Towards High Performance Mobility Simulations

Rodrigo Bruno
Michel Mueller
Gustavo Alonso
Torsten Hoefler

Systems Group, Dept. of Computer Science, ETH Zurich

May 2019

STRC

19th Swiss Transport Research Conference
Monte Verità / Ascona, May 15 – 17, 2019

Systems Group, Dept. of Computer Science, ETH Zurich

Towards High Performance Mobility Simulations

Rodrigo Bruno, Michel Mueller, Gustavo
Alonso, Torsten Hoefler
Systems Group
Dept. of Computer Science, ETH Zurich
Universitatstrasse 6, 8092 Zurich, Switzerland
phone: +41-44-632 68 79
fax: +41-44-632 10 59
{rodrigo.bruno,michel.mueller,alonso,torsten.hoefler}@inf.ethz.ch

May 2019

Abstract

Efficient public transportation plays a key role in addressing crucial societal challenges such as urban planning, pollution, or energy consumption. To understand how transportation systems might evolve and determine the impact of new technologies and planning decisions, traffic simulations are used to explore different scenarios and help with the planning process. Existing tools, however, suffer from scalability and performance problems when it comes to simulating country-wide scenarios with millions of moving agents, scenarios that are needed to understand how complex transportation networks behave.

In collaboration with the Swiss Railway Services (SBB) and IVT (ETH), the Systems Group at ETH is developing a high performance and scalable traffic simulation platform capable of, initially, covering the entire population and transportation network of Switzerland. With current tools, such scenarios are projected to take months of computing time when running on a powerful multi-core server. By redesigning MATSim's simulation engine for the use of highly efficient data structures and algorithms, we have been improving its performance, allowing us to pursue our end goal of simulating a Swiss-scale scenario. Further performance gains are expected as we include improvements on convergence rates, the replanning stages of the simulation, and enable scalability across clusters of machines.

Keywords

Mobility Simulation; Agent-based Simulation; High Performance Computing; Large Scale

1 Introduction

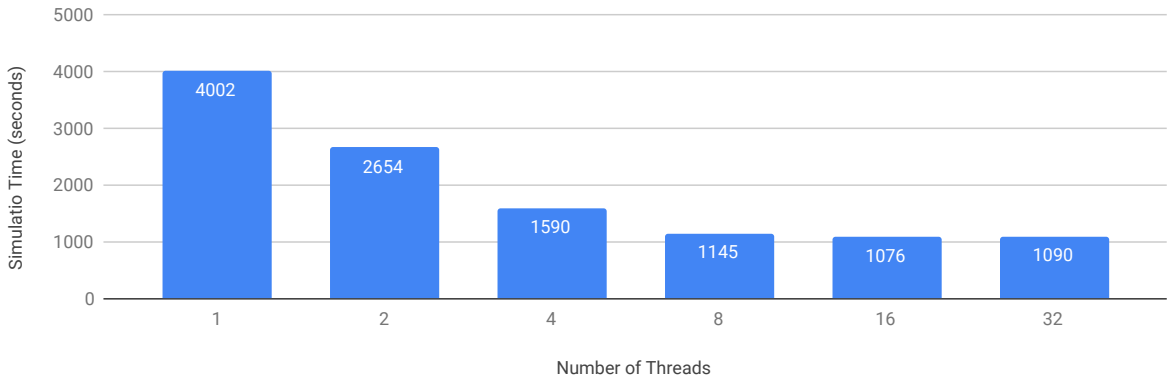
Efficient public transportation plays a key role in addressing a number of crucial challenges to society such as urban planning, pollution, or energy consumption. Achieving efficiency and keeping public transportation systems competitive is challenging, though, given the fast changes in vehicle and transportation modalities, in demographics, and the urgent need to reduce the human impact on global climate. To understand how transportation systems might evolve and determine the impact of new technologies and planning decisions, traffic simulations are often used to explore different scenarios and help with the planning process. Existing tools, however, often exhibit scalability and performance limitations when it comes to simulating country-wide scenarios with millions of moving agents, scenarios needed to understand the behavior of complex transportation networks and how they might evolve in the future. A deeper look into current simulation works is presented in Section 2.

