*Proceedings of the 2015 Winter Simulation Conference*
*L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.*

# TUTORIAL ON A MODELING AND SIMULATION CLOUD SERVICE

Daniel Zehe

TUM CREATE
1 CREATE WAY
Singapore 138602

Wentong Cai

School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

Alois Knoll

Technische Universität München (TUM)
Institut for Informatics
Robotics and Embedded System
Munich, Germany

Heiko Aydt

TUM CREATE
1 CREATE WAY
Singapore 138602

## ABSTRACT

For large-scale urban system simulations the computing power of traditional workstations is not sufficient. The move to High Performance Computing clusters is a viable solution. Users of such simulations are domain experts with little knowledge in computer science and optimization of such simulations. The access to HPC resources is also not available. Vendors have not sufficiently addressed this. This leads to the conclusion of moving the computational demand to the cloud, where the on-demand culture for resources has been expanding. In this tutorial we will present an approach of how to work with an entirely cloud-based solution for modeling and simulation, with an exemplary implementation of an urban traffic simulation cloud service. Since the computational offload from the workstation to a remote computing entity also allows the use of novel user interfaces (design and devices), through the use of RESTful interfaces, use-case applicable interfaces for simulations can also be created.

## 1 INTRODUCTION

There have been many simulation studies that use agent-based models to simulate complex systems. Such systems are related to urban systems such as transportation (Daamen 2004) (Yang, Koutsopoulos, and Ben-Akiva 2000) and social sciences (Aydt, Lees, Turner, and Cai 2014), climate sciences (Saitoh, Shimada, and Hoshi 1996) (Wagner, Viswanathan, Pelzer, Berger, and Aydt 2015) or energy studies (Pelzer, Ciechanowicz, Aydt, and Knoll 2014) (Pipattanasomporn, Feroze, and Rahman 2009). The complexity of such simulations has already been recognized as large. Therefore, climate models are already computed on dedicated supercomputers. At the same time, traffic engineers are still using workstation computers to simulate traffic phenomena. With an increasing granularity of models or the size of road networks, workstation computers with their limited computational resources are no longer feasible. Similar developments can be seen with crowd simulations or agent-based models of energy systems. A large-scale agent-based simulation of i.e. an urban traffic system with several hundred thousand agents has to be run on a high performance computing (HPC) system in order to obtain results quickly. Even if the execution time is not of major concern to the simulator, the physical constrains of the hardware resources of current workstation computers is very limited. This means that it is impossible to hold the state of several thousand agents in main memory

and retrieve it when needed. The typical users of such simulation systems are transportation engineers, social behaviour scientists or urban policy makers. These users do not have the expertise in computer science, nor access to high performance computing clusters. With the onset of cloud computing, the availability of high performance hardware has improved rapidly. HPC-like resources are readily available via cloud computing providers. Google Compute Engine (GCE) and Amazon $EC^2$, just to name a few, offer on-demand computational resources without any upfront costs.

The rest of this tutorial paper is structured as follows. We will be introducing the topic of cloud computing and focussing advances in the applications of cloud computing for simulation and research. Afterwards, we are going to elaborate on the back-end services necessary to enable an agent-based simulation cloud services and will then show an exemplary implementation of a cloud service for an agent-based nanoscopic traffic simulation called SEMSim traffic.

## 2 CLOUD COMPUTING

The infrastructure services include (virtualized) hardware resources in form of computing, networking and storage services. Whereas PaaS offers database, APIs or execution run times that can be used by developers to program against. The SaaS takes the services idea one step further by providing web-based application (-suits) like email or document editors. Figure 1 shows how services can be built up from IaaS, with client devices or thin-clients using it at the highest level. This does not mean all PaaS or SaaS have to be built on top of IaaS. At any level, a dedicated solution is also viable. It might even be favourable to base a PaaS on real hardware in order to optimize the performance as opposed to using the general purpose or virtualized hardware.

