Scaling Simulation of Continuous Urban Traffic Model for High Performance Computing System

Mateusz Najdek[1], Hairuo Xie[2], and Wojciech Turek[1]

[1] AGH University of Science and Technology, Krakow, Poland
[2] University of Melbourne, Australia

Abstract. Urban traffic simulation of extensive areas with complex driver models poses a significant computational challenge. Developing highly scalable parallel simulation algorithms is the only feasible way to provide useful results in this case. In this paper, we present extensions of the SMARTS system, a traffic simulation tool, which provides efficient scalability with a large number of parallel processes. The presented extensions enabled its scalability for HPC-grade systems. The extended version has been thoroughly tested in strong and weak scalability scenarios for up to 2400 computing cores of a supercomputer. The satisfactory scalability has been achieved by introducing several significant improvements, which have been discussed in details.

Keywords: Urban traffic simulation · Simulation scalability · Simulation metrics · HPC.

1 Introduction

The domain of Agent-based Modeling has been attracting more and more attention in recent years, providing means for explaining phenomena observed in complex social situations [10]. Microscopic urban traffic is a very interesting real-life case within the domain, where a large number of autonomous agents (drivers) coexist and interact in a common system (road network). The simulation of this phenomenon is a significant computational challenge. Simulation algorithms repetitively modify one large data structure, which, when performed in parallel, must be properly synchronized. Providing parallelization methods for such a task, which could efficiently utilize High-Performance Computing (HPC) hardware, remains an open problem.

The work presented in this paper is the result of cooperation between the creators of the SMARTS traffic simulator [11] from University of Melbourne and the HPC team from AGH University of Science and Technology. The main contribution is the extended version of the SMARTS system, which allows efficient simulation of the IDM [4] model on 2400 cores of a supercomputer. We also describe introduced improvements and provide a discussion of typical issues with creating or migrating spatial simulations for HPC-grade systems.

2 Scalable Traffic Simulations

The need for parallel execution of traffic simulation has been identified at the end of XX century [7]. At least a few attempts to creating a centrally synchronized parallel traffic simulation were tried later on. Work presented in [6] or [9] report scalability at a few nodes. The concept of increasing the time between global synchronization, presented in [2], led to positive impact on scalability and significant errors in the simulation results. Another approach, presented in [13], introduced a complex protocol for correcting the errors resulting from rare synchronizations.

In [12] authors suggested that the global synchronization in parallel traffic simulation algorithm might not be necessary. Removing centralized element from the computation is crucial in terms of HPC-grade scalability, however, achieving this in traffic simulations was not straightforward. Probably the first reports on a method which overcame this problem were presented in [18]. The authors presented an Asynchronous Synchronization Strategy, which assumed that each computing process communicates with limited number of processes – only those responsible for adjacent fragments of the modeled environment. The authors achieved linear scalability with 32 parallel workers. The concept has been extended in [14,15], where a simulation of discrete traffic model scaled linearly to 19200 parallel processes running on 800 nodes of a supercomputer, which is probably the largest setup tested so far in this type of simulations.

The work presented in this paper is the result of merging the experiences from building HPC-scalable traffic simulations with the SMARTS tool, which supports continuous traffic models and manages realistic maps. *Scalable Microscopic Adaptive Road Traffic Simulator (SMARTS)* [11] is a free, light-weight and versatile simulation tool developed at the School of Computing and Information Systems, University of Melbourne, Australia. The simulator has been used in many research projects [8,16,17]. SMARTS builds road networks based on freely-available OpenStreetMap data, which allows researchers to simulate traffic in real road networks around the world. During a simulation, the position of vehicles on roads is updated based on a continuous model (a car-following model), *Intelligent Driver Model (IDM)* [4], which computes the acceleration of the vehicle based on several factors such as the distance to an impeding object. A simulated vehicle is also controlled by a lane-changing model, *Minimizing Overall Braking Induced by Lane Changes (MOBIL)* [3], which is used to determine whether it is safe and beneficial for the vehicle to change to specific traffic lanes. SMARTS also models traffic lights and a number of traffic rules.

SMARTS is a distributed system that has one *server* and one or more *workers*. It divides the simulation area into multiple sub-areas, assigned to different workers that run in parallel and exchange information only with workers responsible for adjacent sub-areas. The workers do not need to communicate with the server during simulation, which helps prevent a communication bottleneck at the server. SMARTS also adopts a Priority Synchronous Parallel model, which uses a two-phase simulation approach to reduce the impact of the slowest worker [5].