In collaboration with SBB (Swiss Federal Railways) and IVT (Institute for Transport Planning and Systems, ETH), the ETH Systems Group is designing a high performance and scalable traffic simulation platform capable of supporting large scenarios as well as repetitive what-if analyses requiring many runs under different parameters to evaluate the impact of intended and unintended changes. The initial goal of the project is to provide complete coverage of the entire population and transportation network of Switzerland. Tools like MATSim, proposed by Horni *et al.* (2016), provide a wealth of functionality having been designed for flexibility rather than high performance. It can take up months even when running on a powerful multi-core server using the current MATSim implementation to run the scenarios we want to support. Figure 1 presents the execution time for simulating 10% Switzerland, for one iteration, with MATSim's default simulator, QSim. From these results, one can roughly estimate that simulating 100% Switzerland, for 1000 iterations (the most commonly used number of iterations at SBB), would take approximately 116 days, an unrealistically long time for any production system.

We have conducted an extensive performance analysis on QSim (the simulator included in MATSim), both looking at raw performance as well as studying the underlying modelling problem through our own prototypes. Our experiments show that the main performance bottlenecks are data structures and algorithms chosen for ease of use by the developer but not for their efficient representation and memory access speed.

To address these shortcomings, we are developing Hermes (High pERformance Multi-mode transport nEtwork Simulation), an alternative simulation engine intended to be a drop-in replacement for QSim compatible with the rest of MATSim. Hermes is built around two main

Figure 1: MATSim simulator (QSim) Scalability simulating 10% Switzerland for 1 Iteration



design decisions. First, Hermes is an event-driven simulator meaning the simulation tracks and produces events that need to occur at a specific time. In other words, Hermes does not process network links or agents individually but simulation events instead. Doing so is important to efficiently support sparse networks, where the average number of events is low, as it is to be expected in large scale scenarios. Second, Hermes is heavily optimized for the common case, i.e., the most common type of network events. Basic network events such as activities, legs, and public transportation related events are handled through a fast code path. Complex and experimental features, such as taxi events, are handled through a slower, not optimized code path. By relying on this execution path split, it is possible to support both large-scale simulations using standard/simpler network features while supporting simulations requiring more experimental features.

Preliminary performance results obtained using SBB's testing infrastructure and utilizing a MATSim scenario of Switzerland with 10% of the population (provided by SBB) show that Hermes is significantly faster than QSim. These results validate both the design decisions and performance techniques used while developing Hermes.

2 Related Work

There have been many proposed simulation engines such as the work by TRANSIM Smith *et al.* (1995), SUMO Behrisch *et al.* (2011), DRACULA Liu (2010), DynaMIT Ben-Akiva *et al.* (1998), and MATSim Horni *et al.* (2016) that, however, suffer from scalability problems and therefore, are unable to simulate large scale scenarios.

Parallelization of the simulation engine is a popular approach as it is the fastest way to scale inside a single machine as presented by Barceló *et al.* (1998) and Aydt *et al.* (2013). However, parallelization has several challenges such as selecting/designing efficient data structures that minimize the overhead of synchronization. Besides, parallel approaches are limited to a single machine, which poses a limit on the number of cores and size of the memory usable by the simulation.

To overcome the resource limitation posed by using a single node, distributing the simulation has been also proposed by Cetin *et al.* (2003), Rickert and Nagel (2001), Cameron and Duncan (1996), Klefstad *et al.* (2005), Suzumura and Kanezashi (2014), and Suzumura *et al.* (2015). While distributing the workload provides much more resources to the simulation, it also poses new challenges such as a the partitioning the network. Such partitioning is usually very difficult as agents might be able to teleport to random locations in the network, making it difficult to estimate a good partition of the network and thus potentially leading to large communication overheads.

