SEMSim Cloud Service: Large-Scale Urban Systems Simulation in the Cloud

Daniel Zehe[a], Alois Knoll[c], Wentong Cai[b], Heiko Aydt[d]

[a]*Daniel.Zehe@tum-create.edu.sg, TUM CREATE, 1 CREATE WAY, Singapore*
[b]*Nanyang Technological University, School of Computer Engineering, Singapore*
[c]*Technische Universität München (TUM), Institute for Informatics, Robotics and Embedded Systems, Germany*
[d]*TUM CREATE, 1 CREATE WAY, Singapore*

Abstract

Large-scale urban systems simulations are complex and with a large number of active simulation entities the computational workload is extensive. Workstation computers have only limited capabilities of delivering results for large-scale simulations. This leads to the problem that many researchers and engineers have to either reduce the scope of their experiments or fail to execute as many experiments as they would like in a given time frame. The use of high-performance computing (HPC) infrastructure offers a solution to the problem. Users of such simulations are often domain experts with no or little experience with HPC environments. In addition users do not necessarily have access to an HPC. In this paper we propose an architecture for a cloud-based urban systems simulation platform which specifically aims at making large-scale simulations available to typical users. The proposed architecture also addresses the issue of data confidentiality. In addition we describe the Scalable Electro-Mobility Simulation (SEMSim) Cloud Service that implements the proposed architecture.

Keywords: Simulation, Cloud-Based Simulation, Performance Evaluation, Urban Systems Simulations, High Performance Computing, Traffic Simulation, Agent-based Simulation

1. INTRODUCTION

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 [1]) , energy studies (e.g., smart grids [2] or vehicle-to-grid [3]), health (e.g., pandemics [4]), social science (e.g., crowd evacuation [5]), and transportation (e.g., public transport [6] and traffic management [7]) to name only a few. Agent-based models are commonly used for simulating urban processes, such as transportation for example, and are often the only feasible way to study the urban systems of interest. With an increasing interest in city science and research, we expect to see more 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 (e.g., 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 Google Compute Engine (GCE) [1] or Amazon EC2[2]. Given the availability of HPC resources, we expect to see more cloud-based simulation services to emerge. It has been shown [8] 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 we want to address in this paper.

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

- Usability – Graphical user interfaces 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 not directly communicate with the simulation but rather through a defined API over the internet. This enables many different UIs for different aspects of the same simulation (e.g., configuration, visualisation, data analysis).

- Data Confidentiality – 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.

In this paper we introduce a general architecture for cloud-based simulation services that addresses the two issues mentioned above. In addition, we introduce the Scalable Electro-Mobility Simulation Cloud Service (SEMSim CS), a proof-of-concept implementation of the proposed general architecture. SEMSim CS is used in the context of electromobility research in order to help answering questions regarding the impact of the introduction of electric vehicles into an existing transportation system and energy infrastructure of a mega-city such as Singapore. We evaluate the performance of the SEMSim CS on a typical cloud-based virtual machine (VM) and compare it to the performance on a dedicated HPC machine.

2. RELATED WORK

2.1. Web-based and Cloud-based Simulations

Web-based simulations have been used in the scientific community for many years. Most of the early applications of combining the web with simulations consisted of providing collaboration platforms or independent front-end interfaces [9, 10]. 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 [11] for load-balancing or load optimization. For collaboration between different institutions web-based repositories and interfaces for consolidating the modeling efforts have always been popular [10].

The question of Modeling & Simulation as a Service (MSaaS) was surveyed by Cayirci [12]. 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

[1] `https://cloud.google.com/products/compute-engine`

[2] `https://aws.amazon.com/ec2`

infrastructure. The survey largely focuses on the security, privacy and trust concerns 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 for general cloud computing instances [13]. Those are then evaluated and reduced to risks which are of concern for MSaaS. Cayirci concludes that it comes down to trust between the experimenter and the different CSPs and a possible lock-in to a specific CSP.

Guo et al. [14] 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 research project CloudSME[3] is concerned with the use of cloud-based simulations for the manufacturing and engineering industry. This project has shown it potential for being a viable option to be used for cloud-based modeling and simulation [15]. SimScale[4] and AutoDesk[5] 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. The *Altair Simulation Cloud Suite*[6] allows researchers and engineers in that have a CAD and CAE work flow for their simulations to move it completely to a cloud based solution. This allows users to import their CAD models into the browser based interface and allows a off-site simulation life cycle management, leveraging on their Hyperworks simulation tools for fluid dynamics and CFD simulation and exploration as well as visualisation. Another example is Rescale[7] 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 [16].

2.2. Cloud and High Performance Computing

The terms *cloud-service* or *cloud-based* have been used in mass-media and mainstream for a while now. But there are subtle differences between the different kinds of cloud computing solutions provided by commercial companies [17]. Also the difference between using dedicated hardware resources to using virtualized resources is important in order to distinguish between using cloud-based virtualized hardware and dedicated hardware. Firstly, we want to introduce the different kind of cloud services before differentiating cloud-based hardware and dedicated hardware.

Figure 1 shows the different stages of cloud services. Infrastructure as a Service (IaaS) is the underlaying foundation for many other services. It provides the virtual back-end resources.

