TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme

Cloud Simulation for Large-Scale Agent-Based Traffic Simulations

Daniel Zehe

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr.-Ing. Nils Thürey
Prüfer der Dissertation:	1. Prof. Dr.-Ing. habil. Alois Knoll
	2. Prof. Wentong Cai, Ph.D.

Die Dissertation wurde am 28.06.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 23.11.2018 angenommen.

Abstract

It has always been the goal of urban system simulators to build ever-more detailed simulation models to understand the complex interactions between the different entities in the urban environment. In order to simulate these complex models, the computational power required is significant. The last two decades have shown that this problem of ever-more demanding software has been solved by increasing the processing power of a single CPU, increasing the number of processors of a CPU die and creating a better interconnect between the hardware nodes.

The setup of such computing systems is an expensive endeavor and not accessible to everyone due to the complexity of setup and maintenance. Domain experts of smaller and medium-sized institutions are not able to conduct large-scale simulation experiments, since only large research centers like universities, large cooperations or the military can afford this kind of infrastructure.

The growth of cloud computing in the last 10 years now allows computing environments to scale on demand. Since the infrastructure and services are comparable in performance to dedicated hardware, not only consumer-oriented web services can be improved, but also scientific simulation experiments can benefit greatly from this development. A high-performance simulation for urban systems is only one example.

This thesis introduces a reference architecture for a cloud-based agent-based simulation platform, which allows domain experts of different fields to collaborate on simulation experiments. Apart from defining an abstract architecture, a specific implementation for traffic-simulation experiments has been developed as part of this thesis. This includes the extension of the SEMSim traffic (now CityMoS) simulation platform to be cloud capable as well as a data analysis framework for online data extraction of simulation data with a low bandwidth footprint. The scalability of cloud resources allows for a concurrent simulation-supported search-space exploration without much overhead. In this thesis an approach to improve the search-space exploration even further by using multi-resolution models to reduce the computational overhead when exploring a multi-dimensional search space is given.

Zusammenfassung

Bei der Simulation von komplexen urbanen Systemen, wie zum Beispiel Verkehr, kommt es auf die Genauigkeit der verwendeten Modelle an. Diese beschreiben den Zusammenhang und die Interaktionen zwischen den einzelnen, im urbanen System beteiligten Entitäten. Eine höhere Genauigkeit bedeutet auch eine größere Anforderung an Hardwareressourcen. In den letzten 20 Jahren wurde das allgemeine Problem der anspruchsvolleren Software durch Hardwareoptimierung angegangen. Mehr Taktgeschwindigkeit bei weniger Leistungsaufnahme und mehr Prozessorkerne auf einem Sockel sind das Resultat. Da jedoch die Grenze der Hardwareoptimierung erkennbar ist, wurde auch die Interkonnektivität zwischen Hardwareknoten verbessert. Auf diese Weise wird die Arbeit in so genannten High-Performance-Clustern auf mehrere Computer verteilt. Somit können Simulationen in sehr viel weniger Zeit durchgeführt werden als noch vor 20 Jahren. Die gestiegene Komplexität der Hardware erschwert den Zugang, da diese Systeme teuer in Anschaffung und Erhaltung sind. Domänenexperten kleiner und mittelständischer Unternehmen versperrt dies den Zugang zu großflächigen Simulationsexperimenten, da sich nur akademische, Regierungs- und Militäreinrichtungen derartige Infrastrukturen leisten können.

Eine mögliche Abhilfe ist Cloud-Computing. Es erlaubt Hardware je nach Anforderung dynamisch zu skalieren. Da die Cloud-Infrastruktur auf die gleiche Hardware setzt wie dedizierte High-Performance-Cluster, können dadurch nicht nur endkundenorientierte Dienste (soziale Netzwerke, E-Mail, Video-on-Demand etc.), sondern auch wissenschaftlichen Simulationen verbessert werden; urbane Simulationen sind dafür nur ein Beispiel.

In der vorliegenden Dissertation wird eine Referenzarchitektur für cloudbasierte und agentenbasierte Simulationsplattformen vorgestellt. Diese ermöglicht es Experten aus unterschiedlichen Domänen kollaborative Simulationsexperimente zu erstellen.

Des Weiteren wird die Durchführbarkeit dieser Architektur anhand einer spezifischen Implementierung für die Verkehrssimulation CityMoS aufgezeigt und evaluiert. Dazu gehört die Erweiterung durch eine Online-Datenextraktion und -analyse Middleware, um den Datenspeicheraufwand zu minimieren. Die Skalierbarkeit von Cloudressourcen erlaubt es, gleichzeitig mehrere simulationsbasierte Suchraumexplorationen vorzunehmen. In der vorliegenden Dissertation wird diese optimiert, indem Techniken der Hybriden-Modellierung angewandt werden. Diese minimieren die Laufzeit, bei gleichzeitiger Aufrechterhaltung der Simulationsergebnisgüte.

Acknowledgements

I would like to thank Prof. Dr. Alois Knoll for giving me the opportunity to conduct research under his supervision. I would also like to express my gratitude towards Prof. Dr. Wentong Cai from the Nanayang Technological University and the Principle Investigator at TUMCREATE.

Additionally, I would like to thank Dr. Heiko Aydt, Dr. Vaisagh Viswanathan and Dr. Suraj Nair for their thematic support while conducting my research at TUMCREATE. While working at TUMCREATE, I had the opportunity to supervise David Grotzky, Marianna Avezum and Akriti Vij, while they were working for their Bachelor's and Master's theses or their final year projects. Their work strongly influenced my research.

Lastly, I would like to thank my colleagues at TUMCREATE, who provided me with a pleasant and friendly working environment as well as may fruitful discussions on scientific matters. The support from Lim Xiu Fen over the last 4 years was exceptional.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

The complexity of simulations has grown exponentially over the last 50 years and so has the computational capability of the hardware running such simulations. Today, chemical and physical processes can be represented using computer simulations to a granularity which was not feasible a decade ago. These include high-fidelity climate simulations that take a single day to simulate the global climate for an entire year, producing 100TB of output data [1]. Biological simulations of pandemics [2] or folding of proteins as in the *Folding@Home*[1] project can be computed in a fraction of the time that was required years ago. The most complex of human organs, the human brain[2], is also heavily researched and will be simulatable soon. The simulation of vehicular traffic in cities, regions and entire countries poses a similar problem.

The main objective of the work presented in this thesis is to explore the requirements and give a possible solution that allows domain experts with limited technical knowledge to use high-performance simulations in the cloud. This includes the creation of a simulation cloud-service reference-architecture, exploring the challenges of simulation templating and data-extraction as well as using multi-resolution modeling and simulation to speed up search-space exploration for agent-based traffic simulations. The computational time required to conduct a simulation of the aforementioned systems depends mostly on two parameters: (1) The size of the simulated

[1]`http://folding.stanford.edu/home`
[2]`https://www.humanbrainproject.eu/en/brain-simulation/`

system and (2) the resolution or granularity at which the domain is modeled and simulated. Controlling these parameters gives users the ability to influence the time required to conduct experiments.

In the age of *Big Data* the need for ever-faster and more reliable information is ubiquitous. At the same time, however, lower-resolution simulations and models which would be computationally fast are in some use-cases too coarse-grained to draw conclusions. High granularity also means that the smallest mistakes in modeling a system component or a misconfiguration of a model parameter can have a significant effect on the result of the simulation, which can lead to conclusions drawn from misinformation. Rerunning simulation experiments can be costly in time as well as financially. Domain experts in a specific field have the knowledge of model behavior for their particular sub-system of a greater, more complex system. The experts understand the complex dependencies and influence of parameters and behavior with neighboring models. The combined knowledge of all domain experts can lead to a complete system model, which subsequently can be simulated in high resolution and with great accuracy.

This presents a dilemma, that the modeled, and ultimately simulated, complex system, might only show its relevant characteristics at high resolutions and the ever-increasing need for fast simulation results, to base decisions on, leads to the conclusion that execution of high-resolution simulations needs to be improved. A solution for this is the use of High-Performance Computing (HPC), also commonly known as super computing, where many highly specialized computing environments execute a simulation task together. However, the problem with HPC centers is that the computers also known as nodes need to be closely interlinked and a large number of nodes are required.

The issue of scale has been resolved by distributing workloads to ordinary computers in projects like *Folding@Home* or *Seti@Home*[1], where smaller work packages are sent out to users who are willing to share the resources of their (personal) computing devices such as laptops, workstations or game consoles, to work on small pieces of data and then send the result back to a centralized computer, integrating the individual results. This approach might be effective in utilizing the raw computing power of modern computing devices. However, the interlinking between these devices is loose. The network connection, in such instances mostly the Internet, cannot be used to exchange state change information between clients quickly, due to bandwidth and latency limitations and inequalities.

[1] `https://setiathome.berkeley.edu`

Nonetheless, this leads to a use-case where each computing device is working on a smaller part of a bigger problem independently of other devices working on the same bigger problem. This type of loose interconnection is useful for specific computing tasks. Unfortunately, complex system simulations cannot work using such paradigms where a high degree of fast interlinking is required. Small changes in one sub-system can affect another sub-system, which cannot be predicted or modeled beforehand. This property is called the emerging behavior of a complex system and can be observed in high resolution complex system simulations. In order to communicate small state changes from one sub-system to another, a fast interconnection between nodes needs to present. As stated above, the Internet is one way of connecting computers to exchange information between distributed computing nodes. Even though the average bandwidth of the Internet has increased (see Figure 1.1), and will further increase, the latency between sending a piece of information and receiving it in a different geographic location is still large. For the time being, the solution to raw computing power and fast interconnection is the use of HPC centers. They provide a homogeneous computing environment of hardware and software, allowing simulations to be optimized to deliver results in the most effective manner.

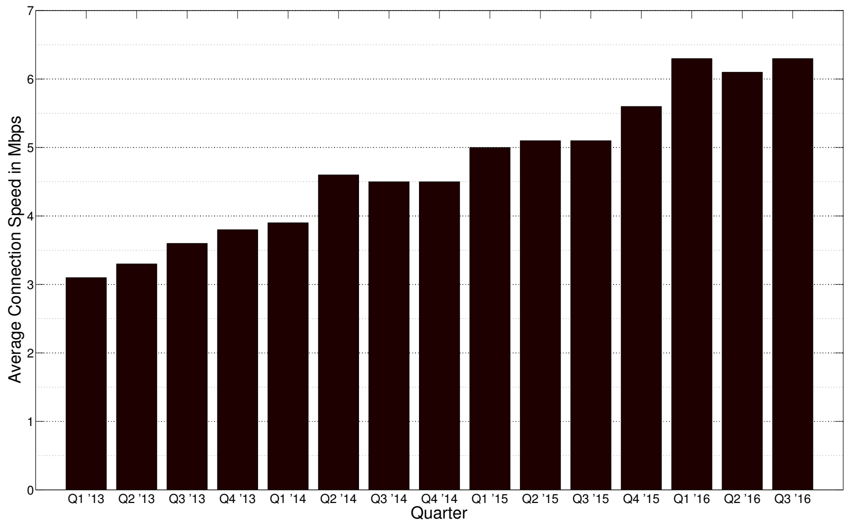


Figure 1.1: Average global connection speed from 2013 to 2016[1]

Setting up a HPC center is expensive. It not only relies on providing the computing hardware and software, but also on other factors. These are, for example, the location of the facility and

[1]`http://stateoftheinternet.com`

its connection to data and power infrastructure. The connection to the Internet and therefore the connection to the users and other data centers is as important as a reliable electrical infrastructure connection. A dependable power supply is required for running the computing hardware (e.g., computers, network switches and storage) but more importantly for cooling the facility to optimal operating temperatures. The prevailing political conditions in a country are also a deciding factor. Data security and privacy depend very much on the jurisdiction the data centers are built in and the stability of the political system in the country. In short, it is expensive and a complex endeavor to build and maintain a HPC center. Therefore, few can afford to have a large computing center. Usually, research institutes, government facilities as well as military establishments have the means and requirements to own a dedicated data center for HPC. This has some drawbacks in advancing research and development of new simulation tools to solve today's and also tomorrow's problems. The capability of simulating the evermore complex world should be open to everyone. It also extends to the individual non-researcher, who can make more informed decisions based on complex system simulations without knowing that a simulation was executed in the first place. In this thesis this decision-making process using simulation tools will be called *simulation-supported decision making.*

Cloud computing offers pay-per-use access to data-center resources that would otherwise not be accessible to such user groups. The Cloud Providers (CPs) offer access to their computing hardware and data center facilities on an abstracted level. They also take care of the prevailing conditions as well as maintenance, which includes updating the software and hardware, of the entire system.

While simulation tools on workstation or laptop computers can offer graphical user interfaces, the execution of the tools in data centers or on the cloud do not offer such an option or only offer it with reduced functionality. While a single executable is run on a local machine where the input data is present and parametrization is also done, the distribution to a remote data center requires a defined workflow and knowledge of all different components involved to ensure optimal simulation execution. This includes data transfer, setup of the environment, observation and monitoring of the simulation progress, and retrieval of simulation results as well as deallocation of used resources.

Such a set of skills cannot be expected of a domain expert. Therefore, the approach of simulation execution needs to be rethought. Researchers should focus on their domain-specific research and use the tools to answer significant scientific questions instead of encountering software related problems. It should be as easy to configure and use high-performance simulation software in the cloud as it has become easier to obtain simulation hardware in the cloud. It has become an everyday tool like web-based email programs or social media. These tools need to be as easy to use as the ones that are currently utilized but at the same time as powerful and computationally optimal as possible. This combination can be achieved by using HPC paradigms, the most advanced hardware, software and interconnections together with rethought interaction paradigms to create, control and manage simulation-based experimentation, using cloud resources.

Unfortunately, this is not as easy as it might sound. The difficulties lie not in the obtaining the respective hardware resources from the CP in an ad-hoc manner, but rather in the automation as well as configuration of the software using such resources. It is not feasible for a domain expert with limited computer engineering knowledge to obtain, automatically allocate and deallocate resources from the CP, transfer the required input data to the instances, start and monitor the running of the experiment and ultimately retrieve the experimental results. The CPs enable each of the steps with available tools and web-based front-ends. Even though these web-based front-ends can be used easily and provide a low entrance barrier, the difficulty lies with optimally using the resources to not waste a lot of time, which in cloud computing can be directly translated into money. Web front-ends are useful for a one-time ad-hoc usage. Once a more elaborate experimentation setup, including several nodes and different services, is required, such work becomes cumbersome. The coordination of such experiments and the structured allocation and deallocation of resources becomes especially important when using the scalability of the cloud for executing several experiments concurrently (e.g., for search-space exploration). The individual cost for using a few resources is relatively low, but using these resources concurrently can become a significant constraint on the budget. A reduction in experimentation time or speeding up the search-space exploration algorithm can have a huge effect.

Ever since the onset of scientific computer networks, the exchange of knowledge has been one of the main objectives. In the field of modeling and simulation, there have been tools and support mechanisms [3, 4, 5] that help experts from different fields to work on interdisciplinary projects like simulations of complex systems. These tools mainly focus on designing databases and ontologies for knowledge collection and distribution. Repositories of models and their parameter ranges are created by domain experts to facilitate scientific exchange within the community. With the onset of cloud computing and larger scientific networks, these databases are made accessible to a wider audience through web services [6]. This is a more convenient and intuitive way since web-services are widely used in professional as well as personal life by many people. The low barrier-of-entry facilitates the exchange and improvement of existing models or the creation completely new simulation models for a specific (sub-)system. For the simulation of mechanical parts where standardized simulation techniques exist, cloud services that offer off-loading of simulation tasks to *Simulation Software as a Service* web-sites have been developed and are in use commercially. These simulation-based testing services for e.g., fluid-dynamic, structural mechanics and thermal evaluation are performed on digital prototypes (e.g., CAD models). The research project CloudSME [7] is an European Union founded project that aims at enabling small and medium enterprises in manufacturing and engineering to use simulation tools. It is using the App-Model at providing different simulation services on the cloud to domain experts from industry with limited technical knowledge.

Another difficulty of using cloud resources to conduct experiments is the technical as well as domain knowledge of different user groups that have an interest on using cloud computing services. These different kind of users who would greatly benefit from having access to high performance cloud resources for simulation experiments are:

1. The **Domain Expert** that has the in-depth knowledge of a parameters and their value range for given complex system component or model. This group of users can estimate the consequences of changing models or model parametrization in a given system.

2. The **Integration Expert**, who has an overview of the entire system and general knowledge of the models and their behavior. This domain expert uses models and parameterizations developed by the domain expert to assemble a high-detailed and extensive simulation experiment.

3. The **Administrative User**, who is not an expert in a specific domain, but uses the results of a simulation experiment to base the person's decisions on. Those can be policy makers or administrative advisors. They have very specific questions they want to have answered by using simulation tools. The number of parameters that the simulation should expose to them needs to be low, in order to not over-complicate the setup process.

4. The **non-researcher** has neither technical nor in-depth domain knowledge. The person uses tools to make informed decisions that affect their life or field of work, but does not know about simulation or the cloud.

Each of these user groups can utilize the power of the cloud in different ways, but the technical entrance barrier into cloud-based simulation should be lowered either way. For some of the user groups. the cloud should be as transparent as possible to have them focus on the goal and not the technicalities.

1.1 Problem Statement

These aforementioned difficulties in using cloud resources for conducting experiments, especially simulation-based experiments, describe a problem that this thesis will try to solve. In the context of this thesis, the focus is not on the hardware configuration or the availability of cloud resources, but rather on the simulation software services that use the HPC cloud resources. The problems that exist for agent-base simulations in the cloud and those need to be overcome in order to fully utilize the computational capabilities that cloud-based resources offer to enable everyone to use simulation tools are:

1. The interoperability between different user groups of varying degrees of domain knowledge
2. The technical difficulty of designing, starting, monitoring and evaluating agent-based simulations in the cloud
3. The ways in which the cost and overall usage of cloud resources can be reduced and experiments optimized
4. The democratization of the decision making process, by allowing everyone to use simulation-based tools on the cloud

1.1.1 Contribution

The main contribution of this thesis is the analysis and design of a simulation cloud service reference architecture which addresses simulation specific research questions. This is tailored towards traffic simulations but can be applied to most other types of high-resolution agent-based simulations as well. Along with the architecture, several problems arise when moving the execution from the local workstation computer into the cloud. One problem is the accessibility of the simulation software by the end-user. User interfaces and interactions have to be rethought since the computer executing the simulation program is not the same as the computer user operates to design, configure and start an experiment. The above-mentioned user groups have the possibility of working together using the same underlying back-end, while the concepts of cloud and specific domain knowledge can be as abstract or as transparent as possible. As a use-case, a Representational State Transfer (REST)full Application Programming Interface (API) for the Scalable Electromobility Simulator (SEMSim) traffic simulation platform was implemented in which the first three groups can work together to model traffic related scenarios and execute such simulations in the cloud.

Also a simulation-experiment templating concept has been envisioned, which allows advisors, policy makers and non-researchers (group 4) to base decisions on simulation-based experiments, where only a small number of parameters important to the question at hand need to be changed, while still utilizing on the computational power and high-resolution models running on cloud resources. This enables novices, non-experts and related domain experts to have a low entrance barrier when using HPC-based simulations.

Another challenge when executing simulations in remote locations, where physical access is not given to the end-user, is data retrieval. Users need a way of retrieving the result datasets of their simulations for post-processing and evaluation. These can be large datasets, which are generated and stored during the simulation and later transfered to a post-processing environment. Within the context of this thesis, a different approach for an online data extraction system is given; in which the post-processing is done while the simulation experiment is still running and thus reducing the amount of data being stored or transfered off-site. For this, a transparent middleware concept was developed, which reduces the data transfer between the simulation and the (post/online-)processing pipeline.

Another key contribution of this thesis is the concept of using multi-resolution modeling of a traffic system to speed up the search-space exploration in the cloud. It utilizes the idea that in many simulations, especially traffic simulation, there are multiple levels in which the same scenario can be simulated. These levels are nanoscopic (sub-microscopic), microscopic, mesoscopic and macroscopic which reach from very fine granularity to very coarse granularity respectively. The presented approach uses nanoscopic and macroscopic models in the same simulation and can switch between them on demand. This reduces the overall computational time, but not significantly reducing the fidelity of the simulation results. This general approach to multi-resolution simulation in traffic was then used to evaluate different city networks from around the world on their optimization potential. This is inherently important for the development of a cost model for large-scale search space exploration in the cloud, as described in the same chapter, because the cost savings are directly related to the amount of time saved by using multi-resolution simulation experiments.

1.1.2 Thesis Outline

After introducing the related work in the fields of traffic simulation and cloud computing, SEMSim traffic (City Mobility Simulator (CityMoS)) platform will be introduced, which is used throughout the thesis as an example for a complex system simulation (CSS) of vehicular traffic in large cities.

The SEMSim Cloud Service in Chapter 3 is a reference architecture implementation for an agent-based simulation cloud service. It allows different domain experts to collaboratively develop component models for traffic simulations and design simulation experiments. These experiments are then executed using cloud resources and the results can be downloaded to the user's computer for further analysis. Within the context of the reference architecture, experiments that compare the hardware performance of dedicated HPC nodes to virtual-machines nodes in the cloud are conducted.

Compared to a generic Cloud Service that enables different domain experts related to traffic simulation to explore their models on the cloud, a more end-user-centric approach to cloud computing is to abstract the cloud and technicality away from the user, while still leveraging on the computational power of cloud computing infrastructure. This allows use of APIs of the SEMSim cloud service to create specialized templates for specific simulation experiments. These experiment templates are pre-composed simulation configurations which allow only a limited number of parameters to be changed. Using these templates, the end-user can explore new possibilities and configurations of traffic systems, while being sure that models from outside their domain expertise behave as verified by the respective domain experts. This allows domain experts and policy makers to focus on their field of research.

Simulation templates can also help non-specialists to use simulation tools without specialized knowledge to draw their own conclusions on current problems like traffic policy decisions. The ubiquitous use of high-resolution simulations on the cloud in all walks of life with easy-to-use interfaces and templates for answering specific questions allows the democratization of the decision-making process at the lowest level.

Since the simulation experiments are executed in the cloud and the extraction of simulation output data is the essence of the entire simulation, new techniques for, not only, post-simulation data analysis but also online data extraction and analysis need to be studied. This is required since the data generated by large-scale agent-based high-resolution simulations can be considered as Big Data. In order to process the data produced in high velocity and in high volume, the storage capabilities of current technologies might not be up to the task in terms of read and write speeds and also total storage capacity. In Chapter 4, an envisioned online data-extraction system, allowing the reduction of data stored and transmitted by designing a stream processing framework, is presented. It enables the user to draw conclusions rapidly, even while the simulation is running, reducing the costs involved (e.g., time and money) when executing the simulation in dedicated data centers or on the cloud.

The seemingly limitless scalability of cloud resources also allows users to do concurrent search-space exploration. Reducing the simulation resolution can be an effective way to decrease execution time as well as cost of exploring a large search space. Chapter 5 introduces such a multi-resolution modeling and simulation technique. These hybrid-resolution simulations can be executed faster, while the high-resolution equivalent simulation is subsequently used to quantify the results in greater detail once an interesting parameter combination has been identified by the hybrid-resolution simulation. In addition to the fundamental research in multi-resolution simulation and resolution switching in hybrid-resolution simulations, a cloud computing cost model for sequential and concurrent execution has been developed. Also, an evaluation of network characteristics towards usability for hybrid-simulation of several metropolitan areas around the world was performed.

The thesis concludes with a discussion on use-cases and future work that can be explored.

Chapter 2

Background of Research

Contents

In this chapter, the related work within the context of this thesis will be evaluated. This includes a deeper insight into cloud computing developments, as well as agent-based simulation of large complex systems. As all of the use-cases presented in this thesis are related to agent-based traffic simulation, an introduction to this will be given as well.

2.1 Cloud Computing

2.1.1 Web-based and Cloud-based Simulations

Web-based simulations have been used in the scientific community for many years; the world wide web was developed through the scientific community. Most of the early applications of combining the web with simulations consisted of providing collaboration platforms or independent front-end interfaces [3, 4]. These systems offered interfaces in the form of web-applications or thin-clients for controlling simulations. Other simulation support systems distribute simulation jobs over a network of computing centers [5] for load-balancing or load optimization. For collaboration between different institutions, web-based repositories and interfaces for consolidating modeling efforts have always been popular [4].

The question of Modeling & Simulation as a Service (MSaaS) was surveyed by Cayirci [6]. In the survey, MSaaS is defined as a "model for provisioning modeling and simulation services on demand from a Cloud Service Provider (CSP)". It is furthermore defined that a user of such simulation systems should not be responsible for the maintenance, licensing of software or scaling of infrastructure. The survey largely focuses on the security, privacy and trust issues connected with using cloud computing as a vital back-end of simulation experiments. The security concerns discussed by Cayirci are the ones defined by the *Cloud Security Alliance*[1] for general cloud computing instances [8]. These are evaluated and transformed into tangible risks which are of concern for MSaaS. Cayirci concludes that it comes down to trust between the experimenter and the different CPs as well as a possible lock-in to a specific CP.

Guo et al. [9] tried to develop a service specification of how a Simulation Software as a Service (SSaaS) and service-oriented simulation experiments should be formally specified. They give a formal description of how service-oriented experiments can be expressed as well as how formal descriptions of a SSaaS can be given. This approach helps to produce a meta-model for simulation experiments.

Cloud-based services have been very successful for Web 2.0 applications as a back-end for mobile applications. There are some commercial applications available that offer SSaaS. The EU-founded research project and now start-up CloudSME[2] is concerned with the use of cloud-based simulations for the manufacturing and engineering industry. This project has shown its potential for being a viable option for cloud-based modeling and simulation [7]. SimScale[3]

[1]`https://cloudsecurityalliance.org`
[2]`http://cloudsme.eu`
[3]`http://www.simscale.de/`

and AutoDesk[1] both offer simulation services for *Computer-Aided-Design* (CAD) models. The simulation services include simulation-based tests for fluid-dynamics, structural mechanics and thermal evaluation on digital prototypes. The *Altair Simulation Cloud Suite*[2] allows researchers and engineers who have a CAD and Computer Aided Engineering (CAE) workflow for their simulations to move them completely to a cloud-based solution. This allows users to import their CAD models into the browser-based interface and allows an off-site simulation life cycle management, leveraging on the Altair Hyperworks[3] simulation tools for *Computational fluid dynamics* (CFD) simulation and exploration as well as visualisation. Another example is Rescale[4] which offers a cloud simulation platform. Their approach is to give researchers an easy-to-use web-interface for creating and starting experiments and simulations as well as allowing some basic data analysis. Their catalog of simulation software includes, among others, tools for finite element analysis, fluid dynamics and molecular dynamics from different software providers. In the scientific community, cloud-based computing infrastructure is used to give researchers the opportunity to execute proof-of-concept studies without the risk of investing in expensive hardware [10].

2.1.2 Cloud and High-Performance Computing

The terms *cloud-service* or *cloud-based* have been used in mass-media and mainstream frequently in the last decade. There are subtle differences between the different kinds of cloud computing solutions provided by commercial companies [11]. The difference between using dedicated hardware resources and using virtualized resources is also important as we should differentiate between using cloud-based virtualized hardware and dedicated hardware. Firstly, the different kinds of cloud services are introduced before differentiating cloud-based hardware and dedicated hardware.

Figure 2.1 shows the different stages of cloud services. Infrastructure as a Service (IaaS) is the underlaying foundation for many other services. It provides virtual back-end resources. These resources are usually shared between many users and applications. IaaS cloud services such as Azure[5] from Microsoft, Elastic Cloud from Amazon, Google Compute Engine (GCE) from Google and others offer computing resources on-demand. The drawback of using cloud

[1]`http://www.autodesk.com/products/sim-360`
[2]`http://www.altair.com/simulation-cloud/`
[3]`http://www.altairhyperworks.com/industry/Automotive`
[4]`http://www.rescale.com`
[5]`http://www.azure.microsoft.com/`

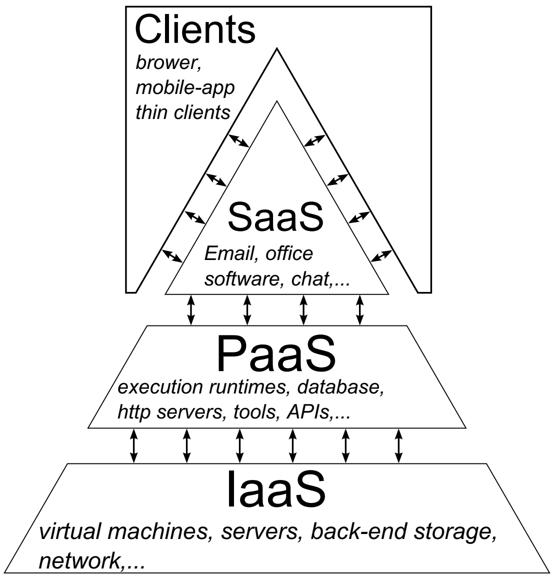

Clients
brower,
mobile-app
thin clients
SaaS
Email, office
software, chat,...
PaaS
execution runtimes, database,
http servers, tools, APIs,...
IaaS
virtual machines, servers, back-end storage,
network,...

Figure 2.1: Different kinds of traditional Cloud Services and their hierarchy

infrastructure resources for a longer period of time is the cost [12]. However, the benefit of using infrastructure resources, is the elimination of initial hardware setup expenses. This makes infrastructure cloud services a great tool for short-term, highly scalable systems and for prototyping applications before migrating them to a dedicated HPC node [13]. On the other hand, HPC nodes offer computing resources at high initial costs but very little operational costs apart from electricity for operation and cooling, housing as well as maintenance. Nonetheless, virtualized hardware has the problem if interference from other guest virtual machines (VMs) on the same physical hardware. Since two or more customers with different application profiles are sharing the same hardware, the mixed-use calculation of the CP, would not allow for full performance of all components to all virtualized instances at the same time. In comparison with rented and shared resources from an IaaS CP, the performance of a dedicated system is deterministic. Applications are executed directly on the physical hardware (bare metal computing) without an abstraction layer for virtualization. The different computing nodes on an HPC are usually co-located and connected via high-bandwidth interfaces such as InfiniBand or 10-Gigabit Ethernet. This connectivity offers the advantage that high-bandwidth for distributed applications which use message passing interface (MPI) protocol can also be run [14].

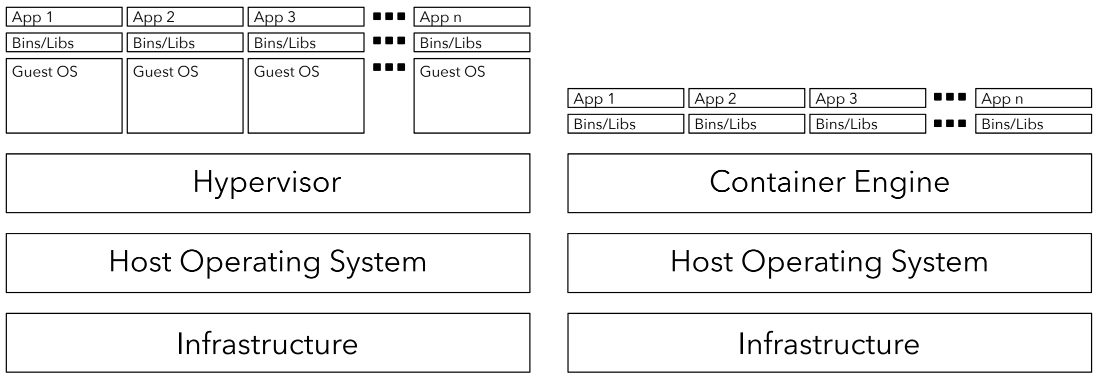
Platform as a Service (PaaS) can be implemented either on top of the IaaS or by using dedicated hardware instead of a virtual machine. These kinds of services are typically APIs that are used for varying use-cases. The Dropbox Datastore[1], as one example, can be considered a PaaS, because developers can use it to build applications that use a key-value-store to exchange small amounts of data. Similar services exist from other CPs like Apple, Microsoft or Google. Web-based email or online text-processing applications are prominent examples of Software as a Service (SaaS). There are also file storage solutions like Dropbox[2] or Google Drive[3] which are widely used. It is possible that different kinds of services are offered under the same name but to different target groups at different abstraction levels. The end-user can access SaaS via a web-browser, mobile applications or thin-clients with limited on-device processing power, whereas application developers can store data directly using PaaS API calls.

The *Anything As a Service* (XaaS) is a special representation of the traditional Services mentioned above with a very specific use case. *Simulation Software as a Service* is just one specific Software as a Service, while *Network as a Service* is a specific IaaS or PaaS, which depends on the amount of control and management over the actual infrastructure the user has.

2.1.3 Docker - Virtualization at Operating-System Level

In contrast to virtual machines, the use of containers to isolate different user-space instances of an operating system reduces the overhead that is introduced when virtualizing entire machines. These containers are a kernel feature of modern Linux kernels and allow processes and software stacks to have a virtual environment with its dedicated namespace. This, of course, adds different security concerns when deploying different containers on the same physical hardware and host operating system [15]. Unlike virtual machines, a container running on the same kernel as the host system has different security concerns to be taken into consideration. Docker [16] is an automation tool that uses different container technologies (e.g., LXC, libvirt and libcontainer) to create Docker containers that can be used to package and deploy pieces of software independent of the underlying host operating system. It is widely used in the cloud computing community for its ease of use in deployment of (micro-) services and its portability. All big CPs offer pre-configured images for their virtual machines to run Docker containers on. Since the processes running inside a Docker container have a very low overhead on native execution on the host OS, but offer a more heterogeneous computing environment on a single piece of hardware, the scientific community can have a great effect on such technology. These two differences in system layers can be seen in Figure 2.2.

[1]`https://www.dropbox.com/developers/datastore`
[2]`https://www.dropbox.com`
[3]`https://drive.google.com`



(a) Virtual Machines to wrap applications **(b)** Container to wrap applications

Figure 2.2: Virtual Machines and Container Abstraction Layers

Docker in the scientific community

The use of containers for emulating different real network components offers a great advantage over simulating and hardware test-bedding for reproducibility of results [17]. Handigol et al. envisioned a research community where published results can be reproduced and checked, which ultimately improves the reliability in the research conducted. Shekhar et al. [18] have introduced a cloud-based *Simulation as a Service* middleware concept, that can be used as a general purpose middleware for different kinds of simulation in order to bring them to the cloud. They have also identified complex system simulations and large search-space explorations as a viable use-case for cloud simulation. In addition to their work in comparing VMs to containers, their middleware concept also includes a scheduling architecture that allows a time- or resource-bounded experiment run. This has the advantage that, if the underlying simulation engine allows it, to either scale the experiment up or down depending on the immediacy of the deadline or the budget allocated for the experiments. The middleware uses the Docker container engine to rapidly replicate simulation experiments for stochastic simulations on different hosts and compared their results with VM-based executions. They also identified that the lightweight of a container to a full featured hypervisor is very appealing to the HPC community.

2.2 Agent-based Traffic Simulations

In order to gain knowledge about the traffic of a region (city or country), experiment within the traffic system have to be performed. This is not always easily possible since the sheer size of a traffic network makes it realistically unfeasible. There are two important problems that have to be addressed when trying to understand the traffic system of a region. One is the hard data that the traffic is producing. How many vehicles? How fast are they driving? From where to where are they going? The second is the complex interactions that all the individual entities in a traffic system are engaging with. When does a vehicle accelerate? When does it decide to overtake or to break? These are behavioral aspects of a traffic system that is influenced by human decisions.

If the traffic system is to be understood as well as experimented with, models of each individual component of the system need to be created. The models can later be combined to draw a big picture and can be used to recreate existing conditions for validation and verification, as well as to estimate how the system behaves when other models are introduced or changed.