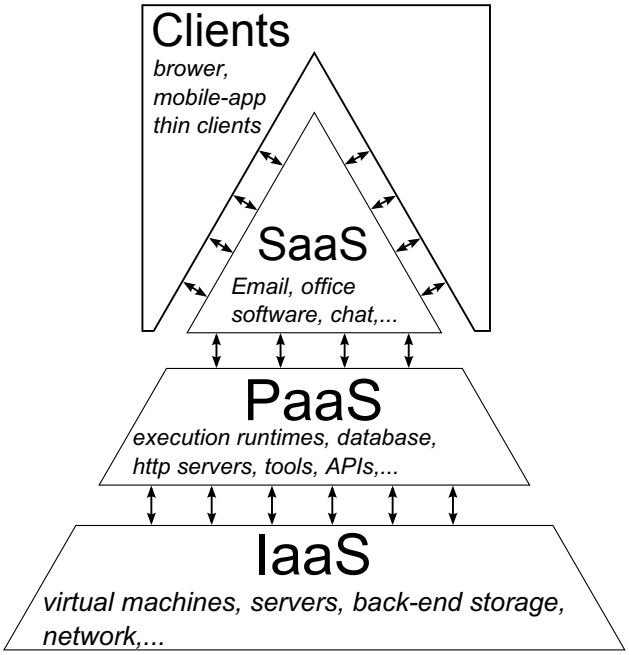



Figure 1: Hierarchy of current cloud services

All *Cloud Service Providers* (CSP) that offer infrastructure resources offer APIs and SDKs for convenient access to create, maintain and deallocate resources on demand as well as a web interface for manual control. This allows developers to dynamically scale their applications on demand. Such scaling and deployment strategies are highly flexible and can be adopted by the scientific community, where sometimes a lower budget for experiments can be overcome by longer running experiments and vice versa. The pricing model is usually time-based pricing, on actual usage. This is schematically illustrated in Figure 2. A

continuous-use policy is also in place for some providers, where a discount is offered when resources are used for a larger period of time. This is visible in the different slopes between the continuous and non-continuous cloud-usage model. When using cloud-based resources continuously for a longer period of time, it will eventually equalize the higher upfront costs and the maintenance cost of a dedicated HPC. Should the resources only be used when needed, the cloud solution provides a better cost model. There is also no maintenance and upgrade necessary that would increase the cost while at the same time having the resources unavailable to use. It has also been shown that the performance of cloud-based infrastructure for simulations is not much worse then dedicated hardware (Taylor, Anagnostou, Kiss, Terstyanszky, Kacsuk, and Fantini 2014),(Zehe, Grotzky, Knoll, Wentong, and Aydt ).

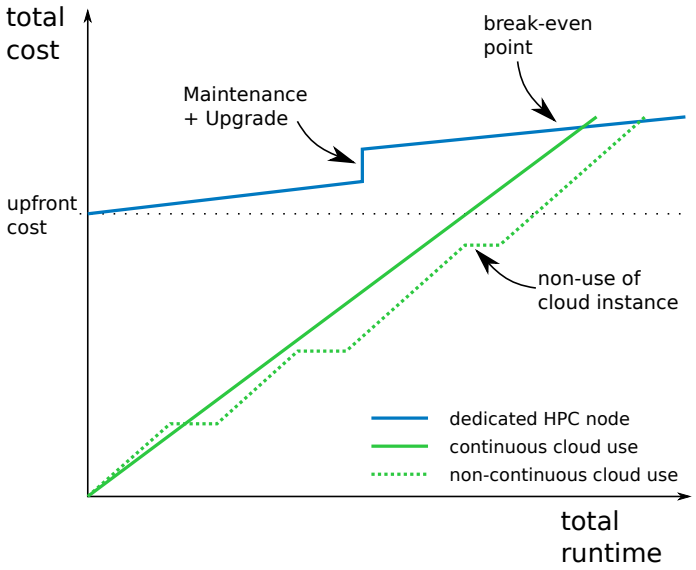



Figure 2: Schematic cost-difference model between dedicated resources and cloud resources in continuous and non-continuous use

A rather new technology connected to cloud computing is *Docker* (http://www.docker.com). It is a container engine that runs (but does not have to) on IaaS and encapsulates a full runtime operating system stack without the overhead of a virtual machine. This allows a user to run many containers on the same physical or virtual machine with completely different software dependencies. Since this offers a standardized interface to the underlying OS, it runs on any computational resources that uses docker without any configuration or installation. This enables developers not only to ship their applications, but also to ship the entire execution environment with it.

### 2.1 Simulations in the Cloud

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 *Simulation Software as a Service* (SSaaS). The research project CloudSME (http://cloudsme.eu) 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 to be used for cloud-based modeling and simulation (Taylor, Anagnostou, Kiss, Terstyanszky, Kacsuk, and Fantini 2014). SimScale (http://www.simscale.de/) and AutoDesk (http://www.autodesk.com/products/sim-360) both offer simulation services for computer aided design models. The simulation services include simulation-based tests for fluid-dynamics, structural mechanics and thermal evaluation on digital prototypes. Another example is Rescale (http://www.rescale.com) 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 catalogue of simulation software includes among other 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 (Huqqani, Li, Beran, and Schikuta 2010).

## 3 BACKEND SYSTEM DESIGN

In this section we are going to introduce the back-end components of a Cloud service for agent-based simulations. An illustration on how the individual components work together in the back-end is given in Figure 3. It shows that the user input from the front-end in the form of RESTful API calls can trigger different back-end responses.