[3]`http://cloudsme.eu`
[4]`http://www.simscale.de/`
[5]`http://www.autodesk.com/products/sim-360`
[6]`http://www.altair.com/simulation-cloud/`
[7]`http://www.rescale.com`

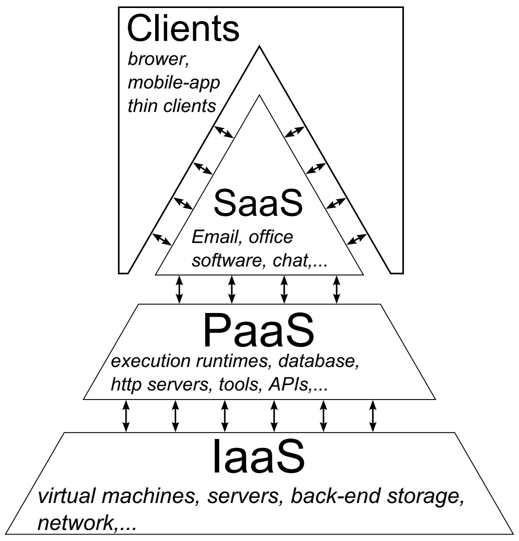

Clients
brower,
mobile-app
thin clients
SaaS
Email, office
software, chat,...
PaaS
execution runtimes, database,
http servers, tools, APIs,...
IaaS
virtual machines, servers, back-end storage,
network,...

Figure 1: Differences between different kinds of cloud services

These resources are usually shared between many users and applications. IaaS cloud services such as Azure[8] from Microsoft, Elastic Cloud from Amazon or Compute Engine from Google offer computing resources on-demand. The drawback of using cloud infrastructure resources for a longer period of time is the cost [18]. However, the benefit of using infrastructure resources, is the elimination of initial hardware setup costs. 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 [19]. On the other hand HPCs offer computing resources at high initial costs but very little operational costs. In comparison with rented and shared resources from an IaaS CSP, the performance of the system is deterministic. Applications are executed directly on the physical hardware 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 or distributed applications which use message passing interface (MPI) technology or can also be run [20].

Platform as a Services (PaaS) can be implemented either on top of the IaaS or by using dedicated hardware instead of a virtual machine. These kind of services are typically application programming interfaces that are used for varying use-cases. The Dropbox Datastore[9] can be considered as a PaaS, because developers can use it to build applications that use a key-value-store to exchange small amounts of data. 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[10] or Google Drive[11] 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

[8]http://www.azure.microsoft.com/

[9]https://www.dropbox.com/developers/datastore

[10]https://www.dropbox.com

[11]https://drive.google.com

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.

2.3. Simulation User Interfaces and Data Security

In this section we would like to present the current state-of-the-art of traffic simulation front-end applications. We will focus on two traffic simulations: MATSim and SUMO, because they are microscopic traffic simulations similar to our own SEMSim Traffic. MATSim does not offer a standard graphical user interface. The main configuration method is configuration files that are loaded at the start of the simulation program. For outputting data MATSim writes the simulation results into files and these can then be visualized using external visualization tools. Having a command line interfaces allows MATSim to be run on various hardware configurations, including head-less 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, be also 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 run on a head-less 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. SUMOs 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 MATSim nor SUMO offers any built in data de-/encryption functionality for input or output data. For this reason we decided to use our own SEMSim traffic simulation, which has support for AES encrypted input data as well as secured output data storage and streaming. In addition, it also offers, similar to the two other simulation engines, a head-less execution mode with file-based configuration. Together with the SEMSim platform tools which provided (REST) APIs to generate such input files with any kind of (graphical) user interface, SEMSim traffic can be easily configured.

3. General Architecture for Cloud-based Simulation Services

The intention of a cloud-based simulation service is to migrate the simulation software into the cloud and provide users with appropriate graphical user interfaces and application programming interfaces. 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. 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 virtual machines. 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 approximate idea about where the virtual machines are located and what type of physical resources they run on.

3.1. 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. Generally, all front-end applications interact with the back-end by means of corresponding application programming interfaces (APIs). 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[12]). 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 2 illustrates an overview of the proposed architecture.

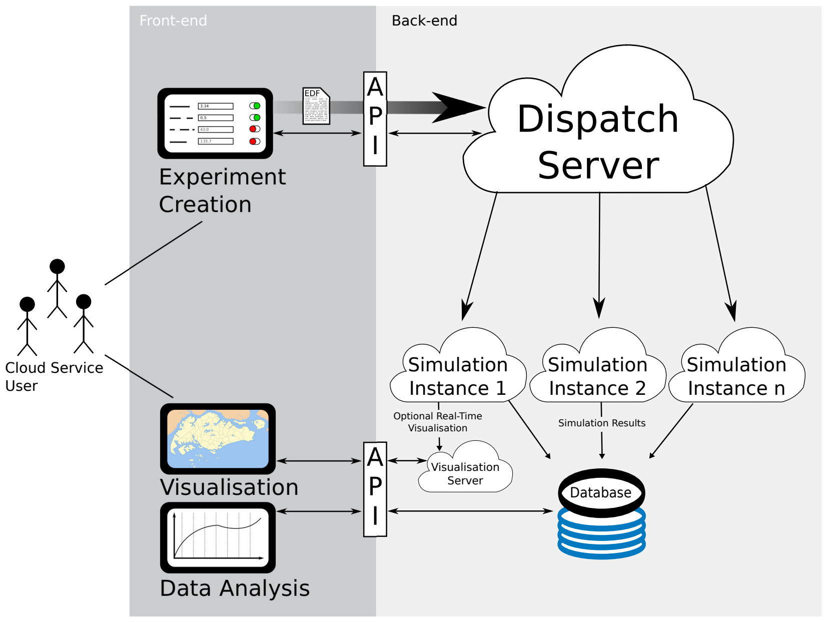


Figure 2: Overview of proposed architecture for a cloud-based simulation service

The front-end consists of the following components:

- *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) 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)

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

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.

- *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.

The back-end consists of the following components:

- A *Dispatch server application* is responsible for executing a simulation experiment according to the specification in the EDF. Generally, this is done in the following steps: the dispatcher (1) provisions a number of VMs in the cloud that will host the simulation instances, (2) initializes the VMs (i.e., copies files, setup the run-time environment, etc.), (3) starts execution of the simulation instance in the VMs, 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.

