APPROACHES TO RUN SIMULATIONS OF BUSINESS PROCESSES IN A GRID COMPUTING NETWORK

Christian Müller
Department of Management and Business Computing
Technical University of Applied Sciences Wildau
Hochschulring 1
D-15745 Wildau, Germany
christian.mueller@th-wildau.de

KEYWORDS
Event driven simulation, business processes, grid computing

ABSTRACT

Two approaches to run simulations of business processes in a grid computing network are compared. Based on special properties of simulation models, the approach that use a grid framework is more stable than the web service approach.

INTRODUCTION

Business processes of modern companies are characterized by a huge complexity which is caused for example by quickly changing markets, short product life cycles or dynamic interactions between particular subsystems of a company. Business process management is intended to implement efficient and customer-oriented processes whereby the simulation of business processes can be used to evaluate the quality of processes and to identify areas of improvements. To analyze these models we must run many of experiments, which needs a lot of time on a single computer. In this paper we will discuss two approaches to parallelize the running of the experiments in a grid network. For modeling of business processes as an event driven simulation model we use the Epc-Simulator as simulation system (Müller 2012), (Müller 2014).

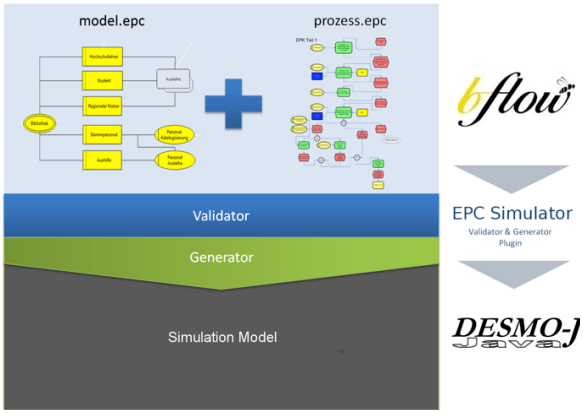

Figure 1: Epc-Simulator concept image

The Epc-Simulator is a plugin of the EPC modeling toolbox bflow* (Kern et al. 2010), (Bflow 2014). As shown in Figure 1, the first step to create a simulation model is the specification of a model document and several process documents in bflow*. The model document contains information about the model infrastructure (simulation time, available resources, inter-arrival times of entities, etc.). The process documents contain the process descriptions in EPC notation. Based on these documents, the Epc-Simulator can generate a simulation model. This is a Java Application that uses the DESMO-J Framework, whereby DESMO-J provides the basic functionality of a simulation. (Page and Kreutzer 2005), (DesmoJ 2014)

Each business process described with Epc-Simulator contains a model and some process documents. The idea of modeling with Epc-Simulator will be introduced with the following supermarket example.

A supermarket is entered by customers. The inter-arrival time between two customers that enter the shop is given by a probability distribution. The process that each customer runs is described in a process document named customer (Figure 3). In the supermarket work one or more entities named cashier. This is described in the model document in Figure 2.

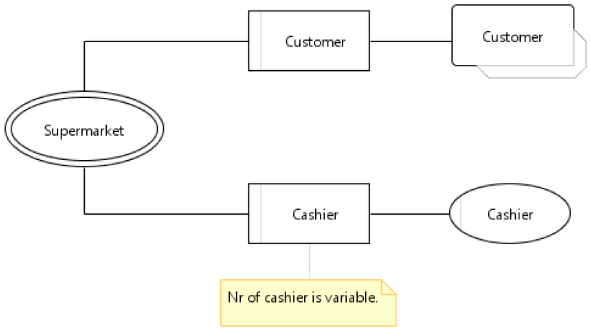

Figure 2: Model document

Every model is parametrized by some standard parameters, like a seed value for the random generator and some optional model-dependent parameters, like the number of available cashiers. On the other hand the simulator computes some performance indices from each simulation run, e.g. the utilization rate of the cashiers.

Proceedings 28th European Conference on Modelling and Simulation ©ECMS Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani (Editors)
ISBN: 978-0-9564944-8-1 / ISBN: 978-0-9564944-9-8 (CD)

The Epc-Simulator produces an executable jar archive with the generated simulation model and a parameter file. Before the simulation runs, this file contains the input data and after the run the file is extended by the output data.

