Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2021) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 9th International Workshop on Agent-based Mobility, Traffic and Transportation Models, (ABMTRANS) March 23 - 26, 2021, Warsaw, Poland

Hermes: Enabling efficient large-scale simulation in MATSim

Dan Graur*,[a], Rodrigo Bruno[a,b], Joschka Bischoff[c], Marcel Rieser[d], Wolfgang Scherr[c], Torsten Hoefler[a], Gustavo Alonso[a]

[a] *ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland*
[b] *Oracle Labs, Hardstrasse 201, 8005 Zürich, Switzerland*
[c] *Swiss Federal Railways SBB, Hilfikerstrasse 1, 3000 Bern 65, Switzerland*
[d] *Simunto, Riedgrabenweg 49, 8050 Zürich, Switzerland*

Abstract

Large scale simulations of transportation networks can yield highly valuable insights into the design decisions needed to deliver the best possible transportation service. Current technologies, however, are generally unable to provide support for large scale simulations as they often require impractical amounts of time to compute results. To address this issue, we have developed Hermes, a novel simulation core for MATSim designed as an alternative to QSim. Using an event-driven approach, as well as numerous optimizations to the data structures and the simulation logic, Hermes is capable of completing simulations of larger scale scenarios within feasible time frames. We demonstrate Hermes' capabilities through extensive experiments as well as through production level results obtained from the Swiss Federal Railways (SBB), where Hermes is currently being used in production.

Keywords: MATSim; Event-Driven Simulation; Scalability; Efficiency; Transportation Simulation Software

1. Introduction

Agent-based traffic simulation is used to forecast how transportation systems evolve in the future and how infrastructure and service improvements will impact travel. However, large scale simulation models struggle with long computation time. One example is SBB's model SIMBA MOBi [11], which simulates all residents and visitors of Switzerland over 24 hours. MATSim [7] is an agent-based traffic simulation framework written in Java, and while it enjoys wide-spread utilization, it suffers from scalability issues. Using MATSim's default simulator, QSim, SBB's model would take several weeks of simulation if 100% of the population was to be simulated until acceptable results

* Corresponding author.
E-mail address: dan.graur@inf.ethz.ch

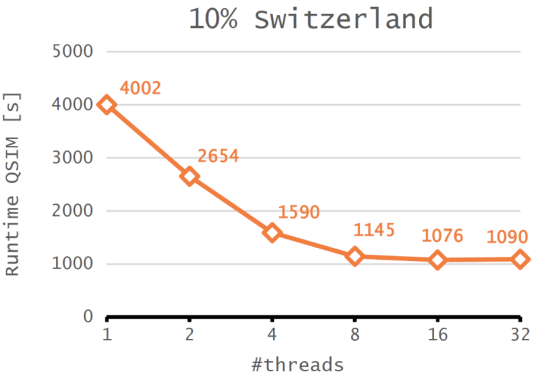


Fig. 1: Per iteration runtime across varying number of threads for a 10% population simulation of the on the 2018 model of Switzerland.

are achieved. Figure 1 shows the seconds per iteration across a varying numbers of threads in a simulation of 10% of Switzerland's population. Past 8 threads, the runtime plateaus, hence no benefit can be obtained from more threads. Plateauing occurs at one point regardless of the machine's hardware, and is due to the design choices and algorithms of the current simulation core. In order to enable better scalability, these issues need to be tackled directly.

After analyzing MATSim, we found that the data structures and algorithms used in QSim, are a deterrent to scalability. QSim is a simulation engine tasked with traffic flow simulation across the network. Based on this analysis, we developed a new, alternative simulation engine for MATSim called Hermes. Hermes produces results equivalent to QSim but is far more efficient in terms of simulation speed. In this paper we briefly describe the design of Hermes and present experimental results that demonstrate its advantages. Hermes is open-source and is released as part of MATSim. Today, it is used in production at SBB.