This methodological approach from the real world to an abstraction is called modeling. A model is a formal description of a (complex) system. The model can be a mathematical equation, a flow diagram or a source code of a computer program, just to name a few. In order to "... build a model of a system, we need first to acquire knowledge of the system what we could translate in terms of assumptions." [19]. The assumptions are usually simplifications of the internal workings of the system. Depending on the assumptions we decide on for our model, the performance of the models will differ. For computational models, performance can be measured by the runtime or the memory usage, while in other disciplines, the correctness or error is used to measure performance of a model. Describing a ray of light only as a particle will give us a close enough estimation of where the ray of light will illuminate a surface, but the wave characteristics of the light have been totally ignored. This will lead to an error in the smaller scale of the system, but will be correct for a macroscopic experiment, where the microscopic characteristics are not important. This leads to the conclusion that there is not one unique model of a physical system but multiple, correct models for the given use-case and degree of knowledge of the modeler. When the people first turned their interest to the planets of our solar system, some thought the earth was in the center, while others thought the sun was. Both groups formed a model of their surroundings and made assumptions based on their knowledge and beliefs. The accepted model by today's standard is the helio-centric model of the solar system. However, this is only because there are more models supporting this theory than the geo-centric model. Thought, as well as, real experiments were used to support the idea of the helio-centric solar system. These thought-experiments were simulations that tried to answer "What happens if" questions. Later, they were then verified by real experiments. Due to the lack of 100 percent certainty in any model, every model "...will only be a partial representation of reality" [19]. The performance of a model is always relative to the real system, therefore these performance measures or utility functions help experimenters to evaluate the quality of the model within the use-case.

In today's research world, the use of computer simulations is inevitable and researchers have proclaimed that modeling and simulation is the "third paradigm of science by observers who think it adds something beyond the paradigms of experiment and theory" [1]. The mostly computerized simulations are becoming a "laboratory" for conducting searches of complex or even dangerous systems from the desk in an office. The "What-if" questions can now be answered in the span of seconds due to the advances in computational capabilities. These give researchers and engineers an invaluable tool for advancement in science and society.

In traffic simulations, the number of models are as different in complexity and performance as the vehicles or pedestrians on the streets of today's cities. There are four main groups of modeling vehicular traffic flow. Macroscopic, mesoscopic, microscopic and sub-microscopic models for describing traffic in cities or countries are used today. These are explained in more detail in Section 2.3.

2.3 Traffic Simulations

Traffic simulations can be classified into macroscopic, mesoscopic and microscopic simulations. Some classifications even define nanoscopic or sub-microscopic simulations. This section will give an overview of the different model characteristics and some existing implementations and tools.

2.3.1 Macroscopic Traffic Simulations

Macroscopic modeling uses the analogy of hydrodynamics to describe the traffic flow in a given road segment. This means the traffic is considered a particular type of fluid process. The defining characteristics are the density $k(x,t)$, the volume of traffic $q(x,t)$ and the speed $u(x,t)$ as the average speed of vehicles on a given road segment. Each of the characteristic values are defined for the time t and every point in space x. Gerlough and Huber [20] as well as Kühne et al. [21] have proposed a continuity equation that links the characteristic values together:

$$\frac{\delta q}{\delta x} + \frac{\delta k}{\delta t} = 0 \tag{2.1}$$

[1] `http://www.educause.edu/ero/article/scientific-research-how-many-paradigms`

$$q(x,t) = k(x,t)u(x,t) \tag{2.2}$$

It assumes that between two points where vehicles are counted, no vehicle is entering or leaving the road segment. There have been extensions to the basic macroscopic models within the late 1980s and throughout the 1990s. Many of these models have been collected and explained by Barcelo [19].

Examples of traffic simulations that use macroscopic models are:

- **Aimsun**: Enables users to have simultaneous macroscopic and microscopic simulations running at the same time. This allows users to define areas of the road network to see in higher microscopic or lower macroscopic resolution, depending on the use case[1].
- Many researchers implement macroscopic traffic models themselves, since the basic models only consist of the above equations.

2.3.2 Mesoscopic Traffic Simulations

In mesoscopic traffic flow modeling, each vehicle is modeled in detail, but the behavior is described in an aggregated manner [22]. These plateaus or groups of vehicles are considered to be homogeneous in type and behavior, but within the simulation there can be different kinds of vehicles with different behaviors. The limitations and advantages of mesoscopic models in comparison to microscopic and macroscopic models are shown by Burghout [23].

Examples of traffic simulations that use mesoscopic model are:

- **DYNAMIT**: It is designed to offer a simulation-based approach to estimate the current and future state of a transportation system. It uses historical and surveillance data to predict future traffic conditions [24]. The simulation engine uses a mesoscopic model of the vehicles' speed-density-distribution, as well as a queuing model for the interaction of vehicles.
- **MEZZO**: It integrates a mesoscopic model with a microscopic model to form a hybrid simulation model. The use of a mesoscopic part of a link and a queuing model before an intersection (microscopic part) simplifies the turning behavior of vehicles in large networks [23].
- **MatSim**: Apart from having a mesoscopic traffic flow model it includes a microscopic demand model [25].

[1]`http://www.aimsun.com/`

2.3.3 Microscopic Traffic Simulations

In microscopic traffic-flow modeling, each individual vehicle's motion is regarded and modeled in respect to other vehicles on a given road segment. Over the years there have been many different models that involve the flow and behavior of individual vehicles. One of the most-used approaches toward modeling traffic flow microscopically is the application of agent-based modeling and simulation techniques. This allows the simulation scenario to have each individual vehicle in the traffic network assigned with different models that behave differently in situations. One of the goals in creating a more realistic traffic scenario is a heterogeneous agent population. It can be achieved by implementing numerous models for each agent or to parametrize and stochastically distribute the input parameter in a certain range such that agents behave similarly but not exactly the same. This process of model selection and parametrization not only functions well for creating models for freeway traffic scenarios but also for inner city traffic scenarios as explained in detail by Barcelo [19]. Two of the more commonly used models are the *Intelligent Driver Model* (IDM) [26], which is explained in more detail in Chapter 5, and the *Gipp's model* [27], which pioneered in factoring in different behaviors like the desired speed of individual vehicles. For agent-based simulations, microscopic simulations are mostly used, because each vehicle's behavior, perception and movement can be translated into an agent with states and state-transfer algorithms.

Examples of traffic simulations that use microscopic models are:

- **SUMO**: The *Simulation of Urban Mobility* offers, in addition to detailed vehicle models, pedestrians as well as public transport system within the same network. It is shipped with a user interface to load existing road networks as well as generate standard artificial networks (e.g., grid, circular, random)[1].

- **Aimsun**: Since it enables users to use macroscopic as well as microscopic models at the same time in a user defined region of the network, it can be considered a microscopic simulation software.

- **SimMobility**: The simulation suit comprises of different modes from an activity-based microscopic simulation that works within a day's time frame to mid-term and long-term simulation. The mid-term and long-term simulation tools are used hierarchically, generating input to the lower models[2].

[1]`http://sumo.dlr.de/`

[2]`https://its.mit.edu/research/simmobility`

- **ViSSim**: The market leader in microscopic traffic flow simulation includes high resolution models for vehicles, public transport, cyclists and pedestrians. It is shipped in an end-user-friendly interface to design, observe and evaluate different simulation experiments[1].

2.3.4 Sub-Microscopic Traffic Simulations

In sub-microscopic traffic flow modeling the model granularity is even higher in comparison to microscopic traffic models. This is sometimes also called *nano-scopic* models, since the individual components of the vehicles are also modeled and simulated individually and not entire the vehicle as one monolithic entity. In traffic simulation, one can differentiate between component models of the individual components in and outside the vehicle, as well as the behavioral model of the driver that influences such models.

Example traffic simulations that use nanoscopic models are:

- **SEMSim Traffic/CityMoS**: Traffic simulation that uses component models of vehicles as well as individual driver behavior models to build up agents in a lane-level nanoscopic simulation. A detailed description follows below, since SEMSim traffic (now CityMoS) is used in almost all use cases throughout this thesis.

- **PELOPS**: It combines sub-microscopic models of vehicles with microscopic traffic flow models to analyze longitudinal dynamics of vehicles together with traffic flow [28].

2.3.5 SEMSim Platform

The SEMSim platform is used in the context of electromobility research and aims at helping to answer various questions regarding the impact of large-scale electro-mobility on a city's infrastructure. With large-scale electro-mobility, it is meant to replace all (or a majority of the) vehicles in a city. Although there are test fleets of electric vehicles all over the world, there are only a few cases of a city where electromobility is used on a large scale. This presents a problem for policy makers and city planners because there are no reference cases available. For that reason, one approach is to use a holistic electromobility simulation platform that allows us to answer some of the important questions in this context. The scope of this research is currently limited to the case of Singapore. The simulation methods developed as part of this project can also be used with other cities.

[1]`http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/`

The extension of the SEMSim platform is City Mobility Simulator (CityMoS). It extends the functionality of SEMSim traffic to include more aspects of urban mobility. The additions include the simulation of public transport options like buses and other road-based transport vehicles, as well as the coupling capabilities with communication network simulation engines such as veins [29] for the evaluation of smart-city scenarios.

Unlike existing research, this approach aims at studying large-scale electro-mobility in a holistic manner, i.e., not investigating individual sub-systems (e.g., power system, traffic system, vehicle system) in isolation but in a holistic simulation that allows to investigate the complex interactions and inter-dependencies between the various sub-systems and their components. For this purpose, many different simulations can be combined or coupled together in order to use the best technique for modeling a certain behavior. An example in the domain of electromobility research is the coupling of an agent-based traffic simulation (SEMSim Traffic) and a discrete-event power system simulator [30]. The main question in creating simulation scenarios that not only exchange information unidirectionally but in two ways is the data-exchange mechanism. One standardized way of coupling various simulation entities together is using the High Level Architecture (HLA) [31]. This allows the investigate of the impact of thousands of electric vehicles on the power grid depending on their charging behavior. Figure 2.3 illustrates the SEMSim platform with its major simulation components. In order to couple the traffic simulation with a communication network simulation the TraCi protocol is used [32]. It is used with other traffic simulation platforms and CityMoS can therefore act as a drop-in-replacement for existing simulation tools, while offering the simulation capabilities for other aspects of urban mobility.

2.3.6 Yet Another Traffic Simulation?

Since the SEMSim Platform is designed to answer electromobility-related questions for mega-cities using multiple tools like SEMSim Traffic and SEMSim Power, the computational needs are significant. One major difference between SEMSim\CityMoS and the existing traffic simulations is the level of detail at which the internal state of vehicles is simulated. In CityMoS, every agent represents a driver-vehicle unit which consists not only of driver-behavior models but also vehicle component models (e.g., air conditioning, drive train, battery). CityMoS is a simulation that has a lane-level resolution which indicates that it is a microscopic simulation. However, due to the fact that the internal state of vehicles is also simulated, it is referred to as

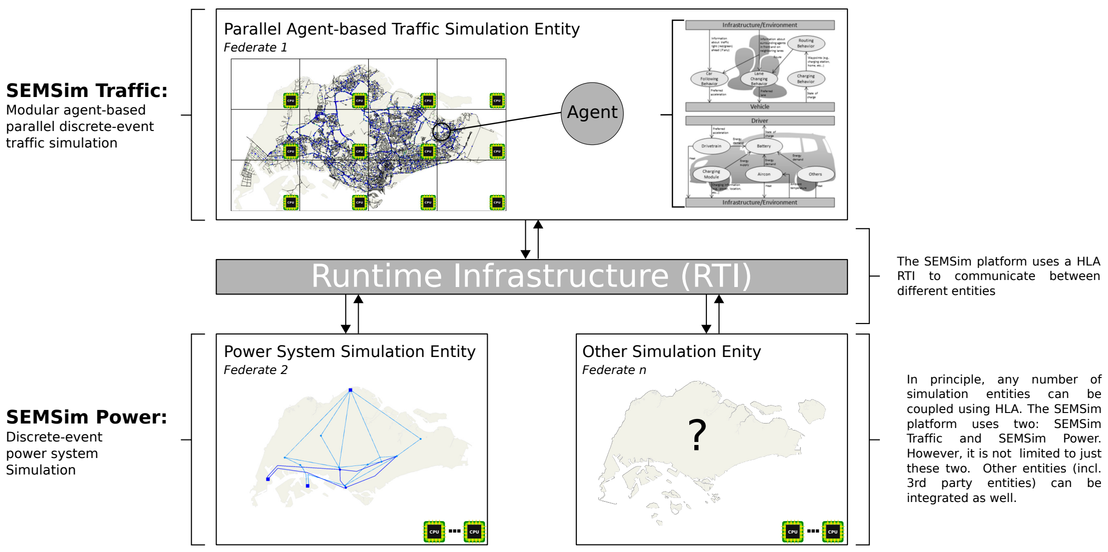

SEMSim Traffic:
Modular agent-based parallel discrete-event traffic simulation
Parallel Agent-based Traffic Simulation Entity
Federate 1
Agent
Runtime Infrastructure (RTI)
The SEMSim platform uses a HLA RTI to communicate between different entities
SEMSim Power:
Discrete-event power system Simulation
Power System Simulation Entity
Federate 2
Other Simulation Enity
Federate n
?
In principle, any number of simulation entities can be coupled using HLA. The SEMSim platform uses two: SEMSim Traffic and SEMSim Power. However, it is not limited to just these two. Other entities (incl. 3rd party entities) can be integrated as well.

Figure 2.3: Overview of simulation entities that are part of the SEMSim Platform

a nanoscopic simulation in order to differentiate it with regards to existing microscopic simulations. Unlike existing simulations tools (e.g., MatSim [33], SUMO [34], DynaMIT [35]), where some use a fixed time-stepped simulation engine, we use a discrete event simulation engine. In a discrete event simulation (DES) engine, the state of the simulation is only advanced when an event is executed (as compared to every time step in a time-stepped simulation). Events in a DES can be scheduled at arbitrary times; this provides the needed flexibility as some models need more frequent updating (e.g., moving of agents on the road network) while others are updated only infrequently (e.g., internal vehicle models or behavior). Figure 2.4 illustrates the various sub-systems and components that are part of the SEMSim platform and in particular, SEMSim Traffic. Due to the high computational requirements of simulating a large number ($> 100k$) of agents and their interaction in a nanoscopic simulation, the execution environment for the SEMSim platform is a high-performance computing cluster that supports highly parallel execution. Workstation computers, as are used by other simulation tools, are not capable of delivering simulation results quickly enough for a megacity-sized experiment. The simulation resolutions of MATSim (meso and macroscopic, see Section 2.3.2), SUMO (micro and mesoscopic, see Section 2.3.3) and DynaMIT (mesoscopic, see Section 2.3.2) are not in the scale of nanoscopic simulation that SEMSim Traffic is offering but are necessary for researching the interactions between vehicles and other simulations of a complex system. The simulations are also not designed to be run on a multi-process/-core execution environment that is crucial for a

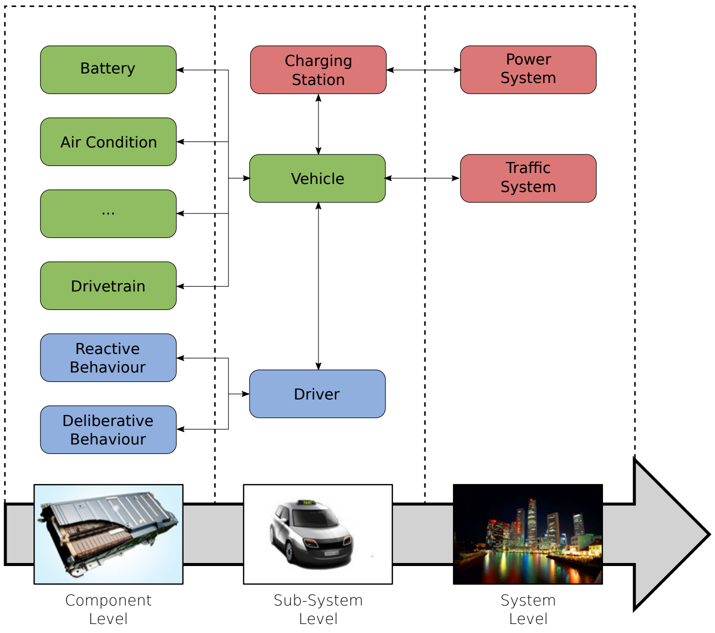


Figure 2.4: Overview of sub-systems and components that are modelled as part of the SEMSim Platform

fast execution on a HPC. The target user-group for the SEMSim Platform are traffic researchers and policy-maker advisors. These user groups have no, or only little, expertise in executing such complex simulation experiments and usually have no access to high performance computing hardware. Therefore, the SEMSim Platform aims to include Cloud Computing capabilities that make it easy to configure, setup and execute simulation experiments to study large urban systems. SEMSim Platform is not a single simulation tool. Instead it is a set of individual simulations, middlewares and UI tools. Among these tools, there are currently SEMSim Traffic and SEMSim Power, which allow to study the interactions between the urban transportation system and a corresponding power system. This is made possible through coupling these individual tools by using HLA. The SEMSim Platform thus provides various tools which can be used depending on the requirements of the simulation study or application that needs to be realised. This differentiates SEMSim Platform in the sense that it is not a monolithic simulation tool in the back-end but looks and feels as one to the user.

Due to the nanoscopic modelling approach in SEMSim Traffic and the fact that we simulate a realistic number of agents, the computational requirements for a megacity like Singapore are high. In Singapore, there are more then 500,000 privately owned vehicles [1] which produce an approximate peak-hour traffic of 100,000 vehicles. This means that the simulation needs to be able to simulate the same magnitude of agents. Because of the computational requirements, SEMSim Traffic has been specifically designed for HPC systems. In order to simulate a realistic amount of agents, the simulation has to be performed either on a dedicated HPC or the cloud. For this reason, most of the examples and use-cases in this thesis will focus on SEMSim Traffic and the SEMSim Cloud Service. Chapter 3 will introduce this cloud service in more detail as a proof-of-concept implementation of the reference architecture envisioned in the same chapter.

2.4 Multi-Resolution Modeling and Simulation

Within the following section, the work of other researchers in the field of multi-resolution modeling and simulation is reviewed. The focus will be on applications concerning traffic simulation models.

Multi-resolution modelling and simulation was, amongst others, introduced when different kinds of military-related simulations should work together. The coupling was done using the High Level Architecture (HLA). The different simulations were also on multiple resolutions. Battlefield generals wanted to have a different view on the simulated battlefield than a commander of a tank division. Aggregation and disaggregation of simulations was necessary and are, until today, the biggest challenge in multi-resolution modeling and simulation. Other fields, where multi-resolution modelling and simulation is used are, amongst others, bio-medical systems [36] or train control system [37].

2.4.1 Multi-resolution in traffic simulation

There have been several approaches to multi-resolution modeling of traffic simulations. Two main approaches towards the topic have been (1) the use aggregation and disaggregation at runtime and (2) the use of multi-resolution entities.

The first approach, which aggregates a high resolution model at runtime to a lower resolution and then disaggregates from the low resolution to a high resolution, is trivial on the aggregation side. There is, in most cases, an aggregation function (e.g., SUM, AVG) that can be used to aggregate existing high-resolution agents' states. The disaggregation algorithm on the other hand, has to interpolate or create/reconstruct information from the low-resolution model

[1] `http://www.lta.gov.sg/content/ltaweb/en/publications-and-research.html`

in order to obtain the state information for the high-resolution model. It has been shown that a frequent transition between high- and low-resolution models through aggregation and disaggregation is not good practice [38]. One problem is the *thrashing effect*, where agents cross between resolution boundaries often. This can create a large overhead in computation, especially when running a distributed simulation due to synchronization between non-local nodes.

This method of aggregation and disaggregation was initially developed for military simulations [39, 40], since it follows very well with the command security structure in the military. Battalions, for example, are disaggregated divisions and allow a manageable view of certain battlefield scenarios for different key personnel in the chain of command, whereas a disaggregated view of individual fighters is necessary for group leaders.

An approach to multi-resolution traffic modeling was discussed by Burghout [41, 22], where a *ghosting* method was used to ease the transition between microscopic and macroscopic boundaries in the aggregation-based approach.

In order to avoid the aforementioned *thrashing effect* of frequent model-switching, Natrajan et al. [42] proposed the concept of *Multi-Resolution Entities* (MRE). A MRE consolidates the properties of several resolutions in one object. Those properties are then kept consistent by design. This requires a larger memory footprint and individual operations require more computing cycles for each of the resolutions, but there is no consistency problem.

Another approach to mitigate the *thrashing effect*, especially for traffic simulations, while avoiding the use of MRE, has been presented by Chua and Low [43], proposing a set of predictive algorithms. These algorithms are used to project an agent's future position on a road segment.

The multi-fidelity modeling approach presented by Choi et al. [44] describes generally how to convert an existing model into a multi-resolution model. This methodology can be used to increase the simulation speed for a given simulation as well as to give measures for fidelity derivation. While their approach generalizes a methodology, within the relevant chapters of this thesis, the focus is on using multi-resolution modeling to investigate its potential in regard to usage in high-performance traffic simulations, mainly SEMSim\CityMoS.

Chapter 3

Simulation Cloud Service

Contents

Large parts of this chapter have been published in the *Simulation Modelling Practice and Theory* Jornmal in a Special Issue on Cloud Simulation [45].

As stated in the introductory chapter, urban processes have been the subject of many simulation studies in the past. This includes topics that cover diverse aspects of urban life, such as climate science (e.g., urban heat island effect [46]) , energy studies (e.g., smart grids [47] or vehicle-to-grid [48]), health (e.g., pandemics [2]), social science (e.g., crowd evacuation [49]), and transportation (e.g., public transport [50] and traffic management [51]), to name only a few. Agent-based models are commonly used for simulating urban processes such as transportation and are often the only feasible way to study the urban systems of interest due to emerging behaviors of complex systems. With an increasing worldwide interest in city science and research, we can expect to see more of such simulation studies in the future.

Large-scale agent-based simulations, i.e., simulations that include several hundred thousand agents, can be compute-intensive which is the reason why they are ideally performed on High-Performance Computing (HPC) systems. Domain experts which are mainly but not limited to transportation engineers, who are the typical users of simulation tools, may not have access to in-house HPC resources and/or may not have the necessary skills to work with an HPC system. With the onset of cloud computing, this situation is rapidly changing. HPC resources are now readily available via cloud computing providers such as the GCE[1], Amazon EC2[2] and others. Given the availability of HPC resources, it is expected that more cloud-based simulation services will emerge. It has been shown [52] that the move to using cloud-based simulation hardware can offer opportunities for vertical and horizontal scaling of hardware resource usage at runtime and is an interesting field of research that will serve as the foundation of the following chapters.

With large-scale simulations being migrated to the cloud, users will have to face new challenges including (but not limited to):

- *Usability*, where graphical user interfaces (GUIs) of many simulation software tools have been designed for use on workstations by a single user. Although existing user interfaces can be re-used even if the simulation is executed in the cloud, it would be better to think about more suitable user interfaces that are specifically designed for cloud-based simulation services. User interfaces should be able to communicate through a well-defined API instead of with the simulation directly. This enables many different user Interfaces which can include GUIs for different aspects of the same simulation (e.g., configuration, visualization, data analysis).

[1] `https://cloud.google.com/products/compute-engine`
[2] `https://aws.amazon.com/ec2`

- *Data Confidentiality*, since virtually all urban systems simulations rely on a large amount of data (e.g., road network data, population data, traffic data, cell phone activity data). This data may be sensitive and data providers may not permit the use of cloud-services due to concerns regarding confidentiality. This issue can be addressed using standard data encryption techniques to ensure that RAW data is never stored on external servers.

Within this chapter, a cloud-based simulation service is introduced. It addresses the above-mentioned issues and presents the Scalable Electromobility Simulator Cloud Service (SEMSim CS), which is a proof-of-concept implementation of the proposed reference architecture. SEMSim CS is used in the context of electromobility research for answering questions regarding the impact of introducing large numbers of electric vehicles into an existing transportation system. For such use case, a comparison in simulation runtime, between dedicated hardware as it is found in HPC centers and VMs that are acquired when using cloud resources, is performed.

3.1 Related Work

In addition to the general related work section in Chapter 2, in this specific section, the existing research surrounding the development of a simulation cloud service is discussed.

In this section, the current state-of-the-art of simulation front-end applications is presented. Two traffic simulation engines will be focused on: *Multi-Agent Transport Simulation* (MATSim) and *Simulation of Urban Mobility* (SUMO), because they are microscopic traffic simulations similar to SEMSim Traffic\CityMoS. Additionally, for both simulation platforms the source code is available, which would allow interested parties to extend the initial capability and convert them into cloud-ready simulations. MATSim does not offer a standard graphical user interface. The main configuration method is using configuration files that are loaded at the start of the simulation program. For outputting, MATSim writes the simulation results into files and these can then be visualized using external visualization tools. Having a command-line interface allows MATSim to be run on various hardware configurations, including headless server hardware as it can be found in HPC or cloud environments. The *Simulation of Urban mobility* (SUMO) comes with a graphical user interface as well as a command-line interface. SUMO can, similar to MATSim, can be configured using a series of input files. Since it can be run in command-line-only mode and configured using configuration files, it can be executed on a headless HPC node or cloud VM. When running in GUI mode, the simulation output can be seen in the program window. It also offers a number of post-processing and plot-generation scripts. SUMO's current GUI could be adapted to work with a cloud-based simulation when changing the input method to the GUI, from direct input from the running simulation, to an stream-based method from a middle-ware or the cloud-based execution itself.

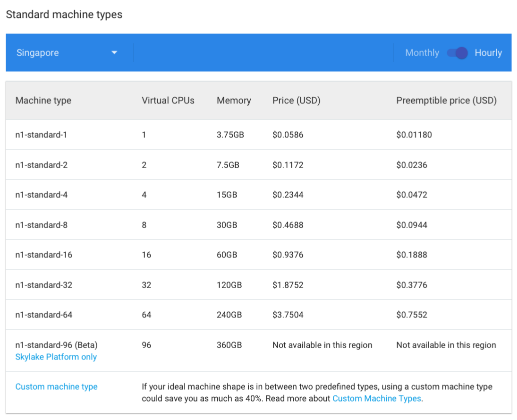
In terms of data confidentiality for simulation data, neither of the comparable simulation tools (e.g., MATSim or SUMO) offer any built-in data de-/encryption functionality for input or output data. For this reason, the use of SEMSim traffic (now CityMoS) as a basis for a traffic simulation cloud service is favorable, since it offers support for AES-encrypted input data as well as secured output data, using AES-encryption for persistent-storage output and transport encryption (e.g., TLS) for steaming output. In addition, it also offers, similar to the two other simulation engines, a headless execution mode with file-based configuration. Together with the SEMSim platform tools which provide a RESTful APIs to generate such input files with any kind of (graphical) user interface, SEMSim traffic can be easily configured.

The CityMoS application architecture allows the simulation to be executed with or without a 3D graphical interface. The underlying models and configurations are exactly the same. The graphical components also use internal APIs to obtain data from simulation entities.

3.2 General Architecture for Cloud-Based Simulation Services

The purpose of a cloud-based simulation service is to migrate the simulation software into the cloud and provide users with appropriate graphical user interfaces and APIs. A cloud-based simulation service as described here can be classified mostly as SaaS with only some components of the architecture that can be classified as PaaS (compare with Figure 2.1 on Page 14). One notable difference between using dedicated HPC machines and virtual machines in the cloud is that users of a dedicated HPC machine know the exact hardware configuration as well as the location of the data center that hosts the HPC machine. Generally, this is not the case with infrastructure provided by cloud-based VMs. Although IaaS providers provide information regarding geographic locations for load-balancing and fast connectivity, the information is mostly on a (sub-)continental level. Therefore, users usually only have an vague idea of where the VMs are located and idea of the type of physical resources they run on. In Figure 3.1, it can be seen that the prices for different pre-configured machines are increasing linearly with the capabilities (e.g., memory and CPU). It can also be seen that the hardware between different regions are not homogeneous. While the Singapore region (asia-southeast1) will only use

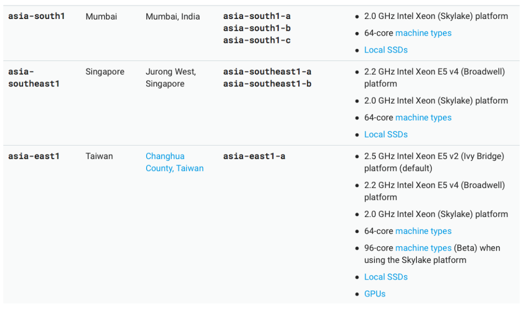
Broadwell and Skylake physical hardware, the Taiwan region (asia-east1) has CPUs from Ivy Bridge, Broadwell and Skylake family. Additionally, the maximum CPU (currently 96) is not rolled out over all regions.


Standard machine types

Singapore | Monthly | Hourly

Machine type	Virtual CPUs	Memory	Price (USD)	Preemptible price (USD)
n1-standard-1	1	3.75GB	$0.0586	$0.01180
n1-standard-2	2	7.5GB	$0.1172	$0.0236
n1-standard-4	4	15GB	$0.2344	$0.0472
n1-standard-8	8	30GB	$0.4688	$0.0944
n1-standard-16	16	60GB	$0.9376	$0.1888
n1-standard-32	32	120GB	$1.8752	$0.3776
n1-standard-64	64	240GB	$3.7504	$0.7552
n1-standard-96 (Beta) Skylake Platform only	96	360GB	Not available in this region	Not available in this region
Custom machine type	If your ideal machine shape is in between two predefined types, using a custom machine type could save you as much as 40%. Read more about Custom Machine Types.			

(a) Pricing for different standard predefined VMs on the GCE for the Singapore region (Feb 2018).



asia-south1	Mumbai	Mumbai, India	asia-south1-a asia-south1-b asia-south1-c	• 2.0 GHz Intel Xeon (Skylake) platform • 64-core machine types • Local SSDs
asia-southeast1	Singapore	Jurong West, Singapore	asia-southeast1-a asia-southeast1-b	• 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform • 2.0 GHz Intel Xeon (Skylake) platform • 64-core machine types • Local SSDs
asia-east1	Taiwan	Changhua County, Taiwan	asia-east1-a	• 2.5 GHz Intel Xeon E5 v2 (Ivy Bridge) platform (default) • 2.2 GHz Intel Xeon E5 v4 (Broadwell) platform • 2.0 GHz Intel Xeon (Skylake) platform • 64-core machine types • 96-core machine types (Beta) when using the Skylake platform • Local SSDs • GPUs

(b) Hardware configurations and limitations in different locations (selection) on the GCE (Feb 2018).

Figure 3.1: Pricing structure and region hardware availability on GCE

3.2.1 Components

The proposed architecture of a cloud-based simulation service consists of front-end applications (i.e., user interface applications) that can be used by the end-users to use the simulation service and back-end applications which provide the necessary logic that realizes this service. In general, all front-end applications interact with the back-end by means of corresponding Application Programming Interface (API). The separation between front-end applications and back-end applications is a notable difference between cloud-based simulation software and a workstation-based simulation software where the user interface and simulation engine often come as monolithic applications (e.g., Arena[1]). Due to this separation, it is possible to develop entirely new concepts for user interfaces that are not limited to the traditional user interfaces used by standalone workstation applications. Figure 3.2 illustrates an overview of the proposed architecture.

The front-end consists of the following components:

[1]`https://www.arenasimulation.com/`

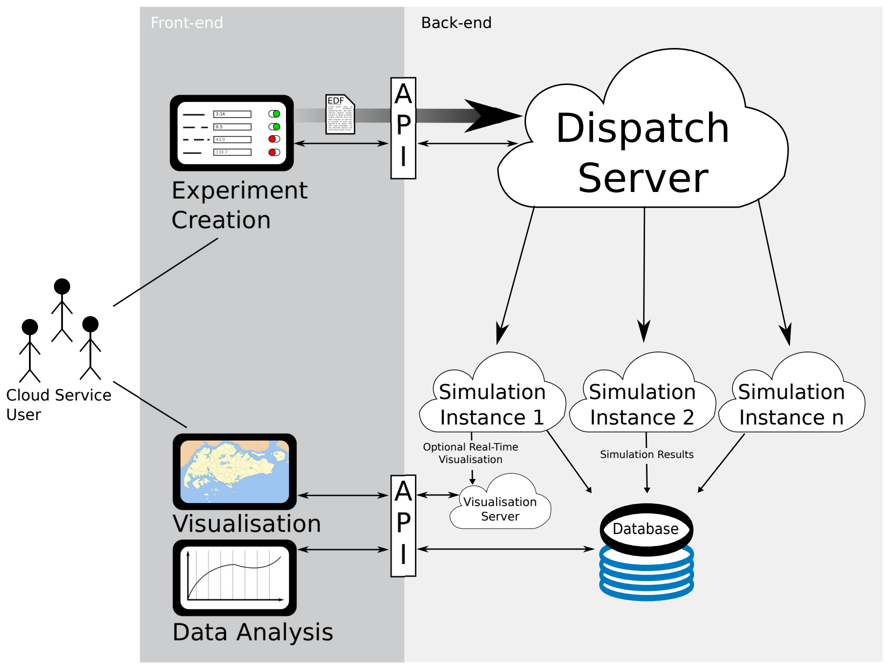


Figure 3.2: Overview of proposed architecture for a cloud-based simulation service

- *Design-of-experiment applications* are used by the user to specify the experiment that is to be performed. This may include specification of a number of experiment parameters as well as specifying which data sets and what models to use in order to perform the simulations. The design of experiments is highly use-case specific. In general, this kind of application needs to generate an *Experiment Description File (EDF)* which contains all the necessary information and data (if applicable) required in order for the back-end to perform the simulation experiment.

- *Visualisation applications* are used to perform real-time visualisations of a currently-running simulation instance. This can be useful for troubleshooting and to gain a better understanding of the current state of the simulation. Depending on the simulation, real-time visualisation is likely to be a performance bottleneck not only because of (possibly) large amounts of data that need to be transferred but also because the simulation instance may have to be temporarily throttled. This is necessary in order to better understand the progress in a faster-than-real-time simulation. One way of accomplishing such inter-activity with high performance simulations running on cluster hardware is to decrease the execution speed when connected with a visualisation client and increasing the speed when running headlessly. One way of reducing the performance bottleneck is the usage of a smart middleware and/or online data-extraction techniques as described in Chapter 4.

- *Analysis applications* are used to analyse the simulation results. Similar to the design of experiment applications, analysis applications are highly use-case specific. Depending on the purpose of the simulation and the aim of the experiments, the analysis process differs from case to case. Here, too, the online data-extraction middleware described in Chapter 4 can be applied.

The back-end consists of the following components:

- A *Simulation instance* executes a single simulation according to the specification provided in the EDF. Depending on the underlying simulation software, a simulation instance may or may not provide certain features. For example, some simulation tools may not provide real-time visualisation capabilities. Since an experiment may require an arbitrary number of simulation runs, each simulation run is executed in a separate VM. When using containerized execution (e.g., Docker) of simulation experiments, multiple simulation instances can be executed on a single physical or virtual hardware, should the resources allow it. This is useful for small experiments that need to be run multiple times to increase the confidence in the results generated.

- A *Dispatch Server application* is responsible for executing a simulation experiment according to the specifications in the EDF. Generally, this is done in the following steps: the dispatcher (1) provisions a number of VMs or containers in the cloud each of which will host one instance of the simulation software (simulation instance), (2) initializes the run environment (i.e., copies files, setup the run-time environment, etc.), (3) starts execution of the simulation instance in the run environment, and (4) notifies the user when the experiment has been completed. It should be noted that an experiment may consist of an arbitrary number of simulation runs, each being executed in its own simulation instance. While this is true for non-distributed simulation, distributed simulation might require multiple simulation instances to form a single simulation-experiment run.

- A *Visualisation server application* enables a specific simulation instance to connect to it and the real-time data stream is then forwarded to the visualisation application in the front-end. In addition, it relays control commands from the visualisation application to the simulation instance. For example, this can be used to control the visualisation (e.g., slowing the simulation down or speeding it up). It can also be used to manipulate objects and parameters in the simulation and thus realize sophisticated interaction between the user and the simulation. In order to reduce complexity and confusion between multiple connected clients, one client connected to the visualisation server application has to become the master-client, which can then interact with the simulation, while all other connected clients have access to a read-only simulation output from the simulation instance. This can be realised by using a middleware as explained in Chapter 4, a simulation specific solution like HLA or a simple publish/subscribe solution.