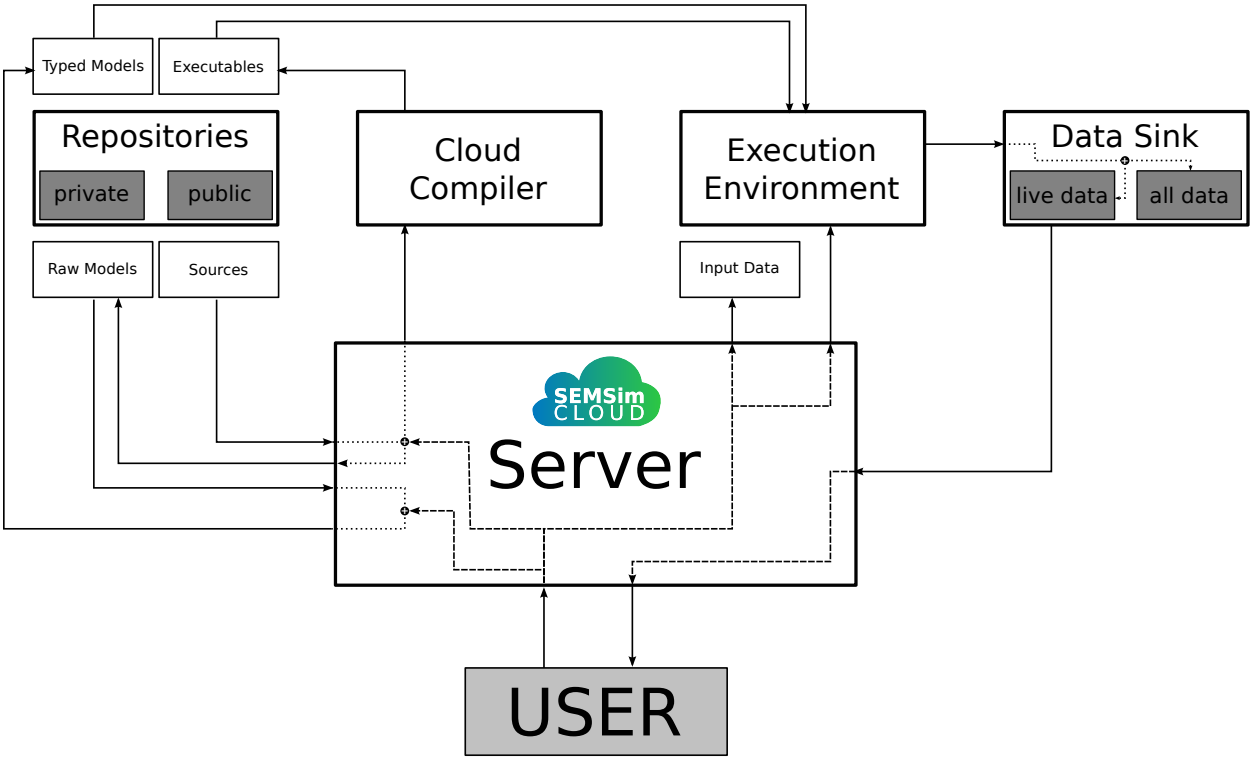



Figure 3: Cloud Simulation back-end services. This includes the coordinating server, repositories, a cloud-based compiler, the actual execution environment and a data sink that will hold the output of the simulation experiments.

### 3.1 Model and Type Repositories

A model in the cloud-based simulation service is defined as a class or a set of classes written in C++ that define a certain behavior depending on the parametrization and the current values of the input parameters. Such models of a specific component in an agent-based simulation can be developed by domain experts. These models are to be parametrized in order to be used in a simulation. The parametrization has to be done in a way that the execution of the model represents the modeled real-world as closely as possible. This process is called calibration. Since calibration of a model always depends on the real-world scenario that it is intended to be used in, there can be different parameterizations of the same underlying model. Such parametrized models are called types in our simulation. Since both, the raw models and the types can

later be used to compose simulation experiments, a repository of storing them is necessary. The description of a raw model is a list of parameters a certain model can have in the simulation, whereas the type is one combination of these parameters with values. There are public and also private repositories for models and types. A public repository allows text-book models and their types to be used as a starting point or for simulations that do not have the focus on this model. It allows users to concentrate on developing novel models instead of recreating trivial models over and over again.

### 3.2 Cloud Compiler

In order to develop models that can be used for the cloud-based simulation platform, models can be developed by domain experts in the high or low level language in which the simulation is written. Should the source code be available to the developer of the model, this can be done on the respective computer of the developer. In order to reduce the installation time overhead, a compilation environment in the form of a pre-configured Docker container can be obtained and used for compilation checking. The source file should then be submitted to the cloud compiling platform, which takes care of creating the final executable. Base source files are modified by the user as shown in Figure 3 and then compiled in order to create executables that can be run later. If the entire source code cannot be made available to all developers, an alternative option would be to offer a web-based model editor and compilation checker, that allows to create or extend models. Along with a new or changed model, the developer has to provide a list of parameters, that can be used for parametrization of the model. This list should also contain a hint to what kind of input is to be expected by the simulation at runtime, reaching from different data types (int, double, float, string, boolean,...) to a fixed value or a given distribution. In addition to the pure source code, a model directory for each binary is created that is then used to determine which types are accepted by the simulation at execution.

