Using FMI in a cloud-based Web Application for System Simulation

Stefan Bittner Olaf Oelsner Thomas Neidhold

ITI GmbH, Germany, info@iti.de

Abstract

This paper presents a generic approach to combine cloud computing and system simulation. It shows the benefits of using FMI to deploy self-executing simulation units on multiple machines. Besides managing the calculation itself, we also present a web interface for uploading, managing and analyzing simulation models. To benefit from available hardware resources in the cloud, an engine is integrated which allows the definition of multiple simulations with different parameter sets in a single step.

Keywords: FMI, Cloud, Web, System Simulation, Parameter Study

1 Introduction

Simulation calculations can be very demanding for hardware resources, such as computing power and disc space. If these resources are not needed permanently, there may be a conflict of insufficient resources to finish a project within a certain time on the one hand, and paying too much for resources on the other hand. Cloud computing could be a solution here: cloud providers offer a wide range of resources which can be allocated on demand and used from anywhere around the globe.

However, reserving raw resources, such as a plain virtual machine, and setting the simulation environment up that is needed for a calculation can be a time consuming task. Let alone additional licenses for the simulation tools which would be necessary in order to run them on the remote machine. Although easy to setup, remote resources in the cloud are not a trivial task to deal with, especially not for simulation engineers. Offering simulation services over the web, which make use of available cloud infrastructure, seems to be a much better approach. The advantages and possibilities of such a service have been presented in (Tiller, 2014). We are extending this idea to a more generic approach: a web service based on FMI itself instead of a specific simulation model where the users are able to upload and share their own models and results.

2 System Perspective

We have implemented different approaches in two projects we are involved in, Cloud4e (Cloud4e, 2012), funded by the Federal Ministry of Economics and Technology, and CloudFlow (CloudFlow, 2013), funded under the 7th Framework Programme of the European Commission. While in Cloud4e, a REST service has been developed which communicates via the Open Cloud Computing Interface (OCCI), in CloudFlow a SOAP service is embedded in another service architecture as part of workflows. Instead of explaining these implementations in detail, which lies outside the scope of this paper, we want to present a brief overview of the underlying service architecture and how it works. For a more detailed description, see (Limmer et al., 2014) and (CloudFlow, 2013).

2.1 Architecture

The web service is consists of different services, each of which is necessary to perform a specific task. A storage service stores models and results and protects them against unauthorized access. The calculations are run by a calculation service which has access to the storage in order to obtain the model, read its configuration and write back result data. Everything is monitored by the simulation web service. It manages models and data, authorization and cloud resources. This service is the heart of the architecture organizing everything that is necessary for this service to work. Different application types, such as web applications as well as desktop applications, can communicate with the simulation web service to request data, upload models or submit simulation tasks to be calculated by the underlying infrastructure.

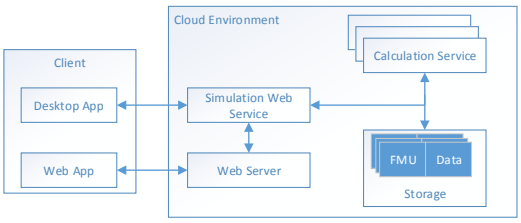

Figure 1. Service architecture

Every component in the architecture may reside on a different logical computer, but does not have to. For example, the web server can be hosted on the same machine as the simulation web service, while calculations can be run on separate machines simultaneously or only on one machine, one at a time. The web server shown in figure 1 provides a web interface, which serves as the front end for the user, and is discussed in detail later on.

The future architecture might include other services as well, for example external authentication and storage services.

3 User Perspective

The user does not have to care about the underlying system, with one exception: since allocated resources cost money, it is desirable to know and control how much resources are in use or will be in use. This is only one example that shows that a stable communication between system and user has to be established. This is realized through a web application which is connected to the service and provides a web platform based on HTML5 and JavaScript. As such, it is accessible through any modern browser from any device connected to the internet. It provides the user view and control of the simulation service: its content and its state.

Models, Tasks, Simulations When an FMU is uploaded, it is embedded in a model where it serves as the description of the simulation model. This description can then be used to create multiple instances of the model called "'tasks"'. A task contains a description for each parameter representing a configuration of the model. They belong to one model at any time.

For the system simulation service, a task is an executable unit: it references an existing FMU and contains a description of its parameters. It is used as input for the calculation service. The results are assigned to the task as simulations which in turn contain the corresponding parameter and result set. Why is there a parameter configuration in the task and a parameter set in the simulation? They may differ, and a task may contain multiple simulations as described later on.