2. Related Work

A straightforward way to speed up simulations is to use parallelism [1, 2]. Nevertheless, one needs to carefully make use of the available synchronization mechanisms and data structures, as they can potentially lead to performance loss or even deadlocks. It should be noted that MATSim already supports parallelism, but as can be seen from Figure 1, there is little return on investment past a certain amount of threads.

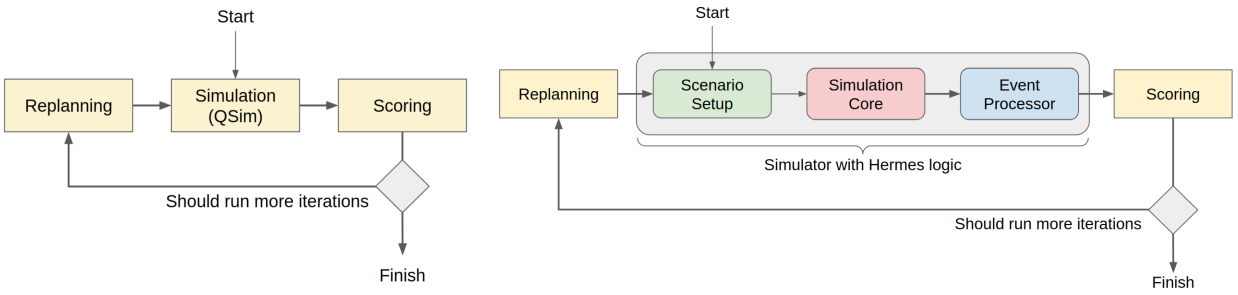
Distribution is another possible approach [3, 4, 8, 9, 14, 15], but given that agents may teleport to an arbitrary location in the network means that efficiently partitioning the simulation becomes a challenging task. We are planning to visit this avenue in the future, and have already began exploring ideas such as Controller-Worker architectures, where the Controller is in charge of replanning and aggregating simulation scores, while the Workers execute the simulation and produce individual simulation scores. We have also started exploring efficient means of object serialization and communication in Java.

Finally, making use of accelerators, such as GPUs, is a viable avenue for speedups [6, 10, 12, 13, 17, 19]. In this respect, there has been some effort to also introduce the use of GPUs in MATSim. However, employing accelerators can make it more difficult for community-driven projects, such as MATSim, to find support, as it requires both specialized hardware and expertise in programming the accelerators efficiently.

JDEQSiM [18], an event-driven alternative to QSim, provides some speedups, but comes with major drawbacks, such as the lack of public transport simulation. Other attempts on parallelizing MATSim and distributing replanning and simulation stages have been conducted, but are not currently usable or well-documented[1].

As results will show, with Hermes as the simulation core, the Replanning Step in MATSim dominates the runtime. Hence, benefits can be obtained by improving this step. There has already been research in the direction of replanning methods which strive to increase the convergence rate [5, 16].

[1] Parallel MATSim URL: https://github.com/matsim-org/parallelMATSim



(a) MATSim's default architecture.

(b) MATSim's architecture with Hermes.

Fig. 2: MATSim's high-level architecture with and without Hermes.

3. MATSim Architecture

MATSim features three main steps in its simulation architecture: Simulation, Scoring and Replanning. Figure 2a presents a diagram of the high level architecture. We further describe these steps in more detail.

Simulation: Carries out an iteration in the simulation (e.g. moves agents on links, manages interactions between agents, etc.). This step receives a sequence of actions called agent plans, which need to be performed by each agent. It then outputs a set of network events, which are further processed by the Scoring Step.

Scoring: Processes the network events, and generates a simulation score. This step also checks if termination conditions apply, and if so, ends the simulation. Otherwise, the simulation moves on to the next step.

Replanning: Generates new plans for each agent, with the aim of improving the simulation score.