### 3.3 Execution Environment and Encrypted Data Repository

All simulations are executed on infrastructure resources obtained from CSPs. Depending on the size and the estimated resource consumption, a virtual machine will either be entirely used by one simulation experiment run in a pre-defined docker container or several docker containers will be run on one virtual machine. This is only advisable for simulation with a low resources footprint. The resource specifications of the VM also depend on the size of the experiment, the time and financial constrains given by the experimenter. Since each simulation needs input data, this will be provided by the input data repository, which stores user-specific, but also general encrypted input data. This data will then be decrypted at runtime, where a decryption key needs to be provided through the front-end user interface or a Machine-to-Machine interface in the form of a key server.

### 3.4 Data Output

Data output can be provided using either a real-time stream of certain variables from the simulation or as part of a data dump for post simulation analysis. For the first option, a mechanism within the front-end interface, as well as the simulation, has to be implemented in order to allow for such data extraction. The experimenter needs to provide a (TLS-)secured end-point for the real-time data stream to which the data will then be streamed. The second option, of storing the simulation results locally only requires the specification of the state variable that is required. This will then be written to a file or database and stored on a cloud storage solutions for easy access by the experimenter. All persistently stored simulation data will be encrypted, using a key that is provided to the user at the start of the simulation. A download link will has to be provided to the user, from which a download or a transfer to a different cloud resources will be initiated. The simulation results will be deleted shortly after the simulation ends since the space constraints are also a factor that increases the cost.

### 3.5 Server

In order to coordinate the compilation, the storage and retrieval of the models as well as types, the preparation of the execution environment for simulations and the data output, a centralized server has to be in place. This server also takes care of starting multiple instances for a given experiment in order to reduce the error in a stochastic simulation or to do a parameter sweep of a given input variable. Another responsibility of the server is the allocation of output data storage and user management in order to notify a user of simulation status and provide information about the location of output data. All user interactions with the server are done through RESTful API calls from front-end applications.

## 4 EXPERIMENTS

In this section we want to show an exemplary demonstration of a large-scale agent-based simulation modeled, parametrized, executed and analyzed in the cloud. For this, we are using an agent-based traffic simulation engine. The entire front-end is communicating via a RESTful API with the back-end server as outlined in Section 3. This enables different user interfaces (apart from the web-based shown in this section) to be developed by third parties. Since all steps related to running a simulation experiments are not run on a workstation computer, the user-interfaces can be different. A web-based approach is the smallest common denominator for a large variety of devices from small form factor mobile phones to tablets and even TV screens or video walls. A custom interface for the specific input method is always preferred, since a better user experience increases the acceptability by the respective user. This is especially important for the data analytics and visualization components. A novel approach to not only display but also interact with live or history data can help users to understand the complex results generated by large urban-systems simulations.

### 4.1 SemSIM Traffic

The *Scalable Electromobility Simulator* (SEMSim) *Traffic* is an agent-based traffic simulation that is used in the context of the SEMSim platform, which also includes a power system simulation. This allows for a holistic electromobility simulation of an entire city, not only looking at the traffic but also the power systems angle of electrification of current and future mega-cities. The scope of our electromobility research is currently limited to Singapore as a current mega-city, but the developed methodology can be easily transferred to other cities and regions as well.

In order to evaluate the complex interactions between the traffic of a city and its power system, SEMSim Traffic's major difference is in the level of detail in comparison to existing traffic simulation platforms (e.g., MATSim or SUMO). Within a SEMSim Traffic simulation every agent is represented as a so-called driver-vehicle-unit (DVU) which consists of driver behavior models as well as vehicle component models. These models can include drive-train models, models for air-conditioning or the battery pack of an electric vehicle. SEMSim Traffic simulates the traffic on a lane-level resolution which qualifies it to be a microscopic simulation. However, due to the fact that the internal state of a vehicle is also expressed through a model, we refer to it as a nanoscopic simulation. Unlike in time-stepped simulation tools, a discrete event simulation engine allows to not recalculate the state of the simulation every time step, but when events occur at arbitrary times. This gives the flexibility to vary the the update of certain models more infrequent than others (e.g., moving agents on the network vs. charging behavior). The interaction of the different possible models within the traffic simulation SEMSim Traffic and a power system simulation on the different abstraction levels is shown in Figure 4.