3.2.2 Workflow

In general, the workflow can be distinguished between two use cases: (1) manual use-case and (2) automated use-case.

The workflow of the manual use-case starts with the user designing an experiment and submit an EDF to the dispatching server in the cloud. The *Dispatch Server* interprets the EDF and performs the necessary tasks in order to start the various simulation instances. Once the simulation instances are running, the user may connect to a simulation instance for real-time visualisation and/or to interact with the simulation run. Depending on the simulation type, the output of the simulation run is directly stored in a database. This database may reside on the user's site or be provisioned from a CP's PaaS portfolio. Once the experiment is completed, the user is notified. By using an analysis tool, the user can analyze the simulation results. The analysed data can then be used as an input for an adjusted experimentation run. Figure 3.3 illustrates the workflow of the manual use-case.

In principle, the workflow of the automated use-case is the same as for the manual use-case. The difference is that the user of the simulation service is not a human user but yet another software tool which generates experiments algorithmically. For example, this can be used to perform simulation-based optimization by employing an optimization algorithm (e.g., evolutionary or genetic algorithms) that works in an iterative manner. For each iteration, an experiment is defined according to the specifications provided by the algorithm (utility or fitness function).

3.2.3 Data Encryption Mechanism

For a cloud-based simulation service, it is important that the confidentiality agreements with data providers for input data are respected by the user. On the other hand, the generated

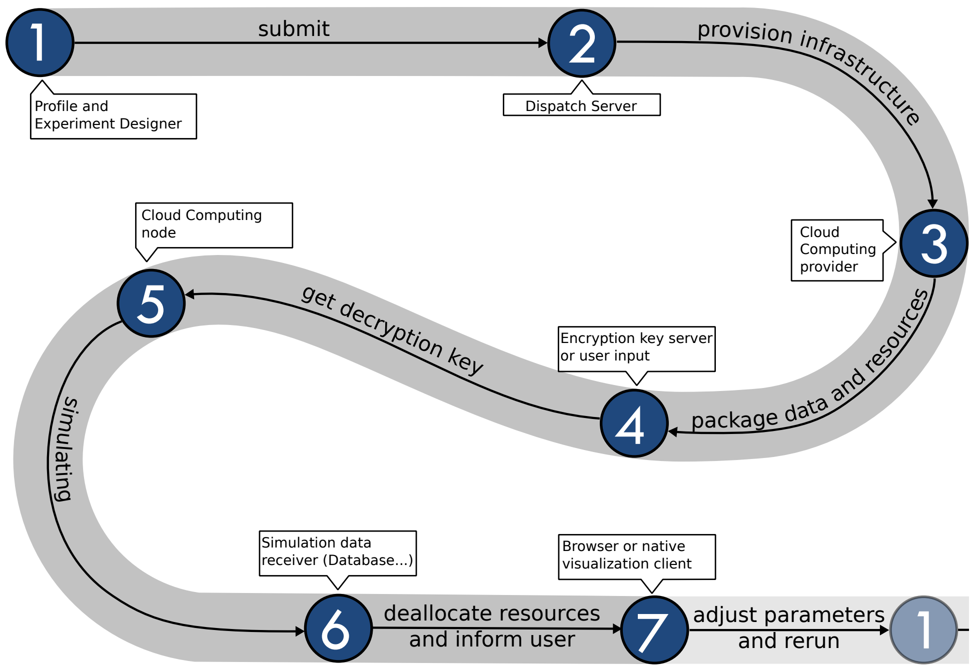

1
submit
2
provision infrastructure
Profile and Experiment Designer
Dispatch Server
3
Cloud Computing provider
package data and resources
4
Encryption key server or user input
get decryption key
5
Cloud Computing node
simulating
6
Simulation data receiver (Database...)
deallocate resources and inform user
7
Browser or native visualization client
adjust parameters and rerun
1

Figure 3.3: Overview of general workflow for the manual use-case.

simulation data and results are of importance to the experimenter. Therefore a simulation cloud service should use secure transfer mechanisms (e.g., TLS [53]) to communicate between cloud computing nodes, the user and the dispatch server. All input data should be encrypted and a secure channel, to either a key-server or a client device (e.g., mobile handset, browser) of the researcher, is opened to allow reliable key exchange (e.g., elliptic curve Diffie-Hellman [54]). The decrypted input data will only exist in the main memory of the underlying resources and will never be available in its unencrypted form on the persistent storage. As in any connected system, there are attack vectors that a potential malicious entity could use, but they are more complex than reading plain text data from persistent storage. This is primarily done for input data since the output data is streamed to a visualisation client or stored in a database. Database connections by all major database providers are encrypted and HTTPS connections between the visualisation server and the actual visualisation client are used. If local data is generated, the same key is used for decrypting the input data is used to encrypt the resulting data before it is written to a file on the cloud node's persistent storage.

A workflow of decryption is depicted in Figure 3.4. When the simulation application starts after the simulation bundle (collection of input data and EDF) has been opened, the first step of the application is to read in the EDF and find out which data needs to be decrypted. The

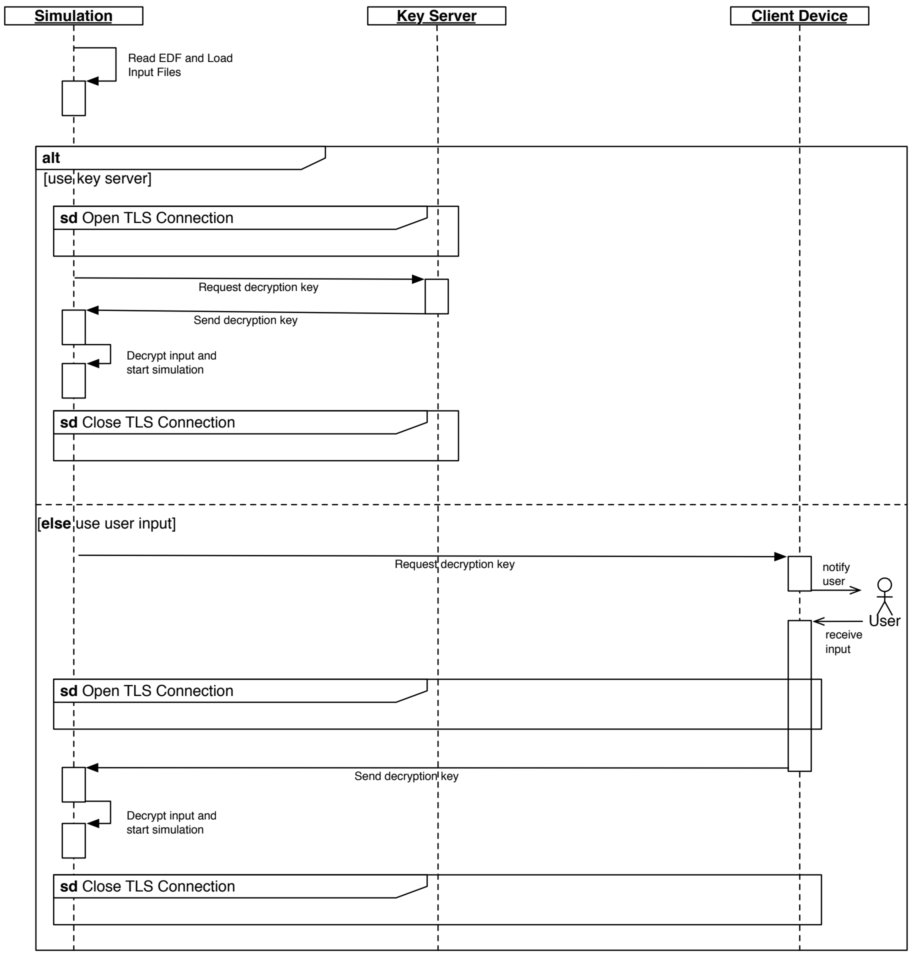


Figure 3.4: Simulation Data Decryption Workflow

key acquisition method is given in the EDF. It can have two options. The first one includes a key server which serves out keys to decrypt the input data (Figure 3.4 left side). For this method, no user interaction is needed and it can also serve as a method of limiting access to foreign resources (e.g., number of accesses, group access policies). The second option is to notify the experimenter of a simulation run and request for him/her to enter the decryption key into a client device application or a website (Figure 3.4 right side). Both the connections to the key server and from the user's client device are required to use TLS for establishing a secure connection. After the keys have been received successfully, the input files are decrypted, the actual simulation is started and the secure TLS connection is closed.

3.3 SEMSim Platform

3.3.1 Front-end Applications

SEMSim (now CityMoS) Cloud Service (CS) provides a series of front-end applications. The general idea behind the user interface concept of SEMSim CS is to make use of modern input devices (e.g., devices with touchscreen) that are not limited to typical workstation applications and standard input methods. The goal of SEMSim CS is to make all technical aspects of the HPC-based simulation in the cloud transparent to the user (typically a domain expert with limited experience in working with HPC environments).

- *Design-of-Experiment Applications:* Design of experiments is concerned with two parts – the Profile Designer (PD) and the Experiment Designer (ED). One part is the parametrization of the models used within a nano-scopic traffic simulation. For this, the PD application is used. Figure 3.5 shows the PD where the Driver-Vehicle-Unit (DVU) models are parametrized in a tree-like structure as an example user interface. More use-case specific user interfaces can be thought of. The model parameter can either be a fixed value or a distribution from which a value is drawn when the specific model instance is created. These profiles are stored in a repository and will be used when creating the actual experiment. This is done by using the ED application shown in Figure 3.6. The ED application is divided into two sections. In the upper section of the application window, general experimentation parameters are configured (e.g., road network, number of agents, simulation time span); whereas in the bottom part, the agent population is configured. The output of the ED is an EDF which is a configuration that serves as input to the simulation experiment. This configuration uses the profiles created in the PD to specify the different combinations of driver-vehicle-unit profiles with ratios on the total population in order to create a heterogeneous agent population.

- *Visualisation Applications:* SEMSim (now CityMoS) CS provides a set of visualisation applications. The individual applications can be classified as browser-based and touch-based. Each type has different features. For example, the browser-based visualisation allows the user to connect to a running simulation instance and provides features such as coloring agents by a given criterion. This should give a broad overview of the simulation's state. For performance reasons, showing an entire map with several thousand agent representations with all state information is not possible in the browser-based visualisation.

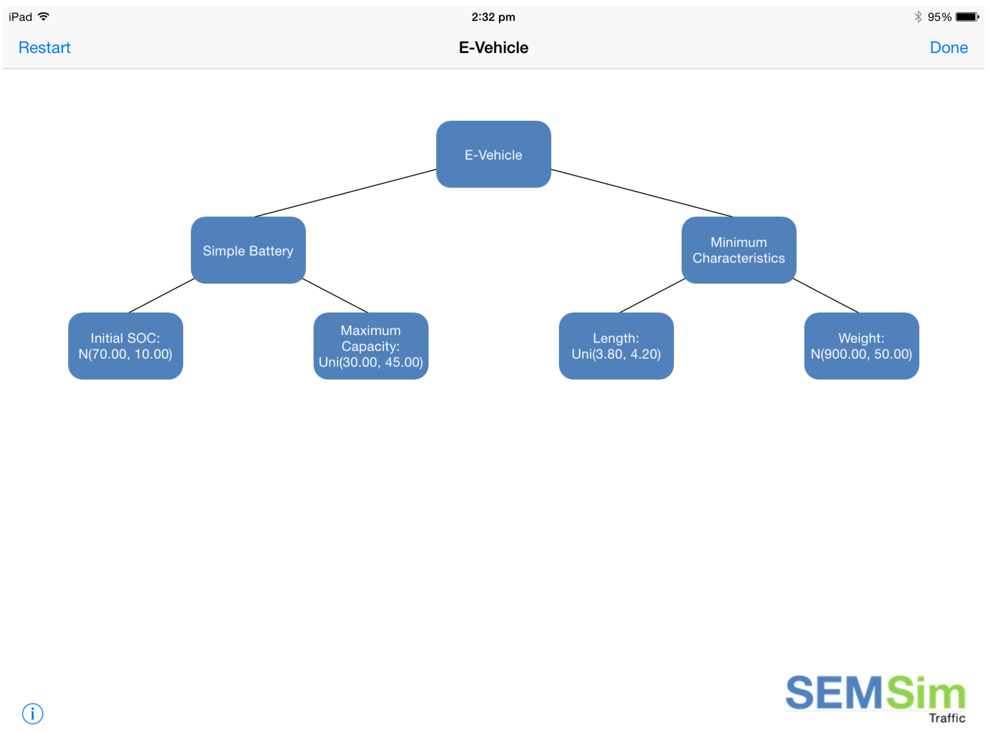
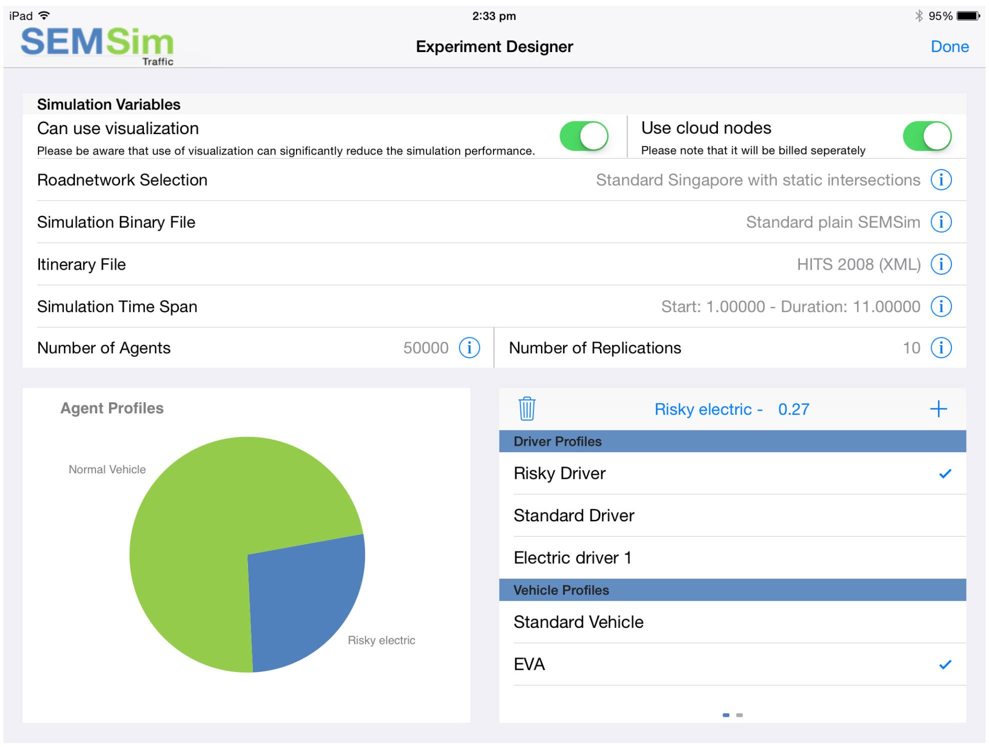


Figure 3.5: SEMSim Cloud Service Model Composition Tree

Rendering several thousand agent representations at a time exceeds the capabilities of most browsers. While browser capabilities are increasing steadily (e.g., WebGL or WebASM), the performance depends greatly on the underlying individual hardware of the client. Therefore, the zoom level is restricted to several streets at a time. Instead, advanced features such as selecting agents and receiving real-time state information about an individual agent or agent tracking are only provided by the touch-based visualisation applications which have been developed for the iOS platform[1]. Since the visualisation does rely on state information of agents and other simulation entities, a simulation could also use the *Online Data Extraction* methodology introduced in Chapter 4. Figure 3.7a and Figure 3.7b show screenshots of the browser-based and touch-based visualisation applications.

- *Analysis Applications:* SEMSim (now CityMoS) CS currently does not come with a dedicated analysis application. In Chapter 4, an online data extraction mechanism is explored

[1]In principal it is also possible to provide visualisation applications on any platform due to the use standard APIs



Experiment Designer

Done

Simulation Variables

Can use visualization
Please be aware that use of visualization can significantly reduce the simulation performance.

Use cloud nodes
Please note that it will be billed seperately

Roadnetwork Selection	Standard Singapore with static intersections
Simulation Binary File	Standard plain SEMSim
Itinerary File	HITS 2008 (XML)
Simulation Time Span	Start: 1.00000 - Duration: 11.00000
Number of Agents	50000
Number of Replications	10

Agent Profiles

Risky electric - 0.27

Driver Profiles
- Risky Driver ✓
- Standard Driver
- Electric driver 1

Vehicle Profiles
- Standard Vehicle
- EVA ✓

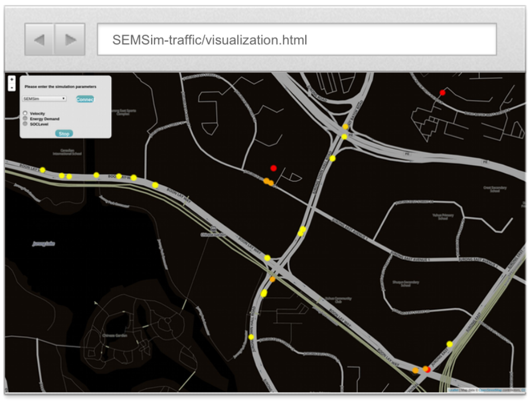
Figure 3.6: SEMSim Simulation CS Experiment Designer

which reduces the amount of data being transfered and ultimately written to persistent storage. It envisions a middleware system that provides data to external processing applications in a similar way how a post-simulation data-base would.

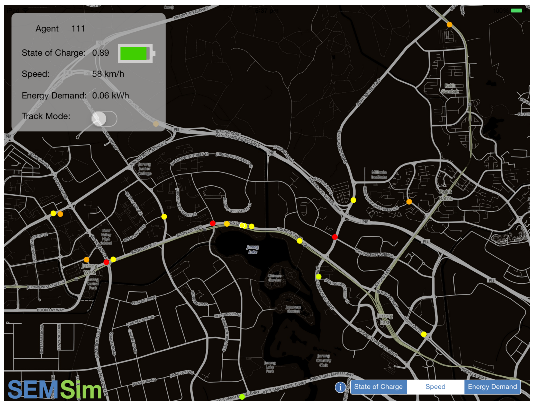
3.3.2 Back-end Applications

The back-end applications used to conduct experiments with the SEMSim (now CityMoS) CS are transparent to the user and run entirely on cloud resources. There are several API endpoints with which the user interacts through the applications. This is mainly for submitting an experiment to be run, to retrieve the data after the simulation is done and, if possible, to access the real-time visualisation of a running simulation.

- *SEMSim (now CityMoS) CS Dispatch Server (DS):* After an experiment has been submitted to the DS, the DS partially reads the EDF and bundles all necessary data into one self-contained simulation bundle. The input data in this bundle can either be stored on the DS instance or can come from an arbitrary source (e.g., database, file server).



(a) Screenshot of browser-based visualisation app.



(b) Screenshot of touch-based visualisation app.

Figure 3.7: Visualisation applications

Since the data is encrypted, the dispatch server has no knowledge of the content of the bundle. Only meta-information such as filename or file size is given. As described in the general system description, the DS is also responsible for creating virtual machine (VM) instances or containers as a run environment for the simulations and keeping track of a simulation's individual progress. If a simulation fails to report its progress, this instance is deallocated, a run environment with the same simulation bundle is created and the user is informed in order to take action.

- *Visualisation Server (VS):* This component of the back-end applications is created by the DS when a real-time visualisation is requested by the experimenter in the EDF. It also runs on a VM in order to reduce the performance penalty of running it on the same machine as the DS. The VS acts as a middleware application between simulation instances and visualisation clients. In order to reduce the overhead on the simulation side due to visualisation output when multiple clients are connected, each VS instance only connects to one simulation instance. It also pre-processes (e.g., aggregation) some visualisation data before sending it to the connected clients. This is especially important for browser-based visualisation clients since their computational capabilities are limited. With a high number of agents, the performance would suffer. Depending on the performance characteristics, a single VS can be used for multiple simulation instances concurrently. The DS also keeps track of the utilization of the VS and provisions an additional one if necessary.

- *Simulation instance:* A simulation instance is executed in one run environment (VM or container) from a CP. In case of a VM, it uses a pre-configured persistent storage medium (e.g., snapshot) with the required libraries and simulation dependencies already installed. If a container is used as the run environment, a pre-configured image is used. The container has been created similarly to the VM by installing required libraries and simulation dependencies. This image could be stored on a private or public repository (e.g., Docker Hub) or loaded from a file. A startup script retrieves the self-contained simulation bundle from the DS and starts the simulation process. It also reports the progress of a given simulation back to the DS in order to verify whether the simulation experiment is still running as planned.

3.4 Performance Evaluation

3.4.1 SEMSim Traffic

SEMSim Traffic (now CityMoS) has been introduced above as part of the SEMSim Platform. We use SEMSim Traffic for the performance evaluation discussed in this section.

The main component of SEMSim Traffic is the High Performance Traffic Simulation (HPTS) core. This consists of a shared memory multi-threaded simulation execution core as introduced in [55]. This offers an efficient way to handle a large number of agents on a multi-core system. Furthermore, it supports routing for individual agents. When updating an agent during the simulation, there are two steps as explained by Aydt et al. [55]. The first step is to calculate a route for a newly created agent or to update the route if the agent needs re-routing. In the second step, the position of the agent is updated and the vehicle and behavior models are evaluated. Amongst others, these models include component models of the battery, and an electric drive train for an electric vehicle as well as behavior models like charging or route choice behavior. The (re-)routing operation is far more computationally expensive than the movement and update of the vehicle models. This assumption is very much dependent on the implementation of the routing algorithm used. A simple unaltered Dijkstra algorithm usually performs poorly in larger graphs. Many forms of optimization have been proposed to decrease time for a single shortest path calculation [56, 57, 58]. Nevertheless, in a road network like the one of Singapore with over 215,000 nodes and 245,000 edges, using a bi-directional Dijkstra algorithm with landmarks [56] and reaches [57], one routing operation costs between 0.1 ms and 100 ms. A very promising approach to speed up the routing is a Contraction Hierarchy routing algorithm proposed by Geisberger et al. [58]. This offers a 100 fold performance increase compared to other implementation (e.g., boost library). With thousands of agents being updated simultaneously, a large number of concurrent threads increases the overall system performance.

3.4.2 Design of Experiment

In order to evaluate the performance of cloud-base simulation, a simulation composed of the HPTS core and a lean extension for basic vehicle characteristics (e.g., kinematic model) and driving behavior (e.g., car following and lane changing) was chosen and executed. The simulation experiments were executed on a virtual computing node composed of cloud computing resources from the GCE as well as on a dedicated HPC node. For the performance analysis, the road network of Singapore with about 80,000 trips during a period of 5 hours between 5 am and 10 am was used. This time frame covers the morning rush hour as it can be seen in Figure 3.8. These times have been obtained by using the Household Interview Travel Survey (HITS) data from 2008 [1].

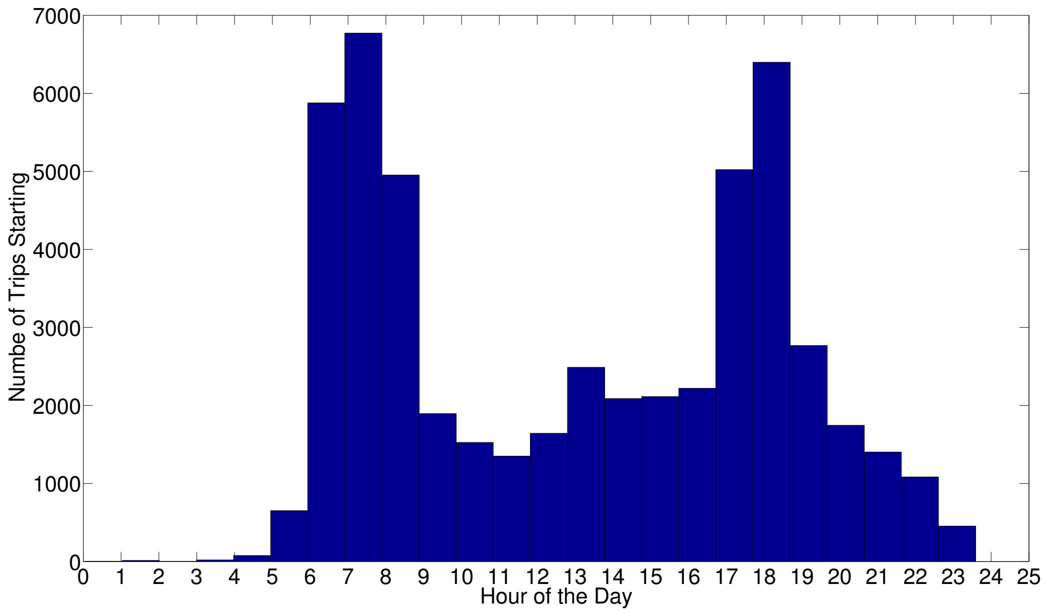


Figure 3.8: Histogram of Trip Start Times

The GCE offers different configurations of VMs ranging from sharing one virtual CPU core to up to 96 virtual cores (compare Figure 3.1 on page 31). The availability of certain

[1]`https://www.lta.gov.sg/apps/news/page.aspx?c=2&id=hgsaeh28887x80v8eyw74vjf02tnhy0xnv36xqcuc87g8r9qk4`

virtual resources depends on the different geographic location of GCE data centers. For our experiment, we chose a data center in the *asia-east region* for all VMs in order to reduce latency when transferring the payload simulation bundle from the DS to the cloud node. The important specifications relevant for this experiment are given in Table 3.1. For the dedicated HPC node, there are 16 physical cores located on 2 separate processor dies on the same motherboard. The processor information within the VM of the CP (e.g., /proc/cpuinfo) indicates that the virtual machine's underlying hardware is a computer with two 6-core processors. The processor information has to be regarded with caution, since it can easily be changed or emulated when running in a virtualized environment. This leads to the conclusion that the 16 virtual cores are not mapped one-to-one to physical cores but rather represent hyper-threading threads. Similar to other cloud infrastructure providers, Google, operator of the GCE, has defined a performance unit to describe the computational capabilities of their virtual machines that is not intuitively mappable to specific hardware resources (e.g., CPU, Memory) as one *Google Compute Engine Unit* (GCEU). It describes the computational capability of $\frac{4}{11}$ hardware hyper-threads (2.75 GCEUs = 1 hardware hyper-thread) [1]. Since the simulation can run with different numbers of

	HPC Node	GCE VM node *(n1-standard-16)*
CPU	2x 8 Intel Xenon E5-2670 @ 2.60 GHz	16 x Virtual Intel Xenon E5-2630 @ 2.50 GHz
Memory	192 GB	60GB

Table 3.1: Computational resources specifications of a dedicated HPC node and a GCE virtual node

threads, the experiment has been conducted using different configurations. Each experiment has been executed on a virtual cloud node and dedicated HPC node with 16 processors in the configuration of 1, 2, 4, 8 and 16 threads. The execution time for the same simulation run has been recorded. In addition to measuring the actual execution time of the simulation, the following times were also measured:

- *Provisioning time:* the time for provisioning a virtual node including hard drive and network from the GCE. This serves as an indicator of whether the availability of a certain configuration can be different.

[1] https://cloud.google.com/compute/docs/machine-types

- *Startup time:* the time for transferring the payload in the form of the self-contained bundle, as discussed in Section 3.3 on Page 37, to the virtual cloud node. This can be used to evaluate the network performance at the geographic location of the data center.

- *Deallocation time:* the time for deallocation of the resources. This has to be considered in order to have a complete time span for an experiment.

These time measurements are not necessary when executing on a dedicated HPC node since there is no provisioning and deallocation of hardware. When the execution environment is a shared high-performance cluster, there might be some waiting time to get required resources assigned, but these considerations are not part of this experiment.

In order to create a new VM from the infrastructure resources offered by the GCE, we must first create a virtual startup disk from a pre-configured base-image. Together with standard network resources, this image is attached to a VM of the type *n1-standard-16*[1]. This is initiated by the DS through an API provided by GCE. The download of the payload data to the VM is initiated by the startup script of the VM. This script has been included in the aforementioned base-image that is used to create the VM and is run every time the VM is (re)booted. Once the simulation terminates, the deallocation process is triggered by the DS through the API. The deallocation process works opposite to the creation. Hence, the virtual machine and all resources (e.g., network) are deleted first and the virtual disk afterwards.

3.4.3 Experimental Results

The experimental results have to be differentiated between the overhead introduced by provisioning, start-up, decryption and deallocation on a GCE VM node and the actual execution time of the simulation.

3.4.3.1 Cloud Computing Overhead

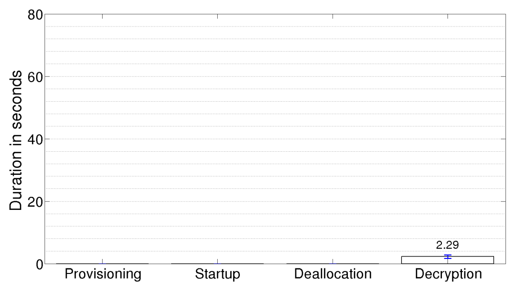
The decryption of 4 files of input data, ranging from only several kilobytes (intersection control) to hundreds of megabytes (road network file), with a total file size of 489.02 megabytes took only 1.26 seconds with a standard deviation of 0.049 seconds. The number of threads used had no influence on the decryption performance, since the implementation used is only single-threaded. A parallel execution of independent input files can speed up the overall decryption process, but

[1] https://developers.google.com/compute/docs/machine-types

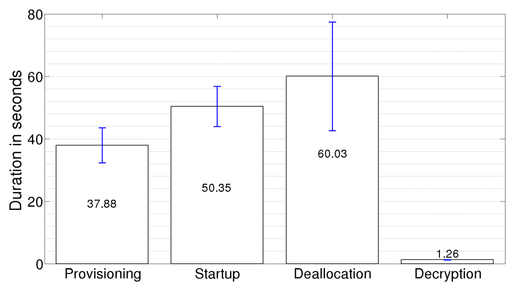
since there are interdependencies between the input files (especially the road network), this was not possible in this case.

For the tested GCE of the machine-type *n1-standard-16*, the average total overhead from 90 samples is 149.52 seconds. The individual times can be seen in Figure 3.9b. Since the provisioning, startup and deallocation operations are highly dependent on the GCE service availability, the execution time of the individual operation can vary due to external factors (e.g., network congestion or high demand) as well. The API calls using the GCE SDK are asynchronous, but have to be finished before advancing to the next operation and cannot be done in parallel.

It can be observed that the decryption of the input data is insignificant compared to the rest of the overhead. On a dedicated HPC node, the times for provisioning, start-up and deallocation can be disregarded since the system already exists. The decryption of the input files nonetheless, adds 2.29 seconds with a standard deviation of 0.56 seconds to the total execution time (see Figure 3.9a).



(a) Simulation overhead when running a simulation instance on HPC node



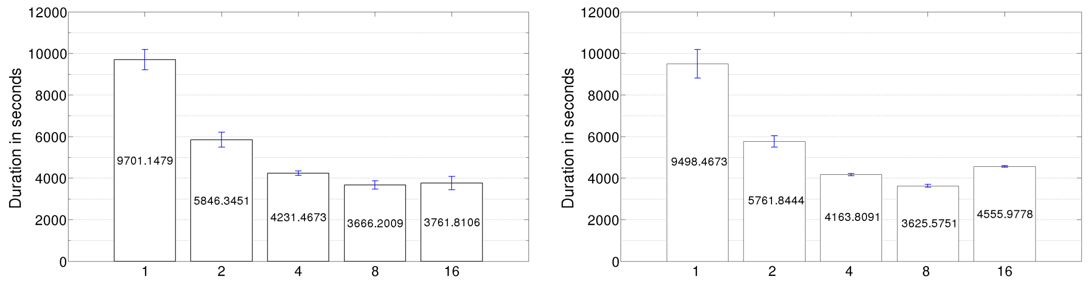
(b) Simulation overhead when running a simulation instance on GCE node

Figure 3.9: Simulation overhead on different execution environments

3.4.3.2 Simulation Runtime Comparison

The execution time on a dedicated HPC node for different numbers of threads is given in Figure 3.10a. This shows that on a dedicated HPC node with 16 physical processing cores, the execution time decreases when the number of threads increases. This trend reaches its minimum execution time with a configuration of 8 threads and has a slight increase when using 16 threads (+2.6%).

The results for executing the same experiment on a comparable virtual node are shown in Figure 3.10b. There, the same general characteristic of a decreasing execution time can be



(a) Simulation execution time in seconds depending on the number of threads on 16 physical cores

(b) simulation execution time in seconds depending on the number of threads on 16 virtual cores

Figure 3.10: Simulation execution time on different execution environments

observed. Rather than having a slight increase in execution time, the increase between using 8 and 16 threads is significant (+25.6%).

3.4.4 Discussion

Since the overhead imposed by provisioning, start-up and deallocation is independent from the actual execution time, its influence on the total experimentation time gets smaller when the simulation workload increases. The increase in workload can be due to various means. The increase of concurrently active agents as well as increasing the complexity (computational requirements) of the internal agent models influence the workload. An average of 149.52 seconds makes up between 1.57 (for 1 thread) and 4.12 percent (for 16 threads) of the given execution time depending on the number of threads used. In comparison, the decrease in execution time by using a virtual node is between 1.11 (for 8 threads) and 2.10 (for 1 thread) percent. This makes the use of virtual nodes very much comparable to the dedicated HPC nodes. Even for such short-running simulation examples, the imposed overhead can be neglected. The only exception to the comparable performance is the use of 16 threads on the virtual node. There, the performance decreases by 21.1 percent compared to a dedicated node.

The apparent increase in execution time in both configurations when 16 threads are used can be explained by evaluating the underlying hardware architecture. The dedicated HPC node has two eight core CPUs whereas the VM from the GCE has two six core CPUs. Since hyper-threading was used on the dedicated HPC node as well, the simulation execution on the GCE virtual machine should result in a comparable outcome. The 16 threads used with two eight core CPUs (32 hyper-threads) leaves more headroom for scheduling of threads to a certain processor than on a machine with two 6-core CPUs (24 hyper-threads). The overhead due to cache misses in the hardware architecture and the programming model can lead to the biggest performance decrease. Simulating thousands of agents in a nanoscopic manner leads to cache misses, since the processor which will need the same agent data during the next calculation cannot be predicted. As Figure 3.11 shows, there are two CPUs on the same motherboard connected via the main system bus on which other peripherals such as the main memory is also connected. An access to a memory address from a CPU core on the same physical CPU results in using the cached value in the CPU L1 to L3 cache, due to the locality principles. Accessing a memory address that has not previously been requested on the same physical core as the requesting core, the chance that the required data will no be present in the cache and therefore has to be retrieved from main memory, using the system bus, is higher. This operation costs many computing cycles as Hennessy et. al. [59] describes it, which slows down the simulation.

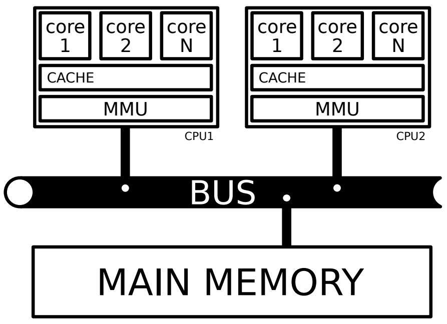


Figure 3.11: Shared memory on multi-core and multi-processor systems[1].

Adding the encryption layer to the input files does not add any overhead to the execution time. The prolonging of the simulation by 1.26 seconds or 2.29 seconds for a virtual and a dedicated HPC node respectively is not significant considering the overall run-times of several hours for a given simulation. The visible discrepancy between the dedicated HPC node and the virtual node of about 1 second can only be inferred by differences in speeds in the persistent storage medium. It is not necessary to encrypt the data on a simulation system that is under the control of the simulator. Nevertheless, if data providers want to control the use of their data, a key server can revoke access to the input data if a usage limit has been negotiated between the simulator and the date provider. This is similar to the digital rights management (DRM) used to protect music, books or movies.