By default, QSim is tasked with carrying out the Simulation Step. To do so, at each time step, it processes every link on which there is at least one agent, regardless of whether the agents on the link are ready for their next interaction. This approach is one of the main reasons behind its lack of scalability, especially considering that large scale simulations feature millions of agents and links.

4. Hermes

4.1. Overview

Compared to other approaches, such as QSim which strives for flexibility, Hermes optimizes the most common operations in transport simulation by carefully designing efficient data structures and algorithms. It also allows users to easily extend the simulation engine with new experimental features. Figure 2b shows the new MATSim architecture with Hermes.

Hermes is built around two critical observations. The fist observation is that, in large-scale realistic simulations, events are concentrated around a small number of areas in the network. For example, cities with high population density will generate many more simulation events than other areas of the simulation graph. Hence, traversing the entire network is wasteful. Hermes capitalizes on this point by exploiting an event-driven execution, where only the events taking place at a specific time step are processed. The second observation is that most of the information contained in the simulation events (the output of the simulation) can be extrapolated by looking at the agents' plans (input of the simulation). Hermes takes advantage of this fact by partially pre-generating the simulation events' data structures during the agents' plans loading phase. This speeds up the simulation by removing the overhead of creating and registering new events during simulation time. What cannot be pre-computed is inferred during simulation time.

Hermes also introduces additional improvements, such as: optimized code paths for common events, optimized data structures, pre-computation, memoization, as well as efficient code which makes extensive use of primitive types. These features, together with the event-driven approach, enable Hermes to deliver efficient, large-scale simulations, with a significantly smaller memory footprint than QSim. It is worth noting that Hermes is single-threaded. We have considered a multi-threaded version of Hermes, however we have decided against it since Hermes is bottlenecked by

memory rather than CPU. Due to this, the addition of threads would yield marginal performance improvements at the cost of a much greater code complexity.

It should be noted that Hermes does not compromise on extensibility, and allows users to plug in experimental events into the simulation. To the best of our knowledge, this is the only simulator algorithm in the MATSim ecosystem which is both scalable and abides to the ecosystem's principles of extensibility and community-driven development.

Hermes does not provide all the functionality that QSim has to offer, but focuses on the core, *standard* functionality. Different vehicle types and both private and public transport can be simulated in a similar way as with QSim. Special extensions such as dynamic vehicle routing, within-day replanning or traffic signals are however not supported. There are slight differences in the way Hermes processes events relative to QSim (e.g. processing stops for public transportation vehicles). Ultimately, this produces slight differences between the simulation outcomes across the two cores. Nevertheless, using both Hermes and QSim together in the same simulation is possible, by alternating which simulator executes which iteration.

4.2. Architecture and Workflow

Hermes' workflow consists of three main steps: the Scenario Setup, Simulation Core, and Event Processor, as shown in Figure 2b. We further describe what each stage does.

Scenario Setup: Prepares the data structures used by the Simulation Core. These structures refer to the links that form the network, the agents that participate in the simulation, as well as other helpful data structures. Considering that an event defines the *what*, *when* and *how* something happens in the network, and that we can readily extract the *what* and *how* from an agent plan, it follows that we can pre-compute all the simulation events by initially omitting the *when*. This latter piece of information is added later, at simulation time.

It is important that the data structures generated during this step are as compressed and efficient as possible, as they are frequently accessed during simulation. Moreover, by compressing the data, we save memory which in turn also enables the cache to store more information. We additionally avoid nesting data structures, in order to decrease the number of expensive memory hops required on access.