- A *Visualisation server application* enables a specific simulation instance 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 interact with the simulation whereas all other connected clients have only read input from the simulation instance.

- 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.

3.2. Workflow

In general, it 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 to design an experiment and submit an EDF to the dispatching server in the cloud. The dispatching 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 in order to interact with the simulation run. Depending on the simulation, 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 CSPs' PaaS portfolio. Once the experiment is completed, the user is notified. By using a corresponding analysis tool, the user can analyze the simulation results. The analyzed data can then be used as an input for an adjusted experimentation run. Figure 3 illustrates the workflow of the manual use-case.

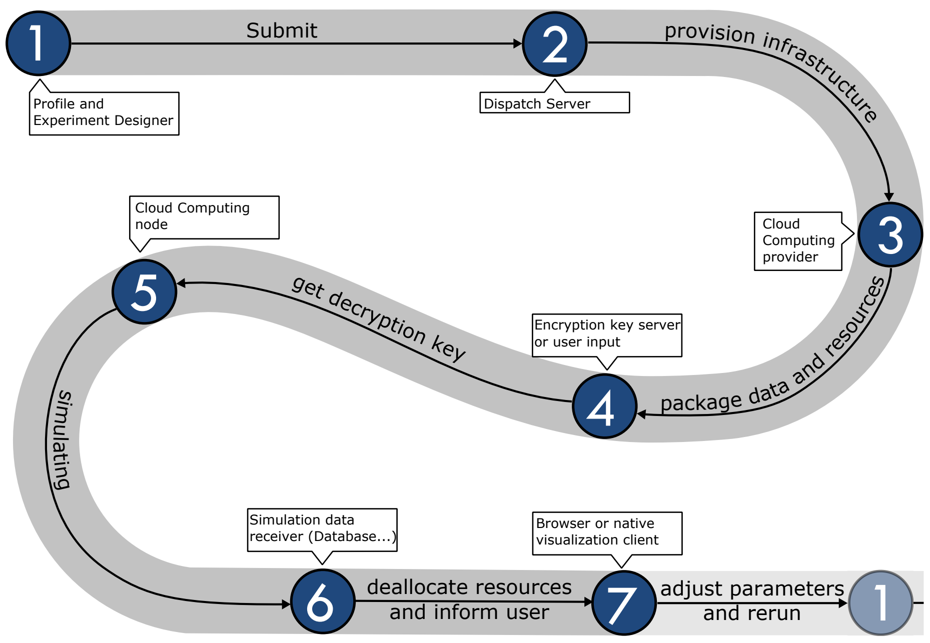

1
Submit
2
provision infrastructure
Profile and Experiment Designer
Dispatch Server
Cloud Computing node
Cloud Computing provider
3
5
get decryption key
Encryption key server or user input
package data and resources
4
simulating
Simulation data receiver (Database...)
Browser or native visualization client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

Figure 3: Overview of general workflow for 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 that works in an iterative manner (e.g., evolutionary algorithm). For each iteration, an experiment is defined according to the specifications provided by the algorithm.

3.3. Data Encryption Mechanism

For a cloud-based simulation service it is important that the confidentiality agreements with data providers for input data are obliged to by the user. On the other hand, the generated simulation data and results are of importance to the experimenter. Therefore a simulation cloud Service should use secure transfer mechanisms (e.g., TLS [21]) to communicate between cloud computing nodes, the user and the dispatch server. All input data is encrypted and a secure channel, to either a key-server or a client device (e.g., mobile handset, browser) of of the researcher, is opened to allow reliable key exchange. 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. 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 as for decrypting the input data is used to encrypt the result data before it is written to a file on the cloud node's persistent storage.

A workflow of decryption is depicted in Figure 4. When the simulation application starts after the simulation bundle has been opened, the first step of the application is to read in the EDF and find out which of the data needs to be decrypted. The key acquisition method is given in the EDF.