Finally, more recently, a number of works also proposed the use of GPUs to accelerate the simulation effort as proposed by Strippgen and Nagel (2009), Shen *et al.* (2011), Xu *et al.* (2014), Vu and Tan (2017), Heywood *et al.* (2015), and Saprykin *et al.* (2019). However, these works require even more complex algorithm implementations (specially designed for GPUs) which will most likely break the MATSim goals of keeping the code extensible and maintainable by the community.

In sum, this work is proposing a realistic implementation of a MATSim simulator, one that could be integrated in MATSim without breaking the MATSim ecosystem, while providing high scalability and performance. To the best of our knowledge, this is the first work to comply with both goals, stay compatible with MATSim and still allow the execution of such large scale scenarios (such as the Switzerland scenario).

3 Hermes - a high performance mobility simulator

Hermes is a new simulator for MATSim developed with performance in mind. The goal is to scale to very large scenarios (nation-wide scenarios, for example) and in order to achieve such goal, Hermes is built around the event-driven simulation design principle, where the simulator processes events that are scheduled to occur at every simulation step. This is a radically different design choice compared to link-driven simulation (currently used in QSim, the default

MATSim simulator) which processes all the network links at every simulation step. While more challenging to implement correctly, event-driven simulation algorithms can be significantly faster when compared to link-driven simulation algorithms because the network does not have to be traversed by the simulation algorithm on every simulation step. For example, if there are no agents interacting with a specific link, this link will not be visited by the event-driven simulation but it will still be visited by a link-driven simulator. This difference has a significant impact on sparse networks (networks which contain hot clusters of very utilized links separated by mostly underutilized links). Sparse networks are specially relevant as they approximate real world large scale scenarios such as the Switzerland scenario.

Besides the type of simulation algorithm to use (either link-based or event-based), a second fundamental challenge arises when highly optimized code requires low maintainability and high extensibility. In particular, MATSim code is required to be highly extensible and accessible to a large community of developers that might not be experts on High Performance Computing (HPC, for short). To address this problem, Hermes provides a fast/optimized simulation path for the most common/standard simulation features and allows plugins, that implement more advanced/experimental features, to extend Hermes. This way, Hermes supports efficient execution of very large scenarios that use the most common simulation features, and also more research-oriented scenarios that might use more experimental simulation features.

In the next sections we analyze in deeper detail the design decisions of Hermes and how it integrates with the MATSim simulation framework.

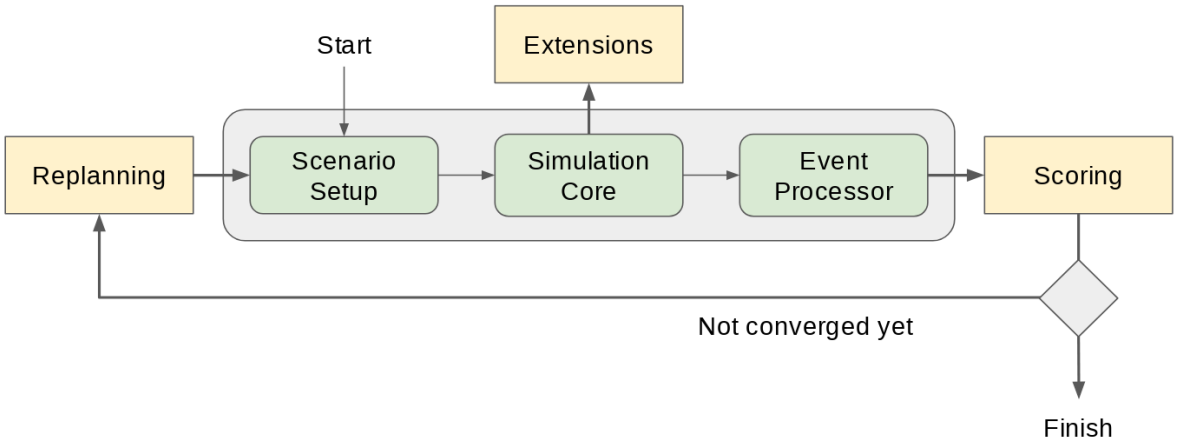
3.1 Simulation Workflow

Hermes is developed as a simulator inside the MATSim simulation framework. MATSim is a widely used tool that allows users to simulate agent-based transport scenarios. A MATSim execution has three main conceptual steps : Simulation step, Scoring step, and the Replanning step.