[1]Figure adapted from Understanding Parallel Hardware: Multiprocessors, Hyperthreading, Dual-Core, Multicore and FPGAs at `http://www.ni.com/tutorial/6097/en/`

3.5 Templating for Simulation Experiments

Templates can be seen as pre-configured instances of an object that needs to be completed by adding or changing parts of the object to obtain a specific result. Websites, for example, are usually built using templates, which consist of static segments of HTML code as well as exchangeable sections. Such sections are filled when the user requests a certain website with his/her user-account. An example could be a web-mail client in a browser or in a web-wrapper in a multi-platform application (e.g., Electron[1]). The construct of how the elements are arranged on the screen are equal between different users, but the actual emails that are displayed are dynamic content. Therefore, the skeleton of the email page is the template to a certain user-specific functionality.

For simulation experiments that are closely related, the amount of input parameter changes between two experiments is rather small. Therefore, simulation configuration templates could be used to reduce the configuration complexity of simulation experiments significantly. This also applies for the easy reproducibility of experiments. Additionally, in a complex-system simulation, like traffic, many different models from different domains play together and their possible interconnection and interdependency cannot be evaluated by a single user in a specific (sub-)domain. Therefore, providing simulation experiment templates in which the static parameter values are validated, by the domain expert, is a big plus in making complex system simulation much more reliable and helps to keep researchers focused on their domain research. In addition to researchers, non-experts can also benefit from templating. Complex scenarios can be simulated using the latest models and technology and the effect of an abstracted change (e.g., change in road infrastructure) of a single or a couple of tangible parameters can be observed, interpreted and explained to the non-expert.

Templates and applications that allow for easy use of simulation templates in for simulation experiments, as presented in this section, can be seen as a targeted *Experiment Designer* (ED) as described in Chapter 3.3. It allows researcher and non-exeperts to quickly prototype models or run manual parameter sweeps without going through all parameters of an Experiment Description File (EDF). The partial EDF (template), where only a limited number of parameter have not been filled in, is used as an input and a completed EDF, with all parameter and runtime values filled out, is the output.

[1]`https://electronjs.org`

3.5.1 Templating for agent-based Traffic Simulations

For agent-based traffic simulations like SEMSim Traffic (now CityMoS), a configuration of the different model parametrization is vital. This also means a template for a simulation can easily be created by using an existing configuration from a validated simulation and converting it into a template.

An example would be to have the user change the population sizes for the different heterogeneous agent populations. While this small change can have a large effect on the overall system performance and can lead to different conclusions to be drawn in the real world, all other models in the templated simulation experiment are already verified and validated to work properly by the respective domain expert. This experiment template is very useful to administrative advisors or city planners that need to base their future infrastructure decisions on these holistic simulations of large urban systems like traffic and its interplay with other systems (e.g., energy, smart city).

The example of agent-based traffic simulation templates will be further explored in order to demonstrate the complexity and therefore necessity of templating. For a holistic agent-based traffic simulation experiment, different model categories are required to adequately represent the real world. For each of these model categories and sub-categories, different domain experts are to be consulted in order to end up with the most accurate model. The vehicle component of such simulation requires mechanical and vehicle engineers which can be further divided into specialized engineers for electric and internal combustion engine vehicles, should the simulation experiment require electric or conventionally powered vehicles respectively. This can be extended to any level of detail, depending on the depth that the simulation needs to go into. Additionally, behavioral models for both drivers as well as other traffic system participants need to be present.

Experts in the field of developing such models have to verify and calibrate models that range from travel patterns (e.g., activity base) to create traffic up to the individual decision model for changing a lane or deciding to refill/charge the vehicle. This is also dependent on the type of vehicles in the simulation and the experiment's focus. Since a traffic simulation experiment would not be possible without infrastructure data like traffic signals, bus stops, car parks and especially the road network, to name a few, experts in data preparation, fusion and processing are required to make the simulation experiment as close to the real world as possible. The above-mentioned domain experts cannot be combined into one single person while expecting the results to be meaningful. Integration experts can be called into action to create experiment scenarios for the end-user, since they have high-level knowledge of all the different components involved. Therefore, the simulation cloud service model and profile repository as mentioned in Chapter 3.3 can be used to generate templates for end-users. While the *Profile Designer* and the *Experiment Designer* help to create entire simulation scenarios, in a templating context these designed profiles and experiments can be simplified into templated scenarios for different user groups. These users can be manifold, consisting of policy makers as well as related-field domain experts up to the non-expert wanting to explore their local traffic-system. This would improve the knowledge of the non-experts and democratize the decision-making process by allowing everyone to have access to easy to use and understand simulation tools and scenarios.

3.5.2 Templating workflow

In order to develop a template for a given scenario that helps non-domain experts to explore a complex system like traffic, several steps have to be taken. These steps are visualized in Figure 3.12 and will be explained as follows.

The three involved user groups are (1) the respective domain experts that create, calibrate and validate the models within their domain's expertise. Furthermore, they are providing a set of parameters and input value ranges for which the models are validated. This allows the (2) integration expert to use such models and combine them into scenarios. This includes taking models from different domains and combining them into a single simulation experiment, where most of the model input parameter values are pre-determined by the scenario creator. The interplay between different models have to be checked for basic interoperability. Since the integration expert is in most cases not a simulation-execution expert, using a cloud service to obtain the required computing power is the most favorable option. The checked model, together with the changeable parameters is then pushed to the template repository. From there (3), the end-user can use the template to explore various scenarios given the limited input parameter-value range. These end-users can be related-field domain-experts, policy makers or non-experts as mentioned above. Since the end-user can be assumed to be the least technical person with regards to executing simulation experiments efficiently in the cloud or on an HPC, tools such as a cloud service or a specialized templating application can be used to abstract the technicalities from the user. The results from such experiments in the cloud are sent back to the end-user and evaluated. These evaluation steps can be part of the templating application to present a non-expert or non-technical person with simple and easy-to-understand answers.

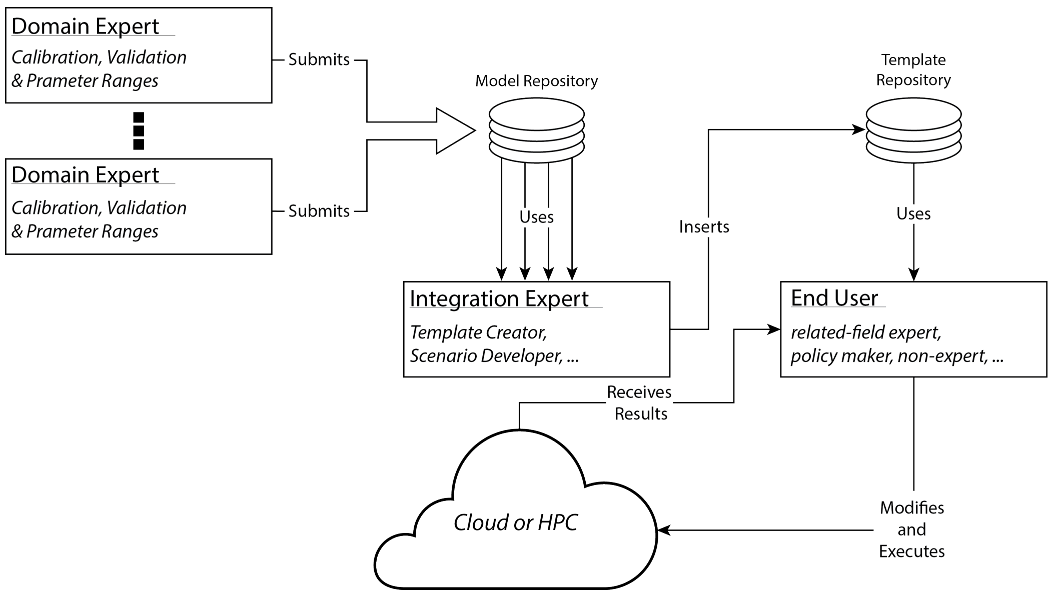

Domain Expert
Calibration, Validation & Prameter Ranges
Submits
Domain Expert
Calibration, Validation & Prameter Ranges
Submits
Model Repository
Uses
Template Repository
Inserts
Uses
Integration Expert
Template Creator, Scenario Developer, ...
End User
related-field expert, policy maker, non-expert, ...
Receives Results
Cloud or HPC
Modifies and Executes

Figure 3.12: Workflow starting from domain experts to receiving the templated experiment results.

3.5.3 Transparent Cloud Realization

The use of templating for simulation experiments increases the usability for non-experts and novice users. It allows the user to focus on the important aspects of the researcher's field or the interest of the user. The calibration and validation of the models and components is done by the domain experts. Nonetheless, the computing infrastructure needs to be present in order to execute such a complex-system simulation, which requires a significant amount of computing resources. One solution as presented in this thesis is the use of cloud resources to do so. The setup and maintenance of such computing resources has become relatively easy, but still nothing a domain expert that is not in computer science should be tasked with. Therefore, cloud services as presented in this thesis abstracts the complexity away from the user. One other option when not relying on cloud-service APIs is the use of lightweight containers to create VMs runtime environments on the local computer and push them to a cloud service when a large-scale experiment, that requires more computing power, is to be run.

Lightweight containers and their most prominent implementation, Docker, are an emerging technology that has changed the landscape of deploying software stacks, not only to the cloud but also local testing environments. The Docker software family and services have grown significantly over the last years and have a very rapid development cycle with new features added on a monthly basis. One of the software options to allocate Docker-capable computing nodes is *Docker Machine*.

This is possible via a standardized (command-line) interface that is able to create VMs, provision them and use them for simulation experiments. The Docker Machine comes pre-configured with multiple common cloud-providers drivers like Amazon, GCE or Microsoft Azure. A driver describes to the Docker Machine application how to interface with the specific CP's backend to create VMs. Should a driver no bin in the list, users can also add them manually. Using this technology, the execution of a simulation experiment follows the same workflow, where it does not matter if executed on a local machine, running bare-metal Linux, a virtualized Linux, using type 1 or type 2 hypervisors, or a cloud VM, it will look and feel the same to the user. In conjunction with Docker and Docker Machine, the distribution and execution on heterogeneous hardware configuration has been much simplified due to automatic configuration on different computing environments. When you include the post-processing into the templating workflow, an entire decision support system can be built up using templated complex-system simulation. In the following section, an exemplary application for using a templated workflow is presented. This also includes information about the back-end system and repositories.

3.5.4 CityMoS Cloud App

The CityMoS Cloud App (CCA) is an application that allows a novice user of simulation to get started quickly. It is used to conduct traffic simulation experiments using the SEMSim traffic (now CityMoS) simulation engine. In Figure 3.13, the work-flow of executing a simulation using CCA is shown, as well as the involved components. The main control component is the CCA Application window. From there, the user can browse the different templates available in the repository and gain an overview of the running experiments. The template repository is a (mongoDB) database that stores the templates, the binaries that can be used with a given template as well as meta-information such as descriptions of the experiments that are possible with the given template. This can either be hosted locally on the user's computer or centralized, allowing multiple users to have access to the templates. Another repository is the private or public docker repository for Docker container images. These are images, that contain a configured runtime environment for the simulation engine and a second container contains the standard input for experiments that can be done with the given templates. The runtime environment has all the dependencies and libraries installed that the SEMSim traffic (now CityMoS) executable can run. Since Docker container can have different versions, this allows for updating the environment independently from the executable. This applies to changes to the input data (container) as well. The most important component is the hardware the simulation experiment is actually executed on. Since provisioning, starting, controlling and deallocation of these resources is done through the Docker Machine, the virtual resources are called machines. The user does not need to know what kind of hardware, physical or virtual, is used to run the experiments.

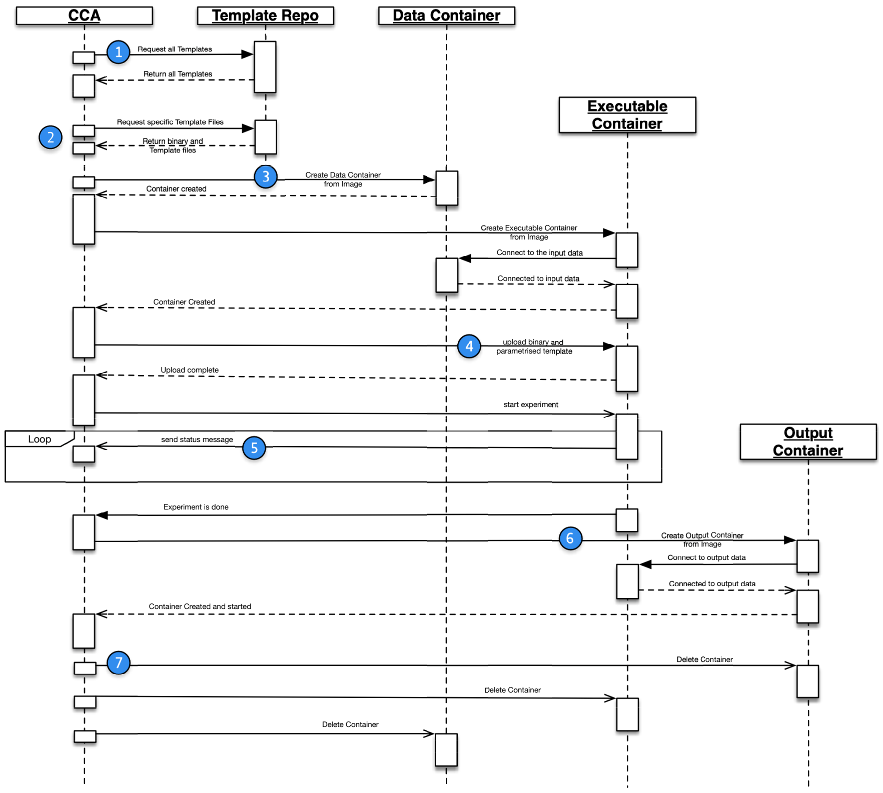


Figure 3.13: Workflow diagram for the CCA overall architecture

The different steps of a templated simulation work-flow are:

1 Download of the template information into the CCA. This mainly includes the meta-information about the experiments possible with this template as well as the parameters that can be changed. The actual template as well as the binary are kept in the repository until needed.

2 The experimenter enters the parameter values into the mask and selects the machine the experiment will run on. The machine can be the current workstation, a Docker-enabled compute-machine in the network or a cloud-allocated resource.

3 This selection, together with the parameter values, is then combined with the template from the database to form a proper configuration file for the simulation experiment.

4 A bundle containing the binary (also downloaded from the database) and the configuration file is created and pushed into the selected Docker container on the selected machine.

5 The simulation starts to produces output that is stored in the container itself. The console output can be taken and used by the CCA to estimate the progress of the simulation experiment.

6 Once the simulation ends, the container stops as well, but is not yet discarded. Furthermore, another container is started that provides a HTTP-based front-end to download the output data contained in the execution container.

7 The experimenter decides when to discard the containers as well as when to deallocate the Docker Machine.

In the execution of a template-based simulation experiment using the CCA, there are three containers involved. The schema and their interconnection is depicted in Figure 3.14.

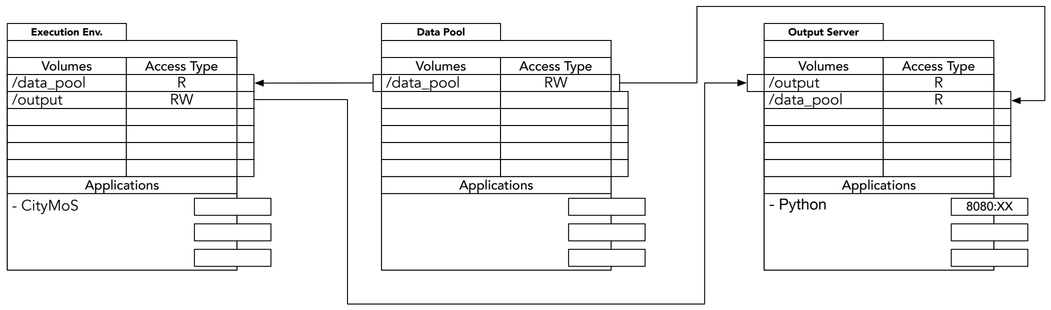


Figure 3.14: Docker container configuration with exposed ports and linked volumes.

3.6 Summary

In this chapter, an architecture for a cloud-based simulation platform for urban systems that enables the user to execute large-scale simulation experiments without the need for dedicated hardware was introduced. It has been shown that the use of IaaS from CP is feasible and relatively easy to do. In addition, the performance of a virtual computing node from the GCE is comparable to that of a dedicated node of a HPC cluster.

In contrast with other approaches, the proposed service acquires computing resources on-demand and therefore has no upfront costs. The presented approach also explicitly takes into account data encryption to ensure confidentiality of data used to perform simulations. Through the use of a self-contained package, a heterogeneous execution environment composed of different hardware and software components can be used and is transparent to the user. Using Docker images or pre-configured VMs as run environments makes the deployment even easier and reduces the overall administrative overhead. Since the actual simulation is not run on workstation computers but rather in a local HPC or virtual compute center, the interaction methodology has been substituted, leading away from mouse-and-keyboard-based input devices on workstation computer and towards application-specific (e.g., touch-based) input devices as well as a standard web-browser through some kind of API. Such simulations are not limited to the presented urban system simulation, but can be manifold.

Based on the presented methods, developing a framework to enable researchers to run their simulations in the cloud is a next logical step. PaaS Application Programming Interfaces that are tailored towards running simulations have to be implemented. This enables developers to utilize the APIs to develop more forward-thinking user-interaction methods and application support systems.

Chapter 4

Online Data Extraction for Large-Scale Agent-Based Traffic Simulations

Contents

Large parts of this chapter have been published in the proceedings of the *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation 2016* [60].

The computational resources required for modern urban-system simulations [61, 62] are increasing; this has also resulted in more high-resolution output data being generated. Moreover, the availability of much cheaper storage options[1] has resulted in data being generated at unprecedented velocities, volumes and varieties [63, 64]. Depending on the run time of a single simulation, the design decision is often taken to record as much as possible in order to reduce the number of simulation runs.

However, despite the price reduction in persistent storage, analyzing large amounts of data can still be problematic. Large data sets have to be loaded into the memory of one or multiple machines in order to do the necessary post-processing of the data. If the main memory of the system is exhausted, a distributed or stream-processing workflow (e.g., Apache Hadoop) has to be used. This adds an additional overhead for the user.

There is a need for more efficient methods for handling the large datasets that are generated, especially with the increasing availability of cloud computing resources and cloud-based simulation services [65] as presented in this thesis. As stated in [66], "transferring data-sets to a centralized machine is thus expensive (due, for example, to network communication and other I/O related costs)".

A possible solution to this problem is to conduct the data analysis while the simulation is running and store only the processed data-set necessary for the given experiment, i.e. the *result data-set*. Such a solution would be most useful for large workflows where, traditionally, huge amounts of data have to be transfered between two consecutive steps. A possible approach to this would be to leverage on the *IEEE 1516* High Level Architecture (HLA) [31] which is popular in the simulation community. A data processing federate could be created which collects the data from the running simulations and processes and writes it to persistent storage as required. However, the limited data transfer rate and the large overhead of publishing all information either reduces the simulation performance significantly or necessitates a larger simulation "cool-down" phase during which all data is analyzed and transmitted to a single data analysis federate.

Most simulation experiment post-processing workflows start with obtaining data-sets from databases, but since the size of these data sets is increasing steadily, an online data-extraction methodology, as presented in this chapter, can be useful. A formal description of the components and advantages involved in an online data extraction methodology for agent-based simulations (specifically cloud-based simulations), in which data processing at simulation time

[1]`http://www.mkomo.com/cost-per-gigabyte-update`

is used to minimize the use for slower persistent storage options, is presented and executed. Using relational algebra in the context of modelling the data output of a simulation does not break the post-processing workflow of many experiments. Agent-based simulations have been taken as an example, since the state of agents of the same (or similar) type in a simulation experiment can be interpreted as a table or and the interaction between such agents as relations.

Using relational algebra to model the data output of a simulation is beneficial, since it does not interfere with the established post-processing workflow of many experiments. Agent-based simulations are a good example for the use of a relational representation of the output data, since agents can be grouped together and be seen as a relation with tuples of state-variables.

As part of the literature review in this chapter, the current research in big data and stream processing as well as data-description-languages is explored. Subsequently, the structure and overall workings of data-analysis components for a cloud-based simulation system as presented in Chapter 3 is introduced. This is followed by a formal relational-algebra-based description of that data output. Unlike traditional relational algebra, the proposed data description can be used to describe time-variant data. Finally, two traffic simulation examples are used to demonstrate the workings for the system as well as the data description using time-variant relational algebra before concluding this chapter with outlook on further research that can be conducted in this field.

4.1 Related Work

4.1.1 Relational Data Description

Relational data models are widely used in the database domain where they describe the relations of data by using tables from which data can be accessed and operations can be executed on. The data entries are called *tuples* and share the same structure (i.e., fields) within a single database table. The advantages of the relational data description language (DDL) and relational algebra are that they enable the analysis and optimization of complex and large amounts of data by using formal mathematical methods. Despite the limited number of operations, complex manipulations of structured data can be achieved through combination. Relational databases and data structures were first introduced by Codd [67] and later refined by Darwen and Date [68]. In this chapter, the well-developed work in relational algebra is extended with a temporal component in order to deal with time-variant data sources that characterize simulations.

4.1.2 Big-Data and Stream Processing

Ranja [66] commented on the fact that the data that is being generated on the internet has been and will keep increasing rapidly over the next couple of years. This is similar to the data being generated by simulations. Individual simulations produce a lot more data than the average server on the internet in the same time-span. The use of cloud-based simulation services [45] can result in multiple concurrent simulations at unprecedented rates which will increase the data output significantly. Ranja [66] postulates further that relational databases will be unable to cope with the massive data. In contrast, state-of-the-art data mining algorithms work on main memory. However, main memory is much more expensive and invariably inadequate to store the amount of data generated.

Another point to be considered is the network and I/O costs involved in the transfer of data between the storage location and the analytics computer [66]. Ranja proposes an ecosystem for data processing that involves a high velocity *data ingestion layer* that communicates with the actual *data analytics layer*, which then pushes the resulting data sets to a *data storage layer*. These kinds of data analytics frameworks have different advantages and drawbacks. While the Apache Hadoop [69] distributed stream data processing toolchain is more suited for historic/existing data analysis, Spark [70] and Storm streaming processing are better suited for online data-streams with highly variable (in terms of amount and type) data.

For general big data processing, different tools already exist. Tools like Apache Mahout[1] and GraphLab[2] have a large number of implemented data analysis algorithms available for use. These offer an easy-to-use, off-the-shelf experience for researchers. On the other hand, building a system for simulation data analysis with such tools might be outside the realm of standard big data processing and presents its own set of challenges. A distributed system that relies on message passing and queuing for a general purpose API is more applicable for simulations. Apache Kafka[3] for distribution of incoming data streams is an example of such a system. This usually works in conjunction with Hadoop, Storm[4] or Spark[5] to distribute high-velocity data processing payloads. Such data needs to be stored in a NoSQL database structure like MongoDB [71] or Casandra [72]; in these systems large amounts of data are stored in

[1]http://mahout.apache.org
[2]http://graphlab.org
[3]http://kafka.apache.org
[4]http://storm.apache.org/
[5]http://spark.apache.org

easily-accessible structures that are advantageous for the application to retrieve and work on the data.

Babcock et al. [73] give an overview of data-streaming models and current issues in data stream processing systems. They acknowledge that there are separate groups of data-streaming models that handle data differently and differ from conventionally stored data in (1) availability (online vs. offline), (2) order of data elements, (3) undefined or unbound size of the stream and (4) the unavailability of historical data, since after processing, the input data is discarded.

4.1.3 Data Extraction Techniques for Simulations

For many simulations the data export focuses on writing to a comma-separated, XML-based or application-specific binary data format. This might be acceptable for small amounts of data coming from small-scale simulations or rarely updated variables, where the final amount of raw data is rather small. For large-scale simulations, saving raw data might be desirable but unfeasible due to I/O constraints. Many simulation tools output only aggregated values or rely on visualization to transport information in the form of recordings (animation or video) or images. Some prominent examples are material simulations, where the physical or thermal forces on a digital workpiece are shown as an overlaid heat map [74, 75].

Schützel et al. [76] describe a stream-based reference architecture for a data-management system that interacts with all the components of a simulation workflow. The described approach starts from the experiment setup and the actual execution of the simulation, up to adding a "processing graph" and a storage engine. Data management has a mediating role between all steps of a simulation and passes data between the different steps of an experiment. This mediator role is also active for different simulations within one experiment run.

In a different publication, Schützel et al. [77] have presented that the extraction of simulation data is dependent on the structural dimension of the simulation entities as well as the sequential dimension that describes the order in which the data is being generated. They describe the *ML-Rules Data Extraction Language* and also the *SystemXtract Language*. The latter uses sequential logs to reconstruct the structural and sequential information of the simulation state and passes it to the analysis application.

4.2 Formal Simulation Data Representation

The data in an agent-based simulation can be described in multiple ways. It is proposed that one way data can be represented is via relation algebra. This is used in this section. This provides the advantage that queries composed using the *structured query language* (SQL) can be used to retrieve data from the simulation. In agent-based simulation, each set of agents of the same type can be perceived as tuples in a relation of a database. Therefore, the approach representing the output data-model in the form of relations is favorable, since many (post-processing) algorithms and work-flows already work on "database structured data" as input. Should there be a very heterogeneous agent-population, where agents cannot be grouped together into relations, each agent will then form its own relation with only one time-variant tuple.

Since some of the data will change over the course of the simulation, a temporal component needs to be added to the relational schema. Mahmood et al. [78] have given an overview of how to encode temporal information into a relational data model with a focus on database use cases. This approach is not restricted to databases and can be applied to generic relational data. This temporal relational data model can be adapted to be used in describing output data from an agent-based simulation. The difference between a simulation that data is described using relational algebra for database used is that there can't be any data of the past delivered to the requester, by simply changing the query as possible with data bases, where all the data is stored and available to be requested. The data analysis middleware will only receive data from the simulation when the requested time is reached. This means, in practical use, a request on a future state of an agent in the system can trigger an event to start output of the data at that event and not sooner.

4.2.1 General Simulation Data Representation

Relational data models, are often used to characterize the well-defined structure of databases, called a *database schema*. One could describe the state variables of the agents in an agent-based simulation as a schema with time-varying data. The formal description of an n-ary tuple of states is usually done by writing the name of the tuple (class name) followed by a comma-separated list of the state variable names in parentheses [67]. Primary keys describe a field or a set of fields which characterizes the tuple unambiguously. The notation for a primary key is underlining the respective variable names. For the time-varying relational data models as described in [78], an extra field is added to each relation entry to describe its temporal activation (when the value is valid). It is also possible to add a field for each variable of the relation, but this increases the number of variables in a relation significantly, since for each variable field an extra field for activation is required. Since the actual time when a state variable is active is not very important, an indication that the field might change over the course of the simulation is sufficient and adding an extra field of each of the state variables indicating the time of change is unnecessary.

It is proposed to indicate time-variance by adding that the variable is a function of time (e.g. $variablename(t)$).

$$A(\underline{f_1, f_2, ..., f_k}, f_{k+1}, f_{k+2}, ..., f_{k+n}, f_{k+n+1}(t), ..., f_{k+n+m}(t))$$

shows agent A with $n \in \mathbb{N}_{>0}$ static state variable names, $m \in \mathbb{N}_{>0}$ time-variant state variable names and $k \in \mathbb{N}_{>0}$ state variable names that unambiguously describe one $(m + n + k)$-ary tuple. Time-variant data cannot be part of the primary key tuple since it would additionally describe the relation uniquely in time, which is not a desired attribute to track the changes of this entry over time.

Time-variant Relational Algebra

Since traditional relational algebra has no concept of time-variant fields or tuples, the individual simple functions like projection (Π), selection (σ), Cartesian product ($\times$) and natural join ($\bowtie$) have to be translated to equivalent time-variant operations. In all following operations the n-ary tuple for a given relation has to be determined, such that the tuples that are included in the result set of tuples are the most recent tuples for a given primary key in respect to the time τ given. This is also expressed in Equation 4.2. The Domain of operation α on relation R at time τ finds all entries where the timestamp is smaller or equal to τ and there is no other timestamp (t_2) that is closer to τ than t_1. Additionally, there is no other entry with the same id and which has a timestamp between t_1 and τ.

$$
\begin{aligned}
A(id_i, t_i) :& \text{Entry in a relation with id and time} \qquad (4.1)\\
TS(id_i) :& \text{all Timestamps for a given } id_i\\
ID(ts_i) :& \text{all IDs for a given } ts_i\\
\alpha :& \Pi_{a_1,...,a_n}, \sigma_{a\theta v}, \bowtie, \times, ...\\
Dom(\alpha(R(\tau))) :& \text{Domain of operation from } \alpha\\
& \text{on relation R at time } \tau\\
Dom(\alpha(R(\tau))) =& \big[\forall A(id_1, t_1) : [t_1 \in TS(id_1), t_1 \leq \tau\\
& \wedge \nexists t_2 \in TS(id_1), t_1 < t_2 \leq \tau]\\
& \wedge [\nexists A(id_2, t_2) : t_2 \in TS(id_2), t_1 < t_2 \leq \tau\\
& \wedge id_1 = id_2]\big] \qquad (4.2)
\end{aligned}
$$

Table 4.1 shows two exemplary relations that are used to demonstrate the time-variant operations in this section. The column TS is the time-stamp of the respective values.

Table 4.1: Example Relations

Relation1				*Relation2*		
ID	**V1**	**V2**	**TS**	**ID**	**V3**	**TS**
1	A	C	0	1	E	0
2	B	H	0	2	F	0
1	J	C	1	1	G	1
3	C	D	2	2	L	1
2	I	C	2			

Time-variant Projection

A projection $\Pi_{a_1,...,a_n}(R)$ returns a set of data items that contain the components $a_1, ..., a_n$ of relation R and disregards all other fields in the set of tuples. The time-variant projection includes the indication for what time τ this operation should be applied. $\Pi_{a_1,....,a_n}(R(\tau))$ returns all n-ary tuples from the time-variant set R that satisfy the result of Equation 4.2. The projection in Table 4.2 outputs the variable V1 and the ID at time $\tau = 1$ from Relation1. The result shows that only the entries from Releation1 are shown, that have or had a value prior to the requested time $\tau = 1$. Even though ID=2 has no timestamp TS=1, the information is in the result projection because information exists for TS=0. The opposite is true for ID=3, which doesn't have any information prior to TS=2 and is therefore not in the resulting projection.

Table 4.2: $\Pi_{\mathrm{ID,V1}}(\mathit{Relation}1(1))$

ID	V1
1	J
2	B

Time-variant Selection

A selection $\sigma_{a\theta v}(R)$ returns a set of data items from the set of n-ary tuples R that satisfies the restriction expressed by attribute a, the binary operation $\theta \in \{<, \leq, =, \neq, \geq, >\}$ and the constant value v. The time-variant version of a selection needs to include the indication for what time τ this operation should be applied. $\sigma_{a\theta v}(R(\tau))$ returns all tuples that satisfy the restriction, but are also part of Domain specification given by Equation 4.2. The selection in Table 4.3 outputs the variables of Relation1 with the condition that variable V2 is equal to D and τ has the value 2.

Table 4.3: $\sigma_{V2=D}(\mathit{Relation}1(2))$

ID	V1	V2
3	C	D

Time-variant Natural join

A natural join $R \bowtie S$ returns a set of data item from the sets of tuples R and S that have at least one matching attribute ($\exists a_r \in R \wedge \exists a_s \in S; a_r = a_s$) from both relations in a resulting tuple. The time-variant version of this natural join includes indications for what times τ_S and τ_R this join should be applied for. $R(\tau_1) \bowtie S(\tau_2)$ return all tuples as the standard join would, but under the constraint that the relations R and S are first reduced to the result generated by Equation 4.2. The natural join exapmle in Table 4.4 joins Relation1 and Relation2 at $\tau_1 = 1$ and $\tau_2 = 2$. The matching attribute is the ID of Relation1 and Relation2. Additionally, the entry $\mathit{Relation}1(2, I, C, 2)$ is not included because it does not meet the ts criterion, becuase it only activates at timestamp 2.

Table 4.4: $Relation1(1) \bowtie Relation2(2)$

ID	V1	V2	V3
1	J	C	G
2	B	H	L

Time-variant Cartesian product

The Cartesian product of $R \times S$ returns a set of data items from tuples R and S in which does not required a matching attribute. The time-variant version of the Cartesian product needs to include in what times τ_1 and τ_2 the tuples need to be included in the operation. $R(\tau_1) \times S(\tau_2)$ returns all standard Cartesian product tuples that are reduced by using Equation 4.2 for τ_1 and τ_2. The Cartesian product in Table 4.5 joins Relations1 and Relation2 at $\tau_1 = 1$ and $\tau_2 = 0$.

Table 4.5: $Relation1(1) \times Relation2(0)$

ID	V1	V2	V3
1	J	C	F
2	B	H	F
1	J	C	E
2	B	H	E

This extension to the standard relational algebra can be used when the data in a database contains additional information about the time of activation of a respective tuple.

4.3 Online Data-Processing System

The SEMSim (now CityMoS) Cloud Service, a simulation cloud service for agent-based simulation with special focus on traffic simulations, has been presented in Chapter 3. In this chapter, the SEMSim (now CityMoS) Cloud Service is used as the reference model for giving context and for describing the proposed model. However, the methods developed are not SEMSim (now CityMoS) specific in any way. In the next section, we give a brief overview of the Cloud Service [45] and its extensions. This is a functional implementation and extension of the reference architecture for an agent-based simulation cloud service presented in Chapter 3.

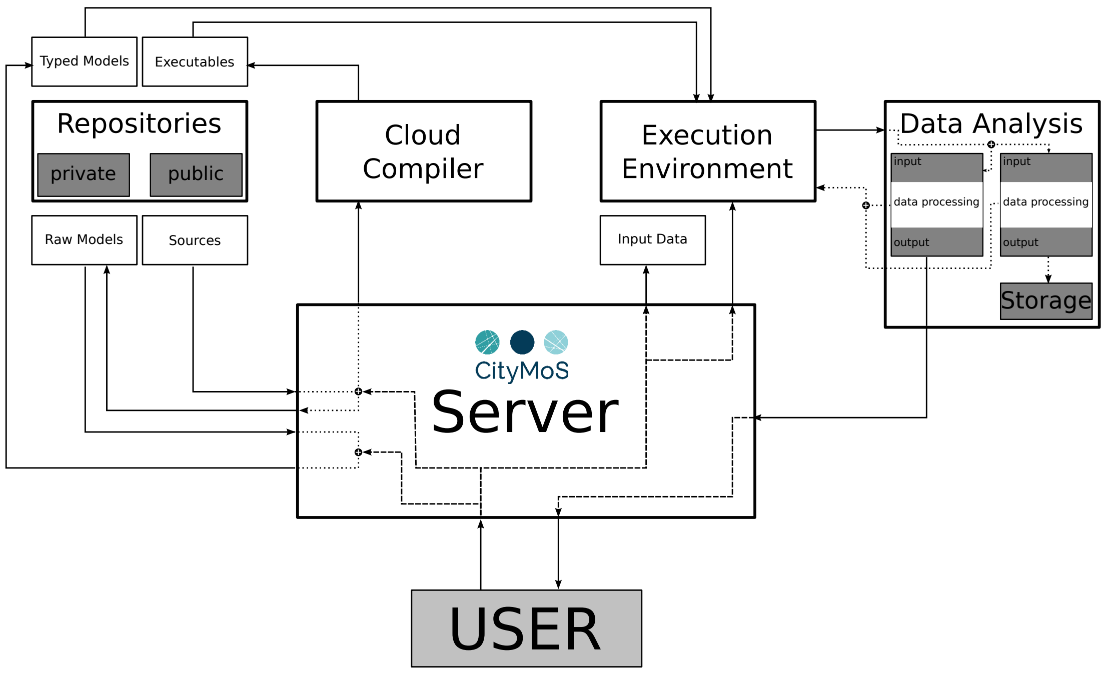


Figure 4.1: Cloud-based Simulation Reference Architecture. The user interacts with the system through RESTful APIs and the entire system runs on public or private cloud instances.