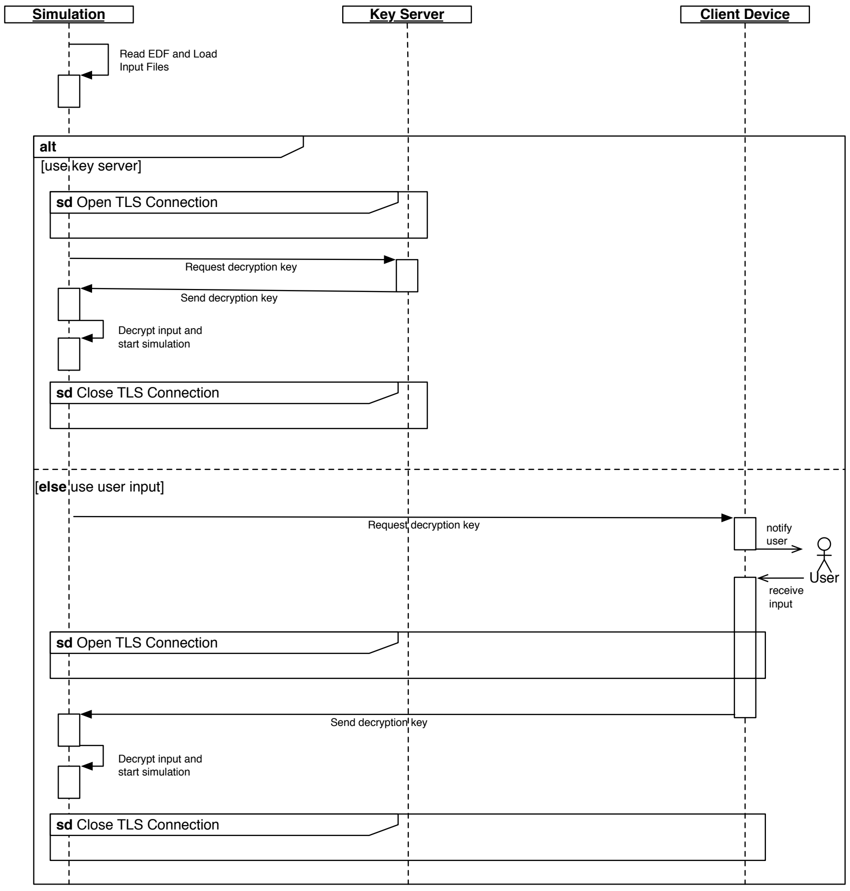


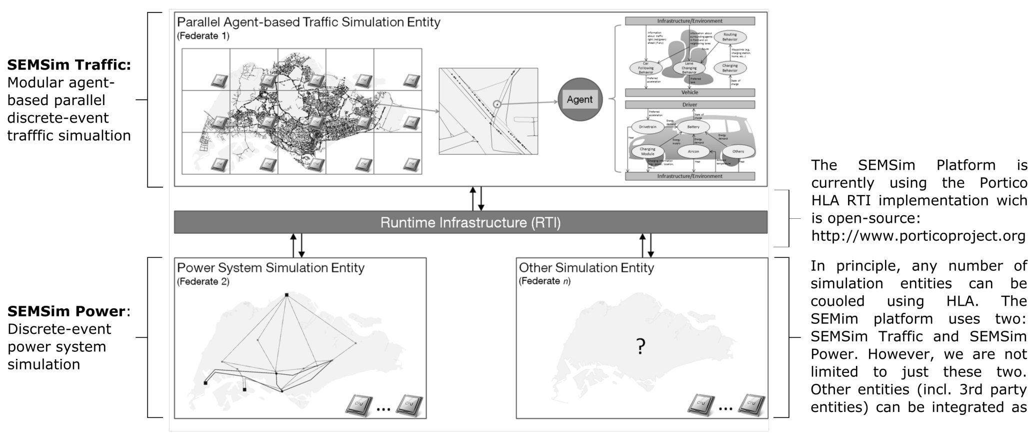
Figure 4: Data decryption workflow

It can have two options. One includes a key server which serves out keys to decrypt the input data (Figure 4 left side). For this method no user interaction is needed. The second option is to notify the experimenter of a simulation run and request him/her to enter the decryption key into a client device application or a website (Figure 4 right side). Both connections to the key server and from the users 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.

4. 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, we mean the replacement of all (or a majority of) vehicles in a city. Although there are test fleets of electric vehicles all over the world, there is no case 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, our 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 transferred to other cities.

Unlike existing research, our approach is aiming at studying large-scale electro-mobility in a holistic manner, i.e., we do not investigate individual sub-systems (e.g., power system, traffic system, vehicle system) in isolation but in a holistic simulation that allows us to investigate the complex interactions and inter-dependencies between the various sub-systems and their components. For this purpose, we are using primarily two kinds of simulations, an agent-based traffic simulation (SEMSim Traffic) and a discrete-event power system simulation (SEMSim Power [22]). These two simulations are coupled using the High Level Architecture (HLA) [23]. This allows us to investigate the impact of thousands of electric vehicles on the power grid depending on their charging behaviour. Figure 5 illustrates the SEMSim platform with its major simulation components.


SEMSim Traffic: Modular agent-based parallel discrete-event trafffic simualtion
Parallel Agent-based Traffic Simulation Entity
(Federate 1)
Agent
Infrastructure/Environment
Vehicle
Driver
Runtime Infrastructure (RTI)
SEMSim Power: Discrete-event power system simulation
Power System Simulation Entity
(Federate 2)
Other Simulation Entity
(Federate *n*)
?
...

The SEMSim Platform is currently using the Portico HLA RTI implementation wich is open-source: http://www.porticoproject.org

In principle, any number of simulation entities can be couoled using HLA. The SEMim platform uses two: SEMSim Traffic and SEMSim Power. However, we are not limited to just these two. Other entities (incl. 3rd party entities) can be integrated as

Figure 5: Overview of simulation entities that are part of the SEMSim Platform.

4.1. *Yet Another Traffic Simulation?*

Since the SEMSim Platform is designed to answer electromobility questions for mega-cities using multiple tools like SEMSim Traffic and SEMSim Power, the computational needs are large. One major difference between SEMSim Traffic and the existing traffic simulations is the level of detail at which the internal state of vehicles is simulated. In SEMSim Traffic, every agent represents a driver-vehicle unit which consist not only of driver behaviour models but also vehicle component models (e.g., air conditioning, drive train, battery). SEMSim Traffic is a simulation that has a lane-level resolution which indicates that it is a microscopic simulation. However, due to the fact that we also simulate the internal state of vehicles, we refer to it as nanoscopic simulation in order to emphasise the difference to existing microscopic simulations. Unlike existing simulations tools (e.g., MatSim [24], SUMO [25], DynaMIT [26]) which use a 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 gives us 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 behaviour). Figure 6 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 other simulation tools use, are not capable of delivering simulation results quickly enough for a mega-city sized experiment. The simulation resolutions of MATSim (meso and macroscopic), SUMO (micro and mesoscopic) and DynaMIT (macroscopic) is not in the scale of nanoscopic simulation that SEMSim Traffic is offering and necessary for researching the interactions between vehicles and other simulations. The simulations are also not designed to be run on a multi-process/-core execution environment that is crucial for a fast execution on a HPC. The target user-group for the SEMSim Platform are traffic researchers and policy-maker advisors. These 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 has to include Cloud Computing tools that make it easy to configure, setup and execute simulation experiments to study large urban systems such as the influence of electric vehicles on the power system and vice versa. SEMSim Platform is not a single simulation tool. Instead it's a set of individual simulations, middlewares and UI tools. Among these tools, we have currently defined SEMSim Traffic and SEMSim Power, two individual simulation tools that can also be coupled together by means of 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 mega-city like Singapore are high.