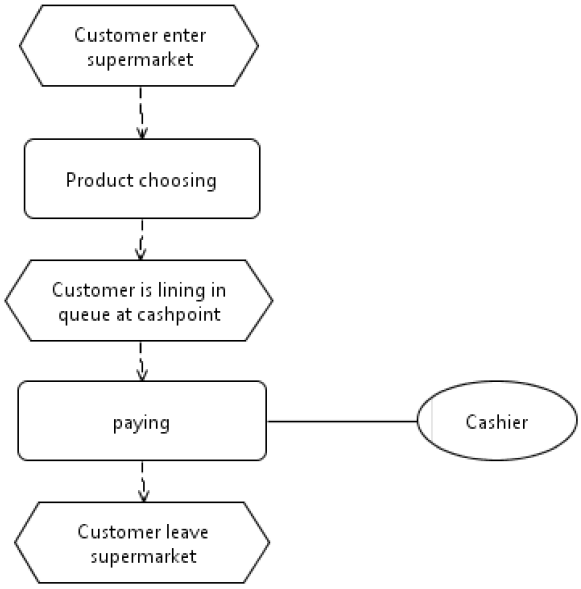

Figure 3: Process document

Before running a row of experiments, the parameter file for each experiment must be customized with an individual parameter file and specialty with its own seed value.

STRUCTURES OF DISTRIBUTED COMPUTING NETWORKS

Distributed computing networks can be organized as Peer-to-Peer, as Cluster or Grid networks. In a Peer-to-Peer network (Figure 4), all nodes in the network are equal and each node can cooperate with an other. This can produce a huge organization overhead. For this see (Bengel 2004), (Bengel et. al. 2008), (Dunkel 2008, (Schill 2012).

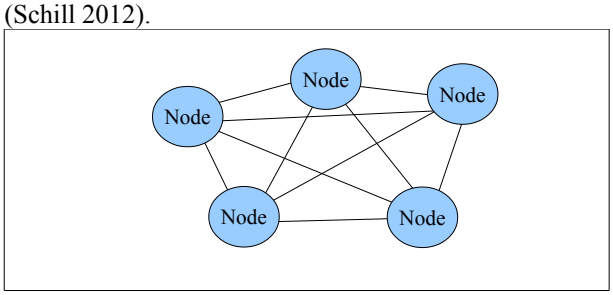

Figure 4: Peer to Peer Network

In a Cluster (Figure 5) we have a client and he can distribute his tasks to nodes that are associated with him exclusively. A cluster works inside of an organization and has no access to external nodes. On the other hand in a grid network a client works with nodes that can be located everywhere. For this, the grid network uses an open protocol stack.

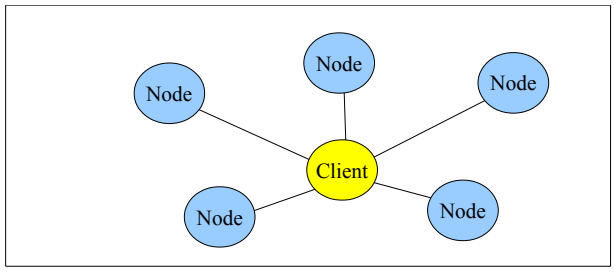

Figure 5: Cluster Network

In this paper we compare two approaches to distribute simulation tasks in a cluster. For this we use techniques that are working on a high abstraction level. The first is a web service approach and the second uses a grid framework. Both approaches are tested only in a cluster, but the principles there are working in a grid too. Approaches that are working on a low abstraction level, like RMI, CORBA or RPC are not examined.

SIMULATION TASKS AS DISTRIBUTED SOFTWARE OBJECTS

A simulation job contains a common jar archive with the simulation model and a parameter file for each simulation task. The client distribute the tasks of a job on a set of nodes. The following executing properties of a task depend on its parameter file and are in general not predictable:

- The *elapsed time of the simulation* depends on the length of the simulation period and the event frequency of the model. The frequency increases with the model complexity.
- The *memory (RAM) requirement of the simulation* model depends on the individual model properties. When e.g. in the example above the inter-arival times of the customer are shorter than the service times of the cashier, then the queue in front of the sale station will increase. Consequently, there are more entities in the simulation and the simulation model needs more memory. In complex models it is not easy to extrapolate the memory requirement.
- The simulation model can produce a *data file to animate the simulation*. The length of this file depends on the length of the simulation period and the event frequency of the model. This file can be big. Hence the size of response data to send from node to client can be big.

