Executing Multiple Simulations in the MERPSYS Environment

Paweł Rościszewski

Abstract

The chapter investigates the steps necessary to perform a simulation instance in the MERPSYS environment and discusses potential limitations in case when vast numbers of simulations are required. An extended architecture is proposed which includes a JMS-based simulation queue and multiple distributed simulators, overcoming the potential bottlenecks. The chapter introduces also methods for preparing suites of multiple simulations for parallel execution. Additionally, potential use cases of the framework are discussed, including a concept of parallel application optimizer.

9.1 Introduction

Following the first book in this series [1], the described models of large-scale parallel systems and concepts for simulating application executions on those systems [2] have been implemented into a software environment called MERPSYS.

The environment comes with a database of hardware elements available to use within a graphical tool in order to model large-scale computing systems. The tool allows also to prepare application models using a Java-based meta-language. The systems and applications prepared in the tool, along with sets of specific parameters form instances of simulations, which can be performed using a simulator program.

Some capabilities of the system have already been presented in the example of an application for parallel similarity measure computations [3]. The developed system model reflected an existing cluster of 16 workstations. The results showed that after calibration of the models based on a subset of real execution results, the MERPSYS system can predict the execution times for modified application parameters.

Since that work, we have dealt with moving the simulation to the large scale, developing models of various application classes, as well as considering factors different than execution time, such as energy consumption and reliability. The experiments require running multiple simulations, often differing only in values of single variables.

In Section 9.2, we describe the aspects of preparing simulations in the MERPSYS environment, which are important from the perspective of running multiple
executing multiple simulations at the same time. To improve the system capabilities in this matter, we developed a simulation framework based on a JMS (Java Message Service) queue, described in Section 9.3. We summarize the Chapter and propose future work in Section 9.4.

9.2 Executing a single simulation instance

9.2.1 System architecture

There are three main components of the MERPSYS environment required to perform a simulation:

- Editor – a JavaFX application with graphical interface for editing system and application models, as well as executing simulations
- Server side – a Java EE application deployed on an application server, containing:
  - Model – a set of entity classes for representing and storing hardware models, systems and applications
  - Engine – this is the central component providing communication between other components
  - Web – a web application for editing hardware models, browsing simulation instances and results
- Simulator – a Java application for actual execution of the simulations

The relationships and communication flow between these components are depicted by Figure 9.1.

9.2.2 Editor

One of the components necessary to perform a simulation is the Editor. The current version is always available for downloading from the MERPSYS project website. This section describes the elements of the Editor crucial for running simulations, with particular focus on the system model, application model and simulation execution.

System model

The first thing that needs to be done in order to perform a simulation is preparing a system model and an application model in the Editor application. The system model represents available hardware as a graph, with computing and network devices as graph nodes connected by edges representing network connections. The system modeling view of the Editor application is shown in Figure 9.2. The model under construction can be seen on the canvas in the middle. Adding new components can be done by dragging from the list of available components in the
Figure 9.1: Basic architecture of the MERPSYS simulation environment

Components panel in the upper right corner. After selecting a particular component on the canvas, the user can modify its details using the Properties panel on the right, where the component can be assigned a name, multiplicity and suitable hardware models.

Figure 9.2: Screenshot of the Editor in the system modeling view
The *Labels* panel in the lower right corner is particularly important in the context of executing the simulations. This is the place where the system model should match the application model. The user can define here so called *labels* connected with the currently selected hardware component. The *labels* specify the names and multiplicities of application processes that can be executed on this component.

It should be noted that in the example presented in Figure 9.2 a specific method of assigning *labels* has been used. Instead of creating a label called "slave" with a multiplicity of 408, the user decided to use the "slave[1-407]" notation. This formula tells the simulator to create 408 *labels* with names "slave1", "slave2", ..., "slave407", each with multiplicity of 1.

This method allows to perform the simulation with fine-grained granularity, where each process is simulated by a separate simulator thread, rather than one thread acting as many processes. Such granularity does not mean that this type of simulation is better. In some cases it might be desirable in order to increase the accuracy of the simulation, but the simulation time would also significantly rise. An example of such approach has been described in Chapter 1.

Either way, at this point it is important to remember that the *labels* only determine the possible sets of processes, but the exact names and multiplicities of processes to be run depend on another setting in the simulation view that will be described later.