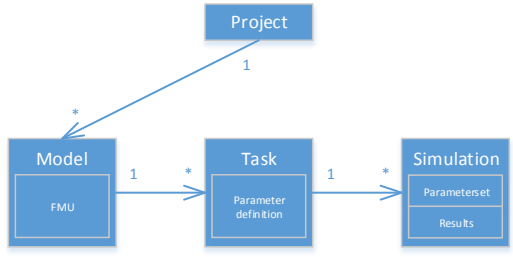

Figure 2. The FMU is embedded in a simple structure to support access management and multiple simulation results.

Projects and Users Models can be assigned to projects in order to organize and share them. While this mechanism can be used for collaborative work and project management, it is also possible to give other users restricted access to models, for example, to the simulation results without rights to modify the model. Possible restrictions are shown in table 1.

Project	**Model**
General	
Create Delete	Upload Open Delete
Contents	
Add models Remove models	Modify tasks Execute tasks
Right management	
Add/Remove users	Modify access rights

Table 1. Access rights for projects and models which can be granted to users and user groups

We think that implementing a reliable project and user management adds a significant value to simulations in the cloud. Sharing models and presenting results are true benefits besides the use of external resources for the calculations themselves. However, user and access management is still under development.

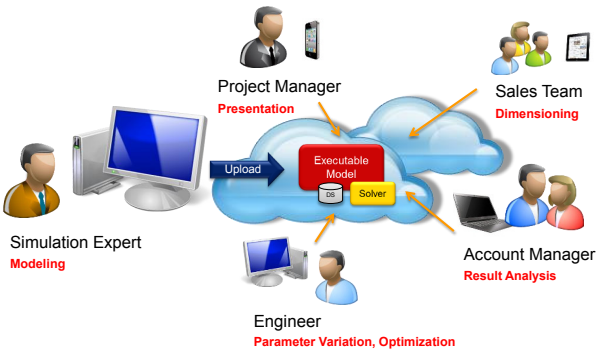

Figure 3. Scenarios for sharing simulation models in the cloud.

Configuring multiple Simulations The parameter configuration of a task may contain range expressions for one or more parameters. A range expression defines more than one value for this parameter. There are currently two options available: comma separated list (`<value>,<value>`) and sequence constructor (`<startValue>:<stepSize>:<endValue>`). When multiple parameter values are defined, the simulation service creates a parameter set for each combination, which

DOI 10.3384/ecp15118845

results in a vast number of simulations. Each of them is calculated and, after that, part of the result of the corresponding task.

Figure 4. Defining multiple parameters values with the result of 200 different parameter sets.

3.1 Web Interface

The web interface, a .Net MVC web application, gives access to the contents of the simulation service through any modern web browser. After logging in, the user has access to the contents and functions he is authorized to use. The SignalR library allows for a bidirectional communication between client and server, providing a user experience similar to desktop applications. As the main interface between user and simulation service, the web application allows the user to manage models and tasks, request calculations and results.

Model Management The model list shows projects and models and allows the user to create new projects, move models from one project to another and upload an FMU from the local disk to create a new model. A preview is provided for each model showing the embedded image of the model and information, such as author, date of creation and number of tasks. Each model can be deleted and opened here.

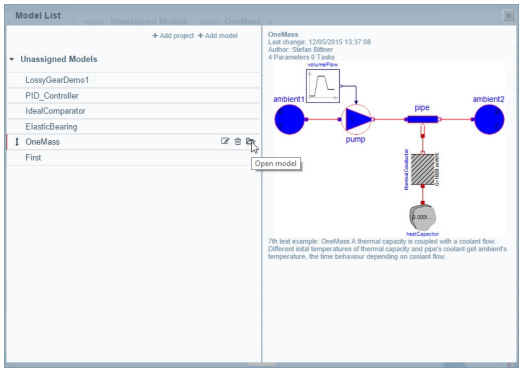

Figure 5. Web interface: List of models and model preview.

Task Management Once opened, the model contents are displayed in the web application. The taskbar allows the user to create and delete tasks as well as start, stop and reset the calculation state of each task. Each task contains a parameter configuration which can be modified as long as the task has not been executed. If available, the model structure is displayed in the center. Model elements can be selected to filter the list of parameters.

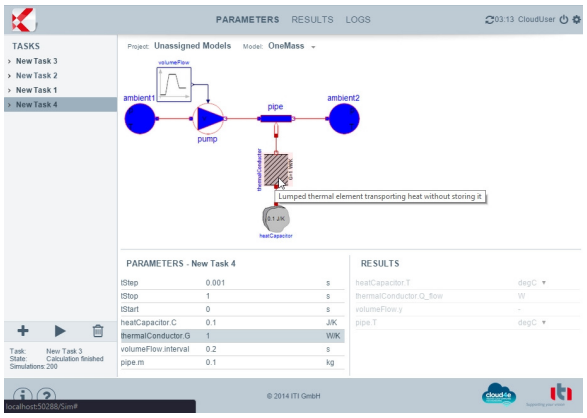

Figure 6. Web interface: Task list and parameters, interactive image of the model in the center.