In these points the properties of simulation models differ from common distributed software objects, like Mandelbrot sets. There are fast, short and their response is only a number. On the other hand, there are more distributed computations than simulation experiments in our case.

USING WEB SERVICES TO ORGANIZE THE GRID

In this approach, the simulation models are enveloped in a wrapper that includes web service functionality. We have developed a wrapper for the REST and one for the SOAP protocol. On each node runs a servlet engine and the models must be deployed on the servlet engine of this node. The client sends simulation tasks, together with their parameter files to the nodes. A manager of the node collects all simulation tasks and organize the sequence of there execution. The client asks after a while about the status of his tasks. When a task is completed, it can download the results.

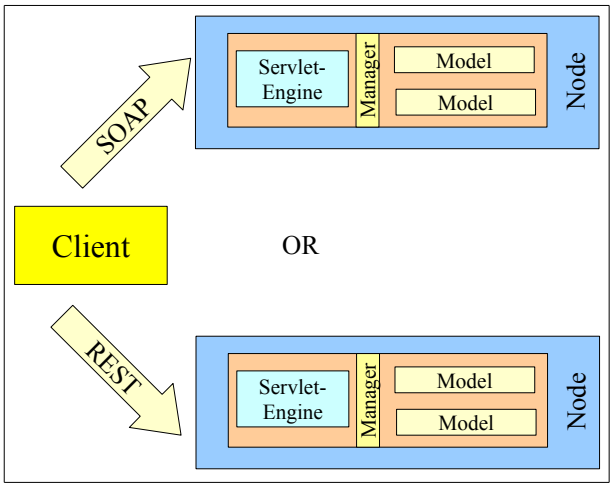

Figure 6: Using web services

The servlet engine, here we used a Glassfish, and their deployed applications, here the simulation models, are running in the same java virtual machine. Sometimes, when the simulation model is badly parametrized, the model runs in an out of memory exception. In consequence of that, the servlet engine runs after the next client access also in a exception and hence the node is no longer accessible from the client (Zimmermann et. al. 2012).

Its not unlikely that a simulation model breaks with a out of memory exception. We will illustrate this on the example above. When the model runs over a long time and more customers enter than leave the shop, then the queue in front of the cashier will go longer and longer. Since every customer entity needs a bit of memory, the model requires after a while more memory than accessible.

USING THE GRID FRAMEWORK-JPPF TO ORGANIZE THE GRID

As an alternative for the web service approach to distribute simulation tasks in a grid, we use JPPF as a grid framework. JPPF stands here as an example of a light weight grid framework(Jensen 2013), (JPPF 2014). JPPF is a open source framwork and runs with the Apache 2 licence (Apache Licence 2004). An overview and comparison of such frameworks is given in (Azadzadeh 2006)

A client application sends a job with a row of simulation tasks to a JPPF server. The server distributes the tasks to nodes. When the tasks on a node are completed the server collects the simulation results from the nodes and delivers them to the client application. On the server and on all nodes runs a small sized JPPF management software. When in a simulation task an exception occurs, the simulation is finished and is marked as not runnable. In our experiments the break of a simulation model has no side effect on the stability of the node management software.

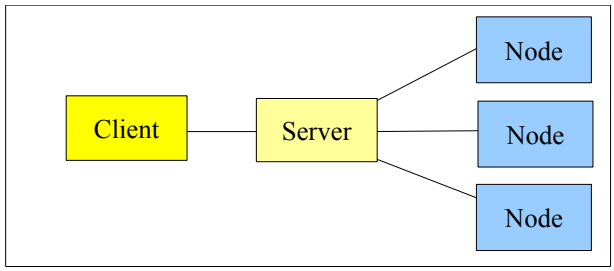

Figure 7: Using a grid framework

In experiments with 1000 simulation tasks and 5 nodes, all results arrived the client nearly at the same time. Hence, the result handling in the client can be a bottleneck. To avoid this, we can use some JPPF notification mechanism to inform the client when a single task is finished. With this information we can distribute the activities in the client to handle the results on the whole processing time of the distributed simulation.