Simulation Core: Performs the simulation itself using an event-driven approach. This stage employs a number of other optimizations which are particularly useful in simulations involving millions of entities. These optimizations are further detailed below:

- *Avoid Objectification*: Objects carry metadata with them, and require that their memory location be loaded when they are accessed. Operations that manipulate such entities will thus incur larger overheads in terms of time and memory. The solution to this is to make use of primitive data types wherever possible, as they are more efficient.
- *Avoid Non-Contiguous Data Structures*: Collections such as Lists or Maps may be non-contiguous. This means that an element may be fragmented across multiple memory locations, hence requiring multiple memory accesses. The solution to this issue is to make use of arrays or specialized data structures whenever possible.
- *Avoid Polymorphism*: Since polymorphic methods have multiple implementations, the decision of which version of the method to call can only be done at run-time. This disallows the compiler from making compile-time optimizations, such as method inlining. Moreover, such run-time decisions can generally be very costly. The solution is to avoid the use of polymorphism as much as possible.
- *Compact Hot-Code Paths*: These are fragments of code which are executed frequently. We make such pieces of code as efficient as possible, leaving in only the relevant instructions which cannot be pre- or post-computed. Indirectly, this also allows the compiler to make potential optimizations to these fragments.

Event Processor: This stage makes heavy use of the event structures pre-computed during the Scenario Setup stage. This is consequently one of the great benefits of the aforementioned pre-computation, as it significantly reduces the load of the Event Processor whose main task is to processes events generated by the Simulation Core, and convert them to MATSim compatible structures.

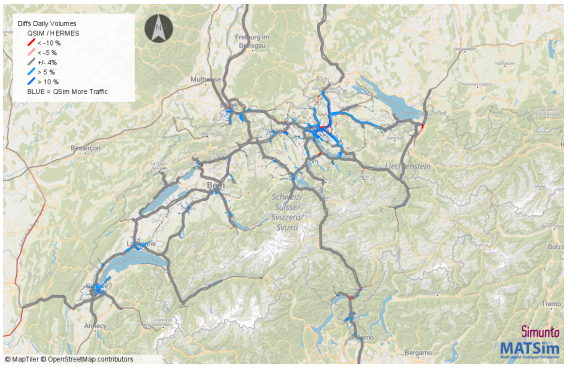
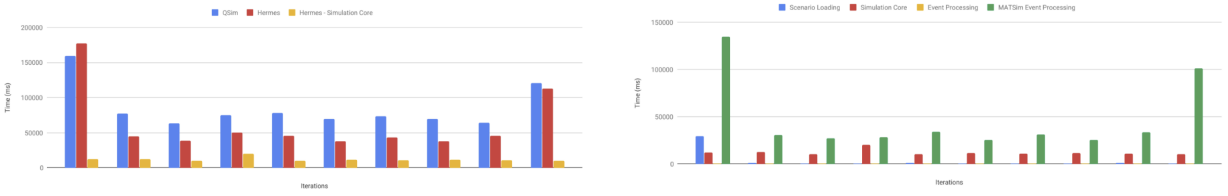


Fig. 3: Daily Traffic Flow difference between QSim and Hermes runs



(a) Hermes vs QSim.

(b) Time spent in Hermes stages.

Fig. 4: Time analyses across the 10 iterations of the Berlin scenario.

5. Results

5.1. Production Results

Hermes has already been adopted by SBB where it is being used in SIMBA MOBi, a simulation model covering Switzerland. In a benchmark simulation run, results and computation times of both a QSim and a Hermes run are compared under a similar setup using Amazon Web Services (instance type *r5.8xlarge*; 32 vCPUs, 256 GB RAM). The overall run time for 301 iterations is 83 hours using QSim. With Hermes, the simulation takes 47 hours to complete, or **roughly 43% less time**. In direct comparison, one iteration using QSim takes 12:09 minutes on average, whereas with Hermes it only takes 4:52 minutes to complete. In terms of simulation results, both mobsims achieve comparable, but not similar results. In Table 1, the modal split differences of both simulations are shown. In tendency, both show very similar results, however the share of car use is slightly higher when using QSim. In figure 3, the daily volume differences in the motorway network are shown. While in most areas the flow difference is small, the links are especially more frequently used around bigger cities in the QSim run. An explanation for these differences might lie in the way how both mobility simulations cope with the downsampling of flow and storage capacities for 10 % scenarios. For example, in QSim link lengths are sometimes virtually increased. Overall, the switch from QSim to Hermes thus requires a minor recalibration of a scenario.