Result Analysis The result panel contains a graph where the transient results of one or more output parameters are displayed. This may be a set of curves per output parameter when multiple tasks are selected for result analysis, while each of them may contain multiple simulations. Such a set of results also contains a set of parameter configurations, one for each simulation. The range of varying parameters can be modified using a slider control which changes the number of curves visible in the result graph. A CSV file download is available for further analysis.

Figure 7. Web interface: result view.

4 The Role of FMI

In this scenario, two sides had to be addressed: creating a scalable service structure capable of running in a cloud environment, and creating a web application to access simulation parameters and results. FMI works either way: while the XML model description can be used to create a generic interface to access model contents, the integrated solver can be embedded in web services and deployed on any machine. Furthermore, there is no restriction to the contents and size of the model as long

as it meets the FMI specification. No additional dependencies or libraries are needed: everything is packed into the FMU which, of course, can be downloaded and integrated in other toolchains as well.

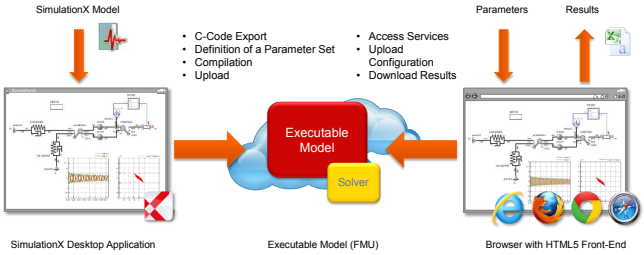

Figure 8. FMI in the cloud: scenario using SimulationX

FMI in the Cloud To run an FMU on a machine in the cloud, a web service had to be implemented which is able to consume the C interface, run as a master for the functional mockup unit and communicate over the web. Since the simulation solver is embedded in the FMU, this service is only an adapter between the FMU and other web services.

Model Description and User Interface The model description itself can be used to create a generic user interface to visualize and gain access to parameters and result variables exposed by the FMU. Different units can be used to display/define parameter values if these units are available in the model description. Optional contents can be added to the FMU's resources folder to be used in the web interface. For example, SimulationX adds an image and interaction map of the model.

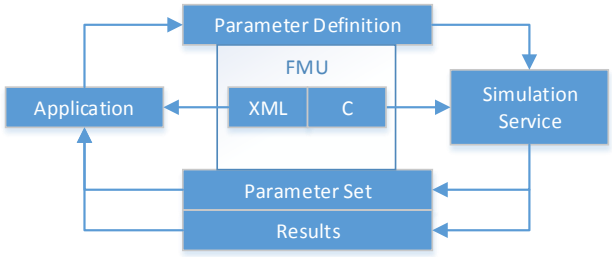

Figure 9. FMU between application and simulation service

5 Conclusion

This paper presents a web application using a cloud service architecture to manage, store and perform simulation calculations and their results. FMI 1.0 for Co-Simulation is the key technology for the simulation part of the service. It offers an easy-to-use scenario to deploy calculations on remote machines and contains a description to generate a generic web interface. This allows us to support a wide range of simulations, especially because creating FMUs using the FMI specification is supported by a growing community of simulation tool vendors.

There is currently still some work to be done, especially regarding user access management. For the future, this service can be extended, for example, to combine FMUs for online co-simulations, or integrating other services, such as PLM services, where FMUs are integrated together with other information to represent a product lifecycle.

References

Cloud4e. Trusted Cloud Computing for Engineering, 2012. URL `http://www.cloud4e.de/`. Funded by the Federal Ministry of Economics and Technology (BMWi).

CloudFlow. Computational Cloud Services and Workflows for Agile Engineering, 2013. URL `http://www.eu-cloudflow.eu/`. Funded under the 7th Framework Programme of the European Commission.

S. Limmer, A. Ditter, M. Srba, S. Thomas, A. Schneider, S. Rülke, O. Oelsner, A. Uhlig, S. Schmitz, D. Fey, and C. Boehme. The Project Cloud4E – Cloud Solutions for Engineers. *Lecture Notes in Computer Science*, 2014.

T. Neidhold, S. Bittner, and O. Oelsner. SimulationX Goes Online – A Web Platform for Cloud-Based Simulation. In *ITI Symposium*. ITI GmbH, 2014.

T. Neidhold, O.Oelsner, S. Bittner, A. Ditter, and D. Frey. Rechnen in der Wolke. *Digital Engineering*, 2015.

M. Tiller. Vehicle Thermal Management – A Case Study in Web-Based Engineering Analysis. *Proceedings of the 10th International ModelicaConference*, 2014.