CONCLUSION

In this paper we compared two approaches to distribute simulation tasks in a grid network. One approach uses a web service to organize the communication between client and computing nodes and the other uses the grid framework JPPF.

For the web service approach, on each computing node runs a servlet engine. This is a heavy-sized application. When the run of a simulation model breaks with an out of memory exception, the servlet engine may break by the same reason.

On the other hand, the distribution approach with the JPPF grid framework runs stably, also when the simulation model breaks with an out of memory exception. For the management of a node, there runs a light weight management software. The less memory requirements of the node management system may be a reason of their stability.

This investigation shows that the approach with the grid framework is more stable than the web service approach. In further investigations we will have a look

at other grid frameworks. Furthermore, more numerical experiments are required.

REFERENCES

Asadzadeh, P; Buyya, R; Kei, C; Nayar, D; Venugopal, S: Global Grids and Software Toolkits – A Study of Four Grid Middelware Technologies 2006 in High-Performance Computing: Paradigm and Infrastructure eds: Yang, L; Guo, M; John Wiley & Sons

Apache License: Apache License, Version 2.0, 2004 http://www.apache.org/licenses/LICENSE-2.0.html

Bengel, G: Grundkurs Verteilte Systeme 2004 Friedrich Vieweg & Sohn Verlag

Bengel, G; Baun, C; Kunze, M; Stucky, K-U: Masterkurs Parallele und Verteilte Systeme 2008 Vieweg +Teubener | GWV Fachverlage GmbH

Bflow: Bflow* Toolbax – Geschäftsprozessmodellierung als Open Source 2014 http://www.bflow.org/

DesmoJ: A Framework for Discrete-Event Modelling and Simulation 2014 http://desmoj.sourceforge.net/home.html

Dunkel, J; Eberhard, A; Fischer, S; Kleiner, C; Koschel, A: Systemarchitekturen für Verteilte Anwendungen 2008 Carl Hanser Verlag

Heiko Kern, Stefan Kühne, Ralf Laue, Markus Nüttgens, Frank J Rump, Arian Storch: bflow* Toolbox - an Open-Source Business Process Modelling Tool 2010, Proc. of BPM Demonstration Track 2010, Business Process Management Conference 2010 (BPM'10), Hoboken, USA

Jensen, A: Ein Ansatz zur Parallelisierung von Geschäftsprozessen mit dem Programm EPC Simulator an Beispiel des Grid-Frameworks JPPF 2013, TH Wildau Masterthesis

JPPF: JPPF Homepage 2014 http://www.jppf.org/

Müller, Chr. : Generation of EPC Based SimulationModels 2012 In: Proceedings 26th European Conference on Modelling and Simulation 2012, Koblenz, http://dx.doi.org/10.7148/2012-0301-0305

Müller, Chr. EpcSimulator Project Homepage 2014 http://www.tfh-wildau.de/cmueller/EpcSimulator/

Page, B; Kreutzer; W : The Java Simulation Handbook, Shaker 2005

Schill, A; Springer, T: Verteilte Systeme – Grundlagen und Basistechnologien 2012 Springer Verlag

Zimmermann, S; Sobottka, M; Szott, E: Fehlerbehebung in komplexen IT-Systemen am Beispiel von Simulation on Demand Webservices auf einer GlassFish-Serverumgebung 2012 Belegarbeit TH Wildau

AUTHORS BIOGRAPHIES

CHRISTIAN MÜLLER has studied mathematics at Free University Berlin. He obtained his PhD in 1989 about network flows with side constraints. From 1990 until 1992 he worked for Schering AG and from 1992 until 1994 for Berlin Public Transport (BVG) in the area of timetable and service schedule optimization. In 1994 he got his professorship for IT Services at Technical University of Applied Sciences Wildau, Germany. His research topics are conception of information systems plus mathematical optimization and simulation of business processes.

His email address is: christian.mueller@th-wildau.de and his web page is http://www.th-wildau.de/cmueller/ .