In Singapore there are more then 500,000 privately owned vehicles [13] 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, in the remainder of this paper, we focus on SEMSim Traffic as SaaS in the cloud (that is SEMSim Cloud Service (CS)).

4.2. Front-end Applications

SEMSim 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.

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

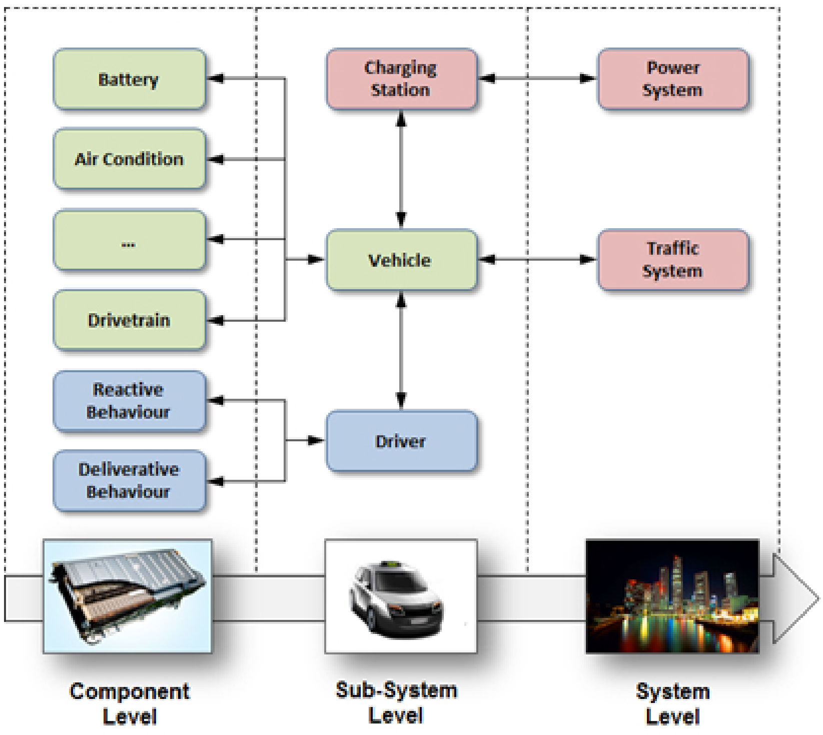


Figure 6: Overview of sub-systems and components that are modelled as part of the SEMSim Platform.

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 7 shows the PD where the driver-vehicle-unit models are parametrized in a tree-like structure. This 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 8. 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. 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.

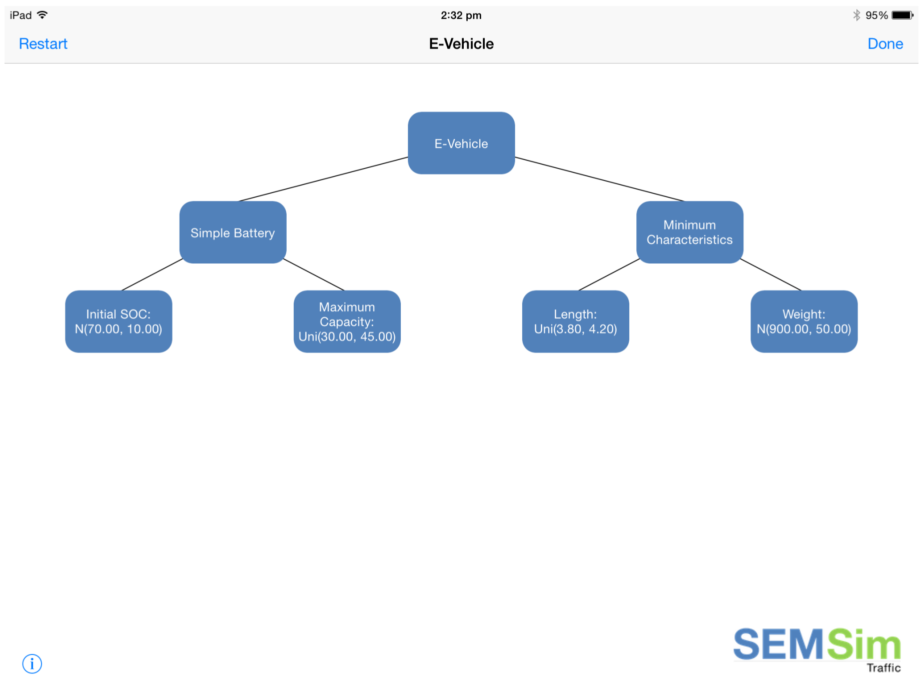
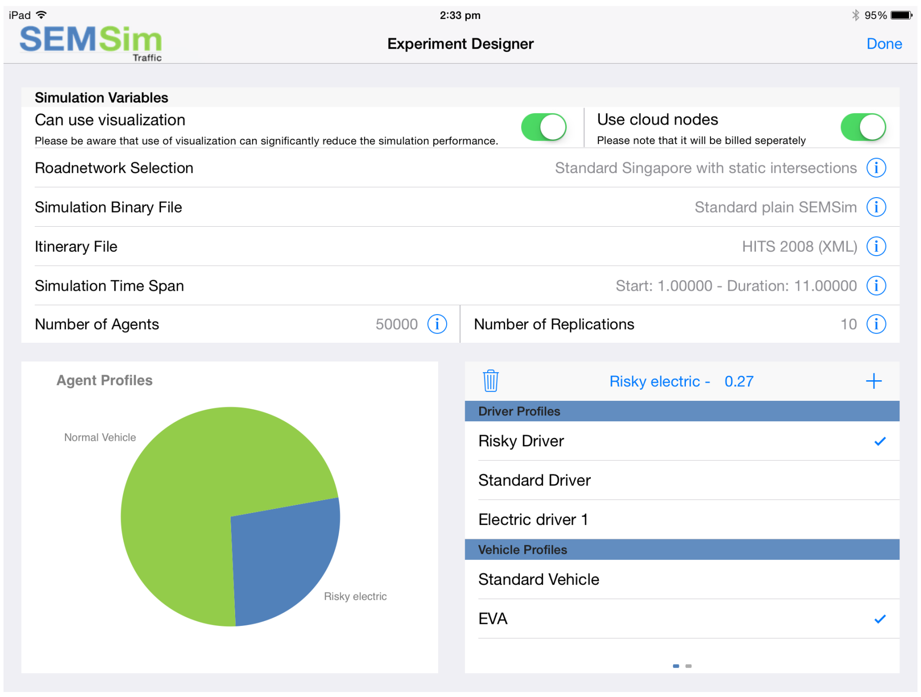