The first, Simulation step, is responsible for simulating the agents on the network, i.e., moving agents between links, processing interactions between agents and public transportation, etc.. The Simulation step receives as input a specific sequence of actions that must be performed by each agent on the network (called plans) and produces, as output, a number of network events. Network events describe when, how, and what happens in the network.

These network events are then processed by the Scoring step which produces a final score for

Figure 2: Hermes Simulation Workflow in MATSim



the simulation executed with the current set of agent plans. This score is then analyzed by a termination criteria which decides if the simulation should continue with one more iterations or if it should terminate. New plans for each agent are generated by the Replanning step. The goal of the Replanning step is to create plans that improve the overall score of the simulation.

Hermes is inserted in MATSim as a new drop-in replacement for the Simulation step. Figure 2 presents the simulation workflow of Hermes integrated in MATSim. The green boxes represent code developed in Hermes while the yellow boxes represent MATSim's code. Note that the Extensions component is provided as a way to install plugins that implement advanced simulation features, which are not implemented and therefore not optimized in Hermes. For each simulation iteration, Hermes executes three main steps: Scenario Setup, Simulation Core, and Event Processing. Each of these are discussed in the following sections.

3.2 Scenario Setup

The Scenario Setup is the first step of a Hermes simulation iteration and it consists on preparing the data structures that are used by the Simulation Core. These data structures include: i) the links that form the network that is being used for the simulation, ii) the agents that participate in the simulation, and iii) other helper data structures that are used to keep state during the simulation.

Since the data stored by these data structures is heavily accessed during the simulation, it is important to select and design these data structures carefully to allow the simulation code to

be as efficient as possible. In practice, two design goals are present when designing these data structures.

First, data should be compressed as much as possible. This will not only save memory but it will also improve caching as more data will be cached. One simple practical example is using integers for identifiers instead of Strings (as MATSim does). An integer will be stored in a 32 bit word while a String might take an arbitrary amount of space (depending on the number of characters).

The second design goal is to avoid multiple hops in memory when loading data. For example, when loading a value stored in a hash table, several memory hops will be necessary to load the data value. However, if the data is stored in a flat memory array, only one access is necessary.

3.3 Simulation Core

After preparing all the necessary data structures, Hermes' simulation code is invoked to perform the actual simulation. As already discussed, the simulation executes an event-driven algorithm that generates events that are used in the next phase (Scoring) of the MATSim simulation workflow.

In the remainder of this section, the core simulation algorithm is presented and a number of optimization techniques that improve the performance of the simulation are discussed.

3.3.1 Simulation Algorithm

A simplified, yet representative, version of Hermes simulation algorithm is presented in Algorithm 1. The algorithm begins with the main procedure, *Simulate*, that executes all the steps of the current simulation iteration. Then, for each step, each agent that should be processed in the current step is handled by calling the *Process_Agent* procedure (line 4). After processing the agents that need to be handled in the current simulation step, all links that need to be handled in the current simulation step are also processed (line 6).

The *Process_Agent* procedure will handle a single agent and it will try to route this agent in the network according to the specified action in the agent's plan. For example, a *leg* plan entry means that the agent needs to enter a specific link. A special case of the *Process_Agent* procedure occurs when Hermes does not implement a handler for the type of action that the agent needs to

Algorithm 1 Hermes Simulation Algorithm

```
1:  procedure SIMULATE
2:      for step in iteration do
3:          for agent in delayed_agents(step) do
4:              Process_Agent(agent)
5:          for link in delayed_links(step) do
6:              Process_Link(link)
7:  procedure PROCESS_AGENT(agent)
8:      switch agent.plan.top do
9:          case leg
10:             // handle agent leg, push to link
11:         case activity
12:             // handle agent activity, add to delayed_agents
13:         case pt
14:             // handle public transport interaction
15:         case default
16:             // call extension code to process plan entry
17: procedure PROCESS_LINK(link)
18:     while can_process_agent(link.top) do
19:         Process_Agent(link.top)
20:         link.pop()
21:     delayed_links[link.top.finish].append(link)
```

execute. In this case (link 15), Hermes resorts to external code (registered extensions) that will be called to handle this agent.