3 Towards HPC-grade Traffic Simulation

The concept of the simulation method presented in SMARTS is suitable for distributed computing, but for a high level of parallelization at High-Performance Computing (HPC) systems several modifications were required. Considering massive HPC parallelism, where the number of workers can be orders of magnitude larger than in the case of small clusters, one should always focus on minimizing the non-parallel part of process or possibly getting rid of it entirely. Original elements of SMARTS, like workers initialization, task division, communication, and results collecting could significantly reduce overall efficiency and even crash the entire system. To overcome these problems the following improvements were required.

3.1 Initialization of the Simulation

SMARTS imposes on each worker to know metadata of its neighboring workers with which it must communicate. To obtain this information, a connection must be established with a centralized server, shared among all running workers. The server builds a dedicated connection with each worker synchronously. This sequentiality overhead cannot be easily avoided, but the effect is relatively reduced in long simulations. It also does not influence the proper simulation stage.

Another much greater sequential part of initialization, referred to as *setup* in sec. 4, requires all workers to obtain simulation configuration from the server by message pushing model. Conceptually, it is better to aggregate data dedicated to all workers within a single node and distribute them locally, involving their resources for parallelizing the process. In HPC practice, a far more efficient solution is to use the common filesystem for shared data distribution. Memory extensive part of shared message content is then loaded locally in parallel.

3.2 Data Scalability

To be able to simulate large scenarios within the available memory, one has to analyze the volume of data and divide them properly between available computing resources to ensure data-scalability. Generally, two types of data can be distinguished: static and dynamic. The static, such as the model of the environment, should be divided among all processing workers. Currently, SMARTS system balances the workload by generating vehicles with random routes on-the-fly, using the whole map at every worker. The optimal solution is to use predefined routes and remove the need for global planning from workers. This will reduce the setup time and also memory usage within a single node. The second data type (dynamic) can increases at demand and correspond mainly to the mobile entities. To prevent complete memory consumption, a safe estimation of maximal volume for each worker should be performed. Another method of prevention is online monitoring of transmitted data and dynamic load balancing – which is a challenge for further research.

3.3 Massive Parallelism Issues

All operations in a distributed environment which use randomness require special attention (or should be avoided), as they can cause complications visible only in large scale. Defining internal identifiers of individual processes based on a randomly generated string may seem a sufficient approach. However, being a variant of *the Birthday problem*, it can, with an increasing number of processes cause problems, when generated the same identifier. A better approach is to associate an internal identifier with a uniquely defined entity in the cluster topology as it guarantees unambiguity. Another thing to be taken into consideration is access to shared system resources, such as random socket number selection by different processes. It can manifest itself when a larger number of workers is involved, causing not obvious race conditions between separate processes within a computing node. Incoming messages were processed so far by dedicated, emerging threads in a thread-per-message approach. This solution is quickly worn-out at a larger scale and causes a massive decrease in efficiency with more incoming messages. To overcome this problem Thread Pooling based solution was introduced where the creation time of thread is avoided and its total amount is restricted. The messages are stored in a queue and handled by reusable processing threads.

3.4 Simulation results processing

The required details of the collected metrics should be carefully considered since fined-grained metrics cause larger sizes of exchanged messages and also more frequent communication. Where every timestamped vehicle positions are not required or are in final post-processing aggregated anyway, then it is better to perform the aggregation ahead at each worker. In such a case instead of gathering trajectories directly, it is better to transform them into more coarse-grained metrics such as density and flow per road. The original architecture, where trajectories were collected by the server, is not feasible with more than 240 workers for the reason of the limited amount of memory. In order to support this feature in the HPC-grade version, we introduced a solution to bind the trajectory metrics with the corresponding vehicle state. It is gradually growing, burdening workers instead of the server. The proposed solution allows for independent serialization of such results at the end of simulation, fully in parallel, instead of massively time-consuming sequential serialization. It totally eliminates the sequential part of this process and greatly speeds it up by a factor of the number of workers used.

4 Evaluation of Traffic Simulation Scalability

Different test scenarios were prepared with an increasing number of processing cores. The first scenario of strong scalability included a fixed minimum number of 2.4 million vehicles in total. The second one (weak scalability) increased from 24 thousand up to 2.4 million vehicles in proportion to the number of workers. Routes were randomly generated using A-star algorithm. Both scenarios used 59km×55km area of Beijing. The road graph contained approximately 400 thousand directed edges, which correspond to streets and 220 thousand nodes, that represent intersections and points of road division into smaller sections. The average length of a road is about 85 meters and the total length is 58.000 km.

All tests were performed using the Prometheus supercomputer, a part of the PL-Grid infrastructure[1]. Currently, Prometheus is ranked as the 324th computer in the worldwide top 500 list[2]. It contains 2232 computing nodes, HP Apollo 8000 with Xeon E5-2680v3 CPUs working at 2.5GHz. Each node has 24 physical cores – the total number of computing cores is 53,568. Nodes are connected by the Infiniband FDR 56Gb/s network.