5.2. Benchmark Results

We perform a simulation of the Open Berlin[2] scenario with 1% population over 10 iterations. The experiments run on a machine equipped with an Intel Xeon E5-4650 v2 (10 cores per socket, 4 sockets) at 2.40GHz and with 512GB of DDR4 at 1866 MHz.

[2] Open Berlin scenario source code: https://github.com/matsim-scenarios/matsim-berlin

Table 1: Transportation modes share in the 10% Switzerland simulation.

Mobility Sim	Taxi	Bike	Car	Public transportation	Ride	Walk
Hermes	4.09%	5.56%	44.69%	16.43%	4.30%	24.92%
QSim	4.13%	5.47%	45.15%	16.18%	4.13%	24.93%

Figure 4a shows the times spent by Hermes and QSim respectively in the 10 iterations. The first and last iteration require more time since MATSim writes the state to disk at the start and end iterations. Moreover, in the context of more realistic simulations, one uses hundreds of iterations, rendering these two as outliers. Hence, they are discarded from our analysis. The results indicate that Hermes (red bars) cuts the simulation time by 70%, when compared to QSim (blue bars). Moreover, the actual Simulation Core of Hermes (yellow bars), takes up a small fraction of the actual runtime, indicating that auxiliary MATSim related tasks in Hermes take up most of the time.

To better understand what operations dominate the runtime in Hermes, we further profile the simulator. Figure 4b describes the runtimes of each stage in Hermes for each of the iterations. The Event Processing stage (yellow bars) takes up a negligible amount of time. Similarly, the Scenario Setup (blue bars) is inexpensive, and only leaves a footprint in the first iteration when the data structures are prepared. Among the Hermes stages, the Simulation Core (red bars) take up the most time, however they are fairly cheap as well. The MATSim Event Processing stage (green bars) takes up the most time. This stage is not part of Hermes, however, it is necessary in order to carry out a simulation step, as it helps forward all events to the event listeners registered in MATSim.

We also study the travel time skew. We say that an agent has a 1% skew if its travel time in Hermes is 1% greater or lower than its time in QSim. Results reveal that most skews are within the 10^{th} percentile, hence simulation outcomes are very similar. As skews only represent differences between simulation cores, this metric should not be misinterpreted as a skew between real-world ground truths and simulation results, and thus a loss in simulation quality.

6. Conclusions

We have presented Hermes, a novel simulator for MATSim, which delivers high-performance and scalability using an event-driven design and numerous optimizations to the code and the data structures. We have shown Hermes' improved efficiency over QSim through production-level results, and benchmarks. Moreover, Hermes is currently being successfully used in production by the SBB to simulate transportation networks within Switzerland. The simulator is currently integrated in MATSim, and is available via the latest weekly releases of MATSim (https://matsim.org/downloads/).

Hermes helps improve the scalability of MATSim by significantly reducing the overhead of simulation, however, as a result of this, the Replanning Step now dominates the runtime. Further research should focus on finding means of reducing this overhead, as it will have the most impact on MATSim's runtime.

Acknowledgements

We thank the *SBB-ETH Mobility Initiative* for sponsoring this research. We would also like to thank Dr. Michel Müller and Prof. Dr. Kai Nagel for their many contributions and ideas on this project.

References

[1] Aydt, H., Xu, Y., Lees, M., Knoll, A., 2013. A multi-threaded execution model for the agent-based semsim traffic simulation, in: Asian Simulation Conference, Springer. pp. 1–12.

[2] Barcelo, J., Ferrer, J.L., García, D., Florian, M., Le Saux, E., 1998. Parallelization of microscopic traffic simulation for att systems analysis, in: Equilibrium and advanced transportation modelling. Springer, pp. 1–26.

