DEMO: Simulation-as-a-Service for ITS Applications

Jérôme Härri*, Moritz Killat*, Tessa Tielert*, Jens Mittag* and Hannes Hartenstein*†
Karlsruhe Institute of Technology (KIT)
*Institute of Telematics
76131 Karlsruhe, Germany
Karlsruhe Institute of Technology (KIT)
†Steinbuch Center for Computing (SCC)
76131 Karlsruhe, Germany
Email: {haerri, killat, tielert, mittag, hartenstein}@kit.edu

I. Introduction

The development of Intelligent Transportation Systems (ITS) applications requires the evaluation of its impact on vehicular traffic. For logistics, flexibility, and financial reasons, evaluations using simulations are usually preferred over field tests as a first step toward deployment. Simulations offer a large modularity in the evaluated ITS applications and full flexibility in the chosen scenarios. Considering the large scale typically involved in ITS scenarios, simulations also require high computational capabilities including maintenance and technical expertise potentially resulting in significant financial investments on local simulation infrastructures. Moreover, such infrastructures are usually not efficiently used, in a sense that they are overused when simulations are conducted and underused when not.

Cloud computing is a novel paradigm that offers a differentiation of the usage of a resource or a service and its physical location on remote High Capacity Computing (HCC) platforms. Typical benefits from this approach are for instance an improved resource sharing between remote users, user-friendly web-based access to and configuration of a service, efficient maintenance and easier extensibility. Due to its high demand in processing power, the large-scale simulation of ITS applications is likely to benefit from cloud computing.

In this work, we therefore propose a simulation-as-a-service approach to evaluate ITS applications. We segment a simulation process in two building blocks: a web-interface/server front-end used by users to configure the remote simulations of their ITS application, and a back-end consisting of a controller and a HCC platform to conduct the remote simulations (see Fig. 1). The controller is in charge of configuring the HCC platform as well as distributing and scheduling the simulations on it. HCC platforms usually provide users with only a restricted control on the simulation environment and a limited set of available libraries. To mitigate these drawbacks, we make use of the virtualization support of recent HCC platforms and employ Kernel-based Virtual Machines (KVMs). Each virtual machine (VM) is a fully configured simulation environment that is transparent to the computing platform. Our objective is to illustrate how simulations requiring a large number of executions could be configured on a web-interface and remotely run in parallel on HCC platforms, for example on the High Performance Computing (HPC) platform of the Steinbuch Center for Computing (SCC) [1] at the Karlsruhe Institute of Technology (KIT). Our approach may be beneficial to research entities that could federate their simulation capabilities, but also companies providing professional simulators that could offer not only an ownership of a license but also a right-of-use of their simulators on their simulation platform.

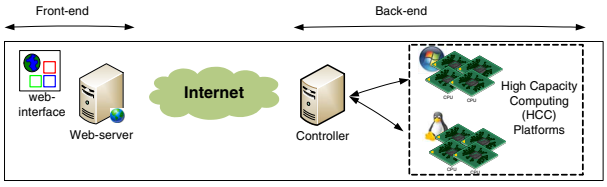


Fig. 1. Simulation-as-a-Service - Front-end and Back-end Concepts.

II. Front-end

The objective of the front-end is to provide a user-friendly web-based configuration of the simulation study, as well as an interface between the user and the simulation infrastructure. A web-interface gathers the inputs from users regardless of their location or used device, while the web-server saves the scenarios and other data transmitted by the users, connects to the controller, transmits the simulation scenarios to the controller and retrieves and stores the simulation results. The current version of the front-end provides the following services:

- upload of simulation scenarios
- specification of the number of simulation executions
- provision of a link to download the aggregated results

The front-end technology is based on the Eclipse framework and on the Eclipse Rich Ajax Platform (RAP) [2]. It has been chosen as it allows the development of AJAX-enabled applications based on the Eclipse Rich Client Platform (RCP) and its comprehensive flexible component-based development model and convenient user-interfaces.

As depicted in Fig. 2, a major benefit of the Eclipse framework and the tight coupling between Eclipse RAP and RCP is

* J. Härri and J. Mittag acknowledge the support of the Ministry of Science, Research and the Arts of Baden-Württemberg (Az: Zu 33-827.377/19,20), the Klaus Tschira Stiftung, the INIT GmbH and the PTV AG for the research group on Traffic Telematics. T. Tielert acknowledges the support of Deutsche Forschungsgemeinschaft within the Research Training Group GRK 1194 Selforganizing Sensor-Actuator-Networks.

the OSGi standard for the development of modular component-based applications, and the Eclipse Modeling Framework (EMF) for its (de-)serialization mechanism and efficient data models composition and validation. Another major benefit, also illustrated in Fig. 2, is RAP's single sourcing capability, which lets the development of rich clients as well as web clients from a single source base. For example, the web front-end implemented in this demonstrator could be converted to an Eclipse RCP application with minor code modifications.