The input data for a SEMSim Traffic based simulation experiment is rather small. A road network that is on a lane-level accuracy and is directly connected to a routing model which works on a link-base and is solely used for calculating routes for the agents in the simulation. Different kinds of routing modes are possible. Depending on the weighting of individual links, a distance or travel time based routing network might be preferred. In addition to this, only two more input files are necessary. One defines the working of

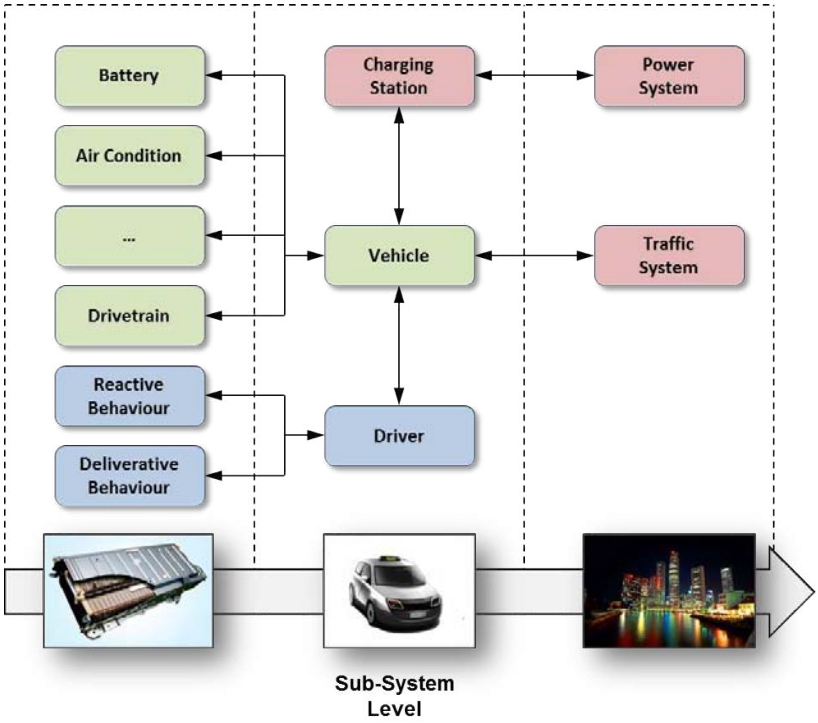



Figure 4: Overview of system, sub-system and component level entities of the SEMSim Platform

traffic lights at intersections and the other one the travel patterns. This is a tempo-spacial origin-destination matrix for the travel behaviors of a megacity like Singapore for a day. It can be differentiated between travel patterns on weekdays and weekends. On weekdays the main travel routes are between the residential areas and work areas, whereas on weekends the leisure and commercial districts are the main destinations.

### 4.2 Experimentation Workflow

Since the simulation is going to be run on infrastructure in the cloud that could potentially be unsafe, data security is crucial to ensure confidentiality between data provider and researchers. Therefore, all input data is encrypted and will never be available in unencrypted form, except in the simulation itself. The only application that is not cloud-based is a small tool, to encrypt input data. It takes any kind of input data and encrypts it using a AES CBC encryption mechanism with a 32 bit Key (Ehrsam, Meyer, Smith, and Tuchman 1978). It creates an encrypted file format that can later be read by the simulation at initialization. Any output data created will also be encrypted the same way should an output file be generated: live-streamed data expects a TLS secured endpoint that at least offers channel security. The decryption key can be provided in one of two ways, either by having a dedicated key callback server that provides the key when needed, or by providing the key on demand on the web-site or a mobile device.

One of the models that can either be uploaded if acquired from a third party or created from scratch is a road network model. Even tough for many experiments a real-world and large scale road network is necessary, for some experiments an artificial traffic network is required. The network editor in the SEMSim Cloud Service helps the researcher to manage, upload, generate and edit networks as well as accompanying data (e.g., travel patterns or intersection controls). This web-based network editor is shown in Figure 5. The editor shown in Figure 5a illustrates the network generation and editing process of the tool. By clicking on the generate button in lower right corner, the back-end generates a random, rectangular or circular network. This generated network can then be edited by selecting nodes or edges to fit the required characteristics. Editing a network allows for specifying the number of lanes on each link as well as other regulation specific

characteristics like the speed-limit. It is also possible to create a network from scratch by adding edges, nodes and lane information in an interactive tool. These networks can then be named and saved in order to be used in subsequent simulation experiment design process. The network editor shown in Figure 5b shows a snippet of an existing network for visualization or editing if necessary. This edited network can also be named and added to the personal collection of the user.