Finally, the Process_Link procedure will process an individual link by going through the agents that are ready to leave the link and trying to route them. The loop stops when one agent cannot be routed to its next hop in the network. After processing all the agents that can be routed, the link is re-inserted in the data structure that holds links until they need to be processed (line 21).

Since this is an event-driven algorithm, there is no full iteration over the network links or agents that participate in the simulation. As demonstrated in lines 3 and 5, only the agents and links that need to be handled in a particular simulation step are accessed and processed.

3.3.2 Core Algorithm Optimizations

To achieve high performance simulation, Hermes employs a number of optimizations that come from understanding how the developed algorithm will be compiled and mapped the to underlying hardware. The main optimizations are depicted below:

Avoid objectification - While Object Oriented languages such as Java heavily rely on objects to couple logic and state, large scale use of such constructs (objects) leads to performance problems/inefficiencies. First, every object requires some memory to accommodate metadata, which becomes non-negligible when many objects are used. Second, every time an object is used, its memory location is loaded, leading to severe performance penalties when many objects are being used. To reduce memory utilization and memory loads, Hermes uses primitive types whenever possible. One simple example is to keep identifiers as ints instead of Strings;

Avoid non-consecutive-memory data structures - Coupled with the objectification problem comes the use of non-consecutive-memory data structures. Such data structures include Lists and Maps which do not place the elements of the data structure consecutively in memory. As a result, accesses to a single element of the data structure might require multiple accesses to memory. To avoid such data structures, Hermes uses arrays of primitive types whenever possible;

Avoid polymorphism - Another performance consequence of the object oriented programming model are polymorphic method calls. A polymorphic method call occurs when a method can have multiple implementations, for example, through the use of class hierarchy or interfacing. When handling a polymorphic call, the compiler does not know which method will be used until the call is performed. In Hermes, we try to remove all the polymorphic calls from the simulation code to allow important compiler-driven optimizations (such as method inlining);

Compact hot code paths - A hot code path is a set of instructions or methods that are executed very frequently. Removing all logic that could live outside the hot code path leaves only essential instructions that could not be pre-computed or post-computed. This improves the ability to further optimize the code. In Hermes, this principle is applied in several places, where the hot code path can be made more compact and efficient. One example is the pre-computation of the data structures that hold the events that need to be fired during the simulation (described next).

3.4 Event Processing

The final step of the Hermes simulation workflow in MATSim is the event processing. An event is a piece of information that can be used to describe **when**, **how**, and **what** happened in the network. These events are important to understand how the agents move in the network and they interact with public transports, for example.

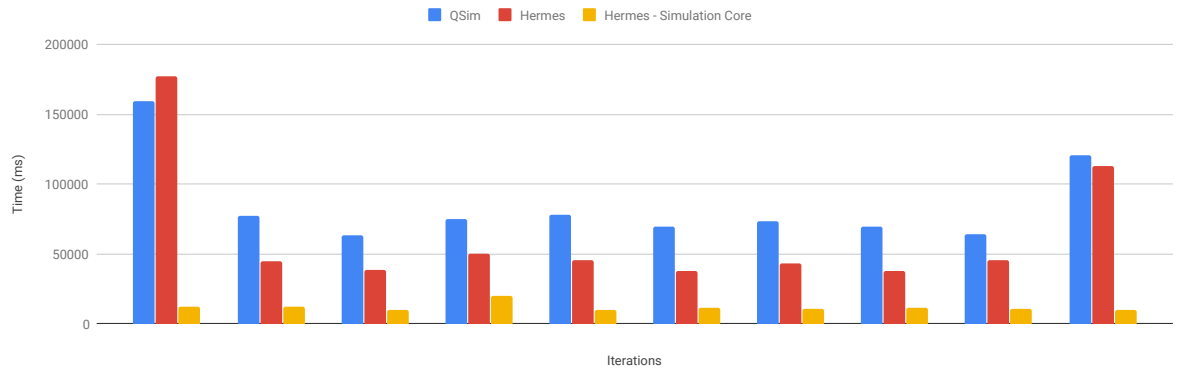
One important observation regarding events is that most of the information that is contained in an event (namely the **how** and **what**) can be predicted from the plans. In other words, the plan of an agent tells **what** the agent will do and **how**. The only missing information is **when** each action will be triggered. Hermes takes advantage of this observation and pre-computes all events during scenario setup. This allows Hermes to mostly avoid event setup during the simulation. During a simulation, the only task is to tag the times at which each event is triggered. At the end of the simulation, only a few operations need to be executed to ensure that events produced by Hermes are compatible with MATSim events.