[3] Cameron, G.D., Duncan, G.I., 1996. Paramics—parallel microscopic simulation of road traffic. The Journal of Supercomputing 10, 25–53.

[4] Cetin, N., Burri, A., Nagel, K., 2003. A large-scale agent-based traffic microsimulation based on queue model, in: In proceedings of swiss transport research conference (strc), monte verita, ch, Citeseer.

[5] Fourie, P.J., Illenberger, J., Nagel, K., 2013. Increased convergence rates in multiagent transport simulations with pseudosimulation. Transportation research record 2343, 68–76.
[6] Heywood, P., Richmond, P., Maddock, S., 2015. Road network simulation using flame gpu, in: European Conference on Parallel Processing, Springer. pp. 430–441.
[7] Horni, A., Nagel, K., Axhausen, K. (Eds.), 2016. Multi-Agent Transport Simulation MATSim. Ubiquity Press, London. doi:10.5334/baw.
[8] Klefstad, R., Zhang, Y., Lai, M., Jayakrishnan, R., Lavanya, R., 2005. A distributed, scalable, and synchronized framework for large-scale microscopic traffic simulation, in: Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005., IEEE. pp. 813–818.
[9] Rickert, M., Nagel, K., 2001. Dynamic traffic assignment on parallel computers in transims. Future generation computer systems 17, 637–648.
[10] Saprykin, A., Chokani, N., Abhari, R.S., 2019. Gemsim: A gpu-accelerated multi-modal mobility simulator for large-scale scenarios. Simulation Modelling Practice and Theory 94, 199–214.
[11] Scherr, W., Manser, P., Bützberger, P., 2020. Simba mobi: Microscopic mobility simulation for corporate planning. Transportation Research Procedia 49, 30 – 43. URL: http://www.sciencedirect.com/science/article/pii/S2352146520307249, doi:https://doi.org/10.1016/j.trpro.2020.09.004. facing the complexity of transport models and innovative developments in sustainable mobility - Selected Proceedings of the 47th European Transport Conference, ETC 2019.
[12] Shen, Z., Wang, K., Zhu, F., 2011. Agent-based traffic simulation and traffic signal timing optimization with gpu, in: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 145–150.
[13] Strippgen, D., Nagel, K., 2009. Multi-agent traffic simulation with cuda, in: 2009 International Conference on High Performance Computing & Simulation, IEEE. pp. 106–114.
[14] Suzumura, T., Kanezashi, H., 2014. Multi-modal traffic simulation platform on parallel and distributed systems, in: Proceedings of the Winter Simulation Conference 2014, IEEE. pp. 769–780.
[15] Suzumura, T., McArdle, G., Kanezashi, H., 2015. A high performance multi-modal traffic simulation platform and its case study with the dublin city, in: 2015 Winter Simulation Conference (WSC), IEEE. pp. 767–778.
[16] Tchervenkov, C., Hörl, S., Balac, M., Dubernet, T., Axhausen, K.W., 2020. An improved replanning strategy for congested traffic conditions in matsim. Procedia Computer Science 170, 779–784.
[17] Vu, V.A., Tan, G., 2017. High-performance mesoscopic traffic simulation with gpu for large scale networks, in: 2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT), IEEE. pp. 1–9.
[18] Waraich, R.A., Charypar, D., Balmer, M., Axhausen, K.W., 2009-09. Performance improvements for large scale traffic simulation in matsim, in: 9th STRC Swiss Transport Research Conference : Proceedings, Swiss Transport Research Conference, Ascona. doi:10.3929/ethz-a-005864320. 9th Swiss Transport Research Conference (STRC 2009); Conference Location: Monte Veritá, Switzerland; Conference Date: September 9-11, 2009.
[19] Xu, Y., Song, X., Weng, Z., Tan, G., 2014. An entry time-based supply framework (etsf) for mesoscopic traffic simulations. Simulation Modelling Practice and Theory 47, 182–195.