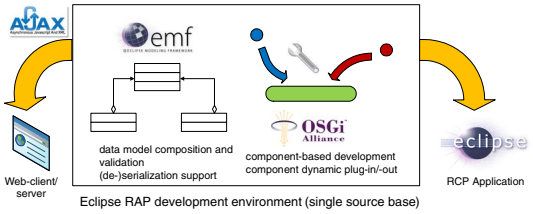


Fig. 2. Rich Ajax Platform (RAP) development highlights. OSGi offers modular component-based developments and EMF modular data model compositions and validations. RAP provides AJAX-based web-applications but it may also generate RCP applications with minor code modifications.

III. Back-end

The back-end is the core of the simulation-as-a-service approach to be demonstrated. It consists of a controller connected to a HCC platform. As illustrated in Fig. 1, the back-end may be physically separated from the front-end, but the controller may also be physically separated from the HCC platform.

Once the controller receives a simulation request from the front-end, it performs the following tasks:

- It configures the scenarios for the HCC platform.
- It distributes the jobs to the required and available cores.
- It gathers and returns the results to the front-end.

The HCC where simulations are remotely run is the HPC of the SCC [1], which offers us a computing capability of approx. 800 GFLOPS on average. Yet, our concept does not restrict the connection to a single HCC platform.

The back-end also employs Kernel-based Virtual Machines (KVM) [3], a virtualization solution for Linux that makes an extensive use of hardware virtualization support in recent x86 and x86-64 processors. This approach offers the possibility to run fully configurable Windows or Linux images adapted to the simulation requirements on the HCC platforms, which allows us to benefit from their high computation capacities regardless of the required simulation environment. Fig. 3 illustrates the conceptual schema of the back-end.

Although the back-end concept uses KVM, it is not restricted to it as virtualizations based on Xen [4], such as Amazon EC2 [5], could also be controlled by the controller.

IV. Demonstration

Our demonstration will consist of three phases: First, we will show the configuration of a simulation scenario for an ITS application called *Green Light Optimal Speed Advisory (GLOSA)*[1] in an urban environment. Second, we will connect

[1]GLOSA is an application making use of signal transition timings transmitted by traffic lights to approaching vehicles in order to provide the optimal approaching speed avoiding a stop at traffic lights.

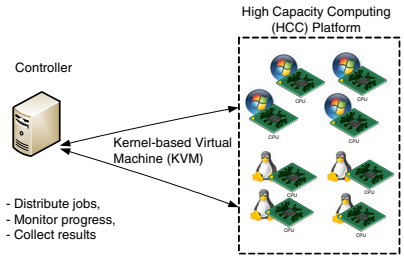
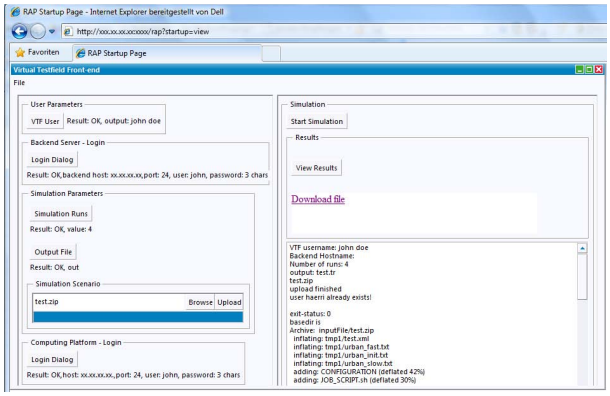
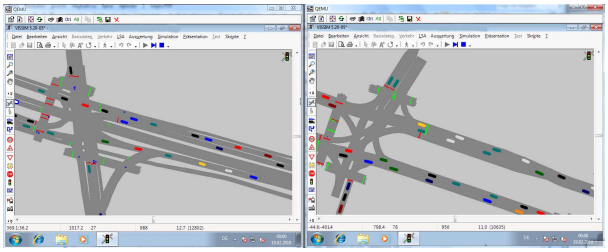


Fig. 3. Back-end concept, composed of a controller and a computing platform. In the demonstrator, Kernel-based Virtual Machines (KVM) run Windows images.

to the web-interface, request the number of simulation executions, and upload the configured scenario to the controller. Finally, the requested multiple instances of the scenario will be run in parallel on multiple cores on the remote computing platform. As GLOSA is simulated using the Windows-based VISSIM traffic simulator [6], we will run Windows as KVM guest platforms and visualize a subset of the running KVMs with a remote desktop connection. Fig. 4 depicts what attendees will be able to see.



(a) Web front-end configuration



(b) KVMs remotely running in the back-end

Fig. 4. Illustration of the demonstrator - a GLOSA simulation campaign is configured on the front-end and remotely dispatched on multiple KVMs running in parallel in the back-end.

Acknowledgments

We would also like to thank Dr. Michael Ortgiese and Dr. Thomas Benz of PTV AG for their help and support in this project.

References

[1] The Steinbuch Center for Computing (SCC), http://www.scc.kit.edu
[2] The Eclipse Rich Ajax Platform (RAP), http://www.eclipse.org/rap/
[3] Kernel-based Virtual Machine, http://www.linux-kvm.org/
[4] The Xen hypervisor, http://xen.org/
[5] Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2/
[6] VISSIM Traffic Simulator, http://www.ptvag.com/