(a) Generated network Editor

(b) Real network Viewer

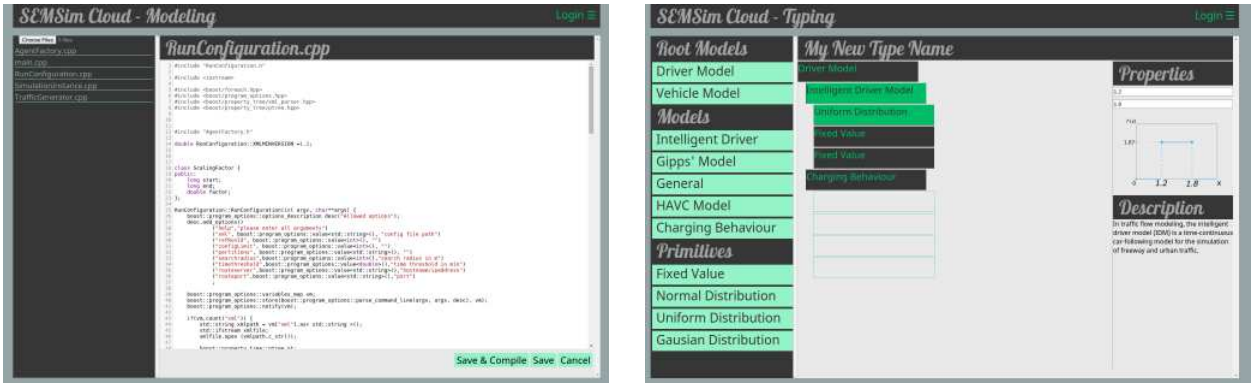
Figure 5: Web-based Network Editor

After the input data has been prepared, the modeling phase of creating a cloud-based simulation experiment begins. For a traffic simulation, standard behavioural models can be parameterized in order to create types. This includes textbook models like *Intelligent Driver Model* (IDM) and Gipps' model. These models can also be extended or different models for driving behaviour can be created. They have the same signature as the existing models but internally the behaviour can be changed. This can be done in a web-based text editor, while a parametrization of a model can be done in a more specialized editor. The web-based editor can function as a code editor that allows the user to write high level C++ models and test if they actually compile within the entirety of the simulation code. A basic web-based editor can be seen in Figure 6a. The compilation test is done by invoking a docker container that compiles the modified source code and returns the compilation status. Only models that pass the compilation test can be saved to be used in a simulation experiment later.

The parametrization process is shown in Figure 6b, where the specialised drag-and-drop typing tool is used to create driver or vehicle types. It starts with dragging a root model (driver or vehicle model) to the empty canvas and then models from different categories that are fitting or necessary for the given model or sub-model. This process can be repeated until only values need to be added to the model. There primitives have to be chosen to reflect either a fixed value or a distribution of values (e.g., Normal, Gaussian, Uniform). By selecting a primitive in the model type, the properties of this primitive can be changed (here the a and b values of uniform distribution). This will also trigger a visual representation of the entered values. Selecting a primitive or a model, a short description about the model and its characteristics is presented. In order to save this type and also to find it later in the experiment design phase, a name has to be given.

Combining these types into a simulation run is done by using the experiment designer component of the web-based application. Until now, the modeling was only done to the extent of creating types of vehicle models or driver behaviour models. These models will now be used to create a heterogeneous population of agents within the simulation, by combining different vehicle models with various driver models. The browser-based application is split into three regions and is shown in Figure 7:

1. *Run-time parameter* consists of the simulation binary selection that is going to be run. This is either a standard SEMSim simulation with minimum text-book models or a self-complied version using the modeling section of the Cloud Service. In addition to selecting a simulation, a network




(a) C++ high level modeling

(b) Model parametrization called typing

Figure 6: Modeling and type creation in browser

on which the simulation is to run has to be selected. Either a standardized public network can be used or a network that has been modeled using the network editor (see above). This includes not only one network file, describing the road network layout, but also a simplified network used for route calculation as well as a traffic generation template. This template allows the simulation to generate traffic specific for this city/region depending on the time and location of the travel demands. Another critical parameter that needs to be configured first is the cloud resources that are going to be used while running this experiment. This starts off with only 2 cores and can go all the way up to 16 cores but could easily be extended to even more. It also determines the price of a single simulation run. The minimal and maximal resources available are dependent on the CSP specified as the preferred CSP in the user profile.

2. *Experiment parameters* are the general parameter that includes the simulation time, the number of agents that will be generated and the number of replications that will be performed. The number of replications has a large influence on the price of the simulation experiment. All replications are executed simultaneously on different machines with the performance characteristic determined in the run-time parameter section. The number of replications is very much necessary, since many of the parameters used are of stochastic nature and a single simulation run, might not produce a usable outcome. Once the simulation is about to start, a confirmation popup window is shown where the estimated price and simulation time is shown. This information is preliminary and changes can be seen in the live data section of the analysis part of the SEMSim cloud service once the actual performance is measurable by the simulation run.
3. *Population Generator* (bottom part of Figure 7) uses the types (driver and vehicle) that have been created using the typing tool and constructs a heterogeneous agent population. Since each agents consists of a DVU, each agent profile consists of a driver type and a vehicle type, as well as a percentage of the total agent population. A name for this agent profile has to be provided as well, in order to later identify specific agent profiles within the simulation results.