Figure 7: Model composition tree

- *Visualisation applications:* SEMSim CS provides a set of visualisation applications. In general, 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 colouring agents by a given criterion. This should give a broad overview of the simulations'



Experiment Designer

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

Risky electric - 0.27

Driver Profiles	
Risky Driver	✓
Standard Driver	
Electric driver 1	

Vehicle Profiles	
Standard Vehicle	
EVA	✓

Figure 8: SEMSim CS Experiment Designer

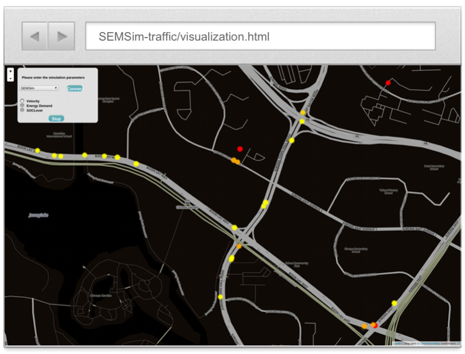
state. Due to performance reasons showing an entire map with several thousand agent representations with all state information is not available in the browser-based visualisation. Rendering several thousand agent representations at a time exceeds the capabilities of most browsers. Therefore, the zoom level is restricted to several streets at a time. Instead, advanced features such as selecting agents and receiving real-time state information about the agent or agent tracking are only provided by the touch-based visualisation applications which have been developed for the iOS platform[14]. Figure 9a and Figure 9b show screenshots of the browser-based and touch-based visualisation application, respectively.

- *Analysis applications:* SEMSim CS currently does not come with a dedicated analysis application. Instead, data analysis is done using Matlab.

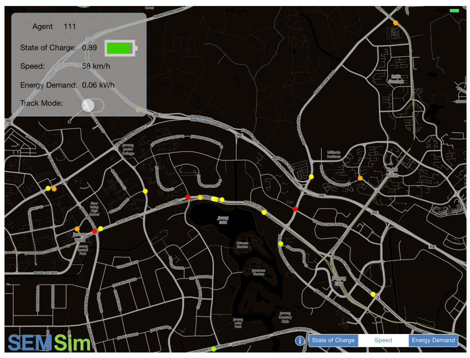
4.3. Back-end Applications

The back-end applications used to conduct experiments with the SEMSim 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.

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



(a) Screenshot of browser-based visualisation app.


Agent 111

State of Charge: 0.89

Speed: 58 km/h

Energy Demand: 0.06 kW/h

Track Mode:

(b) Screenshot of touch-based visualisation app.

Figure 9: Visualisation applications

- *SEMSim 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, fileserver). Since the data is encrypted, the dispatch server has no knowledge of the content of the bundle. As described in the general system description, the DS is also responsible for creating virtual machine (VM) instances for simulation and keeping track of their individual progress. If a simulation fails to report its progress this instance is deallocated and a new VM instance is started with the same simulation bundle.

- *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 and with a high number of agents the performance would suffer. One VS can be used for one or multiple simulation instances. 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 on one VM from a CSP. It uses a pre-configured persistent storage medium with the required libraries and simulation dependencies already installed. 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 or not the simulation experiment is still running as planned.

5. Performance Evaluation

5.1. SEMSim Traffic

SEMSim Traffic 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 [27]. 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 in [27]. 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. The (re-)routing operation is far more computationally expensive than the movement and update of the vehicle models. Amongst other, 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. With thousands of agents being updated simultaneously, a large number of concurrent threads increases the overall system performance. Nevertheless, in a road network like the one of Singapore with over 215000 nodes and 245000 edges, using a bi-directional Dijkstra algorithm with landmarks [28] and reaches [29], one routing operation costs between 0.1 ms and 100 ms.

5.2. Design of Experiment

In order to evaluate the performance of cloud-based simulations, we chose to execute 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). 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 we used the road network of Singapore with about 80,000 trips during a period of 5 hours between 5 am and 10 am. This time frame covers the morning rush hour.

The GCE offers different configurations of VMs ranging from sharing one virtual CPU core up to 16 virtual cores. The availability of certain virtual resources depends on the different geographic location of *Google Compute Engine* (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 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 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. 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 has defined a performance unit to describe the computational capabilities of their virtual machines. One *Google Compute Engine Unit* (GCEU) describes the computational capability of $\frac{4}{11}$ hardware hyper-threads (2.75 GCEUs = 1 hardware hyper-thread) [15].