For evaluation purposes, two main categories of performance metrics were collected depending on the source. The first describes the server perspective. Figures 1, 3 shows execution time of 4 components: global time of initialization, model setup and loading, simulation, serialization of obtained results. The second type describes simulation from the worker's side. Figures 2, 4 present the average execution step time from 50 steps for data collected over the last 1000 out of 1500 steps. The length of a single step was 0.2 second, which is equivalent to 6 minutes of real-time traffic. Synchronization is based on decentralized communication and result are collected in a decentralized manner. All tests were repeated 3 times to confirm the stability of the obtained results. Each node has been configured to run 24 workers and the server was run on a separate node.

4.1 Strong Scalability

As Amdahl [1] pointed out in 1967, for fixed-sized problem overall speedup is limited by the factor of sequential part that cannot be subject to parallelization. Based on that, strong scalability is defined as the variance of the solution's execution time to the utilized number of computing processors. In this experiment, each worker minimally simulated the number of vehicles equivalent to roughly a fraction of the fixed number of 2.4 million vehicles by the number of workers. Since the size of the problem is large, it was not feasible to run scenario using less than 4 nodes. Figure 1 shows results of global execution times for each component of the process.

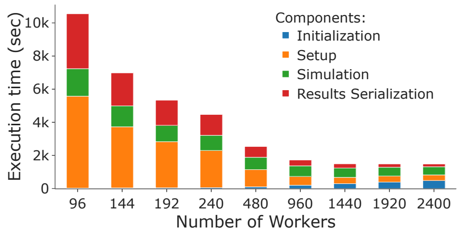

Fig. 1. Global times of process components with increasing number of workers for growing problem size.

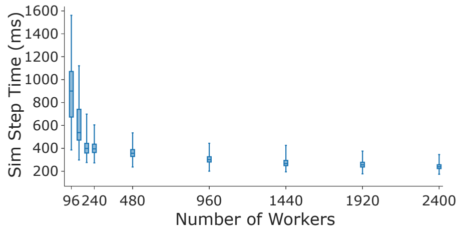

Fig. 2. Simulation step times for increasing problem size in proportion to the number of workers.

The whole process for the same task size benefits greatly from adding up to 1440 workers and further increase causes it to suffer due to the initialization part. In Figure 2 the median of simulation step time is reduced and distribution becomes more convergent with an increasing number of workers.

[1] http://www.plgrid.pl/

[2] https://www.top500.org/lists/top500/list/2020/11?page=4

4.2 Weak Scalability

As Gustafson [1] noticed, in practice predominantly size of the problem scales with the number of available resources. The observation is that if the problem is small, there is no benefit in using large amounts of resources, and a more sensible approach is to use resources according to the problem size. Weak scaling is concerned with how the execution time for solving a scaled size problem varies with the number of CPUs used. The computational costs depend on the number of vehicles and the results are presented in figures below (3 and 4), where each worker initially simulated a minimal number of 1000 vehicles.

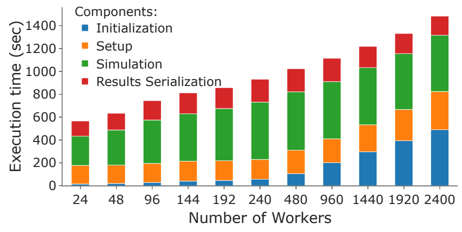

Fig. 3. Global times of process components with increasing number of workers for growing problem size.

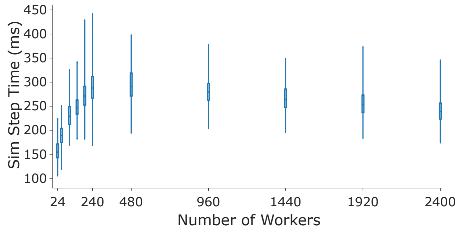

Fig. 4. Simulation step times for increasing problem size in proportion to the number of workers.

The main factor extending the duration of the entire process is the initialization part. As shown in Figure 4 results confirm algorithm scalability as the simulation time of each task remains within a constant range. The conclusion is by increasing the number of workers communication overhead is not strongly affecting this method.

5 Conclusions and Further Work

The presented solution to the problem of efficient urban traffic simulation is most likely the first to efficiently utilize HPC-grade hardware for simulation of continuous traffic model with real-life scenarios. The simulation algorithm itself exposes proper features both in strong and weak scalability tests. Achieved overall scalability of the simulation system is not perfect, however, in all of the analyzed cases addition of hardware improved the simulation efficiency.