Experimentation results are the most important aspect of every experiment run. Therefore, the data analysis section is an important part of the SEMSim Cloud Service. It provides a simple interface to three types of data. One is a live data stream of a specific state variable during the simulation. For a multiple simulation runs executed simultaneously, a data stream of only one simulation is provided. The second data source is the data that has been created during the simulation and marked to be outputted in the modeling phase. This data is already statistically pre-processed. A third data type is the possibility of downloading all the raw data that has been generated by the simulation and evaluating it externally. An exemplary screen shot can


Figure 7: Web-base experiment Designer

be seen in Figure 8, where the population distribution over the simulation time as well as the performance of the simulation is shown for a selected past simulation.

## 5 CONCLUSION

This paper provides a short exemplary tutorial on how a cloud-based simulation service for a large urban-systems simulation running on IaaS solutions should work. It uses an agent-based traffic simulation to present the complexity of modeling traffic simulation agents, design experiments and allows access to live and post-simulation results using a web-based interface that uses a RESTful API as the underlying framework to communicate to a back-end service. The SEMSim Cloud Service provides researchers, policy makers and traffic engineers with an easy-to-use solution to run large-scale simulations without the necessity of investing in expensive computing infrastructure or computer science expertise to set up high performance computing capabilities. The scalable nature of cloud computing resources from resources with low computing capabilities, to high-performance compute nodes, enables the user to scale from a rather inexpensive but long-running simulation experiment to a fast result-oriented research culture. The authors are inviting interested parties to collaborate and test the platform in order to evolve the underlying simulation engine, while at the same time investigate interaction methodologies for next generation research tools.

## ACKNOWLEDGMENTS

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) program.

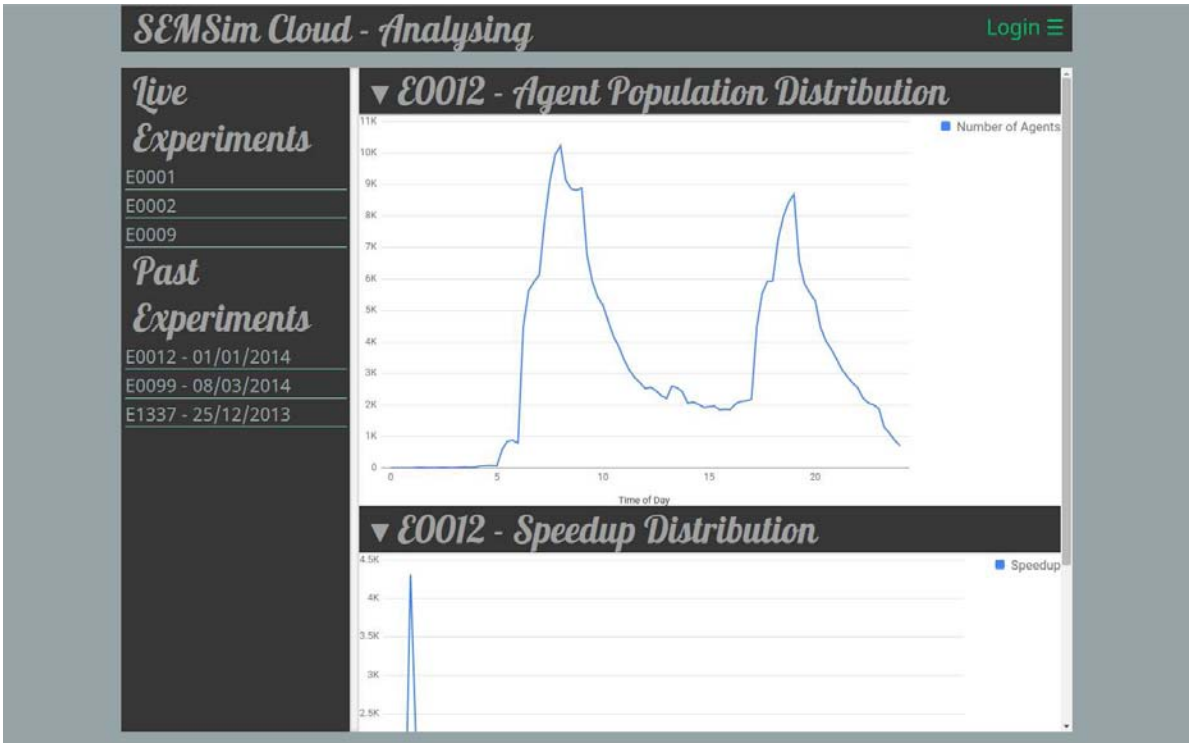



Figure 8: Analysis and data download section

## REFERENCES

Aydt, H., M. H. Lees, S. J. Turner, and W. Cai. 2014. "Toward Simulation-Based Egress Optimization in Smart Buildings Using Symbiotic Simulation". In *Pedestrian and Evacuation Dynamics 2012*, 987–999. Springer.