4 Evaluation Results

Hermes is a new, drop-in, simulator for MATSim and, in order to measure the performance benefits of using Hermes we now present some evaluation results. The goal of this experimental evaluation is twofold. First, analyze the performance impact of Hermes and measure the time it takes to finish the simulation and compare it to QSim, the default simulator in MATSim. Second, measure quality of the simulation results produced by Hermes, i.e., assuming QSim as the point of reference.

All the performance results were obtained in a single node equipped with an Intel Xeon E5-4650 v2 @ 2.40GHz (10 cores per socket, 4 sockets) and with 512 GBs of DDR4 @ 1866 MHz. Several runs were performed to confirm that the presented results are reproducible and are representative of the performance of the system. All runs were executed in isolation. A scenario of Berlin with 1% of the total agents was used for this performance evaluation.

Figure 3: Hermes Simulation Workflow in MATSim



4.1 Simulation Performance

We start the study of Hermes' performance by comparing MATSim simulation workflow time with both simulators: QSim and Hermes. This is an end-to-end metric that shows the global performance of the simulator and abstracts some internal simulation steps, which are discussed in the next section.

Figure 3 presents the simulation time (in milliseconds) for QSim (blue bars), Hermes (red bars), and Hermes Simulation Core (yellow bars). For this experiment, we configured MATSim to run for only 10 iterations. The simulation time for the first and last iterations is higher than the other iterations because MATSim records the state of the simulation to disk. Since a typical simulation incurs into a high number of iterations (1000 iterations, for example), the performance of these two iterations will not be analyzed in this work.

Looking at the execution time of iterations 1 to 9, it is possible see that Hermes (red bar) reduces the simulation type by 70% on average when compared to QSim (blue bar). It is also interesting to note that Hermes Simulation Core (yellow bars) is significantly faster than QSim and Hermes (7x faster than QSim on average). This is an important result as it indicates that a significant portion of time is spent not executing code that is actually moving agents and simulating our scenario but instead, executing internal MATSim code (discussed next).

