ComsolGrid – A framework for performing large-scale parameter studies using COMSOL Multiphysics and the Berkeley Open Infrastructure for Network Computing (BOINC)

Christian Benjamin Ries[1] and Christian Schröder[1]
[1]Department of Engineering Sciences and Mathematics, University of Applied Sciences Bielefeld, Wilhelm-Bertelsmann-Straße 10, 33602 Bielefeld, Germany

Abstract: BOINC *(Berkeley Open Infrastructure for Network Computing)* is an open-source framework for solving large-scale computational problems by means of public resource computing (PRC). In contrast to massive parallel computing, PRC applications are distributed onto a large number of heterogeneous client computers connected by the Internet where each computer is assigned an individual task that can be solved independently without the need of communication upon the clients. Nowadays even small companies hold remarkable computer resources which are not accessible as a whole but distributed over the numerous computers which belong to the standard working environment in today's companies. Over the day, these computers are rarely operating at full capacity and hence valuable computational power is just wasted.
Here, we present a new approach for performing large-scale parameter studies using COMSOL Multiphysics based on the BOINC technology which can be used to utilize idle computer resources that are connected within a companywide intranet. Based on our approach we provide a tool chain called ComsolGrid that allows the COMSOL Multiphysics user to define parameter study by means of a high-level description. ComsolGrid then automatically performs the complete parameter study.

Keywords: BOINC, Grid Computing, large-scale Parametric Studies

1. Introduction

With the help of BOINC [1, 6] it is possible to create and run self-made scientific applications heterogeneous computer networks. However, one can also run legacy applications like Matlab [8] by using so-called wrapper functions within BOINC for performing parameter studies on distributed computer networks [2]. Based on this idea, we have created a framework called ComsolGrid to perform distributed COMSOL Multiphysics *(hereafter referred to as COMSOL)* simulations. The key idea is to run COMSOL simulations with predefined parameter values on different computers. The COMSOL simulation model and parameter values are packed in so-called *workunits* which a server computer sends to the client computers. Each client performs a single simulation according to the parameter set given and returns the result to the server computer. The BOINC project server keeps tracks of the

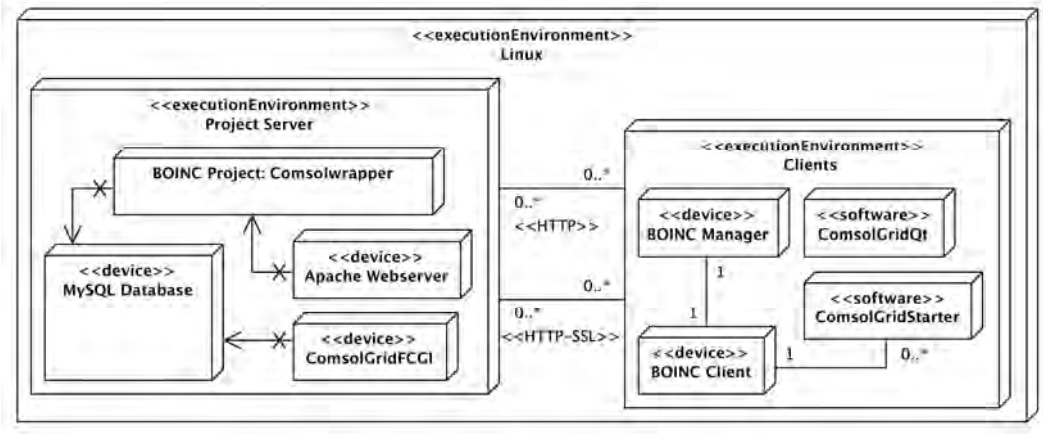


Figure 1: Software architecture of *ComsolGrid*.

workunits and checks whether each result is valid or not (within the so-called *validation* process). Validated results are stored in specified locations on the server computer during the so-called *assimilation process*.

A COMSOL simulation can be started using the within a *batch mode* [7], i.e. a command-line based program call which bypasses the use of the common graphical user interface (GUI). The command-line parameters in the *batch mode* allow the user to control the simulation process. For example, it is possible to define the target processor type, i.e. 32-/64-bit architecture. Furthermore, one can define the parameter names and parameter values which should be used by the COMSOL simulation. This enables one to perform a parameter study based on one specific simulation model by varying the parameter values for each simulation in a distributed computer environment. Within ComsolGrid the complete configuration and communication management is defined by data structures based on the Extensible Markup Language (XML).

