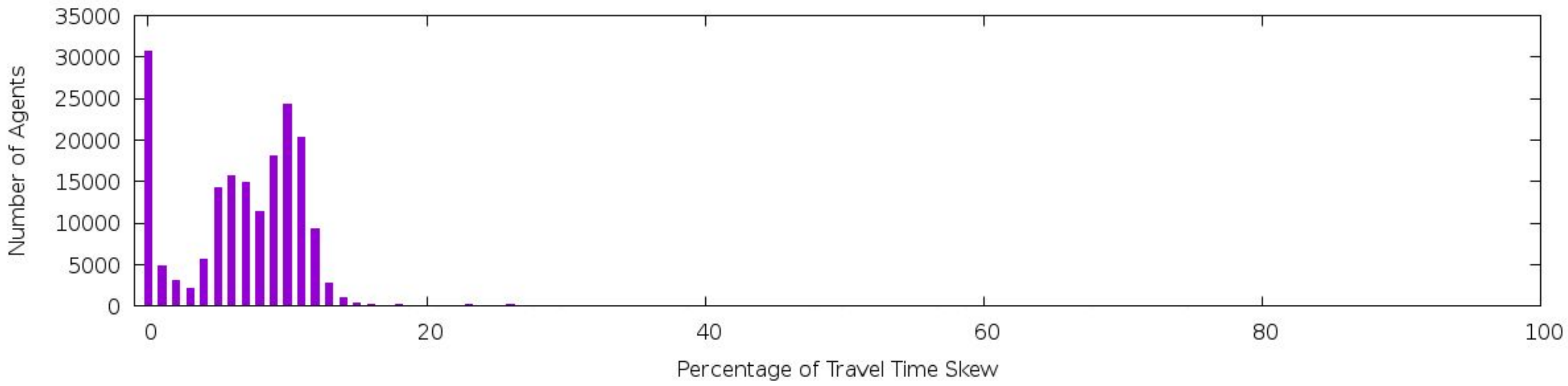
Performance Analysis (Berlin 1%)



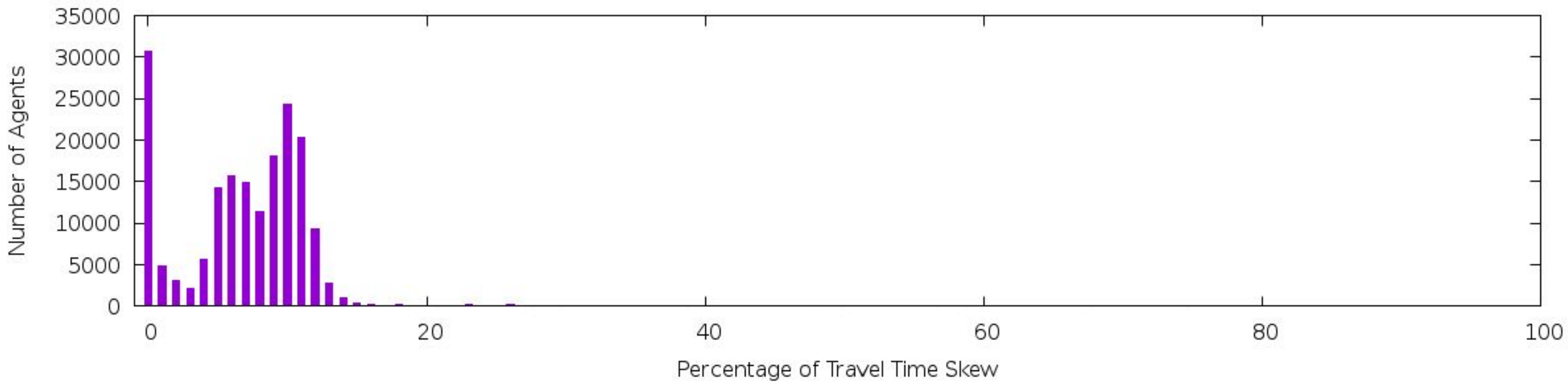
Performance Analysis (Berlin 1%)

Performance Analysis (Berlin 1%)

Performance Analysis (Berlin 1%)



Performance Analysis (Berlin 1%)



Reasons for high skew include (WIP):

- Using integers instead of doubles
- Not waiting for the time of departure
- ...

Performance Analysis

- Multithreaded implementation of Hermes is currently WIP

- Insight so far:
  - After making single-threaded version very efficient
    - synchronization becomes bottleneck quickly
  - Four ways to deal with this:
    - More work (100% scenario, more features)
    - Larger timesteps (tradeoff between error and performance, analysis required)
    - Keep improving workload-balancing, synching overhead
    - Relax synchronization for tbd. time windows (hard, can be research project)

Conclusions

- Hermes is a new simulator for MATSim
- Implementing HPC systems is hard
  - Requires constant re-evaluation and potential redesign to cope with new data/workloads
- The performance delta we see vs. QSIM motivates redesign
- Significant work to fully integrate, but not detrimental to success

Next Steps

- Improve MATSim event processing
- Keep implementing features and validating Hermes
- Propose improvements for faster event processing
- Ensemble runs
- Experiment with larger scenarios
    - HPC requires data - algorithms & data structures - hardware

Try Hermes

```
$> git clone https://github.com/muellermichel/matsim
$> cd matsim
$> git checkout hermes
$> ./Build.sh
$> ./Run.sh
```

Rodrigo Bruno
email: rodrigo.bruno@inf.ethz.ch
webpage: rodrigo-bruno.github.io