Figure 4: Hermes Simulation Workflow in MATSim

4.2 Hermes Simulation Breakdown

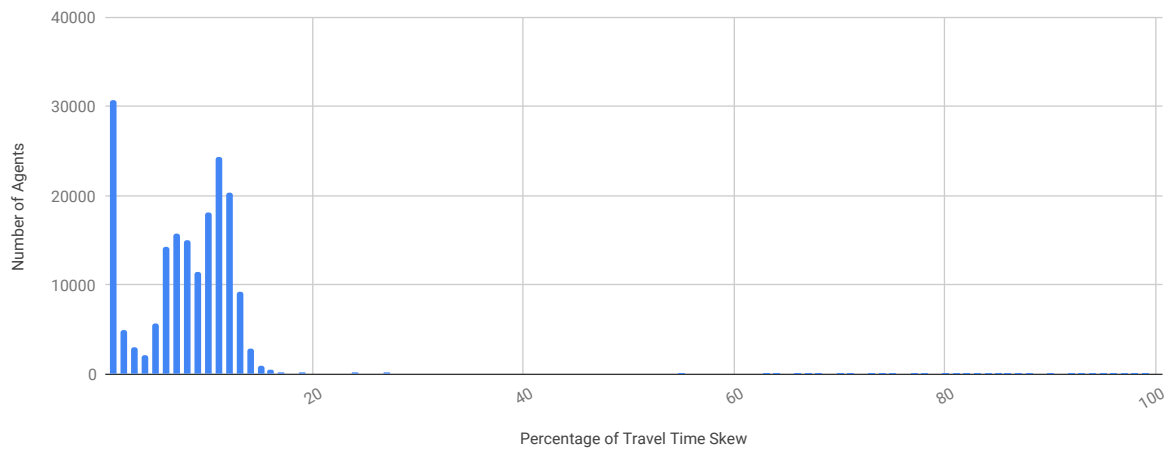
In order to have a better understanding on where time goes inside Hermes' simulation, we conducted further analysis on the iteration workflow in MATSim. The results are presented in Figure 4, which contains performance results for each of the Hermes simulation workflow steps: Scenario Loading (blue bars), Simulation Core (red bars), and Event Processing (yellow bars), as presented in Section 3.1). In addition, we also present results for MATSim event processing, a step that is executed after Hermes Event Processing and before Scoring.

Looking at the Scenario Loading times (blue bars), it is possible to see that for the first iteration, there is a high amount of time required to build all the data structures necessary for the simulation. However, in subsequent iterations, only minor updates to agent plans are necessary and therefore, the time of this step is negligible after the first iteration.

The Simulation step (red bars) represents the time used by the simulator to loop through all the events and moves agents in the network. This is where most of the complexity of the simulation lies. The Event Processing step requires a very low amount of time to execute as very little work is performed at this stage of the simulation (as discussed in Section 3.4).

Finally, the MATSim Event Processing step (green bars) is where most of the time goes. This is step is responsible for forwarding all events to all event listeners registered in MATSim. Although this code is not part of Hermes, is has a significant impact on the overall simulation performance at it has to be executed in between every iteration. Therefore, and since this is the component that is taking the most amount of time (even more than the Simulation Core), this is currently our target for improving the performance of MATSim.

Figure 5: Hermes Simulation Workflow in MATSim



4.3 Hermes Simulation Travel Time Skew

We finish our evaluation section by providing results of the quality of Hermes' simulation. In this experiment, we consider that QSim is our target and we compared the percentage of travel time skew for agents simulated in Hermes. For example, an agent with a travel time skew of 10% means that this agent, when simulated with Hermes, takes either 10% more time or 10% less time to complete its trip.

A histogram of the travel time skew is presented in Figure 5. From these results, one can conclude that Hermes is providing less than 10% skew for most agents. This seems to be a reasonable penalty given the performance improvements that is currently provided by Hermes. Nevertheless, we are working on reducing the travel time skew of Hermes compared to QSim. One main factor for increasing this time skew is the fact that Hermes is using integers for all its internal calculations (length, time to destination, etc) instead of double precision values as used by QSim.

5 Conclusions and Next Steps

In this work, we presented an work-in-progress effort to make large scale simulations possible in MATSim by providing a new simulator implementation, Hermes. Hermes is an event-driven simulator that provides a fast execution path for most common simulation features but still

supports extensions for research-oriented features.

The experimental evaluation shows that Hermes is able to reduce the simulation time when compared to the current default simulator, QSim. It is also demonstrated that the current bottleneck is the MATSim Event Processing step, that takes most of the simulation time for each iteration.

The next steps in the development of this project include the improvement of MATSim Event Processing and the study of the applicability of ensemble runs in MATSim (i.e, running multiple simulations in parallel). By running multiple simulations in parallel, it would be possible to explore multiple replanning options at the same time, thus reducing the number of iterations necessary until the simulation finishes. Hermes is available at github.com/muellermichel/matsim.

6 References

Aydt, H., Y. Xu, M. Lees and A. Knoll (2013) A multi-threaded execution model for the agent-based semsim traffic simulation, paper presented at the *AsiaSim 2013*, 1–12, Berlin, Heidelberg, ISBN 978-3-642-45037-2.