2. ComsolGrid – Components, Architecture, and Definitions

Fig. 1 shows the software architecture of ComsolGrid and gives an overview of the developed software components. There is one component which describes the complete infrastructure of ComsolGrid. Currently, only Linux installations are supported. On the left hand side the project server is shown. This server is responsible for the correct handling of user requests, the creation process of new workunits and the validation and assimilation of completed simulations. On the right hand side the client environment is shown. This environment includes the user interfaces to enable the maintenance process; in order to create new parameter studies and to supervise the server system conditions clients can perform COMSOL parameter simulations as well.

As shown in Fig. 1, there are eight applications defined: **(1)** the BOINC project itself, which is named *"Comsolwrapper"*, **(2)** a MySQL database to store information about users and host computers of the network, **(3)** an Apache web server installation which provides the interface to the client computers for the exchange of information (distributing workunits, gathering results), **(4)** *ComsolGridFCGI* which is described in more detail in Section 2.1, **(5)** the BOINC Manager providing the GUI on each BOINC client computer, **(6)** the BOINC client which performs the actual COMSOL simulation, **(7)** *ComsolGridStarter* is an application which enables the BOINC Manager to start, stop, and pause COMSOL simulations, (see section 2.2), and **(8)** the tool *ComsolGridQt*, which provides the GUI for managing the complete parameter study (see section 2.3).

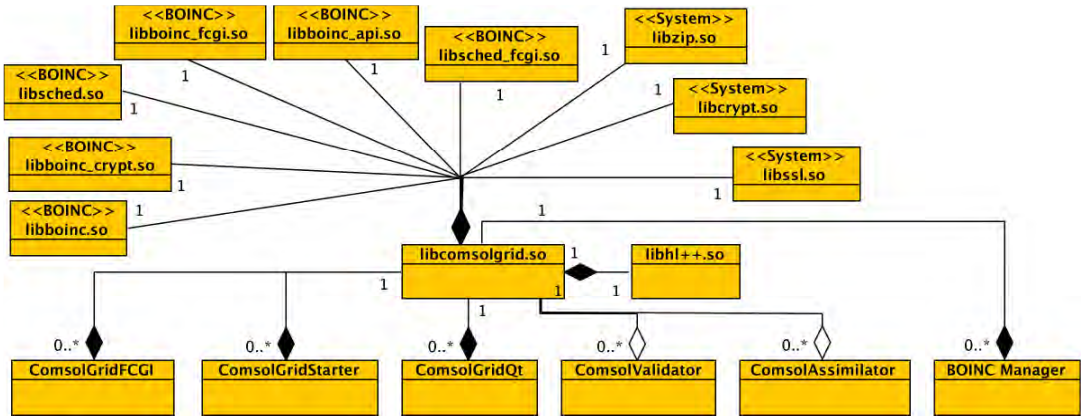


Figure 2: Architecture of software libraries used in the *ComsolGrid framework*. Most of them are implemented by the BOINC framework (indicated by the stereotype: *BOINC*), four libraries are external software libraries (*libhl++.so*, indicated by the stereotype: *System*). The library *libcomsolgrid.so* in a composition of the core functions which are used by all ComsolGrid applications listed at the bottom of the figure. For ComsolGrid the BOINC Manager has been modified in order to handle COMSOL specific parameters.

The server and client processes can have two network communication channels. One channel is encrypted using the Secure Socket Layer (SSL) and the other one is based on plain-text. The BOINC clients do not need an encrypted channel to receive new workunits. For this purpose, BOINC has implemented a hash key verification system. The BOINC project creates a hash key from a combination of the user's personal password and e-mail address and is only valid for one machine. Only this hash key is exchanged between the BOINC clients and the project server and does not enable permissions to administrate the BOINC server. The encrypted channel is used by the user to create new parameter studies.

Fig. 2 shows the software architecture of the *ComsolGrid framework*. At the top the BOINC dependencies to internal and external software libraries are shown. These libraries are used by the software library *libcomsolgrid.so*. Furthermore, *libcomsolgrid.so* is used by the ComsolGrid applications which are implemented for the *ComsolGrid framework* and are described in the following sections 2.1 to 2.4.