4.3.1 SEMSim Cloud Service

The cloud service consists of 6 main components as shown in Figure 4.1, of which the *Data Analysis* component is the focus of this chapter.

1. The **user-interface** is represented through REST APIs. Through the use of APIs and non-fixed front-end, the user can tailor a use-case-specific front-end for the simulation experiments.

2. These API calls are then translated by the **cloud service server** component which executes cloud service actions like allocation and deallocation of cloud resources as well as starting, monitoring and stopping of simulation experiments (comparable to the *dispatch server* in Chapter 3).

3. A **repository** for *raw* models (algorithms with parameters), *typed* models (parameter configurations for models), source code (for implemented models and simulation engine) and executables (compiled source code) is the main component of the cloud service. It can offer private repositories to privileged users or repositories that are publicly available

to all user. This allows different domain experts to develop, validate and test models before usage by a non-domain expert.

4. The **cloud compiler** takes the source code from the repository and compiles it into an executable that can be used by the execution environment.

5. An **execution environment** is the heart of the cloud-based simulation service. It executes an executable from the repository together with *typed* models and input data to form a simulation run (in comparison to run-environment in Chapter 3).

6. **Data Analysis** is an integral part of any simulation experiment. It is connected to the execution environment and uses the data being generated by a simulation run to deduce results by analyzing it directly. This data generated can be transferred to persistent storage or forwarded as a stream to a visualization endpoint. The size can range from megabytes for infrequently changing data to terabytes for a single simulation run. This depends on the configuration of the simulation experiments.

The complete system relies on public cloud resources allocated from cloud service providers (e.g., GCE[1], Amazon AWS[2], Microsoft Azure[3]). These resources, mostly virtual machines but also virtual networks and virtual storage, are allocated according to the specifications of the experiment and deallocated after the experiment has finished. The data generated is stored in the cloud as well. In order to guarantee the security of the input data into the simulation, all data is encrypted until used in the simulation itself. Generated output data can also be encrypted, if required. The focus in this section is on the data analysis part which is described in more detail in the next section.

4.3.2 System Design

In the proposed system, the simulation instance's output data is first formally defined in relational algebra as presented in Section 4.2.1. This allows for leveraging on the effectiveness and power of relational algebra in simulation data manipulation. Once the simulation is running, a high performance simulation can output data in several ways. The naive and standard approach is to output all state variables of all agents whenever they change. However, by doing so, the bandwidth required would be very high (as illustrated in Section 4.4.1). A better strategy

[1]`https://cloud.google.com/compute/`
[2]`https://aws.amazon.com`
[3]`https://azure.microsoft.com/`

would be allowing external programs to subscribe to updates of variables as required for the analysis. This means that subscribers either get an update of the variable whenever it changes, at certain intervals, or, if sufficient for the analysis, just once at initialization (e.g., start of the simulation). This is a contrast to always receiving all data, whenever a state changes in the simulation. A constraint is that the update interval, in which state changes can be received, should not be smaller than the time between two events that change an attribute in the simulation (e.g., if the simulation event occurs every 1s then an update frequency of 500ms is possible but would waste bandwidth by sending each value twice).

This data-stream is then received by stream processing scripts. As the data is received in batches (e.g., every 1 second), and in temporal order[1], an event-based processing framework that instantiates a new thread/process is very favorable. This also allows for scaling the processing to multiple nodes, due to its limited dependency.

Once the data has been processed by the scripts (which may even be a chain of different processing steps), this final resulting data set is either stored in persistent storage, or sent to a visualization client.

In order to extend a simulation engine in the cloud with a stream processing engine, the simulation engine needs to provide an API that exposes the state of the simulation to an external data sink. This API should present the data in such a way, that once requested, each change in the agent's state will be transmitted to the requester. This process continues for as long as the simulation is running, or until the request is specifically terminated by the requester. Additionally, the data sink should have some form of caching to reduce the amount of data being transfered at every change. The actual processing can be done using existing stream processing tools like Hadoop or Spark, as introduced in Section 4.1, which distributes the data to be processed to one or multiple worker nodes.

4.3.3 Implementation

The data request scripts that define what data is streamed from the simulation are, for ease of use, in an SQL-like language called *SimuSQL*, to ensure compatibility with offline analysis of the data. An example query on the output data-model of a traffic simulation requests data of electric vehicles (EV) that have the time variant property *State-of-Charge* (SOC) at an update rate of 1 second (see below).

```
SELECT * from EV WHERE SOC>0.3 AT 1 SECOND
```

[1]note that the transport protocol can change the receive order

The extension `AT 1 SECOND` is added to specify how often an update of the data should be provided by the simulation. When omitting this parameter, a new data item is sent whenever available. The related concepts are described in more detail in Section 4.2. The data analysis middleware should take care of static/non-changing content and inject this accordingly, to ensure that the simulation can omit that data if unchanged. This can be described as *transparent caching*. The receiver gets the entire dataset in order to not break the (post-)processing toolchain.

The most important part of the cloud service, in the context of the proposed data analysis system, is an interface which the data analysis middleware provides, that the processing framework can access and request data from. This interface (connection towards the *Stream Processing* application in Figure 4.2) needs to be separate from the simulation itself, but should use locality in the data center to ensure fast data transfer (e.g., infiniband/10GE) from the simulation (connection towards the *Simulation* in Figure 4.2). The middleware accepts requests in the above defined *SimuSQL* and returns data when available. It also takes care of the transparent caching of the data. The connection to the simulation has to be via open socket connections that support fast data transfers.

As can be seen in Figure 4.2, the data-processing starts with issuing a request towards the data analysis middleware component (step ①). This can be done by using the *SimuSQL* `SELECT` statement from above. Note that the user initiating this data-processing workflow can be different from the user that designed or started the simulation experiment. The `SELECT` statement is furthermore the trigger to instantiate a connection between the simulation and the data analysis middleware. For this specific call, the first callback will be towards the initialization function of the data analysis middleware (step ②). This is different from any other succeeding callback, because the simulation will transmit the entire current state. This includes, unlike all other callbacks, the static data as well. All following callbacks will only include the time-variant data, as well as the key to identify the tuple unambiguously (step ③). Should a new agent be added to the simulation and should it satisfy the conditions stated in the initial request, its data will be fully included. The data-analysis middleware has to ensure that the static data from the initial data set is merged with the time-variant data stream. After the data has been prepared, the entire data set is presented to the stream-processing toolchain. At the end of processing, data can be stored as a result data set or streamed out to a different data-sink. Since there is no historical data (e.g., from previous simulation states) injected into the stream of data, the processing component needs to take care of passing values from one step to the next should such data be required for that analysis process. The data will be sent from the simulation to the data analysis middleware and passed to the processing component until a cancel request is sent (step 4).

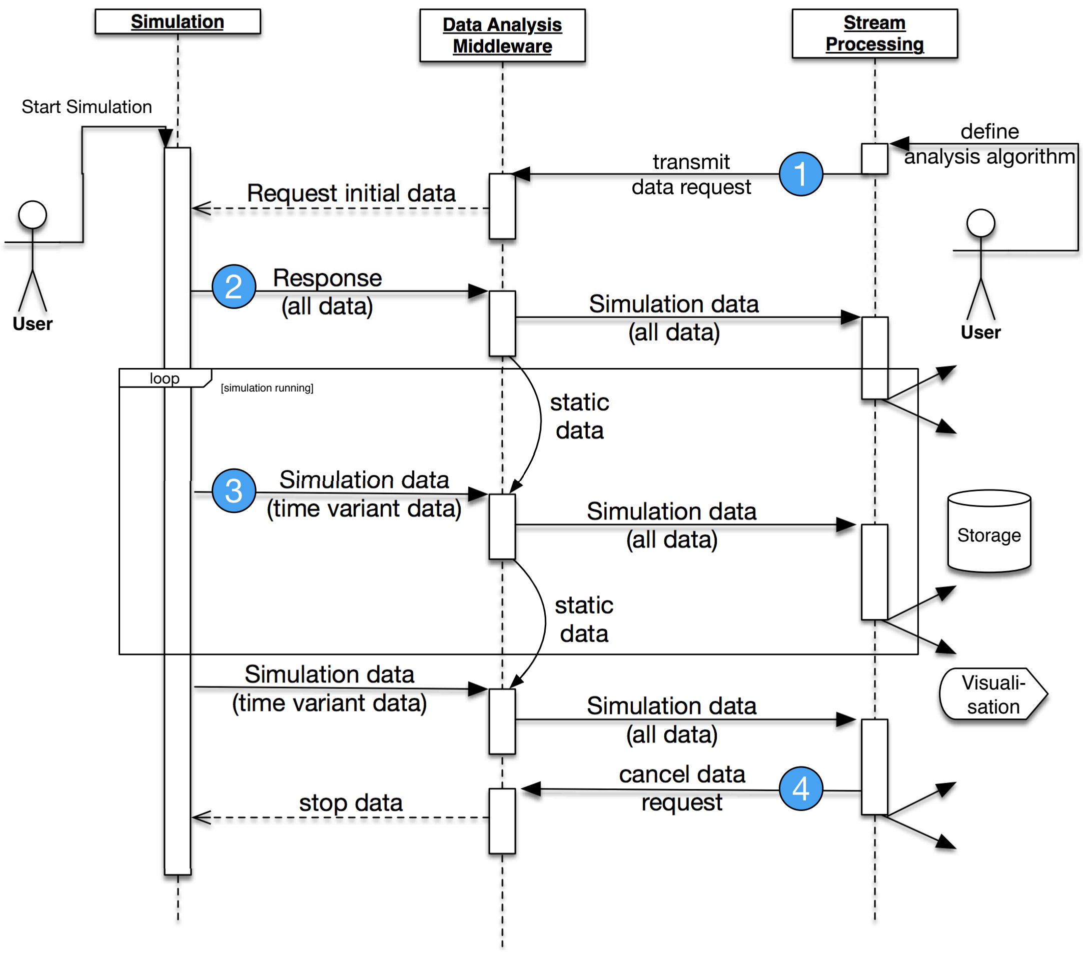


Figure 4.2: Sequence diagram depicting the workflow of the data analysis including the data processing. The Data Analysis middleware combines the time-variant data and the static data and passes it on to the analysis processing. Finally the result is sent to storage or visualization.

Data Order Description

The data transmitted from the simulation is not a constant stream of data. Depending on the execution speed of the simulation itself, the output will change once the state of an agent changes or a new agent is added to the simulation. For example, the position of all vehicles in a traffic simulation, will be updated rather frequently (in the range of a few seconds), while the occupancy of a car park will be updated more infrequently (in the range of a few minutes)[1]. The data-analysis middleware needs to know when all data is in the same logical time step in order to pass this block of data to the processing component. In order to handle this, *landmarks*, a concept proposed by Gama et al. [79], can be introduced into the stream by the simulation. Those are meta-information from which the receiver, in our case the data-analysis middleware, can deduce the end of a concurrent data stream of information. This combats the problem, that between 2 discrete simulation time steps each agent's information is at maximum transmitted once, because the full state of a simulation needs to be consistent at the processing side.

Figure 4.3 shows how the landmarks are inserted into the data stream of serialized data objects from the simulation. Each rectangle represents information from a specific agent instance in the simulation (instances of a class). The *Entity Types* are different classes in the object-oriented framework in a simulation. The order in which entities of different types are put on the stream can vary (see A and B in Figure 4.3), as long as the same state variable of an agent is not updated twice between 2 simulation time steps (landmarks). It can also happen that for different entity types, the amount of data varies. Also, not all entities are updated between 2 landmarks as for more infrequently updated agents (compare B and C in Figure 4.3). The data-analysis middleware needs to keep track of this. This is done to ensure the reduction of bandwidth usage by avoiding unnecessary transfers of information.

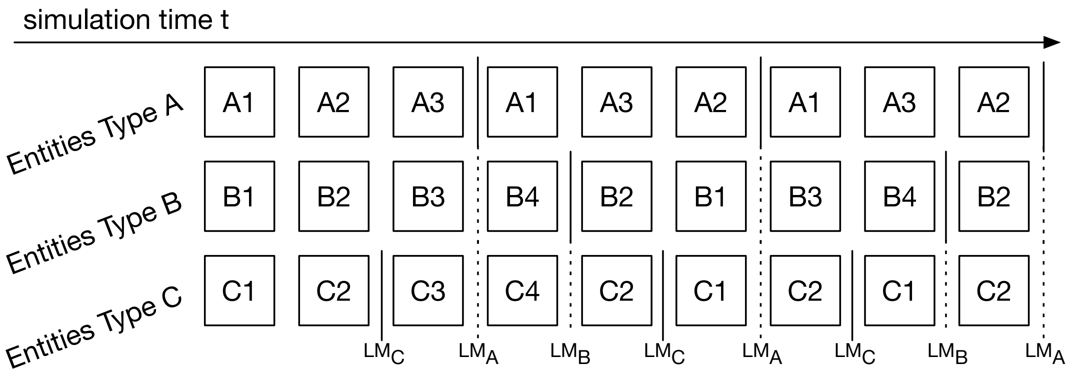


Figure 4.3: The Simulation inserts landmarks into the stream of updates of state changes for each entity type. The data analysis middleware detects the landmarks and discards all old information from the changed entities and replaces it with the newly-received information. Should an agent's state not change, the previous value is kept by the data analysis middleware.

To ensure data consistency between the simulation data model and the output data model,

[1]Note that all time information is simulation time, not real-time

the landmarks play an important role. With regards to the entities that have not been updated since the last landmark, their old values are still to be considered valid, which means they are transmitted to the stream processing component over again, using the data from cache. Between different entity types, the time stamp of the landmarks for each type is compared by the middleware component in order to send a consistent set of data to the processing component.

4.4 Case Study

The necessity for reducing the amount of data stored in persistent storage is presented by first evaluating the bandwidth requirements in which data needs to be written to a storage medium and what current storage technologies support. The measures are the number of agents and the actual simulation speed. Secondly, an example on how the workflow can be used in an actual cloud-based traffic simulation scenario is given.

4.4.1 Data Amount Model

In the following section, a comparison between (different) suitable data-processing frameworks and the amount of data typically generated by simulation is conducted. For this, the writing speed of conventional storage mediums like HDD or SSD as well as network-based distributed data processing solutions is considered. The writing speed is important, since it either slows down the simulation when written inline with the simulation execution, being a blocking system call, or the experiment prolongs after the simulation has ended, since the queued up write commands have to be finished. Both options will result in longer experimentation times and ultimately in higher costs. The best result, that can be expected, is then the writing speed of the medium is similar to the output rate of the simulation. First, a model to estimate the amount of data that is being generated by a simulation is developed.

For this, the assumption is made that for an agent-based simulation with $n_A \in \mathbb{N}_{>0}$ number of agents, the update frequency of the agents t_s (in seconds), the run time of the simulation T (in seconds) as well as the function $g(\tau)$, the percentage of agents updated at a given time. Additionally, the run-time performance as a *real time factor RTF* (how much faster than real-time) needs to be considered, as well as the number of state variables $s \in \mathbb{N}_{>0}$ each agent can have. An exemplary function $g(\tau)$ can be seen in Figure 4.4.

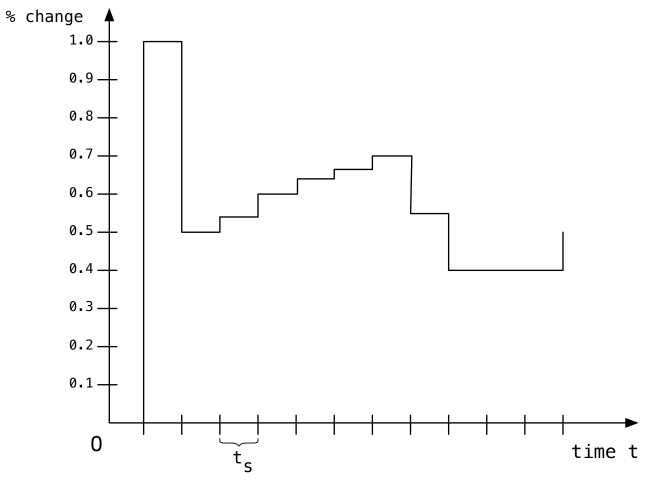


Figure 4.4: Time-variant percentage of agent's state changing in the simulation. t_s describes the time between subsequent simulation steps.

The final mathematical model to determine how many values will have to be saved or transmitted is:

$$TotalValues = \sum_{t=t_0}^{T} g(t) * n_A * s \tag{4.3}$$

All variables are double values on a 64-bit system, the memory footprint is different from the space required when outputting into CSV format. The precision $p \in \mathbb{N}_{>0}$ needs to be regarded as well, since this will influence the number of bytes that need to be written to disk. If the precision is 10 digits, a string representation of the number with ASCII coding [80] would need 11 bytes, due to the decimal point. This is 3 bytes more than that in the computer's main memory.

$$CSVsize = TotalValues * (p + 1) \tag{4.4}$$

In order to evaluate if a certain storage medium can be used, the specification of common hardware write speeds need to be collected. For this, conventional hard drives with a writing speed of 80 - 160 MB/s as well as solid-state disks with a writing speed of 400 - 1000 MB/s are considered. The speeds mentioned are for single disks and not for a redundant array of independent disks (RAID) or other storage solutions/technologies (e.g., NVME or 3D XPoint). Additionally, state-of-the-art network transfer speeds for local networks need to be considered, since the data analysis is not done on the same machine as the one the simulation is being executed on. In case of executing a large-scale, multi-agent simulation that might need more memory, while processing the results in statistical application might be more CPU-bound. Therefore, different machines are required to efficiently work. Current data center interconnection technologies are Ethernet, Infiniband or fiber-optical connections. They range from $10 - 40GB/s$, $56GB/s$ to $> 100GB/s$ respectively. The output bandwidth in bytes per update is defined as

$$OutputSpeed = \frac{CSVsize}{\frac{T}{t_s}} * RTF \tag{4.5}$$

Where the $CSVsize$ is given in bytes and the simulation time T and time step size t_s are given in seconds, and the RTF as no unit. The bandwidth in bytes per second is obtained by multiplying the $OutputSpeed$ by the number of updates per second, which is $1/t_s$. With the overhead of representing numbers as a set of individual characters, the transmission of data over a network connection has some protocol overhead. The more abstraction layers there are between the data representation and the actual bit-level, the more overhead is introduced and the actual transmission speeds not only depends on the performance of the network interface, but also from the processing power of the layers above. For a TCP-based transmission model, as found in most applications, the *maximum transmission unit* (MTU) has the most influence on the overhead, since more byte per packet reduces the ratio between the fixed size protocol data (header) and the payload. For standard installation, the MTU is usually set to 1500 which results in a relative overhead of around 5%. In a data-center environment, larger MTU sizes might not be as much of a problem, since the environment is well known and the interference which can lead to dropped frames and resending of information, is minimal in comparison to a open network environment like WLAN. Conventional ways of processing simulation-output data also requires the system to access data from main memory. There are also limits on main memory of a single system. A distributed processing workflow (e.g., Apache Hadoop [69]) has to be used in such cases. This allows the workload to be distributed to multiple worker nodes and use a single aggregation node to combine the results. This works very well, when the data has little or no dependencies. The developed model for estimating the amount of data produced buy a simulation can help to design the right data analysis workflow for an agent-based simulation.

Data amount example

In a simple example where the number of agents in a 24-hour simulation with each agent having 25 state variables that change throughout the simulation is varied from 5000 to 800000, the time step is 1 second and the RTF is 100. Over the course of the simulation, with each time step 50% of the agents change and their state and the change has to be outputted.

This would result in $87.55GB$ of total data, assuming a precision of 16 digits, and $1.04MB$ per simulation second when considering 5000 agents at real-time execution speed. Multiplying it with the RTF of 100, it results in $103.76MB/s$ that have to be written to storage. This would be possible with all the listed storage options in Table 4.6, especially because sequential writes and reads in a constant stream output usually yield higher speeds than random access. The introduction of more agents into a simulation will have the effect that the simulation data cannot be written out anymore and the system's capability is exhausted. This assumes that there is no non-linear dependency between the agent amount and the data generated. In a simple example when more agents are present in the simulation there might be more interaction and more data being generated through state change.

Table 4.6: Agent count vs. writing speed

	5000	8000	80000	800000
	103.76 MB/s	166.02MB/s	1.66GB/s	16.6GB/s
HDD	✓	-	-	-
SSD	✓	✓	-	-
10G Ethernet	✓	✓	✓	-
56G Infiniband	✓	✓	✓	✓
100G Fibre	✓	✓	✓	✓

In the system presented, the large data would only be transferred from the simulation to a distributed processing workflow.

This means that the large amount of data generated by a simulation is streamed to the processing systems using high-throughput network communications. In contrast to hard-disk-based approach of storing the data, where after the simulation concludes the post-processing step needs to wait for all data to be written, in a stream-processing approach the data is concurrently processed and the final result is ready immediately after the simulation ended. Once the simulation output data bandwidth exceeds the capability of the medium, the I/O presents a bottleneck to the simulation experiment.

4.4.2 Traffic Simulation Cloud Service

The Scalable Electromobility Simulator (SEMSim) (now CityMoS) simulation engine [55, 61] is used in the SEMSim (now CityMoS) traffic simulation cloud service. It is an agent-based traffic simulation engine which is part of a platform, that also includes a power system simulation [62].

This allows researchers to study the holistic effects of electromobility on an entire city/region through simulation.

Within the SEMSim traffic (now CityMoS) simulation, each agent is represented as driver-vehicle-units (DVU) which consists of driver behavior and vehicle component models. Depending on the specific agent, the actual models can vary. The simulation uses a hybrid time-stepped and event-based execution model. Specific events like agent movement have predefined intervals, whereas other events, like the decision-making of the driver or the update of the air-conditioning, are scheduled at different intervals. This gives the flexibility to vary the update of certain models more frequently than others. The following example extracts the data from an electromobility study [62], where the objective is to determine the locations and State-of-Charge (SoC) of all electric vehicles which battery has a capacity of less than 30% of the capacity they started with. For this experiment, the simulation data model needs to be determined, followed by determining the correct time-variant relation algebra equation which is subsequently transformed into an *SimuSQL* query.

4.4.2.1 Traffic Simulation Data

In nanoscopic agent-based traffic simulations like SEMSim Traffic (now CityMoS), the agents are the vehicles and their driver. The behavior of the driver directly influences the state of the vehicle. For example, the decision by the driver model to increase the velocity directly influences the acceleration, fuel consumption and, ultimately, the velocity of the vehicle itself. Since the vehicle is not the only entity in a traffic simulation, the output data representation also needs to consider other entities:

- *Roads*: Consist of links and lanes [61], used for routing and movement.
- *Car Parks*: Hold a certain number of vehicles.
- *Traffic Lights*: Regulate the traffic at intersections and control the flow of vehicles.
- *Other Infrastructure Components*: including Bus stops, Crosswalks and Tram/MRT stations

An agent-based traffic simulation is usually modeled in an object-oriented framework. All the entities can have inheritance to child-models with larger amounts of data (see Figure 4.5). A vehicle only contains the very basic attributes like speed, location and geometric dimensions, while more specific vehicle implementations (e.g., electric, fuel-cell, ICE) can have more specific

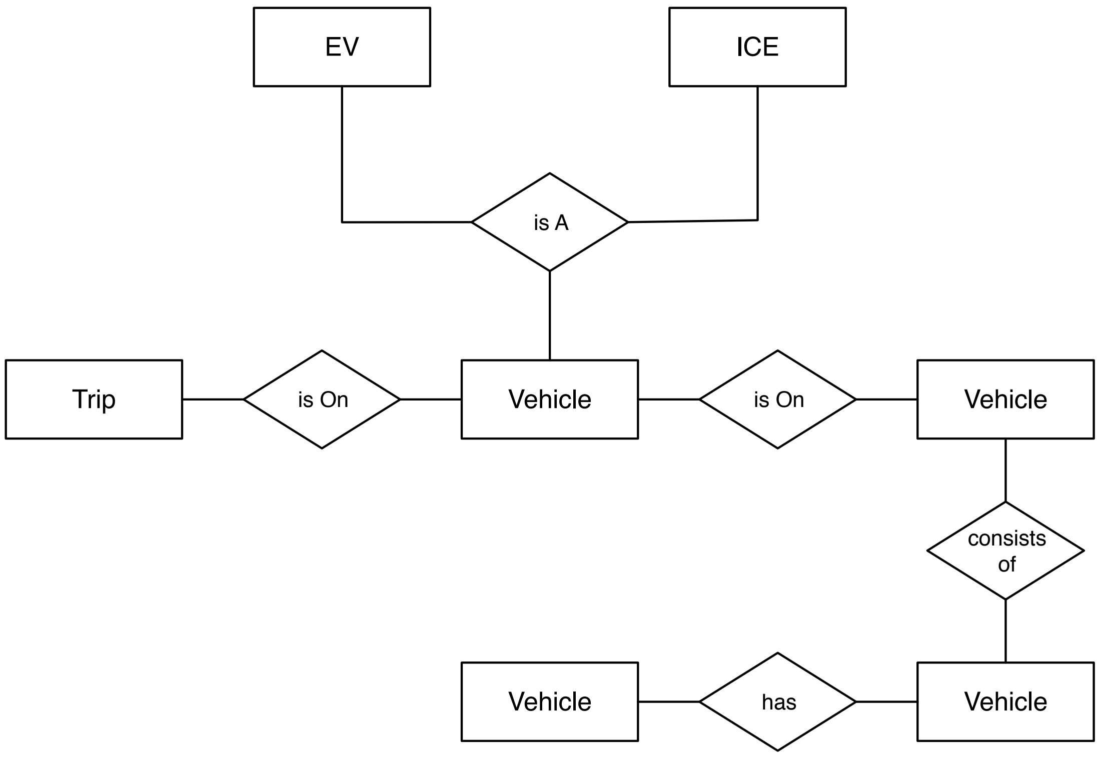


Figure 4.5: ER output data Model for a agent-based traffic simulation

model attributes that influence the speed or location. The same applies for driver behavior. The output data relations for this example, containing only a subset of all entities in the simulation, would be:

```
Vehicle(vehicleID,geometry, velocity(t),location(t))
EV(vehicleID,batterySize,currentCapacity(t))
ICE(vehicleID,MotorPower,tankSize, tankfilled(t))
Road(ID,name,startpoint,endpoint)
Link(ID,startpoint,endpoint)
Lane(ID,startpoint,endpoint,leftlane,rightlane)
CarPark(ID,location,totalSlots, freeSlots(t))
Trip(ID,vehicleID,FromLocation,ToLocation)
```

This relation scheme needs to be known to the user that is developing the processing scripts, because these are the relations a *SimuSQL* query can return results for.

4.4.2.2 Time-variant Relational Data Model

The data required from the simulation is the vehicle location as well as the information of the SoC. A time-variant relation algebra expression would look like the following when considering that t is the current time t_{now}:

$$\text{LowSOCAgents}(t_{now}) = \text{Vehicle}(t_{now}) \bowtie \sigma_{currentCapacity/batterySize<0.3}(EV(t_{now})) \tag{4.6}$$

$$\Pi_{vehicleID,location,batterySize,currentCapacity}(\text{LowSoCAgents}(t_{now})) \tag{4.7}$$

This projection will return the vehicle ID, the location of the vehicle, the current capacity of the battery as well as the total battery size. Due to the selection in equation 4.6, the natural join only includes those agents that have an SoC of less than 0.3. The SoC is defined as the current capacity of the battery over the maximal battery size. The equation 4.7 applies a projection on the result tuples of equation 4.6. This is the formal description and operations that can be used to optimize queries using time-variant relational algebra. The following section expresses the same query as an *SimuSQL* statement which is used by the processing script to request data from the data analysis middleware.

4.4.2.3 Stream Output for Traffic Simulations

After we have introduced the data that a traffic simulation is generating, and the formal time-variant relation algebra optimization that has been performed, the translation into a *SimuSQL* query and the workflow at the data analysis middleware will be shown now.

A *SimuSQL* query to obtain the location and the SoC of all vehicles in the simulation that have less than 0.3 would be:

```
SELECT vehicleID,Location,currentCapacity/batterySize as SOC from
    EV natural join Vehicle WHERE currentCapacity/batterySize <
   0.3;
```

This will return a time-variant tuple of all vehicles containing the location as well as batterySize as a fixed value whenever a vehicle has an SoC (currentCapacity/BatterySize) of less than 0.3. Since there is no update interval (e.g., `AT 1 SECOND`) specified, the data is streamed whenever the value changes in the simulation. This query will be forwarded to the data analysis middleware where it is translated into a statement requesting information from the vehicle object as well as the inherited properties from the EV object in the simulation. The first request will transmit the entire state of the simulation regarding the vehicles and the EVs to the data analysis middleware. The transparent caching capabilities in the data analysis middleware will then cache the time-invariant variables, like the `batterySize` in this case and pass the entire data stream to the processing engine. Here, the processing of spatial information of vehicles is performed and the result of clustering “low SoC agents” is transmitted to the database and/or pushed to a visualization engine, which eventually displays this information. In parallel with the processing, the simulation continues running and data continues being sent to the data analysis middelware. With any subsequent stream of data after the initial one (containing the `batterySize`), only the primary key, in this case the `vehicleID`, together with the time-variant information (e.g. currentCapacity) is forwarded. In the above example, data analysis middleware takes care of inputting the `batterySize` from the transparent cache and forwards it to the processing engine. To the processing engine, the simulation looks like a normal SQL database; however, in the background the simulation is changing the data constantly, while the cloud service is trying to minimize the data transmitted between the simulation and the data analysis middleware and the processing engine.

4.5 Summary

In this chapter, an approach to reduce the data generated by simulation experiments and to couple the data analysis to the simulation execution has been presented. This approach is especially useful for cloud-based simulation experiments, where the costs of running simulation experiments is low, but long-term storage and transfer of large amounts of data can be expensive. The proposed system uses stream processing of data during its generation and a relational data model of the possible output data generated by a simulation experiment. This enables experiment designers to formally model the simulation output data with a time-variant relational data model.

A formalism that indicates time variance on a simple relational model, where all operations have to regard the time at which the data was created or becomes active, has also been proposed in this chapter. Since this method introduces another layer of middleware into the simulation work-flow, the middleware component presents a possible bottleneck to the system. Experimental studies could be used to evaluate the real world performance, but since the middleware discards any historical data, the execution, even of complex queries, is expected to be fast.

Possible use-cases of this model is an online data visualization application that can show results, and possible interactions with the simulation while it is still running. (Semi-)manual exploration of different simulation and model configurations could be much more engaging and fruitful. This can be used for visualization of the data but also for decision-support systems that deliver results while the simulation is still running.

Chapter 5

Cloud-Based Search Space Exploration for Traffic Simulations

Contents

Large parts of this chapter have been published in the proceedings of the *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation 2015* [81].

5.1 Introduction and Motivation

Within the thesis, different aspects of enabling non-researchers, domain experts and policy makers have been shown. In this chapter, the focus lies on speeding up the search-space exploration of large domain spaces using multi-resolution modelling and hybrid-resolution simulations. This is possible, since Cloud Providers (CPs) offer the cloud resources in seemingly unlimited quantities, which can be used concurrently for search-space exploration.

When using cloud resources, the cost for ad-hoc (single use) is slightly higher than using a specific instance for longer times. CPs offer sustained-use policies to give users the incentive to use the resources for longer times. These policies have different implementations. The Google Compute Engine (GCE), for example, offers a pricing philosophy that automatically reduces the cost-per-minute of a machine that runs more than 25, 50, 75 per cent of the month. For this to take effect, it is not the same machine that need to be running, but rather all machines that are of the same type[1]. Amazon's AWS takes a different approach with discounting users for prolonged use. They offer *Reserved Instances*, which are an upfront payment for 1 or 3 years of use. For this period, the price is significantly lower than for ad-hoc instances. The pricing structure needs to be taken into consideration when designing large-scale search-space exploration experiments.

In order to lower the cost, by reducing the runtime of a single simulation configuration, the multi-resolution modelling and simulation approach can be very helpful. It can offer significant speedups to the simulation runtime while at the same time not significantly sacrificing on the simulation results. This can be used to identify interesting combinations of values in a multi-dimensional search-space exploration where the entire search space can be explored in a series multi-resolution simulations and only regions of interest in the search space in higher, but computationally more demanding, resolution simulations. A qualitative visual example can be seen in Figure 5.1. An approach that is in comparable is the use of surrogate models. They are used as a drop-in replacement for more computationally demanding models but also lower the accuracy of the results. Multi-resolution simulations are different they combine the lower and higher resolution models for the same aspect of the simulation, while surrogate models only offer the a single resolution. One could see the lower resolution models in the multi-resolution context as a surrogate model of the higher resolution model.

[1]`https://cloud.google.com/compute/pricing`

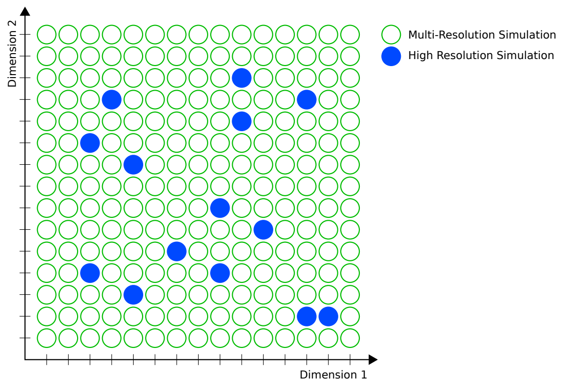


Figure 5.1: Simulation search space with 2 dimensions with regions of interest identified to be simulated in higher resolution

This chapter illustrates the findings of using multi-resolution modelling and simulation for a transportation scenario of switching between a macroscopic and nanoscopic traffic model.

Since for agent-based traffic simulations the level of detail is crucial to the system's runtime performance as well as the fidelity of the results, different model abstractions have been used through out literature to find the best model for a specific use case. Each of the models (macroscopic, mesoscopic, microscopic and nanoscopic) have their use-cases and benefits. Microscopic traffic simulations have a high level of detail but at the same time require a large amount of computational resources. In a large traffic network of a mega-city or an entire country, the use of a complete microscopic simulation is a difficult endeavor. The resources required to do so have, for most use cases, no relation to the actual outcome. A hybrid traffic-simulation model as it has been proposed within this chapter that uses both, a high-resolution agent-based microscopic simulation alongside a lower resolution flow-based macroscopic simulation for specific road segments, might be a solution. Since there are regions of a road network of an entire city, country or region where a lower level of detail is sufficient, computing resources can be saved. In such cases, a multi-resolution model may offer a means to improve processing efficiency while only marginally affecting the overall simulation fidelity. Such region could be a stretch of highway between on and off-ramp or long country roads connecting cites or villages, but other city-roads could be applicable as well. The problem with using different simulation models is the fidelity at the boundary between such simulation models. This fidelity discrepancy is caused by the difficulties with aggregation and disaggregation passing through the boundary. For the evaluation of such discrepancies between fidelity and performance increase, a prototypical design of a multi-resolution agent-based traffic simulation has been developed and its performance with regards to computational run-time and fidelity (deviation from highest resolution) has been evaluated.

The findings have been presented at the PADS 2015 [81] conference and have been extended to include an analysis of the performance benefits and result drawbacks for larger, real-world networks of 17 different city regions, extending the single road example presented in [81]. Saving the computational time can directly be translated into saving money when using cloud resources. Therefore, an example pricing model will be presented that takes the different cloud service providers' pricing strategies into consideration for the use of a large-scale search-space traffic simulation experiment.

5.2 Related Work

There have been several approaches to multi-resolution modeling of traffic. Two main approaches towards the topic have been (1) the use of aggregation and disaggregation at runtime and (2) the use of multi-resolution entities.

The first approach, which aggregates a high-resolution model at runtime to a lower resolution and then disaggregates from the low resolution to a higher resolution is trivial on the aggregation side. This is because there is usually sufficient data to average over the existing high resolution agents' states. The disaggregation algorithm on the other hand has to interpolate or create/reconstruct information from the low-resolution model in order to obtain the state information for the high-resolution model. It has been shown that a frequent transition between high- and low-resolution models through aggregation and disaggregation is not a good practice [38]. One problem is the *thrashing effect*, where agents cross between resolution boundaries often, can create a large overhead in computation, especially when running a distributed simulation.