**Application model**

Another element necessary for the simulation is the application model. The applications are modeled using a Java-based meta-language with a possibility to call the simulation API methods such as computation and various types of communication. A screenshot of the application modeling view is presented in Figure 9.3.

It is crucial that the application model matches the system model in terms of aforementioned *labels*. For each label of the simulated parallel application process, there should be a corresponding fragment of code in the application model. The following code fragment:

```java
else if(tag.equals("master")) { /* Code for the "master" label */ }
```

indicates code for the process with label "master".

The *Methods* and *Editor* panels on the right are very helpful for working on the application model. In the first one, the user can choose from existing code templates (e.g. computation, p2p communication, etc.) defined for all simulator API methods (with description of parameters) and well known application schemes. The latter panel allows to define a path of a program in the operating system and run it as an external editor. The application model will be synchronized using a temporary file. This way, the user can take advantage of a fully featured editor or IDE (Integrated Development Environment) instead of the basic panel in the Editor application.

In the *Variables* panel, the user can specify variables specific for the application model, such as data size, data package size for a master-slave application, etc. In
9.2. EXECUTING A SINGLE SIMULATION INSTANCE

When the system and application models are ready, the user can proceed to the simulation execution view, presented in Figure 9.4.

The Labels panel in the center is the supplement to the aforementioned labels taken into account by both, the system and application model. Unlike in the similar panel during the creation of the system model, this time the meaning of the setting is which and how many processes will be executed in the simulation.

The dependency between these two settings is that in order to simulate execution of a parallel application that consists of processes with the required labels, there has to be a place for them in the system model. On the other hand, there can be more such places than necessary. In such cases, the Scheduler component within the Simulator decides where the required processes should be executed.

This means that for a given system model and application model simulation results depend on:

- application parameters;
- required process labels;
- scheduler mechanism and its parameters.
CHAPTER 9. EXECUTING MULTIPLE SIMULATIONS...

The application parameters can be specified in the Variables panel, which is the same panel as in the application modelling view.

The results and messages from the last executed simulation can be accessed in the Results and Messages panels on the right. An important setting is the Simulator panel over the "start simulation" button. The user should use it to specify the name of the simulator, which should perform the actual simulations, before clicking "start simulation". This name should correspond to the value set during the starting of a simulator. The differentiation between simulator names prevents from using other users simulator, at the same time allowing a group of users to use one common simulator.

9.2.3 The application server side

Another components required to perform simulations are deployed on the application server located in Figure 9.1. As one instance of the application server is maintained within the MERPSYS project, from the users point of view it is only necessary to set (or not change the default) proper settings in the other components configuration.

The Engine component provides an EJB (Enterprise JavaBeans) [7] programming interface for the Editor, which allows sending simulation instances for execution. The simulation instances are represented by an object-oriented model (Model component) and stored in a relational database, where results of previous simulations can be also found using a proper ID number.

The Web component provides access to the aforementioned entities in the database, such as system and application models, requested simulation executions.
and their results. A list of simulators connected to the Engine is also available. Finally, the current versions of Editor and Simulator subsystems are hosted within the application server.

9.2.4 Simulator

The last but not least subsystem needed to perform the simulations is the Simulator itself. The current version of the Simulator program can be downloaded from the MERPSYS project website. It is shipped as a ZIP archive, which contains a Java JAR file, lib directory with necessary libraries and running scripts. An exemplary running script run.sh in bash is as follows:

```bash
#!/bin/bash
/opt/jdk1.8.0_25/bin/java -jar Simulator-1.2.3.jar \
-host astaroth.eti.pg.gda.pl -port 8038 \
-login login -password password \
-truststore merpsys.jks -authconf merpsys.conf \
-name common
```

Besides the parameters locating the Engine component, user credentials and authentication settings, the running script specifies the simulator name that should correspond to the aforementioned setting in the Editor.

9.3 Executing multiple simulation instances

The steps described in Section 9.2 allow performing only one simulation at the same time. In this situation, waiting for results may take considerable time if there are many users, each of which sends simulation instances from her copy of the Editor application. This long time is caused by the architecture presented in Figure 9.1, which assumes only one instance of the Simulator program. In Section 9.3.1 we describe an expanded architecture, which allows performing multiple distributed simulation at the same time.