Since the simulation can run with different numbers of threads, the experiment has been conducted using different configurations. Each experiment has been executed on a virtual cloud

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

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, we also measured the following run times:

- *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.
- *Startup time:* the time for transferring the payload in the form of the self-contained bundle as discussed in Section 4 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 last time measurements are not necessary when executing on a dedicated HPC node since there is no provisioning and deallocation of hardware.

In order to create a new VM from the infrastructure resources offered by the GCE, we 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*[16]. 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 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 is deleted first and the virtual disk afterwards.

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

5.3. Experimental Results

For reporting the experimental results we distinguished between the overhead introduced by provisioning, start-up, decryption and deallocation on a GCE VM node and the actual execution time of the simulation.

5.3.1. Cloud Computing Overhead

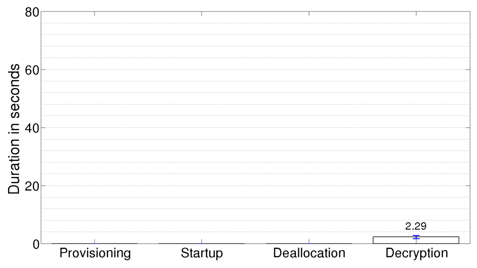
The decryption of 4 files of input data ranging from only several Kilobytes (intersection control) to hundreds of Megabytes (road network file) with a total file size of 489.02 Megabytes took only 1.26 seconds with a standard deviation of 0.049 seconds. The number of threads used had no influence on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

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

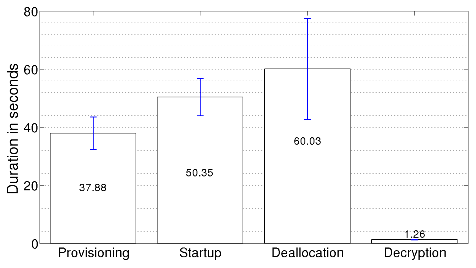
since there are interdependencies between the input files (especially the road network) this was not possible in this case.

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 10b. Since the provisioning, startup and deallocation operations are highly dependant on the GCE service availability, the execution time of the individual operation can vary as well due to external factors (e.g., Network congestion or high demand). 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.

There 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 10a).



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



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

Figure 10: Simulation overhead on different execution environments

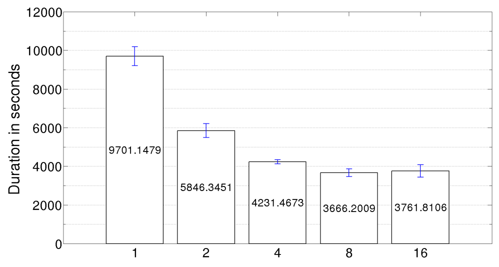
5.3.2. Simulation Runtime Comparison

The execution time on a dedicated HPC node for different number of threads is given in Figure 11a. 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%).

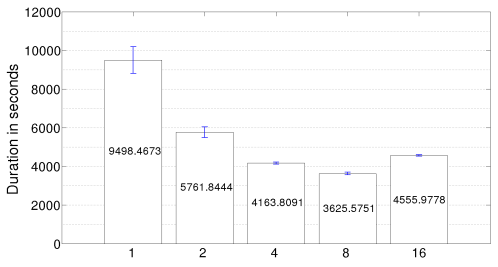
The results for executing the same experiment on a comparable virtual node are shown in Figure 11b. There, the same general characteristic of a decreasing execution time can be observed. Rather than having a slight increase in execution time the difference between using 8 or 16 threads is significant (+25.6%).

5.4. 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 gets heavier. An average of 149.52 seconds makes up between 1.57 and 4.12 percent 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 and 2.10 percent. This makes the use of virtual nodes very much comparable to the dedicated HPC nodes and even



(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 11: Simulation execution time on different execution environments

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 architectures. The dedicated HPC node has two eight core CPUs; whereas the VM from the GCE has two six core CPUs. Since we used hyper-threading 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 head room 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 leads to the biggest performance decrease. Simulating thousands of agents in a nanoscopic manner leads to cache misses, since it cannot be predicted in which processor the same agent data is needed during the next calculation. As Figure 12 shows, there are two CPUs on the same motherboard connected via the main system bus on which also other peripherals such as the main memory is also connected. An access to a memory address from a CPU core on the same CPU results in using the cached value in the CPU L1 to L3 cache, due to the locality principles. If a memory address required by a core is on a different CPU the data will not be in the cache and therefore has to be retrieved from main memory using the system bus. This operation costs many computing cycles as Hennessy et. al. [30] describe it.

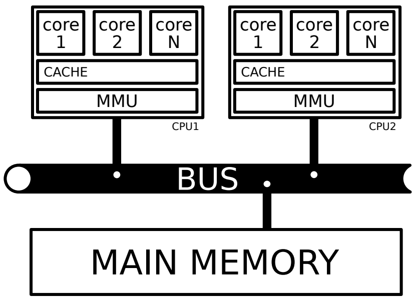


Figure 12: Shared memory on multi-core and multi-processor systems[17].

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.

6. Conclusion

In this paper we have introduced an architecture for a cloud-based simulation platform for urban systems that enables users to execute large-scale simulation experiments without the need for dedicated hardware. We show that the use of IaaS from cloud computing providers is feasible and relatively easy to do. In addition, the performance of a virtual computing node from the GCE is comparable to the one of a dedicated node of a high performance cluster.