This method of aggregation and disaggregation was initially developed for military simulations [39, 40], since it conforms very well with the command security structure in the military. Battalions, for example, are aggregated divisions and allow manageable view on certain battlefield scenarios for different key personnel in the chain of command, whereas a disaggregated view of individual fighters is necessary for group leaders.

An approach to multi-resolution traffic modeling was discussed by Burghout [41, 23], where a *ghosting* method was used to ease the transition between microscopic and macroscopic boundaries in the aggregation-based approach.

In order to avoid the trashing effects of frequent model switching, Natrajan et al. [42] propose the concept of *Multi-Resolution entities* (MRE). A MRE consolidates the properties of several resolutions in one object. Those properties are then kept consistent by design. This requires a larger memory footprint and individual operations require mode computing cycles for each of the resolutions, but there is no consistency problem.

Another approach to mitigate the thrashing effects, especially for traffic simulations, while avoiding the use of MRE, has been presented by Chua and Low [43], proposing a set of predictive algorithms. These algorithms are used to project an agent's future position on a road segment.

The multi-fidelity modeling approach presented by Choi et al. [44] describes how to convert an existing model into a multi-resolution model. This methodology can be used to increase the simulation speed for a given simulation as well as to give measures for fidelity derivation. While their approach generalizes a methodology, this chapter focuses on using multi-resolution modeling to investigate its potential with regard to its use in high-performance traffic simulations.

5.3 Model description

The multi-resolution model is comprised of a microscopic car-following model with lane changes and a macroscopic traffic-flow model. These models are executed simultaneously and are connected through a model-switching strategy to form a multi-resolution model of traffic on road segments. This section will begin with a detailed description of the two model types, followed by a description of the connecting boundaries between two joining road segment zones. Such connections require an aggregation of agents' states at the microscopic-macroscopic boundary and a disaggregation at the macroscopic-microscopic boundary.

The model uses a discrete time advancement in the microscopic model and a fixed time advancement in the macroscopic model. The multi-resolution model functions as a drop-in replacement for either a macroscopic or a microscopic simulation model. This requires the two different models to have no knowledge of the respective other model and be fully self-sufficient traffic simulations.

5.3.1 Microscopic Model

For the microscopic traffic simulation models, a car-following model that determines acceleration and gap calculation between agents is important to express the traffic behavior correctly. The Intelligent Driver Model (IDM) developed by Treiber et al. [26] is widely used in agent-based traffic simulations. Together with the Minimal Overall Breaking Induced by Lane Change (MOBIL) [82] algorithms that extend the capabilities of Intelligent Driver Model (IDM), this model is sufficient to model the driving behavior on a road segment. The acceleration for each agent is given by Equation 5.1;

$$a_{cur} = a \cdot \left(1 - \left(\frac{v_\alpha}{v_0}\right)^\sigma - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right) \tag{5.1}$$

it uses the maximum acceleration (a) of the agent (α) along with the current velocity (v_α) of the agent as well as the position ($s^*(v_\alpha, \Delta v_\alpha)$), derived using Equation 5.2

$$s^*(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha \cdot T + \frac{v_\alpha \cdot \Delta v_\alpha}{2\sqrt{a \cdot b}} \tag{5.2}$$

as well as distance ($s_\alpha = x_{\alpha-1} - x_\alpha - l_{\alpha-1}$) between an agent ($\alpha - 1$) in front of the agent α, and the velocity difference ($\Delta v_\alpha = v_\alpha - v\alpha - 1$) between agent α and agent $\alpha - 1$. Where s_0 denotes the minimum distance between two agents on one lane, T is the time headway in seconds between agents in normal traffic conditions, and a and b are the comfortable acceleration and deceleration of the agent.

In addition to the IDM and MOBIL features, the underlying microscopic model is augmented to support arbitrary obstacles. Intersections, standing vehicles and moving vehicles are considered obstacles. An agent has to assess whether to adjust the acceleration or to change lanes in order to avoid collision with the obstacle (collision-free model), which can lead to an unrealistic braking behavior. Intersections are a special type of obstacle that span the entire width of the road (multiple lanes) and can have a transparency property that can change dynamically. If the state is set to transparent, agents can pass through and normal car-following models apply. Otherwise, agents will adjust their speed to come to a full stop. This allows for manipulation of the traffic simulation without violating any IDM or MOBILE rules.

5.3.2 Macroscopic Model

The macroscopic metrics described by Hoogendoorn and Bovy [83] are one way of expressing a macroscopic view on a road segment. They assume the traffic density k can be derived, as in fluid dynamics, from the number of vehicles that pass through a certain section of road (vc) in a given time and the length (l) of that road segment (Eq. 5.3).

$$k = \frac{vc}{l} \tag{5.3}$$

This density, together with the average velocity ($\overline{v}$) of all agents within the given time frame, produces the traffic flow q (Eq. 5.4).

$$q = \overline{v} \cdot k \tag{5.4}$$

Since the macroscopic model is not aware of individual agents within the segment, the model uses *detectors* at the boundaries to gain knowledge about entering and leaving agents. The time and state of the macroscopic model is only advanced when there is a vehicle entering the macroscopic zone or scheduled to be leaving the zone. A front detector senses a microscopic agent entering the macroscopic zone and saves its ID as well as the cycle number (simulation time). The information about a vehicle entering is gathered by the front-detector is used to calculate the vehicle's velocity at the end of the macroscopic zone and when a single vehicle should emerge from the macroscopic zone into the next read segment. This is done by updating the state of the macroscopic zone and using the newly changed average velocity $\overline{v}$ and the length of the segment to determine the cycle this vehicle should exit the macroscopic zone through the back detector. This next segment can either be a macroscopic or microscopic model. The front and back-detectors are illustrated in Figure 5.2. The three important metrics for macroscopic

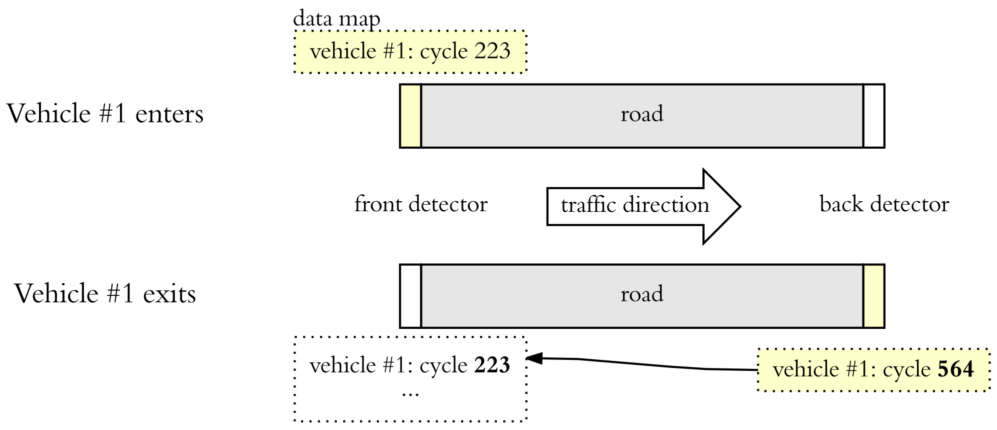


Figure 5.2: Using detectors to record entering and exiting vehicles

traffic state are average velocity ($\overline{v}$), density (k) and traffic flow (q). The average velocity ($\overline{v}$) can be determined solely by the information gathered from the traffic detectors.

$$\overline{v} = \frac{vc \cdot \overline{v}_{old} + v}{vc + 1} \tag{5.5}$$

An update to $\overline{v}$ is done only when the back (or front) detector records a vehicle leaving (or entering) the macro zone. This will implicitly decrease (or increase) the vehicle count (vc) by one, since the vehicle is not (or additionally) present in the data structure map assigned to this macroscopic zone. Because the number of vehicles on the road segment has changed, the k and q values (Eq. 5.3 and 5.4) have to be updated as well.

5.3.3 Multi-Resolution Model Extensions

The microscopic and macroscopic simulations can be run independently in separate simulations. In a multi-resolution simulation, they are executed simultaneously and have to exchange data. The points on a road segment at which data has to be exchanged are the front and back detectors discussed in Section 5.3.2. In our model we use two approaches for aggregation and disaggregation at the boundaries of the different zones. On the boundary between the microscopic and macroscopic zone (front detector), a *position-triggered aggregation* is used. The back-detector's boundary between the macroscopic and microscopic zone uses a *cycle-based disaggregation* scheme. Where one cycle is equivalent to one time step in the microscopic simulation.

There are three different categories in which overall traffic can be categorized. They are dependent on the current traffic density (k). Since they can be observed and/or calculated in macroscopic as well as microscopic zones, they can be used to compare macroscopic and microscopic road segments.

- **Free Flow Traffic** $k \leq k_c$: The vehicles on the road segment are able to achieve their desired velocity by accelerating normally. This critical density (k_c) has been described by Hoogendoorn and Bovy [83] and is the density at which congestion starts.

- **Congested Traffic** $k_c \leq k \leq k_j$: Since traffic density that exceeds k_c leads to congestions, vehicles cannot accelerate as freely as before and might not reach their desired velocity. The vehicles also have not yet come to a full stop, meaning that k would be reaching the jam density k_j. Intermittent braking of agents can be observed. This traffic state has been described by Helbing et. al [84].

- **Jammed Traffic** $k = k_j$: When the density of a road segment reaches k_j as described by Hoogendoorn and Knoop [85]. All vehicles come to a full stop and no movement is possible.

5.3.3.1 Aggregation

In the microscopic area before the aggregation boundary (micro-zone), the simulation checks during each simulation cycle which agents traverse though the defined position where the micro-zone ends and the macroscopic-zone starts. For estimating the projected cycle time for disaggregating the vehicle back into the micro-zone, two operation modes are possible. One is the single-lane mode where overtaking is not possible and the other one is the multi-lane mode.

In the single-lane case, the vehicle's velocity (v) before passing into the macroscopic-zone is taken, together with the length of the macroscopic-zone (l_{macro}) and the duration of one cycle in seconds (t_{cycle}), to determine the number of simulation cycles that have to pass before the vehicle is disaggregated back into the micro-zone at the back detector.

$$\Delta\text{cycles} = \frac{l_{macro}}{v \cdot t_{cycle}} \tag{5.6}$$

When looking at the model for a multi-lane road segment, the aggregation and cycle prediction gets more complicated. Since overtaking is possible, a vehicle's maximum velocity is not restricted by the leading vehicle on the same lane but also by the vehicle furthest away on any of the parallel lanes. Therefore, the projection takes the traffic density (k) immediately after the macroscopic-zone into account as well. This is not required for the single lane case, since the density is used to determine the likelihood of a lane change occurring and the agent disaggregating on a different lane. The three cases of traffic state as discussed above have to be regarded.

For the free flow traffic case, the acceleration (a) is regarded for the entire length of the macroscopic-zone. The total travel time through the macroscopic-zone is determined using Equation 5.7 as a basis and solving it for Δt in Equation 5.8 and ending up with the number of cycles (Δcycles) given through Equation 5.9. The length of the macroscopic-zone (l_macro) can be substituted in for s and the initial velocity (v_0) for v.

$$s = \frac{a}{2}\Delta t^2 + v \cdot \Delta t \tag{5.7}$$

$$\Delta t = \frac{-v + \sqrt{v^2 + 2 \cdot a \cdot s}}{a} \tag{5.8}$$

$$\Delta\text{cycles} = \frac{\Delta t}{t_{cycle}} \tag{5.9}$$

Should the velocity at the end of the macroscopic-zone ($v_{end} = a\Delta t + v$) exceed the desired velocity v_0, the total travel time has to be recalculated by having an acceleration time (Δt_a)

and a constant velocity phase (Δt_{v0}).

$$\Delta t_a = \Delta t - \frac{v_{end} - v_0}{a} \tag{5.10}$$

$$l_a = \frac{a(\Delta t_a)^2}{2} + v\Delta t_a \tag{5.11}$$

The constant velocity phase is calculated using Equation 5.12.

$$\Delta t_{v0} = \frac{l_{macro} - l_a}{v_0} \tag{5.12}$$

Where Δt_{v0} is the time spent in constant velocity v_0. l_{macro} and l_a represent the total distance of the macroscopic-zone and the distance traveled while accelerating respectively.

The cycle count when the vehicle exits the macroscopic-zone is then determined by Equation 5.13.

$$\Delta\text{cycles} = \frac{\Delta t_a + \Delta t_{v0}}{t_{cycle}} \tag{5.13}$$

For the congested traffic state, the acceleration is determined as in the free flow case (Eq. 5.7,5.8,5.9), but then multiplied with a density proportion factor p_{dens} which leads to an adjusted acceleration value $a' = a \cdot p_{dens}$. This is necessary to cope for very slow moving traffic and to counteract unrealistic acceleration and breaking behaviors. It results in the maximum acceleration of an individual agent being proportional to the state of the road. The density proportion factor is calculated using Equation 5.14 with the densities from above.

$$p_{dens} = \frac{k - k_c}{k_j - k_c} \tag{5.14}$$

The cycle count is then determined as it would in the free flow case, except the acceleration is adjusted to a'.

In the traffic jam case, the velocity at the end of the macroscopic zone is 0 and the projection is performed using the vehicle's current velocity only and disregarding any acceleration occurring in the macroscopic zone.

Since both, the single and multi-lane calculation of Δcycles, calculate the time a vehicle spends in the macroscopic simulation and the returned position ($x_{ret} = x + l_{macro}$), which is equal with the back detector's position. The absolute cycle in which the vehicle is put back into the microscopic simulation, after removing it, is determined using Equation 5.15.

$$\text{cycle}_{ret} = \text{cycle}_{now} + \Delta\text{cycles} \tag{5.15}$$

IDM car-following model relies very much on a leading vehicle to determine the new acceleration. When there is no vehicle leading, a free-flow mode is chosen. The free-flow mode in the

IDM model is much simpler and assumes the maximum acceleration. However, once an agent is removed from the microscopic model, a following vehicle loses its leading vehicle to base its acceleration algorithms on (Eq 5.1). The IDM car-following model is very much depended on a leading vehicle, otherwise free-flow traffic is assumed. Free-Flow traffic mode is not applicable to the following vehicle, since it could lead to unwanted acceleration and to wrong overall microscopic model behavior. Therefore, we introduce *ghost vehicles* to mitigate this problem [41]. This allows a following vehicle to determine its correct acceleration values. A ghost vehicle contains only a subset of properties of a "true" microscopic agent and is used to approximate a microscopic model. A ghost vehicle is spawned as soon as a vehicle traverses through the

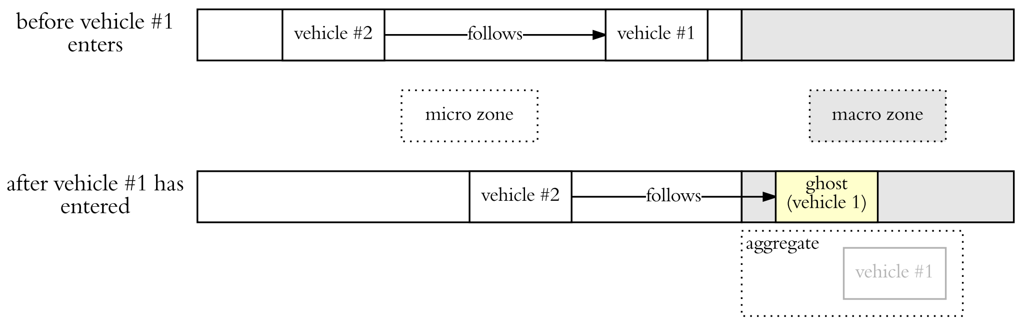


Figure 5.3: Ghost vehicle generation at micro-macro boundary

micro-macro boundary. At this point, the ghost's length, position and velocity are set to the properties of the removed agent's. During each cycle, the ghost's microscopic model states are updated by as if this agent was part of the microscopic model. This tricks the subsequent agent into following into the macroscopic-zone. A ghost moves along the macroscopic-zone until another vehicle enters the macroscopic-zone. At that point the ghost is discarded and replaced by a new one. The movement of a ghost is dependent on the number of lanes (single or multi-lane mode) and the traffic state behind the macroscopic-zone. In single lane mode, the ghost moves along the macroscopic-zone at a constant velocity it had when it was created. In a multi-lane scenario, the traffic state in the microscopic zone behind the macroscopic-zone is used to modify the ghost's velocity. The new velocity $v' = (1 - \text{visc}) \cdot v$ uses a viscosity metric that is derived from the traffic density behind the macroscopic-zone and given by Equation 5.16 (similar to the density proportion factor p_{dense}).

$$\text{visc} = \begin{cases} \frac{k-k_c}{k_j-k_c} & k_c \leq k \leq k_j \\ 1 & k > k_j \end{cases} \tag{5.16}$$

At each cycle the density inside the macroscopic zone is queried and if it exceeds the jam density k_j the ghost is stopped. This leads to agents reducing their speed when approaching the macroscopic zone and eventually stopping before entering the macroscopic zone. Since the macro density is updated whenever a vehicle is disaggregated back into the microscopic model, movement of the ghost continues when k_{macro} falls below k_j. Should the ghost vehicle reach the end of the macroscopic zone before being replaced by a ghost of a newly entered vehicle, it is discarded and the free-flow case can be assumed.

5.3.3.2 Disaggregation

When vehicles are returned to the macroscopic-microscopic boundary, disaggregation has to be performed. There are 3 steps involved to ensure the model integrity, given by the position on the road and the internal state of the agent, is not violated.

1. Depending on the current cycle time, the internal data structure holding all agents currently in the macroscopic-zone is checked, and all agents that have a lower return cycle number (cycle$_{ret}$) are prepared to be returned to the microscopic zone. The data structure is depicted in Figure 5.4

2. If there is a vehicle that is supposed to be returned at the current cycle, the lane(s) are checked for enough free space to insert the vehicle safely.

3. Once a vehicle is supposed to be returned at a specific location, the IDM calculation is executed. It is then checked if an emergency breaking situation would occur. If this is the case, insertion is suspended. This check is not executed when the average velocity of the micro-zone is below 10% of the roads recommended speed, in order to allow for insertion when the traffic state is strongly congested. Alternatively, using the macroscopic metrics k_j and k_c could be used to determine whether this check needs to be performed.

For a multi-lane scenario, all lanes are checked against the free-space and emergency breaking conditions. Should the checks fail for every single lane, the disaggregation is postponed. Vehicles stay in the data structure and are checked again on the next cycle. Upon a successful disaggregation, the vehicles are removed from the data structure and the back-detector is notified.

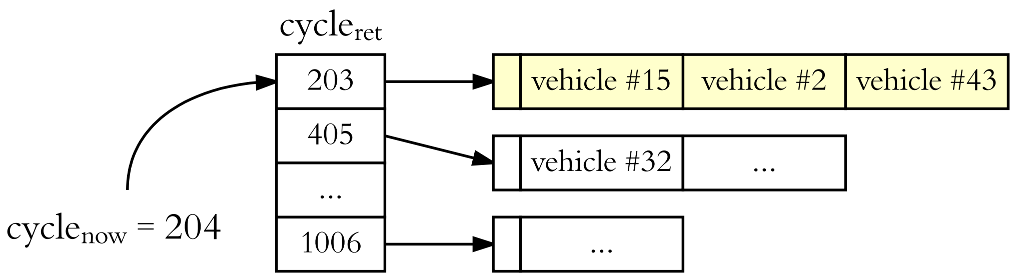


Figure 5.4: Microscopic vehicle date holding data structure

5.4 Model Evaluation of Multi-Resolution Simulation Model

Since the aim of the proposed multi-resolution model is to increase the performance while affecting the model fidelity as little as possible, the multi-resolution performance is compared against a pure microscopic simulation.

5.4.1 Design of Experiments

Three experiments with the following setup in order to establish a microscopic base line for comparison and experiments for evaluating performance and fidelity respectively were designed.

- The **Microscopic Mode** simulates the entire road segment using only the microscopic model. There is no need for model switching and it serves as the base-line implementation used to compare the performance and fidelity (see Figure 5.5a).
- The **Multi-resolution Mode** simulates a road segment as discussed in Section 5.3. Both microscopic and macroscopic models are used and model-switching at the boundaries is executed (see Figure 5.5b). This is done to gain the performance measures that are later compared to the base-line microscopic model experiment.
- The **Dual-Simulation Mode** runs the previous modes simultaneously with identical starting conditions. The mode is used to evaluate the fidelity effects of multi-resolution execution (compare Figure 5.5c).

Four different types of driver-vehicle unit configurations for the microscopic road segments have been chosen. They are normal, timid and aggressive driver of private vehicles as well as a configuration for truck drivers. Each of the configurations have different values for the preferred maximum speed, the time headway, the acceleration and deceleration for normal

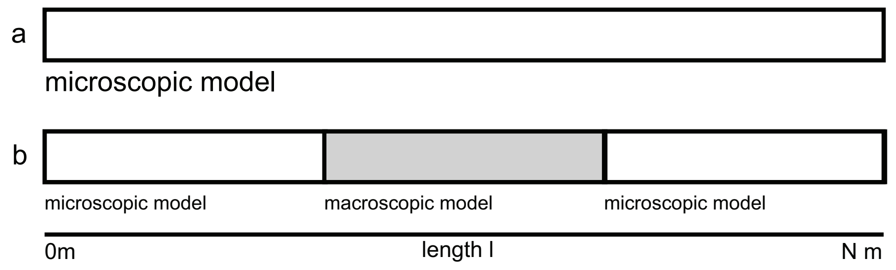


Figure 5.5: Simulation experiment setups

traffic conditions. These values for different types have been identified by Kesting et al. [86]. In order to generate a heterogeneous agent population, the properties are uniformly distributed around the values identified by Kestig et al. with a margin of ±20%.

5.4.2 Model Performance

The goal of this chapter is to evaluate the performance increase when using multi-resolution models for traffic simulations. Therefore, the average processing of one time step in the simulation for a pure microscopic simulation ($\bar{t}_{micro}$) and a multi-resolution simulation ($\bar{t}_{mrs}$) is compared. The performance difference (Δp) was then calculated using Equation 5.17.

$$\Delta p = \frac{\bar{t}_{mrs}}{\bar{t}_{micro}} - 1 \tag{5.17}$$

The experiment runs several (> 20) simulations in microscopic mode (Figure 5.5 a) as well as multi-resolution mode (Figure 5.5 b). Each simulation created 15 agents and stopped them at the same location within each simulation with a non-transparent obstacle (e.g., traffic light at an intersection) to simulate a red traffic light situation, where all vehicles start from a resting position and accelerate after the light turns green. This obstacle is placed in the microscopic-zone before the macroscopic-zone. The setup can be seen in Figure 5.6. Once all agents stop

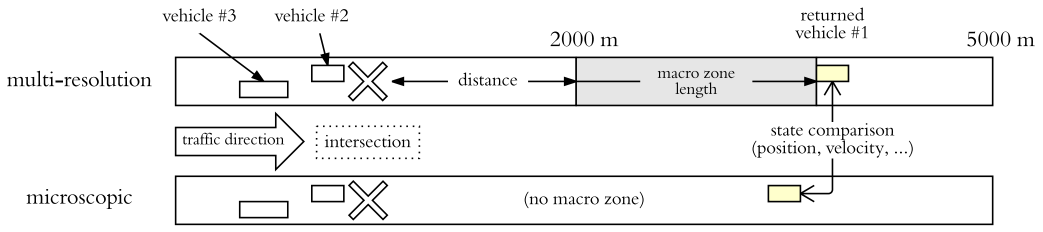


Figure 5.6: Dual-Simulation setup for agent state deviation experiment

fully, the obstacle are made transparent. This allows the IDM to accelerate the agents and move along the road segment. For each simulation cycle, the execution time is recorded and then averaged over the entire simulation until all agents reach the end of the road segment.

A close-to-zero value of Δp indicates that the performance of both simulations is equal, whereas negative values indicates less processing time is spend on a cycle in the multi-resolution simulation. The performances for single- and multi-lane simulations are shown in Figues 5.7

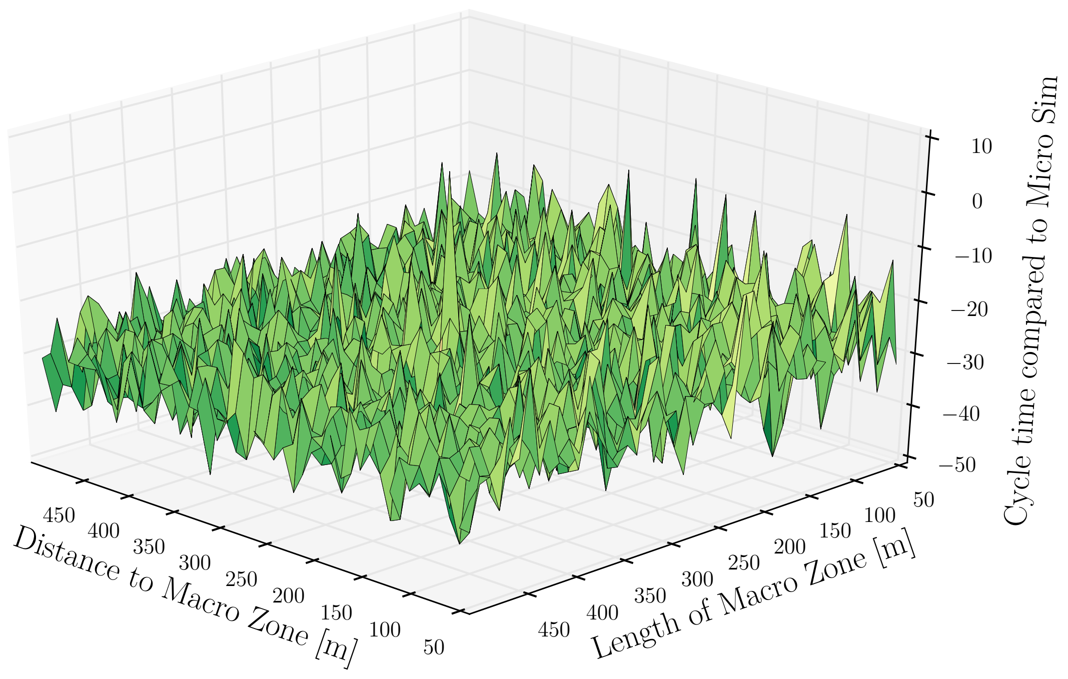


Figure 5.7: Single lane performance comparison: Relative performance between microscopic and multi-resolution mode. Cycle time up to 40% reduced for long macro-zones and long distance to macro-zone.

and 5.8. Each point shows the average relative performance between the microscopic and multi-resolution simulation form more than 20 runs. It shows that for the single-lane simulation the performance increases when the macroscopic-zone becomes longer. An increase in model performance is also seen, even though not so prominently visible, when the distance to the macro-zone gets longer. The same general shape can be seen in the multi-lane simulation experiment. There, the overall performance increase is, averaged over all configurations, bigger(+30% compared to +25%). It also shows that there is a larger deviation from the mean value. The performance increase is bigger due to the more complex model in the pure microscopic simulation. There is of course an overhead due to the model switching mechanism. The mechanism includes the aggregation of the agent at the front-detector, where the agent is

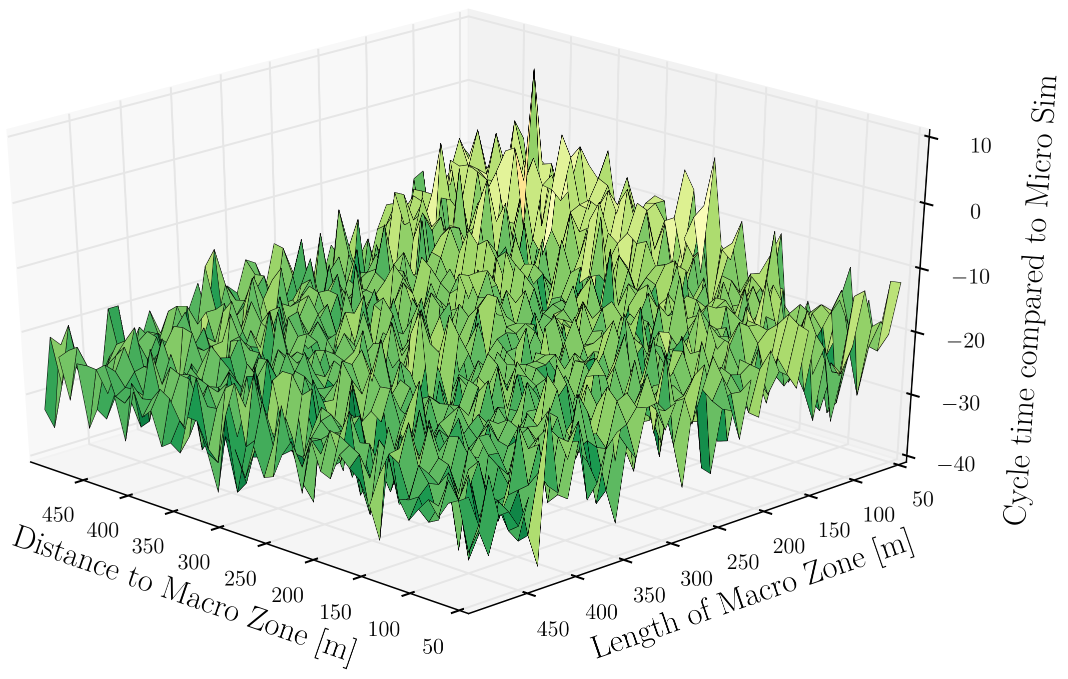


Figure 5.8: Dual lane performance comparison: Relative performance between microscopic and multi-resolution mode. Cycle time reduction lessens when decreasing the length of the macro-zone.

removed from the microscopic simulation, the estimation of the re-entry cycle and the creation of a ghost vehicle; and the disaggregation, where the agents is put back into the microscopic simulation (see Sections 5.3.3.1 and 5.3.3.2). This overhead with regard to the entire simulation run is shown in Table 5.1.

	min%	max%	mean%	σ
single-lane	9.3	14.5	9.4	± 0.39
multi-lane	6.5	10.8	7.6	± 0.74

Table 5.1: Model switching overhead

5.4.3 Model Fidelity

The model fidelity is determined between the multi-resolution simulation and the pure-microscopic simulation. In order to quantify the differences, both simulations with the exact same starting parameters and settings have been executed. This leads to each agent in the microscopic simulation having an exact counterpart in the multi-resolution simulation. Two types of deviations that occurs from the switching and the use of a macroscopic model are evaluated:

- **Agent State Deviation** describes the difference between the individual state variables of an agent (e.g., position on the road).
- **Differences in Traffic Flow** is a more statistical assessment of the influence of the multi-resolution simulation. It is a macroscopic metric used instead of the individual state based agents fidelity.

5.4.3.1 Agent State Deviation

In order to determine an agent's state deviation by the proposed multi-resolution model, two simulations (one micro and one multi-resolution) were used. Both simulation had the exact same parameter. The road length of 5000 meters was used and number of vehicles was set to 100. Vehicles are spawned with a distribution of 3 : 3 : 3 : 1 of the vehicles classes discussed previously. Both simulations contain a non-transparent obstacle (e.g., traffic light) at the exact same position and are only opened when all agents have come to a full stop. This obstacle mimics an intersection, where the vehicles are starting to accelerate at the same time. This intersection is always placed before the macroscopic zone (front detector). In the multi-resolution simulation model a macroscopic zone is placed at the 2000 meter mark. This is not present in the pure microscopic simulation. The experiment setup is illustrated in Figure 5.6 on Page 96 and the distance between the obstacle and the length of the macroscopic-zone are varied as in the performance evaluation above. Once a vehicle leaves the macroscopic zone, its state is logged in each of the simulations and analyzed after the simulation finishes. The root-mean-square for the position (x_{rms}) and velocity (v_{rms}) deviation are calculated.

$$y_{rms} \in \{x_{rms}, v_{rms}\} \tag{5.18}$$

$$y_{rms} = \sqrt{\frac{\sum_{i=1}^{N}(y^i_{mrs} - y^i_{micro})^2}{N}} \tag{5.19}$$

Since the rms value grows with positive and negative value differences, it only gives absolute discrepancy information. A more fitting estimate for over- and underestimation of the agent state variables is the root mean square of the ratio ($y^{\%}_{rms}$) between multi-resolution simulation and pure microscopic simulation.

$$y^{\%}_{rms} = \sqrt{\frac{\sum_{i=1}^{N}\left(\frac{y^i_{mrs}}{y^i_{micro}}\right)^2}{N}} - 1 \tag{5.20}$$

If the root-mean-square ratio is 0.0 then there is no significant difference between the microscopic and multi-resolution simulation, but if the value is negative, the multi-resolution vehicles have a smaller value (e.g., position behind microscopic simulation). This means they were underestimated when aggregating. Should the value be positive, the multi-resolution vehicles overestimated their projection values. These experiments were repeated 20 times and the standard deviation (σ) to all values was calculated.

$$\sigma_y = \sqrt{\frac{\sum_{i=1}^{20}(y^i_{rms} - \overline{y}_{rms})}{20-1}} \tag{5.21}$$

The experiments were conducted with different ratios of microscopic and macroscopic regions. This was done by changing the length of the macro-zones without changing the overall road segment length constant. This leads to an increasing proportion of the road segment being handled by the macroscopic model. Macro-zones length between 50 and 500 meters with 10 meter increments (45 configurations) were chosen. Additionally, the distance from the intersection to the beginning of the macroscopic-zone was varied. The same 50 to 500 meter and 10 meter increment was applied there (45 configurations). This lead to a total number of 40,500 simulation runs ($45 \cdot 45 \cdot 20$) in the single lane experiment and the same amount in the multi-lane experiment.

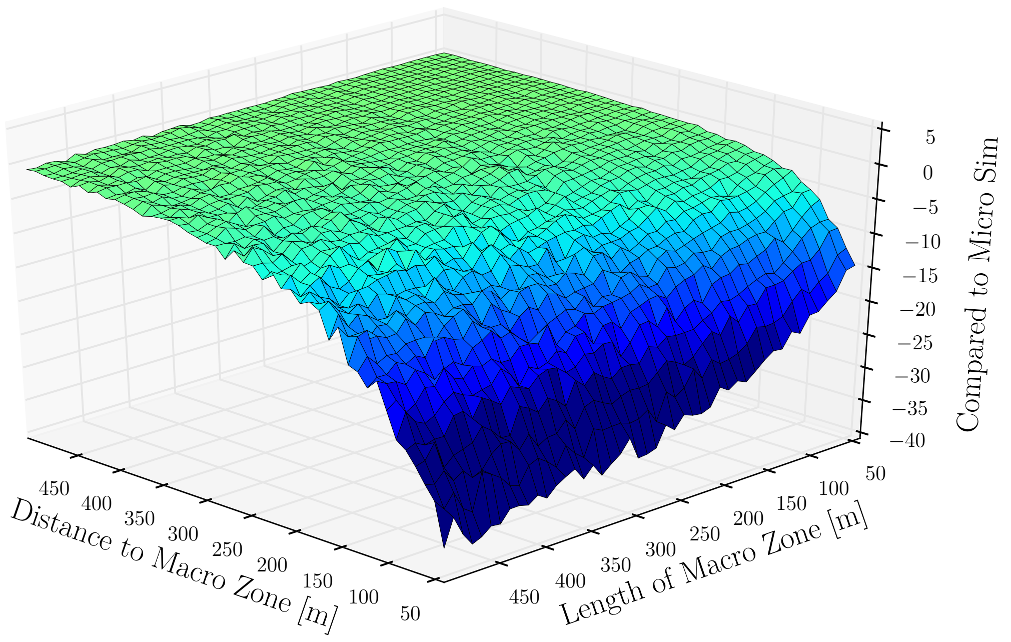
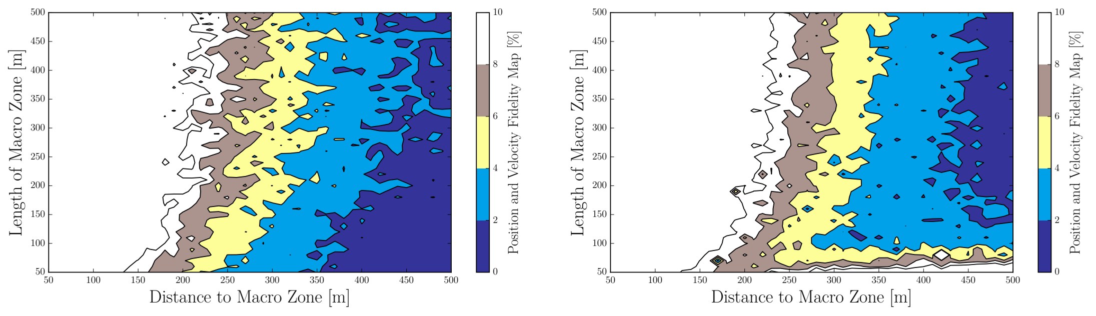


Figure 5.9: Position deviation in percent on single-lane experiments: Longer macro-zones and short distances to the macro-zone show the highest positional difference. Small deviation visible for longer distance to the macro-zone regardless of the length of the macro-zone.