Barceló, J., J. L. Ferrer, D. García, M. Florian and E. L. Saux (1998) *Parallelization of Microscopic Traffic Simulation for Att Systems Analysis*, 1–26, Springer US, Boston, MA, ISBN 978-1-4615-5757-9.

Behrisch, M., L. Bieker, J. Erdmann and D. Krajzewicz (2011) Sumo – simulation of urban mobility: An overview, paper presented at the *SIMUL 2011*, October 2011.

Ben-Akiva, M., M. Bierlaire, H. Koutsopoulos and R. Mishalani (1998) Dynamit: a simulation-based system for traffic prediction, paper presented at the *DACCORD Short Term Forecasting Workshop*, 1–12.

Cameron, G. D. B. and G. I. D. Duncan (1996) PARAMICS-parallel microscopic simulation of road traffic, *Journal of Supercomputing*, **10** (1) 25–53.

Cetin, N., A. Burri and K. Nagel (2003) A large-scale multi-agent traffic microsimulation based on queue model, paper presented at the *3rd Swiss Transport Research Conference*, Ascona, March 2003.

Heywood, P., P. Richmond and S. Maddock (2015) Road network simulation using flame gpu, paper presented at the *Euro-Par 2015: Parallel Processing Workshops*, 430–441, Cham, ISBN 978-3-319-27308-2.

Horni, A., K. Nagel and K. W. Axhausen (2016) *The multi-agent transport simulation MATSim*, Ubiquity Press London.

Klefstad, R., , , R. Jayakrishnan and R. Lavanya (2005) A distributed, scalable, and synchronized framework for large-scale microscopic traffic simulation, paper presented at the *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, 813–818, Sep. 2005, ISSN 2153-0009.

Liu, R. (2010) *Traffic Simulation with DRACULA*, 295–322, Springer New York, New York, NY, ISBN 978-1-4419-6142-6.

Rickert, M. and K. Nagel (2001) Dynamic traffic assignment on parallel computers in TRANSIMS, *Future Generation Computer Systems*, **17** (5) 637–648.

Saprykin, A., N. Chokani and R. S. Abhari (2019) Gemsim: A gpu-accelerated multi-modal mobility simulator for large-scale scenarios, *Simulation Modelling Practice and Theory*, **94**, 199 – 214, ISSN 1569-190X.

Shen, Z., K. Wang and F. Zhu (2011) Agent-based traffic simulation and traffic signal timing optimization with gpu, paper presented at the *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 145–150, Oct 2011, ISSN 2153-0017.

Smith, L., R. Beckman and K. Baggerly (1995) Transims: Transportation analysis and simulation system, 7 1995.

Strippgen, D. and K. Nagel (2009) Multi-agent traffic simulation with cuda, paper presented at the *2009 International Conference on High Performance Computing Simulation*, 106–114, June 2009.

Suzumura, T. and H. Kanezashi (2014) Multi-modal traffic simulation platform on parallel and distributed systems, paper presented at the *Proceedings of the Winter Simulation Conference 2014*, 769–780, Dec 2014, ISSN 0891-7736.

Suzumura, T., G. McArdle and H. Kanezashi (2015) A high performance multi-modal traffic simulation platform and its case study with the dublin city, paper presented at the *Proceedings of the 2015 Winter Simulation Conference*, WSC '15, 767–778, Piscataway, NJ, USA, ISBN 978-1-4673-9741-4.

Vu, V. A. and G. Tan (2017) High-performance mesoscopic traffic simulation with gpu for large scale networks, paper presented at the *Proceedings of the 21st International Symposium on Distributed Simulation and Real Time Applications*, DS-RT '17, 127–135, Piscataway, NJ, USA, ISBN 978-1-5386-4028-9.

Xu, Y., X. Song, Z. Weng and G. Tan (2014) An entry time-based supply framework (etsf) for mesoscopic traffic simulations, *Simulation Modelling Practice and Theory*, **47**, 182 – 195, ISSN 1569-190X.