In contrast to other approaches, the proposed service acquires computing resources on-demand and therefore has no upfront costs. Our approach also explicitly considers 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. Since the actual simulation is not run on workstation computers but rather in a local HPC or virtual compute centers the interaction methodology has been substituted, leading away from mouse-and-keyboard-based input devices on workstation computers and towards touch-based input devices as well as standard web-browsers. This enables domain experts with no or limited computer science background to use high performance computing to use large-scale simulations. Such simulations are not limited to the presented urban systems simulations, but manifold.

Going on from the presented methods, developing a framework to enable researchers to run their simulations in the cloud is a 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.

ACKNOWLEDGMENTS

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

REFERENCES

[1] T. Saitoh, T. Shimada, H. Hoshi, Modeling and simulation of the tokyo urban heat island, Atmospheric Environment 30 (20) (1996) 3431–3442.

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

[2] M. Pipattanasomporn, H. Feroze, S. Rahman, Multi-agent systems in a distributed smart grid: Design and implementation, in: Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES, 2009, pp. 1–8. doi:10.1109/PSCE.2009.4840087.

[3] D. Pelzer, D. Ciechanowicz, H. Aydt, A. 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) (2014) 345–353.

[4] J. M. Epstein, Modelling to contain pandemics, Nature 460 (7256) (2009) 687–687.

[5] H. Aydt, M. H. Lees, S. J. Turner, W. Cai, Toward simulation-based egress optimization in smart buildings using symbiotic simulation, in: Pedestrian and Evacuation Dynamics 2012, Springer, 2014, pp. 987–999.

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

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

[8] G. DAngelo, M. Marzolla, New trends in parallel and distributed simulation: From many-cores to cloud computing, Simulation Modelling Practice and Theory.

[9] F. Seibt, M. Schumann, J. Beikirch, Concept and components for a web-based simulation environment (wbse), SIMULATION SERIES 30 (1998) 189–194.

[10] S. Narayanan, Web-based modeling and simulation, in: Simulation Conference, 2000. Proceedings. Winter, Vol. 1, 2000, pp. 60–62 vol.1. doi:10.1109/WSC.2000.899698.

[11] R. Buyya, M. 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) (2002) 1175–1220.

[12] E. Cayirci, Modeling and simulation as a cloud service: A survey, in: Simulation Conference (WSC), 2013 Winter, 2013, pp. 389–400. doi:10.1109/WSC.2013.6721436.

[13] C. S. Alliance, Top threats to cloud computing v1.0 (2010).
URL `https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf`

[14] S. Guo, F. Bai, X. Hu, Simulation software as a service and service-oriented simulation experiment, in: Information Reuse and Integration (IRI), 2011 IEEE International Conference on, 2011, pp. 113–116. doi:10.1109/IRI.2011.6009531.

[15] S. Taylor, A. Anagnostou, T. Kiss, G. Terstyanszky, P. Kacsuk, N. Fantini, A tutorial on cloud computing for agent-based modeling amp; simulation with repast, in: Simulation Conference (WSC), 2014 Winter, 2014, pp. 192–206. doi:10.1109/WSC.2014.7019888.

[16] A. Huqqani, X. Li, P. Beran, E. Schikuta, N2cloud: Cloud based neural network simulation application, in: Neural Networks (IJCNN), The 2010 International Joint Conference on, 2010, pp. 1–5.

[17] C. N. Hoefer, G. Karagiannis, Taxonomy of cloud computing services, in: GLOBECOM Workshops (GC Wkshps), 2010 IEEE, 2010, pp. 1345–1350. doi:10.1109/GLOCOMW.2010.5700157.

[18] S. Chaisiri, B.-S. Lee, D. Niyato, Optimization of resource provisioning cost in cloud computing, Services Computing, IEEE Transactions on 5 (2) (2012) 164–177. doi:10.1109/TSC.2011.7.

[19] E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good, The cost of doing science on the cloud: The montage example, in: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, SC '08, IEEE Press, Piscataway, NJ, USA, 2008, pp. 50:1–50:12.

[20] S. Sur, M. J. Koop, D. 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, ACM, New York, NY, USA, 2006. doi:10.1145/1188455.1188565.

[21] T. Dierks, E. Rescorla, The transport layer security (TLS) protocol version 1.2 (2008).
URL `http://www.ietf.org/rfc/rfc5246.txt`

[22] D. Ciechanowicz, H. Aydt, A. Knoll, Semsim power as an application of uses, in: Proceedings of the IASTED International Symposium on Power and Energy 2013, 2013.

[23] J. S. Dahmann, K. L. Morse, High level architecture for simulation: An update, in: Proceedings of the Second International Workshop on Distributed Interactive Simulation and Real-Time Applications, DIS-RT '98, IEEE Computer Society, Washington, DC, USA, 1998, pp. 32–.

[24] M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, K. Axhausen, Matsim-t: Architecture and simulation times, Multi-agent systems for traffic and transportation engineering (2009) 57–78.

[25] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, Recent development and applications of SUMO - Simulation of Urban MObility, International Journal On Advances in Systems and Measurements 5 (3&4) (2012) 128–138.
URL `http://elib.dlr.de/80483/`

[26] M. E. Ben-Akiva, Smart–future urban mobility, JOURNEYS (2010) 30.

[27] H. Aydt, Y. Xu, M. Lees, A. 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.

[28] A. V. Goldberg, C. Harrelson, Computing the shortest path: A search meets graph theory, in: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics,

2005, pp. 156–165.

[29] R. 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), SIAM, 2004, pp. 100–111.

[30] J. Hennessy, D. Patterson, Computer Architecture: A Quantitative Approach, The Morgan Kaufmann Series in Computer Architecture and Design, Elsevier Science, 2006.
URL `http://books.google.com.sg/books?id=57UIPoLt3tkC`