Extensions introduced to the SMARTS platform resulted in conclusions regarding HPC-grade software design, which goes beyond the particular application. Identified issues and proposed solutions may become valuable guidelines for researchers willing to scale spatial simulations to HPC-grade. The extended versions of SMARTS system, used for conducting the presented experiments, is available at: https://github.com/mateusznajdek/SMARTS-extension.

A few aspects still can and should be improved. More efficient strategies for initialization are definitely needed. Also, scalable mechanisms for model division should be introduced. These will be the subject of further research.

Acknowledgments

The research presented in this paper was funded by the National Science Centre, Poland, under the grant no. 2019/35/O/ST6/01806. The research was supported in part by PL-Grid Infrastructure.

References

1. Gustafson, J.: Reevaluating Amdahl's Law. Commun. ACM **31**(5), 532–533 (1988)
2. Kanezashi, H., Suzumura, T.: Performance optimization for agent-based traffic simulation by dynamic agent assignment. In: Proc. of 2015 Winter Simulation Conference. pp. 757–766. WSC '15, IEEE Press, Piscataway, NJ, USA (2015)
3. Kesting, A., Treiber, M., Helbing, D.: General lane-changing model MOBIL for car-following models. J. of Transportation Research Board **1999**(1), 86–94 (2007)
4. Kesting, A., Treiber, M., Helbing, D.: Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. Trans. of Royal Society of London A **368**(1928), 4585–4605 (2010)
5. Khunayn, E.B., Karunasekera, S., Xie, H., Ramamohanarao, K.: Straggler mitigation for distributed behavioral simulation. In: 2017 IEEE 37th Int. Conf. on Distributed Computing Systems (ICDCS). pp. 2638–2641. IEEE (2017)
6. Klefstad, R., Zhang, Y., Lai, M., Jayakrishnan, R., Lavanya, R.: A distributed, scalable, and synchronized framework for large-scale microscopic traffic simulation. In: Proc. 2005 IEEE Intelligent Transportation Systems, 2005. pp. 813–818 (2005)
7. Nagel, K., Schleicher, A.: Microscopic traffic modeling on parallel high performance computers. Parallel Computing **20**(1), 125 – 146 (1994)
8. Nguyen, U.T., et al.: A randomized path routing algorithm for decentralized route allocation in transportation networks. In: ACM SIGSPATIAL. pp. 15–20 (2015)
9. O'Cearbhaill, E.A., O'Mahony, M.: Parallel implementation of a transportation network model. Journal of Parallel and Distributed Computing **65**(1), 1 – 14 (2005)
10. Railsback, S.F., Grimm, V.: Agent-based and individual-based modeling: a practical introduction. Princeton university press (2019)
11. Ramamohanarao, K., Xie, H., Kulik, L., Karunasekera, S., Tanin, E., Zhang, R., Khunayn, E.B.: SMARTS: Scalable microscopic adaptive road traffic simulator. ACM Trans. on Intelligent Systems and Technology (TIST) **8**(2), 1–22 (2016)
12. Rickert, M., Nagel, K.: Dynamic traffic assignment on parallel computers in transims. Future Generation Computer Systems **17**(5), 637 – 648 (2001)
13. Toscano, L., D'Angelo, G., Marzolla, M.: Parallel discrete event simulation with erlang. In: Proceedings of the 1st ACM SIGPLAN Workshop on Functional High-performance Computing. pp. 83–92. FHPC '12, ACM, New York, NY, USA (2012)
14. Turek, W.: Erlang-based desynchronized urban traffic simulation for high-performance computing systems. Future Generation Computer Systems **79**, 645–652 (2018)
15. Turek, W., Siwik, L., Byrski, A.: Leveraging rapid simulation and analysis of large urban road systems on hpc. Transportation Research Part C: Emerging Technologies **87**, 46–57 (2018)
16. Xie, H., Karunasekera, S., Kulik, L., Tanin, E., Zhang, R., Ramamohanarao, K.: A simulation study of emergency vehicle prioritization in intelligent transportation systems. In: IEEE VTC Spring. pp. 1–5 (2017)
17. Xie, H., Tanin, E., Karunasekera, S., Qi, J., Zhang, R., Kulik, L., Ramamohanarao, K.: Quantifying the impact of autonomous vehicles using microscopic simulations. In: ACM SIGSPATIAL. pp. 1–10 (2019)
18. Xu, Y., Cai, W., Aydt, H., Lees, M., Zehe, D.: An asynchronous synchronization strategy for parallel large-scale agent-based traffic simulations. In: Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. pp. 259–269. SIGSIM PADS '15, ACM, New York, NY, USA (2015)