Daamen, W. 2004. *Modelling passenger flows in public transport facilities*. DUP Science Delft, the Netherlands.

Ehrsam, W.F. and Meyer, C.H.W. and Smith, J.L. and Tuchman, W.L. 1978, February 14. "Message verification and transmission error detection by block chaining". US Patent 4,074,066.

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

Pelzer, D., D. Ciechanowicz, H. Aydt, and A. Knoll. 2014, 06. "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.

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

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

Taylor, S., A. Anagnostou, T. Kiss, G. Terstyanszky, P. Kacsuk, and N. Fantini. 2014, Dec. "A tutorial on Cloud computing for Agent-based Modeling & Simulation with Repast". In *Simulation Conference (WSC), 2014 Winter*, 192–206.

Wagner, M., V. Viswanathan, D. Pelzer, M. Berger, and H. Aydt. 2015. "Cellular Automata-based Anthropogenic Heat Simulation [Currently Under Review]". In *Proceedings of the International Conference of Computer Science, Reykjavik, Iceland*.

Yang, Q., H. N. Koutsopoulos, and M. E. Ben-Akiva. 2000. "Simulation laboratory for evaluating dynamic traffic management systems". *Transportation Research Record: Journal of the Transportation Research Board* 1710 (1): 122–130.

Zehe, D., D. Grotzky, A. Knoll, C. Wentong, and H. Aydt. "Traffic Simulation Performance Optimization through Multi-Resolution Modeling of Road Segments".

## AUTHOR BIOGRAPHIES

**DANIEL ZEHE** received a B.Sc and M.Sc degree in computer systems in engineering from Magdeburg University, Magdeburg, Germany in 2011 and 2012 respectively. He is currently working as a Research Associate at TUM CREATE, Singapore and pursues a Ph.D. at Technische Universitt Mnchen, Munich, Germany. Prior to this, he worked at BMW Research and Technology. His fields of research include multi-resolution modeling for traffic systems and cloud computing for large-scale agent-based simulation. His email address is daniel.zehe@tum-create.edu.sg.

**ALOIS KNOLL** received a Dipl.Ing. (M.Sc.) degree in electrical/communications engineering from the Universitt Stuttgart, Stuttgart, Germany,in 1985 and the Ph.D. degree (summa cum laude) in computer sciencefrom the Technische Universitt (TU) Berlin, Berlin, Germany, in 1988. Since autumn 2001 he has been a professor of Computer Science at the Computer Science Department of the Technische Universitt Mnchen (TUM), Munich, Germany. He is also on the board of directors of the Central Institute of Medical Technology at TUM (IMETUM); from 2004 to 2006 he was Executive Director of the Institute of Computer Science at TUM. His research interests include cognitive, medical and sensor-based robotics, multi-agent systems, data fusion, adaptive systems and multimedia information retrieval. His e-mail address is knoll@in.tum.de.

**WENTONG CAI** is a Professor in the School of Computer Engineering at Nanyang Technological University (NTU), Singapore. He is also the Director of the Parallel and Distributed Computing Centre. He received his Ph.D. in Computer Science from University of Exeter (UK) in 1991. His expertise is mainly in the areas of Modeling and Simulation (particularly, modeling and simulation of large-scale complex systems, and system support for distributed simulation and virtual environments) and Parallel and Distributed Computing (particularly, Cloud, Grid and Cluster computing). He is an associate editor of the ACM Transactions on Modeling and Computer Simulation (TOMACS) and an editor of the Future Generation Computer Systems (FGCS). His email address is aswtcai@ntu.edu.sg.

**HEIKO AYDT** received the Ph.D. degree in computer science from Nanyang Technological University (NTU) in Singapore and an M.Sc. degree from the Royal Institute of Technology (KTH) in Stockholm, Sweden. He also holds a Dipl.Ing. (FH) degree in information technology from Esslingen University of Applied Sciences, Esslingen, Germany. Prior to his doctoral studies, he gained experience as Software Engineer in the automotive industry. In December 2006, he joined the Parallel and Distributed Computing Centre at NTU as Research Associate where he worked on projects concerned with simulation-based optimization and agent-based crowd simulation. In December 2011, he joined TUM CREATE as Research Fellow where he assumed the role of Principal Investigator for the Modeling and Optimization of Architectures and Infrastructure group in December 2012. His research is concerned with analyzing the potential impact of electromobility on the infrastructure and the environment from a complex systems perspective. His current research interests are agent-based simulation, complex adaptive systems, symbiotic simulation, evolutionary computing and simulation-based optimization. His email address is Heiko.Aydt@tum-create.edu.sg.