The two parameters discussed previously are shown on the x and y-axis of Figure 5.9. The z-axis shows the positional deviation in percent between the microscopic and multi-resolution simulation. A negative value stands for an underestimation of the projected position after disaggregation.

In the single-lane scenario the distance to the macro-zone is more important than the length of the macro-zone itself. This can be observed since the values of the graph decrease when the distance to the macro zone is below 200 meters for the distance to the macro-zone but not when the length is changed. This only has a significant effect when the distance to the macro-zone is already low.

For the multi-lane mode (number of lanes between 2 and 5), the results are similar. It also shows that the distance to the macro-zone has a smaller influence than the length. Short macro-zones and long distances to the macro-zone have the least influence on the deviation of position.



(a) Position and velocity deviation map for single-lane experiments: Maximum percent deviation for either velocity or position dependent on the distance to macro-zone and the length of the macro-zone. Distance to macro-zone bigger influence than the length.

(b) Position and velocity deviation map for multi-lane experiments with three lanes: Maximum percent deviation for either velocity or position dependent on the distance to macro-zone and the length of the macro-zone. Distance to macro-zone has bigger influence than length, except for very short macro-zones.

Figure 5.10: Position and Velocity Deviation Maps

The map in Figure 5.10a shows the absolute maximum deviation from either position or velocity (worst case for deviation) for the single-lane experiments. It can be seen that the distance to the macro-zone has a bigger influence and is more volatile, but for road segments that are at least 250m (50m length of macroscopic-zone + 200m distance to macroscopic-zone), the deviation is below 10%. The same is shown in Figure 5.10b, except that there are fewer areas with less than 2% deviation and more configurations lead to a higher than 10% deviation. Also it can be seen that the distance to macro-zone is of influence since the fidelity gets worse when the length of the macro-zone is really short.

5.4.3.2 Traffic Flow Deviation

After looking at the agents' state deviation, the comparison of the traffic flow gives a more statistical approach to the influence of the multi-resolution simulation. Therefore, the region behind the macro-zone is examined. The tested scenario is depicted in Figure 5.11, where the observed area in both simulations is shown. The traffic flow in both simulations from the cycle when the first vehicle enters the zone in either simulation until the simulation run ended was logged. A student's T test, to disprove the hypothesis that there is no significant difference

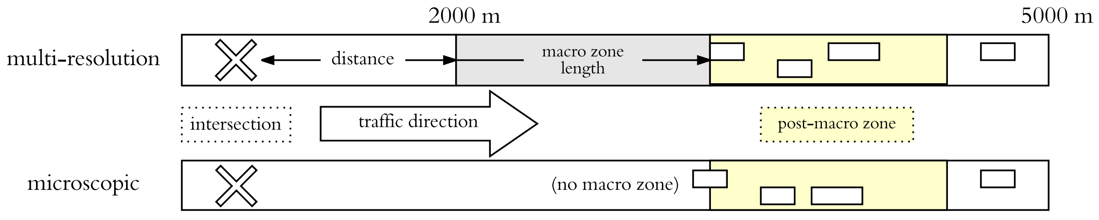


Figure 5.11: Post macro zone flow comparison

between the two zones, has been conducted. This test shows that for a single-lane ($p \leq 0.773$) and a dual-lane ($p \leq 0.879$) the null-hypothesis cannot be discarded and therefore the traffic flow in both simulation shows no significant difference by introducing the multi-resolution model.

5.5 Application to real-world networks

In order to determine how simulation time and ultimately money can be saved when instead of experimenting on a single road segment, a city-scale simulation experiment is to be run, the results from the small-scale evaluation of a single road-segment described in Section 5.3 and evaluated in Section 5.4 are scaled up to a city-scale. Furthermore, utility-function scores for (1) performance increase, (2) fidelity decrease and (3) trade-off (e.g., maximizing difference between performance gain and fidelity decrease) scores are calculated. The 'upscaling' isn't done by implementing the multi-resolution simulation into a full-scale traffic simulation, but rather by taking the results from evaluating different road configurations and fitting them to the network characteristics of different mega-cites, treating each section as independent roads.

Table 5.2: Lane Counts for Different Regions

Region	**Abbrev.**	**# Links**	**1 Lane**	**2 Lanes**	**3 Lanes**	≥ 4 **Lanes**
Bangkok	bkk	201308	195218	1918	1813	2359
Beijing	bei	262818	241519	14040	5815	1444
Chengdu	ctu	86230	77978	2688	4541	1023
Shanghai	sha	313077	275048	21107	10443	6479
Shenzhen	szx	157127	149875	1330	5071	851
Singapore	sgp	122291	118478	1859	1263	691
Tokyo	tyo	864937	782338	74583	4885	3131
Los Angeles	lax	928032	758849	85462	21708	62013
Montreal	ymq	116917	79416	32593	3256	1620
New York City	nyc	221719	197784	10043	9601	4291
Berlin	ber	228086	185333	30834	8425	3494
London	lon	616393	576824	27454	9381	2734
Moscow	mow	401825	340293	31682	13235	16615
Munich	muc	149829	127639	15810	4266	2114
Paris	par	92100	87846	2252	739	1263
Rio de Janeiro	rio	200439	182348	13222	3376	1493
Sydney	syd	462748	430692	17984	11274	2798

Since in the experiments in Sections 5.3 and 5.4 it was assumed that the vehicles will start form a stand-still initial position, similarly to a traffic light, it is assumed that the network can therefore be broken down into road segments where to influence between the segments is regulated through the traffic light. Furthermore, an analysis-based approach, with the results from the previous sections, compared to a full simulation-based evaluation is much quicker and need fewer computing resources. Additionally, the scenario setup of a city-scale simulation is not easy, especially when the objective of the experiment is to find the possible multi-resolution benefit for different regions, as it in done in this analysis. A high resolution (e.g., simulation-based) analysis of major cities as done in this chapter is possible, but not a general methodology for every (custom) network. The selected cities used for comparison are presented in Table 5.2 along with abbreviations and network characteristics used in this chapter.

The cities were chosen on the merit that they present different large cities with different cultural areas. Network data for the given regions was obtained from *OpenStreetMap* (OSM) crowd sourced data. Furthermore, the administrative region that defines the city has been taken to reduce the OSM to the streets that actually belong to the city. This meta information is present in the OSM data. It is necessary, since the export from OSM is always rectangular and includes street that are outside the city boundaries. In Figure 5.12 which shows the exported region of Berlin, Germany, as well as the administrative region Berlin, streets that are located outside the marked area are discarded before the evaluation of a city is started.

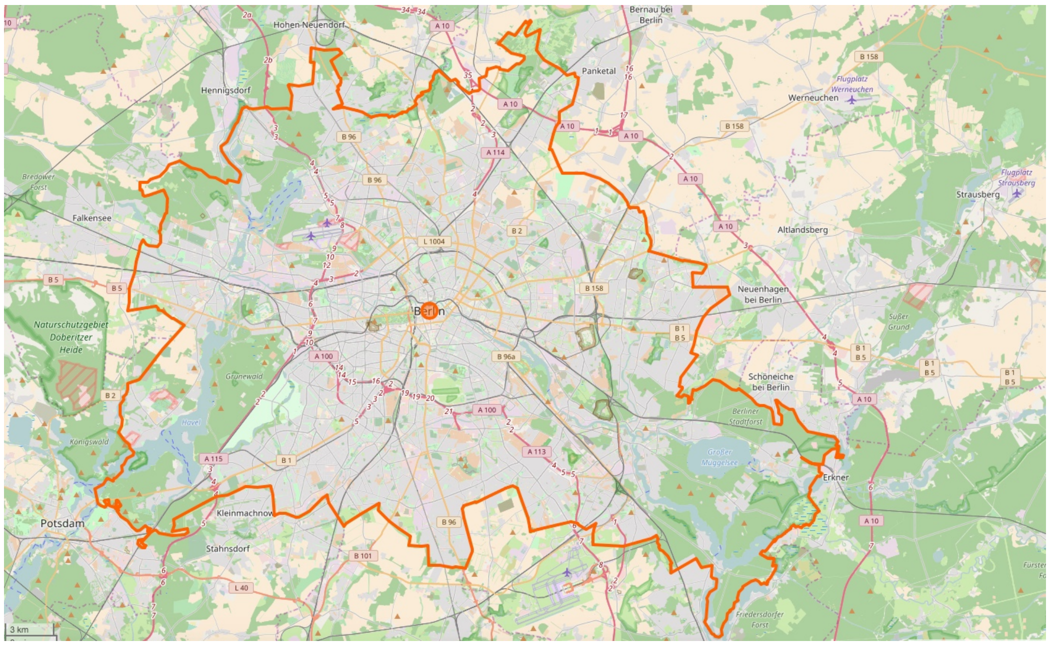
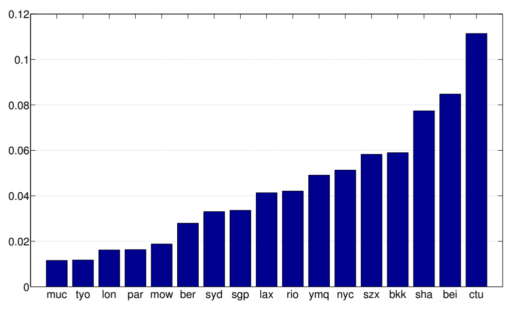


Figure 5.12: OSM export compared to the administrative region of a city

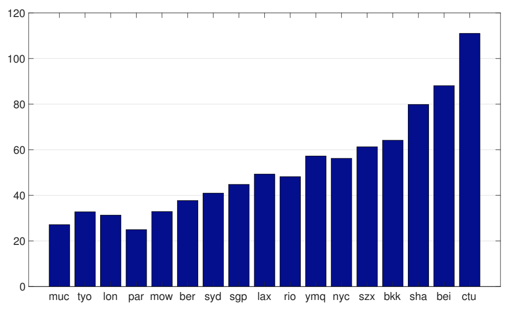
Afterwards, the link information in terms of lane numbers and lane length is extracted. From the multi-resolution experiments (see Sections 5.3 and 5.4), the performance and fidelity information for different combinations of distances from the intersection to the macroscopic-zone and length of the macroscopic-zone have been extracted. This resulted in 2116 $(45 \cdot 45 \cdot 4)$ combinations for lane counts from 1 to 4, where all links with more than 4-lane are counted in the 4-lane category. To determine the different performance and fidelity values, the distance from the intersection to the macroscopic-zone as varied between 50 meter and 500 meter (45 samples), as was the length of the macroscopic-zone (45 samples).

5.5.1 Data Analysis

In a first step of network analysis, all length and lane combinations were categorized into 10 meter bins that range from 100 meter to 1000 meter, whereas all links that are longer than 1000m were added to the 1000 meter bin. The lower limit of 100 meter and upper limit of 1000 meter mark the shortest and longest road segments that the multi-resolution experiment



(a) For the selected regions, the links that are eligible to use for multi-resolution modeling as the percentage of all links in the network are shown.



(b) Average length in meter of links in a given region

Figure 5.13: Links, for which MR-modeling can be applied, have to be at least 100m in length.

has data for (50/500m length to macroscopic-zone and 50/500m length of macroscopic-zone). This gives a count of how many links have a length that is in a certain 10 meter interval. It results in 91 bins for each of the 4 lane counts (1,2,3 and $\geq$4). In order to get the magnitude of each of the links in comparison to all eligible links($length >= 100m$), each count was divided by the total number of links. This resulted in a relative lane-length count for all regions. The percentage of eligible links can be seen in Figure 5.13a and the average length other all lanes in a given region can be seen in Figure 5.13b.

In the next step, the best combination for a certain road length and utility function (performance gain, fidelity lose or trade-off score) had to be determined. This means, iterating over all possible distance-length combinations from the previous experiments, where the sum of which is smaller than the total length of the length-lane value, and finding the best value given by a certain utility function. This is done in order to make sure that the road-segment can be divided into a pre macroscopic and macroscopic-zone. The fidelity of a city (*fidelity(city)*) in all evaluations is the geometric mean of the positional difference and the velocity difference between the multi-resolution simulation and a pure microscopic simulation. Since the fidelity of the pure microscopic simulation is the ground truth, the ratio between that value and the deviation in the multi-resolution simulation is defined the fidelity difference. Performance (*performance(city)*) is defined as how much computational time is required for the execution of the multi-resolution simulation in respect to the microscopic simulation. The performance gain is 1-performance.

For this data analysis two different utility functions where used:

- **Best Performance gain**: determine the combination, which offers the highest performance gain, disregarding the fidelity loss. $(max(performance(city)))$
- **Best Trade off**: maximizing the difference between performance and fidelity. This works because the fidelity is highest when close to 1 and the performance is best when close to 0. $(max(fidelity(city) - performace(city)))$

These utility function values have been multiplied with the relative lane-length count to weight them to all possible eligible links. The sum of all weighted utility values over all eligible lanes is the total performance increase and fidelity loss. This only shows the increase and decrease in respect to all eligible links. Additionally, the overall performance and fidelity values have been calculated by multiplying the total values by the percentage of eligible links as mentioned in Figure 5.13a.

5.5.2 Results and Discussions

Performance and Fidelity

Firstly, the best performance and fidelity measures for a given region are examined. In Figure 5.14 the performance gain values are shown. When simply using the best combination of macroscopic-zone and distance to macroscopic-zone for performance without regarding the fidelity loss, the values can reach up to 40 per cent, but never below 35 per cent (red bars). When looking at the trade-off number, where the utility function maximizes the difference between performance gain and fidelity, the performance for the regions falls below 35 per cent. The same trend between the regions can be observed. The Beijing(bei) and Los Angeles(lax) have the highest values in the *best performance* case as well as in the *trade-off* case.

In Figure 5.15 the fidelity loss values for the same cases are shown. Here, the difference between the *best performance* case and *trade-off* case are more pronounced. Paris(par), for example, has, compared to other regions, one of the lowest fidelity loss values in the *best performance* case, while in the *trade-off* case having the highest value. Overall, the fidelity loss is below 25 per cent in the *trade-off* case but well above 35 per cent in the *best performance* case.

In order to have a closer look, a scatter plot of the trade-off values where the performance gain is shown on one axis and the fidelity loss on the other axis. Figure 5.16 shows that the best performance and fidelity ratio have the cities in the bottom right corner. Beijing, Chengdu

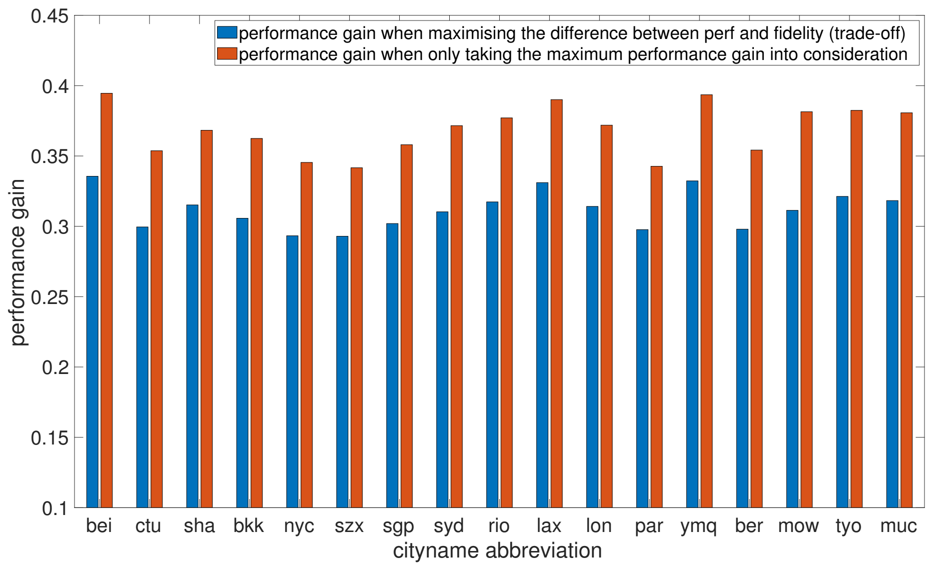


Figure 5.14: The performance gain with respect to different utility values is shown. Once the trade-off value and then the performance-only value.

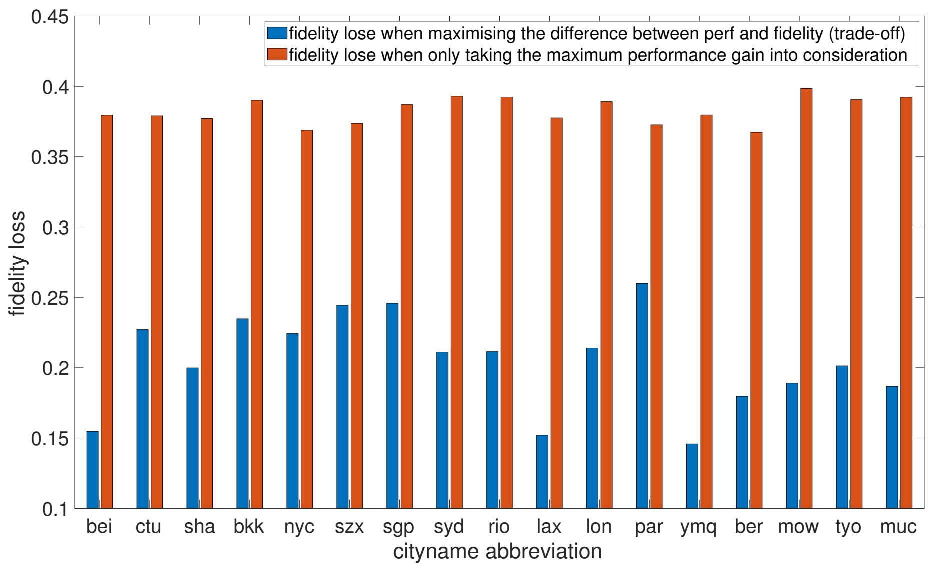


Figure 5.15: The fidelity loss with respect to different utility values is shown. Once the trade-off value and then the performance-only value.

and Shanghai have the highest performance gain while at the same time suffer from the lowest fidelity loss.

The middle cluster consists of North American cities like New York and Los Angeles as well as Singapore and Sydney. This similarity has its roots in similar urban development and industrialization level of the regions. Bangkok, Shenzhen and Rio de Janeiro can be found here as well. All these regions have a higher performance increase than fidelity loss, which is around 20 per cent.

The last cluster consists of European regions like London, Paris and Berlin as well as Tokyo. They score the lowest in terms of performance gain over fidelity loss, which might be due to their historically-grown networks that have been around for more than 100 years, which have rather short links. This is supported by the values shown in Figure 5.13b, where the average link length is depicted as well as Figure 5.13a, which shows the overall low number of links that are longer than 100 meter.

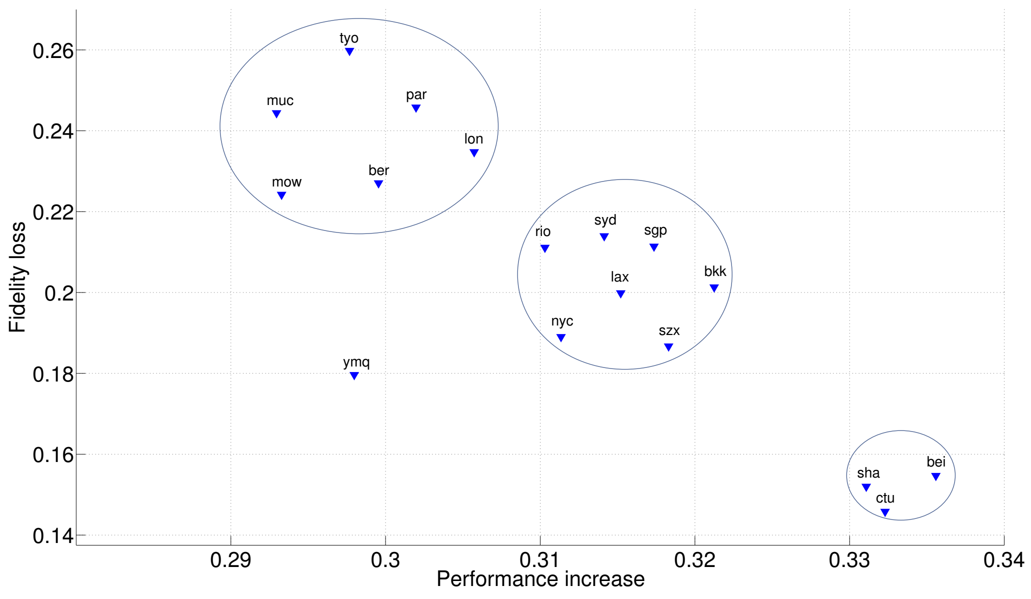


Figure 5.16: Performance gain and fidelity loss on two axis. Three clusters have been identified. Cities within these three main clusters are predominantly from the same cultural or geographical region of the earth.

So far the evaluation focussed on the performance increase of the links that are eligible, but overall performance increase has to take all links within a region into consideration. Therefore, the performance numbers have to be multiplied by the percentage of eligible links from Figure 5.13a. This results in the graph shown in Figure 5.17. It can again be seen, that the Asian regions are leading the statistics with the highest performance increase of up to 7.5 per cent, while the European cities and Tokyo go as low as below 1 per cent of performance increase. The performance is a measure that expresses the computational time saved in comparison with pure microscopic simulation, while the fidelity is the difference between the position and the velocity of an agent after emerging from a macro-zone. An error here can propagate throughout the entire system.

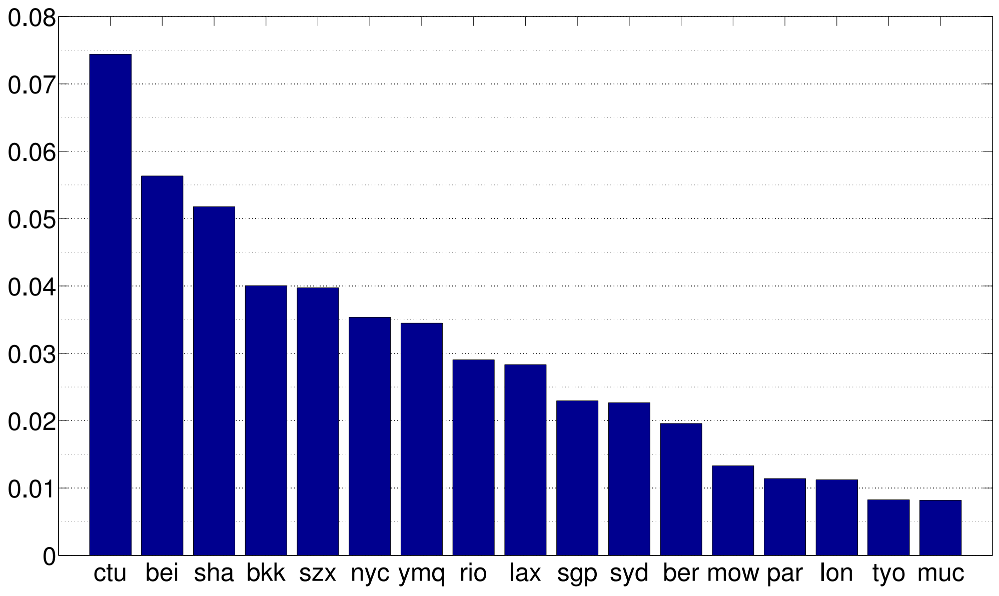


Figure 5.17: Overall performance increase in per cent.

Assumptions and Limitations

Using the OSM data set to determine characteristics of a road network of a city or region is very much dependent on the quality of the data that is being added by the users of the openstreetmap (OSM) platform. In different countries, the quality can vary. This could be one explanation, why Chinese cities have a longer average link length in comparison to the European and North American cities. Since this is the case for many cities within China, the layout and structure of Chinese cities might just be better suited. In European cities, many of the streets and alleys have been around for many 100 years and were converted to motorways with the onset of the last century. This means they were not designed for motorized vehicles but rather horses and are much shorter.

The performance for the different regions is very low, since the number of links that are eligible in the respective networks is relatively small. The actual performance increase might be higher due to the fact that the smaller roads are much less frequented and are therefore not responsible for the majority of the simulation workload. In order to take this into consideration, a detailed experiment with realistic travel patterns needs to be conducted.

5.5.3 Scalability concerns using Cloud Resources

What does that mean for cost savings when using the multi-resolution approach to speed up the performance of the search-space exploration on the cloud? A definite answer to how much cost can be saved can only be given when actually deploying simulations to the cloud, but a rough model for estimating the cost can be developed. First and foremost, it depends on how many interesting combinations have to be re-simulated using a high-resolution simulation. As described in Section 5.1 the execution idea of exploring a large search space is to simulate the entire search space using multi-resolution models and only re-simulating the interesting parameter combinations in high resolution. In the following sections, a run-time model for large-scale agent-based simulations as well as a simple cost model for two CPs (Amazon AWS and Google Compute Engine) will be shown and evaluated on their usability for search space exploration on the cloud.

Time Model

Since the price of cloud resources depends greatly on the duration they are use, a model describing the overall exploration time needs to be developed. The following equation shows how much time a search-space exploration for a given space with n dimensions would take, given these parameters.

$$\begin{aligned} N &: \text{set of dimensions} \\ n_i &: \text{ number of values in dimension i} \\ p_{\text{int}} &: \text{percentage of interesting combinations} \\ t_{\text{MR}} &: \text{time to simulate one combination in multi-resolution} \\ t_{\text{HR}} &: \text{time to simulate one combination in high resolution} \\ p_{\text{perf}} &: \text{performance increase in per cent} \end{aligned} \tag{5.22}$$

The performance increase ratio is calculated as follows:

$$p_{\text{perf}} = \left(1 - \frac{t_{\text{MR}}}{t_{HR}}\right) \tag{5.23}$$

The p_{perf} value is the the resulting performance value for a given region as described in Section 5.5.1. It can be used to determine t_{MR} by rearranging the above equation:

$$t_{\text{MR}} = (1 - p_{\text{perf}}) \cdot t_{\text{HR}} \tag{5.24}$$

The total time to explore the entire search space can be given as the following equation:

$$t_{\text{SSE}} = \left[\prod_{i=1}^{|N|} n_i\right] \cdot ((1 - p_{\text{perf}} + p_{\text{int}}) \cdot t_{\text{HR}}) = \left[\prod_{i=1}^{|N|} n_i\right] \cdot (t_{\text{MR}} + p_{\text{int}} \cdot t_{\text{HR}}) \tag{5.25}$$

This includes the time it takes to simulate all possible combinations in multi-resolution as well as p_{int} percent in high resolution. As long as the overall performance increase using multi-resolution simulation is larger than p_{int} for a given region, the recalculation of an interesting combination of values comes without extra cost.

Take an example city from the above region, cities such as Berlin, Paris or Tokyo, which have a performance increase of less than 2 percent. This method might not be advisable, since the percentage of interesting regions is expected to be higher. This, of course, depends greatly on the use-case of the simulation. On the other hand, simulations of cities such as Chengdu or Beijing benefit greatly.

Until now, the execution time to explore the search space assumed a sequential execution on a single node. Using the scalability of the cloud, this could be done in parallel on many nodes. There are two ways of exploring the search-space concurrently. One is to parallelize the execution of a single simulation run and a second is to have each simulation run on a single node, but to execute multiple parameter combinations at the same time or actually doing both. The time to execute in parallel is a little different than just dividing the total time by the execution time of one simulation. The high-resolution simulation of interest can only be done after all multi-resolution simulations are completed or when an interesting parameter configuration has been identified and resources are available. This also means that there can be different degrees of parallelism, which reduces the overall costs. The total time for a parallel (t_{SSEpar}) version depends greatly on the degree of parallelism for the multi-resolution part DP_{MR} and for the high-resolution part DP_{HR}.

$$t_{\text{SSEpar}} = \frac{\left[\prod_{i=1}^{|N|} n_i\right] \cdot t_{\text{MR}}}{DP_{\text{MR}}} + \frac{\left[\prod_{i=1}^{|N|} n_i\right] \cdot p_{\text{int}} \cdot t_{\text{HR}}}{DP_{\text{HR}}} \tag{5.26}$$

$$\begin{aligned} DP_k &\leq \Big[\prod_{i=1}^{|N|} n_i\Big] \cdot p(k) \\ k &\in \{HR, MR\} \\ p(k) &= \begin{cases} k = HR, & p_{int} \\ k = MR, & 1.0 \end{cases} \end{aligned} \tag{5.27}$$

For most CPs, it is favorable to reduce the number of parallel-used instances to reduce the overall costs. This is especially required when using ad-hoc instances. Since it is simple to increase or decrease the number of instances allocated from CP, DP_k should be adjusted.

Price Model

Following the model to determine the time and parallelization strategy for multi-resolution search-space exploration, the influence in different pricing schemes will be evaluated. For this, three pricing strategies will be taken into consideration.

1. Ad-hoc : Pricing without any saving strategy
2. pre-paid : As a cost saving strategy offered by AWS
3. sustained use: The cost reduction strategy offered by GCE

Each of the CPs have different prices per instance ($ppi(c)$) for virtual machines (VMs) with different capabilities (c) in terms of number of CPUs, amount of main memory and disk sizes. For VMs with higher capacities, the prices increase a lot, since they are running almost exclusively on physical hardware resources in the data centers. This also means that the t_k ($k \in \{HR, MR\}$)is dependent on what hardware-equivalent is chosen. Therefore, the time for executing a multi-resolution (t_{MR}) or a high resolution (t_{HR}) simulation experiment depends on the computational capability of the VM. The times $t_k(c)$ are also expressed as a function of the capabilities c. The total price (P_{total}) of running a search-space exploration is the sum of the price for running the multi-resolution part with a certain degree of parallelism (DP_{MR}) and the price for executing the high-resolution part with a specific degree of parallelism (DP_{HR}). Generically, the price can be state as follows:

$$P_k = ppi(c) \cdot t_{\text{k}}(c) \cdot \frac{\prod_{i=1}^{|N|} n_i}{DP_{\text{k}}} \cdot p(k) \cdot DP_{\text{k}} \tag{5.28}$$

The price P_k is determined by the ppi of a specific category of VM and the time this VM takes to compute this task for the multi-resolution case. This value is multiplied with the number of possible combinations to explore the search space devided by the possibilities that can be done

Table 5.3: Example search space exploration parameter

parameter	description	value
$\|N\|$	number of dimensions	4
n_1	number of vehicles from 100k to 1000k in 50k increments	19
n_2	percent of electric vehicles from 0% to 100% in 5% increments	21
n_3	percent of carparks with charging stations from 0% to 100% in 2% increments	51
n_4	number of different charging behaviors	5
p_{int}	estimated percent of interesting combinations	0.01

Table 5.4: Execution time of multi- and high-resolution simulation as well as cost per instance for different categories of computing capabilities (taken in Feb 2017).

category	$time_{MR}$ $t_{MR}()$	$time_{HR}$ $t_{HR}()$	price in USD $ppi()$
c1	3.7h	3.885h	0.076
c2	1.9h	1.995h	0.152
c3	1h	1.05h	0.304
c4	0.6h	0.63h	0.608
c5	0.35h	0.3675h	1.216

in parallel (DP_k) and the percentage of parameter combinations that will be run in this mode $p(k)$. Lastly, the value is multiplied by the degree of parallelism DP_k.

Example

The following example demonstrates the usefulness of the multi-resolution approach. The above mentioned parameter values can be seen in Table 5.3. Furthermore, the capabilities are taken from the GCE specification of different machine types with high CPU configuration, resulting in 5 different capability categories. Since the execution time and price is a function of the capability, the values for t_{MR}, t_{HR} and ppi have been determined empirically for one of the combination with the assumption that the parameter choice does not influence the execution time and are shown in Table 5.4. Since it is difficult to determine tht exact execution time depending on the model parameter before running the actual simulation, a random combination was chosen.

The price and time for this example is shown in Figure 5.18. The names of specific machines used from the GCE correspond with Figure 3.1 on Page 31 (e.g., c1 corresponds to n1-standard-1). It can be seen that using a higher capability machine increases the costs, but also decreases the time to execute. The degree of parallelization describes how many parameter variations are executed at the same time and also determines the number of VMs that need to allocated from the CP. Parallelization also reduces the overall execution time. There is no upper limit on how many instances can be used a the same time, but in the following experiments it is varied from 1 to 100 in step with the degree of parallelization. Since there is no benefit for reducing the number of allocated instances at a given point in time by the CP, the price does not change, due to the fact that the price scales linearly with the capabilities. The prices for each capability (c1-c5) are constant with the degree of parallelization is due to the fact that it does not matter, if a single instance is run n-times (e.g., n=100) consecutively or n instances are run concurrently (see Table 5.4 on Page 113), when not taking price reduction into consideration.

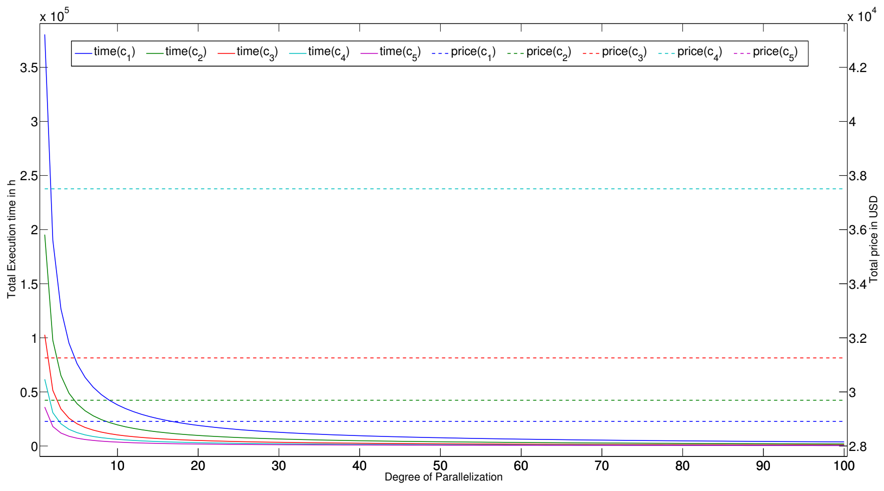


Figure 5.18: Computational time required to finish the experiment with a different number of parallel executions and different capabilities. The price, determined by the pricing model given in Equation 5.28, is also shown, but since the parallel instance costs as much as a sequential used instance, there is no difference in pricing.

Alternatively, the GCE pricing model, which gives discount on sustained-use instances, results in a similar execution time behavior depending on the capability c and the degree of parallelism DP. In Figure 5.19 the effective price per hour for the different configurations is shown instead of the total price of the execution of the search-space exploration. The execution time for exploring the search space does not change and stays the same as shown Figure 5.18. The price reduction scheme has been taken from the GCE pricing structure. There the first 25% of the month are full priced and with each following 25% of usage, the price drops by 20%

for the next 25% (see Table 5.5). The figure shows that for sustained use the effective hourly price is below the ad-hoc pricing.