The limitations of the solution presented in Section 9.2 lay also in the Editor simulation execution view (Figure 9.4). The user interface allows to send only one simulation at a time, wait for the results and then send next. In Section 9.3.2 we describe how to prepare a simulation suite: a set of similar simulation instances, differing by one or more parameters.

9.3.1 Framework for executing multiple simulations

From the users point of view, the task of the Engine component is to send the simulation to an available Simulator component and collect results of the simulation as soon as possible. In order to increase the throughput of the system in a flexible way, we introduced an architecture based on a JMS simulation queue, presented in Figure 9.5.
In this architecture, all of the simulation instances ready for execution are pushed into the simulation queue presented in Figure 9.5. There can be many instances of the Simulator component, which fetch simulation instances from the queue. It is also possible for a Simulator component to perform more than one simulation at the same time. The Web component provides a website with a list of simulators available in the system. An exemplary screenshot of this list is presented in Figure 9.6. The user can set a number of parallel simulation threads per each
9.3. EXECUTING MULTIPLE SIMULATION INSTANCES

available Simulator instance in the "Max threads" column. Such an architecture
allows to easily connect many simulators using available computational resources
in order to compute results of many simulations faster.

Additionally, we plan to introduce an Optimizer component. As presented
in Figure 9.5, this component will also have possibility to send simulations to
the queue. In case of the Optimizer, we plan to prepare the simulation instances
programmatically, by changing chosen parameters of an existing simulation instance
in a dynamic way. This can be done periodically, along with fetching the results of
previously sent simulations in order to find the optimal combination of execution
parameters. Finding optimal values of many parameters with regard to one or
more criteria (for example execution time and energy consumption) may be a
non-trivial task and we consider it as future work.

9.3.2 Preparing simulation suites

The framework for multiple simulations presented in Section 9.3.1 allows running
potentially vast numbers of simulations at the same time. Still, the user inter-
face described in Section 9.2.2 does not allow sending more than one simulation
instance for execution. The typical use case of running multiple simulations con-
cerns situations when the user wants to compare executions of the same application
on the same system, but for different values of certain parameters.

Because of this, we introduced the notion of a simulation suite. A suite can be
created using the web interface in the Web component, upon one already existing
simulation, usually prepared using the Editor. Figure 9.7 presents a fragment of
a screenshot of the simulation suite creation view.

The tool contains tables where for each application variable or label multiplicity
the user can specify a range of values for which the simulations should be executed.
As for now, the ranges are linear: the user provides the first and the last value,
as well as step. In the given example, for the variable "DIM_SIZE" the first
and last value are 100000 and 1000000, and step is 100000. This means, that
the simulations will be performed for all of the following values: 100000, 200000, 300000, 400000, 500000, 600000, 700000, 800000, 900000, and 1000000.

This way, the user can easily put huge numbers of simulation instances into the simulation queue. It should be noted that the ranges can be specified for many variables, which means creating a multidimensional mesh of simulations. Specifying different types of ranges (e.g., geometric) may be introduced in the future.

### 9.4 Summary and future work

In this Chapter, firstly we described the steps required to perform a single simulation within the basic architecture of the MERPSYS system. This description revealed that this basic architecture was limited to performing only one simulation at a time.

We provided predictions and examples why this might be a significant bottleneck of the system and proposed an extended architecture based on a JMS simulation queue. The new architecture allows to connect many simulators to the system in a loosely-coupled manner and thus, perform multiple simulations in parallel, using a distributed computing infrastructure.

The possibilities introduced by the new architecture caused the need for new methods of sending simulation instances for execution. We proposed a method of creating simulation suites, which allow preparing multiple simulation instances in a usable way. The simulation instances in a suite are de facto copies of one
9.4. SUMMARY AND FUTURE WORK

previously prepared instance, differing in values of certain variables, specific for the system and application models.

The mechanisms described in this Chapter form a framework for multiple simulations which could be utilized in various ways. It may serve MERPSYS users to easily experiment with certain pre-defined sets of simulation instance parameters, but also to develop an optimizer, which would choose the parameters automatically, based on criteria such as execution time or energy consumption. Thus, performing the experiments and developing various versions of optimizers are considered as future work.

Bibliography