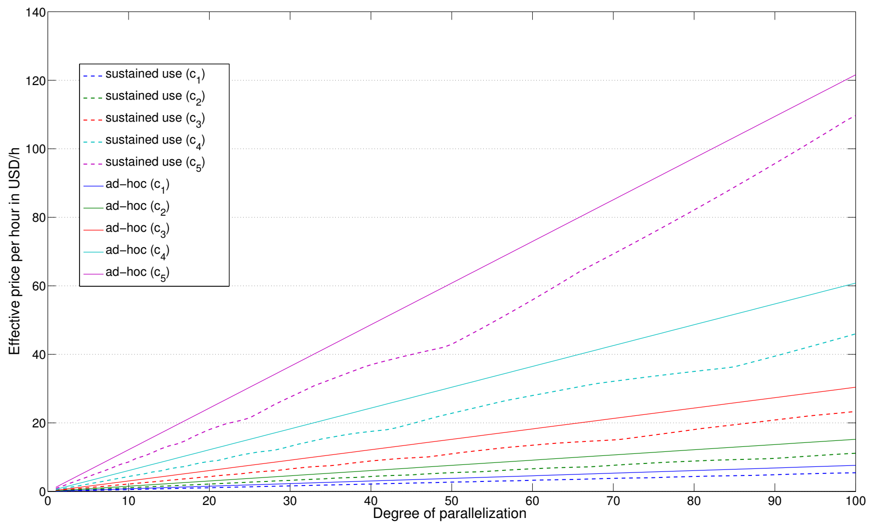


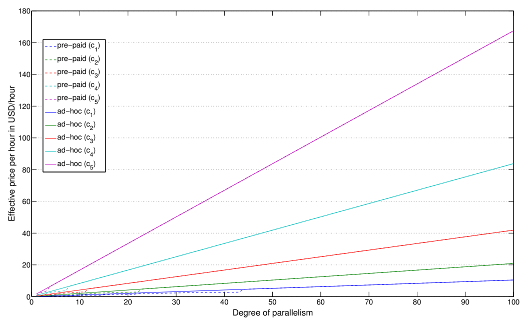
Figure 5.19: Sustained use pricing in comparison to adhoc pricing with degree of parallelization

Table 5.5: GCE pricing scheme for sustained use

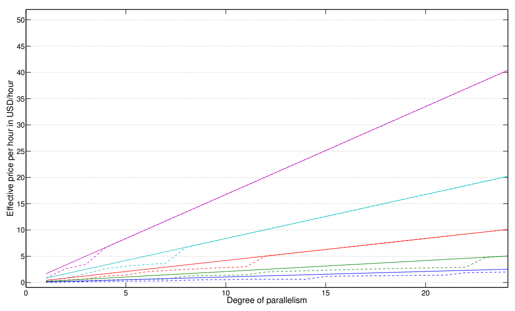
Usage Level	**base rate percentage**
0%-25%	1.0
25%-50%	0.8
50%-75%	0.6
75%-100%	0.4

A similar behavior can be see in Figure 5.20, where the Amazon Web Services (AWS) pricing model is assumed. Under this model, users can pre-purchase VM instances of 1-or 3-year contracts. The effective hourly price for such instances is about 62% or 39% of the ad-hoc base rate.

The results show that the overall price when a large degree of parallelism is assumed is much higher under the ad-hoc scheme. Using the pre-paid scheme with 1 and 3 year contracts reduces the cost significantly, but eventually reaches the same price point as the ad-hoc instances. This can be seen in Figure 5.20b. There, the step pattern in the dashed lines for the different configurations show a significantly lower effective price. This nonetheless stops once the degree of parallelism reduces the overall runtime below 1 or 3 years to have any benefit from the



(a) Pricing for ad-hoc and pre-paid instances in AWS for 1 to 100 concurrent instances



(b) Pricing for ad-hoc and pre-paid instances in AWS for 1 to 30 concurrent instances

Figure 5.20: Pre-paid pricing for different degrees of parallelization

reduced price. Similar behavior can be seen in on the GCE sustained-use pricing scheme as well, but with a higher degree of parallelism since the calculation period is not 1 or 3 years, but rather a month.

While being a rather large, but not necessarily unrealistic example with a total of 101745 different combinations, the costs are high. It should only be considered should there be not a different option on gaining access to a high performance cluster or if the use-case is very singular. Only in this case, will the above benefits of on-demand cloud resources apply. The trade-off between obtaining the search-space exploration results quickly, but paying a high price (high degree of parallelization), or getting cheaper resources that will take longer have to be a budgeting and urgency decision made by the researcher. A mix of resources of ad-hoc and long-term contract (e.g., AWS 1-3 year contract) could be useful, when considering an entire system of nodes for exploring a search space over a longer period of time. The saving using pre-paid resources could be utilized for node that manage the search-space exploration while ad-hoc nodes are used for actual simulations. Here, also *spot instances* from AWS, that have an even lower price, but need to be bidden for in a marketplace, could help in lowering the price even more, but have an even unpredictable long-term price structure that is usually not favorable in a research environment.

5.6 Summary

The evaluation shows that a more than 20% performance increase compared to a pure microscopic model using multi-resolution models can be achieved. The main factors that influence the relative performance (pure microscopic vs. multi-resolution) is the complexity of the high-resolution model and the length of the road segment. The longer the road segment, the more time that can be spent in the macro-zone and therefore the less computing that needs to be done. The side-effect of this is that the fidelity of the result decreases as well. It also shows that a lower than 6% fidelity deviation on single-lane roads can be kept. This is given when the macroscopic zone is at least 150m from an intersection. For a dual-lane road segment, a deviation of less then 6% can be achieved when the macro-zone is at least 300m from an intersection and 200m long. It has also been shown that the introduction of a macroscopic element into a microscopic simulation does not change the traffic flow behind a macroscopic zone modeled.

In this chapter, we have shown that a simple multi-resolution approach to agent-based traffic simulations can have a great performance improvement of the simulation. Nevertheless, the performance improvement comes at a price of model fidelity. It has been shown that with macroscopic metric there is no significant difference between a multi-resolution simulation and a pure-microscopic one. With microscopic measures, a loss in fidelity is observed; however, depending on the road configuration, this can be reduced to a minimum. Therefore, the second part of this chapter dealt with the analysis of different road networks of large cities from around the globe. This has shown that cities in Asia, especially China have characteristics that are better suited for being used in multi-resolution simulations than cities in Europe and North-America. The performance of the simulations can be improved in all tested cities, while at the same time there is some fidelity loss. The amount of performance gain and corresponding fidelity loss differs between the two extreme cases of Beijing, China and Munich, Germany. While the performance gain is between 30 per cent in the Munich network and 33 per cent in the Beijing network, the fidelity loss in Munich is close to 25 per cent and in Beijing only around 15 per cent.

For the Cloud Simulation use-case, to explore the massive parallelism and ease of allocation of additional resources, an evaluation of a simple time model for search-space exploration was performed. This takes into consideration the results from the previous sections about the overall performance gains (using multi-resolution modelling) in different kinds of road networks, to develop a time model for simulations using such methodology to firstly simulate a search space using the faster multi-resolution simulation and then identifying interesting combinations, and re-simulate them using a high resolution simulation. There, the model that determines the actual simulation time for different parameter combinations as well as resource use could improve the time model's accuracy.

Future research should look into the dynamic creation of multi-resolution road segments depending on the road conditions. Also, the static analysis for an entire country could lead to an even bigger performance increase, since connecting highways or country roads are longer which is favorable, since the macroscopic-zones can be longer which reduces the computation time for these road segments.

Chapter 6

Conclusion

The main problems which this thesis tried to explore and give a possible solution to were focused on the issue of how to enable researchers, engineers, policy makers as well as the general users to use the cloud for high fidelity simulation experiments. This encompassed the topic of interoperability between users of different domains and the connected knowledge gaps that are present when complex systems like a traffic system of a large metropolitan area are simulated. Additionally, the problem of different knowledge areas in the domain of computer engineering for setting up high-performance computing infrastructure in the cloud as well as the optimal utilization of such, was identified in the introduction chapter. Apart from the hardware and system design aspect, the difficulty of creating, starting, monitoring and evaluating cloud-based simulation scenarios is the biggest issue. All the above mentioned problems with running agent-based simulation experiments in the cloud, come together into the problem that simulation of complex systems does need expert knowledge and decisions are made within an exclusive circle of people. By moving such decision-support tools to the cloud, where there are little constraints on user-interfaces, hardware and software requirements, everyone can use the tools and will therefore democratize the decision-making process significantly. On one hand, it allows the average interested party to influence the decision by being better informed, but on the other hand also allows decision makers to better explain why decisions have been made in a certain way.

Within this thesis, the use-case was an agent-based simulation system for transport system of mega-cities. This was chosen to illustrate the usefulness of creating a cloud service for such large-scale agent-based traffic simulations. By introducing the general concept and a reference architecture for a traffic simulation cloud service in Chapter 3, the foundation for using such a system to conduct further studies was set. This approach of creating a collaborative and hardware-independent complex-system simulation platform allows users with various forms of domain knowledge but minimal technical expertise in computer engineering to create complex system simulations in the context of traffic simulation. Subsequently, the questions of data extraction of such big-data producing simulations were answered. There, it was shown that it is inherently necessary to find a new paradigm of working and modeling with simulation output data. This includes treating the data as relational data as done in RDBMS, including the introduction of time-variant algorithms, as well as reducing the amount of data persistently stored by the simulation, by doing on-line post processing instead. Ultimately, the main chapters conclude with using cloud resources to conduct large-scale domain-space exploration. This includes an optimization step of using multi-resolution modeling and hybrid-resolution simulation execution to speed up a single experiment run, while only marginally sacrificing on the fidelity of the results. The potential of this method is evaluated on different real-world mega-city scenarios and a pricing as well as exploration-time model has been developed.

The gained insight into the workings of simulation experiment creation as well as collaboration between users and researchers of various, but overlapping fields, can be applied not only to vehicular traffic simulation, but also to general mobility scenarios. There, the challenges lie within the added complexity and diversity of the users. It is even more important that the technicalities of allocating resources and managing the execution of simulations is even more abstracted and the model interfaces are defined properly. This could lead can lead to finding transitive effects, where non-overlapping areas influence each other, but the domain experts are not aware of their influence. This is outside the scope of this thesis, but could be an interesting field for interoperability of simulation models of various fields. Furthermore, the individual aspects of a simulation cloud service can be expanded. While within this thesis an overall reference architecture as well as the topics of data extraction and domain space exploration were covered, optimization of the entire workflow as well as an online cost model for cloud-based simulation experiments are logical extensions. The actual execution optimization, from not only using the a single computing machine for one simulation experiment, but utilizing on distributed computing on cloud architecture with all its challenges of e.g. synchronization and latency as well as data exchange on loosely coupled infrastructure is an interesting research field and needs to be further investigated.

The research questions studied in this thesis focus on removing the technical overhead and offering Application Programming Interfaces (APIs) for users, but it fails to conduct any user studies, if this is what is wanted by the community. It is not out of the realm of expectation that a little complexity in terms of configurability is wanted in order to not hinder the research process by not allowing certain operations. This can only be examined with real-world user studies. That includes human workflow behavior modeling as well as knowledge into human machine interaction for next-generation simulation platforms.

In contrast to other approaches in the field, this thesis tries to give a solution for the specific problem of agent-based simulation in the cloud. The respective problems were identified and addressed, while at the same time offering a reference architecture with regards to how such specific cloud system could look like. Furthermore, the field of mobility and transportation simulation is a highly competitive field where many academic and commercial products are available. Many of these have not yet capitalized on the possibilities of allowing users without powerful computing hardware to execute large-scale simulation experiments by using cloud resources. The approaches within the thesis may help to give developers and platform designers a guideline on how to move such products to the cloud as well as present the complexity of such an undertaking. This will not only help in the development of better simulation tools and services, but ultimately improve the collaboration between related research fields as well as aid in conveying of policy implications to the non-expert by democratizing the knowledge generation from the wealthy institutions to everyone with an Internet connection.

References

[1] Michael F. Wehner, Kevin A. Reed, Fuyu Li, Prabhat, Julio Bacmeister, Cheng-Ta Chen, Christopher Paciorek, Peter J. Gleckler, Kenneth R. Sperber, William D. Collins, Andrew Gettelman, and Christiane Jablonowski. **The effect of horizontal resolution on simulation quality in the Community Atmospheric Model, CAM5.1**. *Journal of Advances in Modeling Earth Systems*, **6**(4):980–997, 2014. 1

[2] J.M. Epstein. **Modelling to contain pandemics**. *Nature*, **460**(7256):687, 2009. cited By 82. 1, 28

[3] Frank Seibt, Marco Schumann, and Jürgen Beikirch. **Concept and components for a web-based simulation environment (WBSE)**. *SIMULATION SERIES 30*, pages 189–194, 1998. 5, 12

[4] S. Narayanan. **Web-based modeling and simulation**. In *Simulation Conference, 2000. Proceedings. Winter*, **1**, pages 60–62 vol.1, 2000. 5, 12

[5] Rajkumar Buyya and Manzur Murshed. **GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing**. *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE (CCPE*, **14**(13):1175–1220, 2002. 5, 12

[6] E. Cayirci. **Modeling and simulation as a cloud service: A survey**. In *Simulation Conference (WSC), 2013 Winter*, pages 389–400, Dec 2013. 5, 12

[7] S.J.E. Taylor, A. Anagnostou, T. Kiss, G. Terstyanszky, P. Kacsuk, and N. Fantini. **A tutorial on Cloud computing for Agent-based Modeling amp; Simulation with Repast**. In *Simulation Conference (WSC), 2014 Winter*, pages 192–206, Dec 2014. 6, 12

[8] Cloud Security Alliance. **Top Threats to Cloud Computing V1.0**, 2010. 12

[9] Song Guo, Fan Bai, and Xiaolin Hu. **Simulation software as a service and Service-Oriented simulation experiment**. In *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, pages 113–116, Aug 2011. 12

[10] A.A. Huqqani, Xin Li, P.P. Beran, and E. Schikuta. **N2Cloud: Cloud based neural network simulation application**. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–5, July 2010. 13

[11] C. N. Hoefer and G. Karagiannis. **Taxonomy of cloud computing services**. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1345–1350, Dec 2010. 13

[12] S. Chaisiri, Bu-Sung Lee, and D. Niyato. **Optimization of Resource Provisioning Cost in Cloud Computing**. *Services Computing, IEEE Transactions on*, **5**(2):164–177, April 2012. 14

[13] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. **The Cost of Doing Science on the Cloud: The Montage Example**. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, pages 50:1–50:12, Piscataway, NJ, USA, 2008. IEEE Press. 14

[14] Sayantan Sur, Matthew J. Koop, and Dhabaleswar K. Panda. **High-performance and Scalable MPI over InfiniBand with Reduced Memory Usage: An In-depth Performance Analysis**. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM. 14

[15] Adrian Mouat. *Docker Security.* OŔeilly Media, August 2015. 15

[16] Dirk Merkel. **Docker: Lightweight Linux Containers for Consistent Development and Deployment**. *Linux J.*, **2014**(239), March 2014. 15

[17] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. **Reproducible network experiments using container-based emulation**. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 253–264. ACM, 2012. 16

[18] Shashank Shekhar, Hamzah Abdel-Aziz, Michael Walker, Faruk Caglar, Aniruddha S. Gokhale, and Xenofon D. Koutsoukos. **A simulation as a service cloud middleware**. *Annales des Télécommunications*, **71**:93–108, 2016. 16

[19] Jaume Barceló et al. *Fundamentals of traffic simulation*, **145**. Springer, 2010. 17, 18, 19, 20

[20] David L Gerlough and Matthew J Huber. **Traffic flow theory**. Technical report, 1976. 18

[21] Reinhart Kuhne and Panos Michalopoulos. **Continuum flow models**. *Traffic flow theory: A state of the art reportrevised monograph on traffic flow theory*, 1997. 18

[22] Wilco Burghout, Haris Koutsopoulos, and Ingmar Andreasson. **Hybrid mesoscopic-microscopic traffic simulation**. *Transportation Research Record: Journal of the Transportation Research Board*, (1934):218–255, 2005. 19, 26

[23] Wilco Burghout. *Hybrid microscopic-mesoscopic traffic simulation.* Doctoral dissertation, Royal Institute of Technology, Stockholm, 2004. 19, 86

[24] Moshe Ben-akiva, Michel Bierlaire, Haris N. Koutsopoulos, and Rabi Mishalani. **Real Time Simulation of Traffic Demand-Supply Interactions within DynaMIT**, 2000. 19

[25] Andreas Horni, Kai Nagel, and Kay W. Axhausen, editors. *The Multi-Agent Transport Simulation MATSim.* Ubiquity, London, UK, 2016. 19

[26] M Treiber, A Hennecke, and D Helbing. **Congested traffic states in empirical observations and microscopic simulations**. *Physical Review E*, **62**:1805–1824, August 2000. 20, 87

[27] Peter G Gipps. **A behavioural car-following model for computer simulation**. *Transportation Research Part B: Methodological*, **15**(2):105–111, 1981. 20

[28] Forschungsbesellscahft Kraftfahrwesen mbH Aachen. **PELOPS White Paper**. 21

[29] Christoph Sommer, Reinhard German, and Falko Dressler. **Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis**. *IEEE Transactions on Mobile Computing*, **10**(1):3–15, January 2011. 22

[30] David Ciechanowicz, Heiko Aydt, and Alois Knoll. **SEMSim Power as an Application of USES**. In *Proceedings of the IASTED International Symposium on Power and Energy 2013*, 2013. 22

[31] J.S. Dahmann and K.L. Morse. **High Level Architecture for simulation: an update**. In *Distributed Interactive Simulation and Real-Time Applications, 1998. Proceedings. 2nd International Workshop on*, pages 32–40, Jul 1998. 22, 58

[32] Axel Wegener, Michal Piorkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. **TraCI: An Interface for Coupling Road Traffic and Network Simulators**. In *11th Communications and Networking Simulation Symposium (CNS)*, 2008. 22

[33] Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, Kai Nagel, and K Axhausen. **MATSim-T: Architecture and simulation times**. *Multi-agent systems for traffic and transportation engineering*, pages 57–78, 2009. 23

[34] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. **Recent Development and Applications of SUMO - Simulation of Urban MObility**. *International Journal On Advances in Systems and Measurements*, **5**(3&4):128–138, December 2012. 23

[35] Moshe E Ben-Akiva. **SMART–Future Urban Mobility**. *JOURNEYS*, page 30, 2010. 23

[36] Katharina Meier, Alexandra Choutko, Jozica Dolenc, Andreas P. Eichenberger, Sereina Riniker, and Wilfred F. van Gunsteren. **Multi-Resolution Simulation of Biomolecular Systems: A Review of Methodological Issues**. *Angewandte Chemie International Edition*, **52**(10):2820–2834, 2013. 25

[37] S. Wei, B. Cai, S. Li, Z. Liu, and J. Wang. **Multi-resolution simulation strategy and its simulation implementation of Train Control System**. In *Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics*, pages 579–584, July 2011. 25

[38] Paul F. ReynoldsJr, Anand Natrajan, and Sudhir Srinivasan. **Consistency maintenance in multiresolution simulation**. *Transactions on Modeling and Computer Simulation (TOMACS)*, pages 368–392, 1997. 26, 86

[39] Gary Tan, WN Ng, and F Moradi. **Aggregation/disaggregation in HLA multi-resolution distributed simulation**. In *proceedings of the IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, pages 76–83, Cincinnati, Ohio, USA, 13-15 Aug., 2001. IEEE. 26, 86

[40] Martin Adelantado and Pierre Siron. **Multiresolution Modeling and Simulation of an Air-Ground Combat Application**. In *proceedings of the 2001 Spring Simulation Interoperability Workshop*, Orlando, USA, 25-30 Mar., 2001. IEEE. 26, 86

[41] Wilco Burghout and H Koutsopoulos. **Hybrid Traffic Simulation Models: Vehicle Loading at Meso–Micro Boundaries**. In *proceeding of the International Symposium of Transport Simulation*, Lausanne, Switzerland, 04-06 Sep., 2006. 26, 86, 93

[42] Anand Natrajan. **MRE: a flexible approach to multi-resolution modeling**. In *proceedings of the 11th Workshop on Parallel and Distributed Simulation*, pages 156–163, Lockenhaus, Austria, 10-13 June, 1997. IEEE. 26, 87

[43] B.Y.W. Chua and M.Y.H. Low. **Predictive algorithms for aggregation and disaggregation in mixed mode simulation**. In *proceeding of the Winter Simulation Conference*, pages 1356–1365, Austin, Texas, USA, 13-16 Dec., 2009. IEEE. 26, 87

[44] Seon Han Choi, Sun Ju Lee, and Tag Gon Kim. **Multi-fidelity modeling & simulation methodology for simulation speed up**. In *proceedings of SIGSIM-PADS '14*, pages 139–150, New York, New York, USA, 18-21 May, 2014. ACM. 26, 87

[45] Daniel Zehe, Alois Knoll, Wentong Cai, and Heiko Aydt. **SEMSim Cloud Service: Large-scale urban systems simulation in the cloud**. *Simulation Modelling Practice and Theory*, **58, Part 2**:157 – 171, nov 2015. Special issue on Cloud Simulation. 27, 60, 66

[46] TS Saitoh, T Shimada, and H Hoshi. **Modeling and simulation of the Tokyo urban heat island**. *Atmospheric Environment*, **30**(20):3431–3442, 1996. 28

[47] M. Pipattanasomporn, H. Feroze, and S. Rahman. **Multi-agent systems in a distributed smart grid: Design and implementation**. In *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pages 1–8, March 2009. 28

[48] Dominik Pelzer, David Ciechanowicz, Heiko Aydt, and Alois Knoll. **A price-responsive dispatching strategy for Vehicle-to-Grid : An economic evaluation applied to the case of Singapore**. *Journal of Power Sources*, **256**(0):345–353, 06 2014. 28

[49] Heiko Aydt, Michael H Lees, Stephen J Turner, and Wentong Cai. **Toward Simulation-Based Egress Optimization in Smart Buildings Using Symbiotic Simulation**. In *Pedestrian and Evacuation Dynamics 2012*, pages 987–999. Springer, 2014. 28

[50] Winnie Daamen. *Modelling passenger flows in public transport facilities.* DUP Science Delft, the Netherlands, 2004. 28

[51] Qi Yang, Haris N Koutsopoulos, and Moshe E Ben-Akiva. **Simulation laboratory for evaluating dynamic traffic management systems**. *Transportation Research Record: Journal of the Transportation Research Board*, **1710**(1):122–130, 2000. 28

[52] Gabriele D'Angelo and Moreno Marzolla. **New trends in parallel and distributed simulation: From many-cores to Cloud Computing**. *Simulation Modelling Practice and Theory*, 2014. 28

[53] T. Dierks and E. Rescorla. **The Transport Layer Security (TLS) Protocol Version 1.2**, 2008. 35

[54] Ram Ratan Ahirwal and Manoj Ahke. **Elliptic curve diffie-hellman key exchange algorithm for securing hypertext information on wide area network**. *International Journal of Computer Science and Information Technologies*, **4**(2):363–368, 2013. 35

[55] Heiko Aydt, Yadong Xu, Michael Lees, and Alois Knoll. **A Multi-Threaded Execution Model for the Agent-based SEMSim Traffic Simulation**. In *Proceedings of the 13th International Conference on Systems Simulation (AsiaSim)*, 2013. 41, 76

[56] Andrew V Goldberg and Chris Harrelson. **Computing the shortest path: A search meets graph theory**. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165. Society for Industrial and Applied Mathematics, 2005. 41

[57] Ron Gutman. **Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks**. In *Proceedings 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 100–111. SIAM, 2004. 41

[58] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. **Contraction hierarchies: Faster and simpler hierarchical routing in road networks**. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008. 41, 42

[59] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science, 2006. 47

[60] Daniel Zehe, Vaisagh Viswanathan, Wentong Cai, and Alois Knoll. **Online Data Extraction for Large-Scale Agent-Based Simulations**. In *Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation*, pages 69–78. ACM, 2016. 57

[61] Vaisagh Viswanathan, Daniel Zehe, Jordan Ivanchev, Dominik Pelzer, Alois Knoll, and Heiko Aydt. **Simulation-assisted exploration of charging infrastructure requirements for electric vehicles in urban environments**. *Journal of Computational Science*, **12**:1–10, jan 2016. 58, 76, 77

[62] David Ciechanowicz, Dominik Pelzer, and Alois Knoll. **Simulation-based Approach for Investigating the Impact of Electric Vehicles on Power Grids**. In *Proceedings of IEEE PES Asia-Pacific Power and Energy Engineering Conference 2015*, Nov 2015. 58, 76, 77

[63] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. **Data mining with big data**. *Knowledge and Data Engineering, IEEE Transactions on*, **26**(1):97–107, Jan 2014. 58

[64] Wei Fan and Albert Bifet. **Mining big data: current status, and forecast to the future**. *ACM sIGKDD Explorations Newsletter*, **14**(2):1–5, dec 2012. 58

[65] Daniel Zehe, Wentong Cai, Alois Knoll, and Heiko Aydt. **Tutorial on a Modeling and Simulation Cloud Service**. In *Proceedings of the 2015 Winter Simulation Conference*, dec 2015. 58

[66] Rajiv Ranjan. **Streaming Big Data Processing in Datacenter Clouds**. *IEEE Cloud Computing*, **1**(1):78–83, may 2014. 58, 60

[67] Edgar F Codd. **A relational model of data for large shared data banks**. *Communications of the ACM*, **13**(6):377–387, jun 1970. 59, 62

[68] Hugh Darwen and C. J. Date. **The Third Manifesto**. *SIGMOD Rec.*, **24**(1):39–49, mar 1995. 59

[69] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. **Apache Hadoop YARN: Yet Another Resource Negotiator**. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5:1–5:16, New York, NY, USA, oct 2013. ACM. 60, 75

[70] Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, and Ion Stoica. **Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters**. In *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing*, pages 10–10. USENIX Association, jun 2012. 60

[71] Kyle Banker. *MongoDB in Action.* Manning Publications Co., Greenwich, CT, USA, jan 2012. 60

[72] Avinash Lakshman and Prashant Malik. **Cassandra: a decentralized structured storage system**. *ACM SIGOPS Operating Systems Review*, **44**(2):35–40, apr 2010. 60

[73] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. **Models and Issues in Data Stream Systems**. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM. 61

[74] P.M. Kurowski. *Engineering Analysis with SolidWorks Simulation 2012*. SDC Publications, apr 2012. 61

[75] Sham Tickoo. *Autodesk Simulation Mechanical 2015 for Designers*. CADCIM Technologies, sep 2014. 61

[76] Johannes Schützel, Holger Meyer, and Adelinde M. Uhrmacher. **A Stream-based Architecture for the Management and On-line Analysis of Unbounded Amounts of Simulation Data**. In *Proceedings of the 2Nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '14, pages 83–94, New York, NY, USA, may 2014. ACM. 61

[77] Johannes Schützel and Adelinde M. Uhrmacher. **Targeted extration of simulation data**. In *Distributed Simulation and Real Time Applications (DS-RT), 2015 IEEE/ACM 19th International Symposium on*, Oct 2015. 61

[78] Nadeem Mahmood, S. M. Aqil Burney, and Kamran Ahsan. **A Logical Temporal Relational Data Model**. *CoRR*, **abs/1002.1143**, jan 2010. 62

[79] JoÃ£o Gama and PedroPereira Rodrigues. **Data Stream Processing**. In JoÃ£o Gama and MohamedMedhat Gaber, editors, *Learning from Data Streams*, pages 25–39. Springer Berlin Heidelberg, sep 2007. 72

[80] Charles E. Mackenzie. *Coded Character Sets, History and Development*. Addison-Wesley, 1980. 74

[81] Daniel Zehe, David Grotzky, Heiko Aydt, Wentong Cai, and Alois Knoll. **Traffic Simulation Performance Optimization Through Multi-Resolution Modeling of Road Segments**. In *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '15, pages 281–288, New York, NY, USA, 2015. ACM. 83, 86

[82] Arne Kesting, Martin Treiber, and Dirk Helbing. **General Lane-Changing Model MOBIL for Car-Following Models**. *Transportation Research Record: Journal of the Transportation Research Board*, **1999**:86–94, 2007. 88

[83] SP Hoogendoorn and PHL Bovy. **State-of-the-art of vehicular traffic flow modelling**. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, **215**(4):283–303, 2001. 88, 90

[84] D Helbing and A Hennecke. **Micro-and macro-simulation of freeway traffic**. *Mathematical and computer modelling*, **35**(5):517–547, 2002. 90

[85] Serge Hoogendoorn and Victor Knoop. **Traffic flow theory and modelling**. In David Banister, Bert Van Wee, and Jan Anne Annema, editors, *The Transport System and Transport Policy: An Introduction*, pages 125–159. Edward Elgar Publishing Ltd, Cheltenham, United Kingdom, 2012. 90

[86] Arne Kesting, Martin Treiber, and Dirk Helbing. **Agents for traffic simulation**. In Adelinde M. Uhrmacher and Danny Weyns, editors, *Multi-Agent Systems: Simulation and Applications*, pages 325–356. CRC Press, Boca Raton, Florida, USA, 2008. 96

Acronyms

API

Application Programming Interface 8, 9, 14, 15, 28, 30, 31, 38, 44, 45, 51, 55, 60, 67, 120, *Glossary:* Application Programming Interface

AWS

Amazon Web Services 115, 116, *Glossary:* Amazon Web Services

CCA

CityMoS Cloud App xi, 52–54, *Glossary:* CityMoS Cloud App

CityMoS

City Mobility Simulator 9, 21, 22, 26, 29, 30, 37–39, 41, 49, 52, *Glossary:* City Mobility Simulator

CP

Cloud Provider 4, 5, 12, 14, 15, 34, 41, 43, 52, 54, 84, 110, 112, 114, *Glossary:* Cloud Provider

CS

Cloud Service 37, 38, *Glossary:* High-Performance Computing

DVU

Driver-Vehicle-Unit 37, *Gloassary:* Driver-Vehicle-Unit

EDF

Experiment Description File 32, 35, 37, 48, *Glossary:* Experiment Description File

GCE

Google Compute Engine xi, xiii, 13, 28, 31, 42–46, 52, 54, 68, 84, 112–116, *Glossary:* Google Compute Engine

GUI

graphical user interface 28–30, *Glossary:* graphical user interface

HITS

Household Interview Travel Survey 42, *Glossary:* Household Interview Travel Survey

HLA

High Level Architecture 22, 24, 34, 58, *Glossary:* High Level Architecture

HPC

High-Performance Computing 2–5, 7–9, 14, 24, 25, 28–30, 37, 42–46, 50, 54, 55, *Glossary:* High-Performance Computing

IaaS

Infrastructure as a Service 13–15, 30, 54, *Glossary:* Infrastructure as a Service

IDM

Intelligent Driver Model 87, 88, 92–94, 96, *Glossary:* Intelligent Driver Model

PaaS

Platform as a Service 14, 15, 30, 55, *Glossary:* Platform as a Service

REST

Representational State Transfer 8, 30, 67, *Glossary:* Representational State Transfer

SaaS

Software as a Service 15, 30, *Glossary:* Software as a Service

SEMSim

Scalable Electromobility Simulator xi, 8, 9, 11, 21–26, 29, 30, 37–39, 41, 49, 52, 57, 66, 67, 76, 77, *Glossary:* Scalable Electromobility Simulator

SSaaS

Simulation Software as a Service 12, *Glossary:* Simulation Software as a Service

VM

virtual machine 14, 16, 29–31, 33, 40–44, 51, 52, 55, 112, 114, 115, *Glossary:* virtual machine

Glossary

Amazon Web Services

An online service for cloud-computing infrastructure provided by Amazon. It offers, amongst others, Infrastructure as a Service (IaaS). 115

Application Programming Interface

Is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. 8, 31, 120

Big Data

The term has been coined in the early 2000s and describes in what velocity, variety and volume data is generated in a and needs to be processed in order to extract viable insights into a system. 2, 9

City Mobility Simulator

An agent-based nanoscopic mobility simulation framework build on the foundation of Scalable Electromobility Simulator (SEMSim). It extends the traffic simulation to a holistic mobility simulation. 9, 22

CityMoS Cloud App

A software that runs platform independently and is used to execute simulation experiments in the cloud or on a local machine. It uses templates to allow the user to get started with simulation experiments quickly, without much configuration. xi, 52

cloud computing

Is the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer. 4, 5, 9

Cloud Provider

A service provider on the Internet that allows users to allocate and use data-center resources on a pay-per-use bases. 4, 84

Cloud Service

A service, usually on the internet, that is made available to users on demand through APIs or web sites. 37

Docker

A lightweight container technology that allows developers to virtualize entire operating systems without the overhead of a virtualization stack. 51–53

Docker container

A container format that is an instantiation of a Docker Image. This can be running or stopped, but until delete or committed into a Docker Image all data is persistent only inside the container. xi, 52, 54

Docker Machine

A software component that is part of the Docker software family. It allows to provision computing resources capable of running Docker container. This can be on the cloud, a local VM or a physical machine. 52–54

Driver-Vehicle-Unit

A concept of agent-based modeling and simulation, where the combination of driver-behavior and vehicle-component models are used together. 37

Experiment Description File

A file containing all the information (e.g., input-file locations, model parametrization, number of repetitions) to create a simulation experiment. 32, 48

Google Compute Engine

Is a service for cloud infrastructure like virtual machines, storage or networking provided by Google Inc. It offers a IaaS. xi, 13, 84

graphical user interface

Is a type of interface that allows a user to interact with an computer system through graphical representation of icons and indicators. 28

High Level Architecture

Is a general-purpose architecture for distribute simulations. The communication between simulation instances is satisfied regardless of the computing platform 22, 58

High-Performance Computing

A system of different computers that are used collectively to execute distributed applications. 2, 28

Household Interview Travel Survey

A reoccurring interview series conducted by the Land Transport Authority (LTA) in Singapore to get insight into the travel patterns of 1 per cent representativity residents of Singapore. 42

hypervisor

A software or hardware component that runs, manages and creates VMs. 16, 52

Infrastructure as a Service

Is the most basic cloud service model and provides the user with bare computing resources (e.g., CPU, RAM, Network, Storage). Theses resources can be physical or virtual. 13

Intelligent Driver Model

Is a car-following model which is used to simulate freeway as well as urban traffic. 87, 88

macroscopic

A mathematical model that formulates the relationship in traffic flow between density, flow and speed. It represents the lowest resolution in traffic simulation models. 8, 18–20, 85, 87–92, 94–101, 104–106, 118

mesoscopic

A simplified but computationally more efficient traffic model category. It takes aspects from both, the macroscopic and the microscopic modeling paradigms into consideration. 8, 18, 19, 85

microscopic

Simulation model or a set of models that are used in the context of traffic simulation to characterize the granularity of the model. A microscopic model include usually only very limited vehicle components and is mostly concerned with driving behavior (e.g., car-following and lane-changing models). xii, 8, 17–21, 23, 85, 87–101, 105, 116, 117

middleware

A piece of software that bridges communication and data exchange between two system that wouldn't share a mutual exchange protocol 8, 33, 34, 39, 40, 70–73, 79–81

nanoscopic

A simulation model or a set of models that are used in the context of traffic simulation to characterize the granularity of the model. A nanoscopic model includes, in addition to driving-behavior also vehicle-component models. 8, 18, 21, 23, 24, 85

Platform as a Service

Is a cloud computing service that provides a platform to develop, run and manage applications on. It removes the complexity and overhead of building and maintaining the infrastructure associated with developing cloud-based services. 14

Representational State Transfer

Is an architectural style, and an approach to communications that is often used in the development of web services. It can be seen as a API. 8

Scalable Electromobility Simulator

An agent-based nanoscopic traffic simulation framework as well as a power system simulation platform for holistic transportation related questions. xi, 8, 76

Simulation Software as a Service

Is a specialized Software as a Service (SaaS) in the simulation domain. 12

Software as a Service

Is a software licensing and delivery model in which the software is provided on a subscription bases and is centrally hosted (usually cloud). The users access the software through thin clients or a web browser. 15

virtual machine

Is an emulation of particular computer system in software. 14, 112