2.1 ComsolGridFCGI – Server interface for Parameter Studies

The *ComsolGridFCGI* is an interface for the

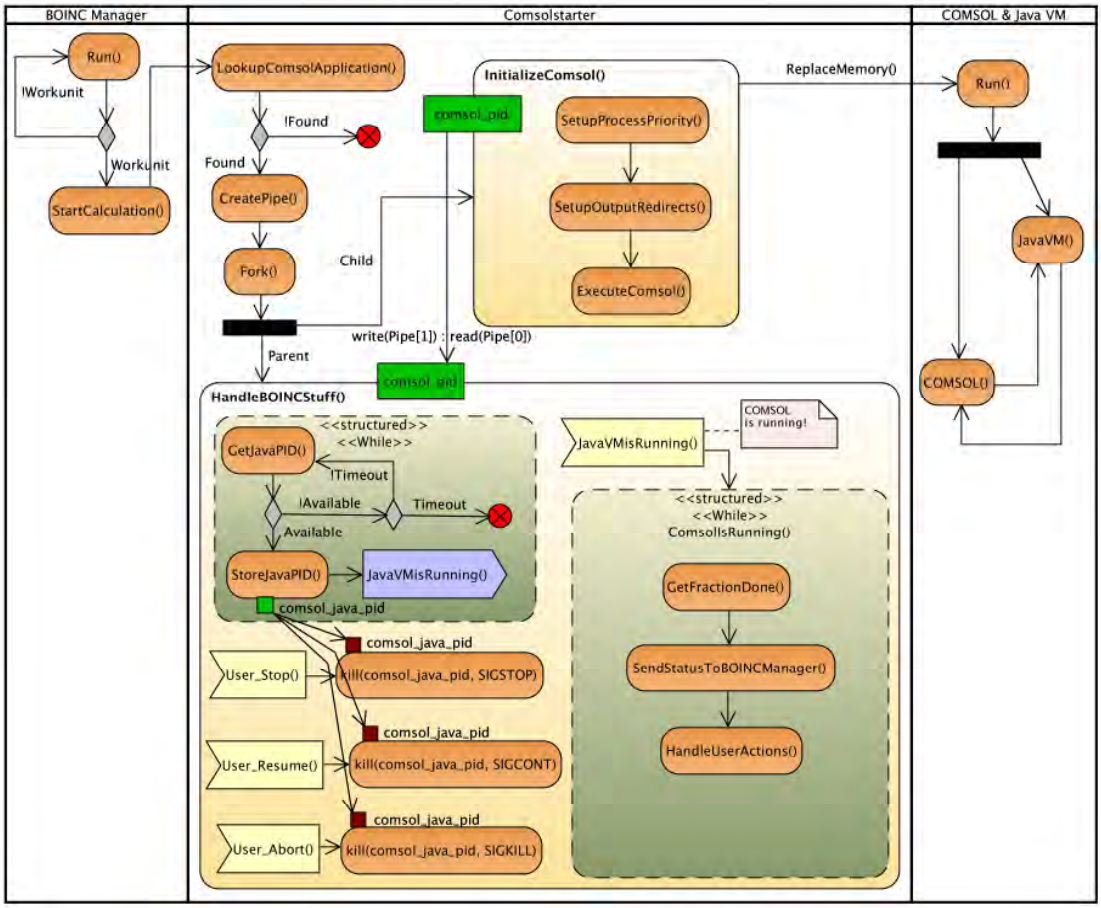


Figure 3: Architecture of the *ComsolGrid* wrapper: **(BOINC Manager)** checks for existing workunits and calls *Comsolstarter*, **(Comsolstarter)** is our wrapper implementation, **(COMSOL & Java VM)** is the native COMSOL Multiphysics application process.

user. The interface is installed on the project server computer and is directly connected to the MySQL database using the FastCGI library [3]. FastCGI provides an easy to use application programming interface to implement server applications with a full support of web communication by plain-text communication protocols like Hypertext Transfer Protocol (HTTP).

Each framework user must be assigned a correct user role. User roles are an extension to the BOINC framework. We have defined three roles: **(1)** *Administrator*, **(2)** *Developer*, and **(3)** *Scientist*. Users assigned the role *Administrator* can start, stop and maintain the BOINC project server. The *Developer* role defines a user who can do everything, just like a Linux root administrator. Users assigned the role *Scientist* can create and run new parameter studies; they are allowed to define or change the COMSOL simulation model, parameter names and values, and they can upload this configuration to the BOINC project server. In Section 2.4 we describe details of the GUI which allows a high-level description of a new parameter study.

2.2 ComsolGridStarter - BOINC Wrapper for COMSOL Multiphysics

The *ComsolGridStarter* is a wrapper routine handling the BOINC client and BOINC user commands which are used to control a COMSOL application. The BOINC user can send commands using the BOINC Manager; these commands are sent via predefined BOINC communication channels [1]. There exists one channel between the BOINC Manager and the BOINC client and another one between the BOINC client and each running scientific application.

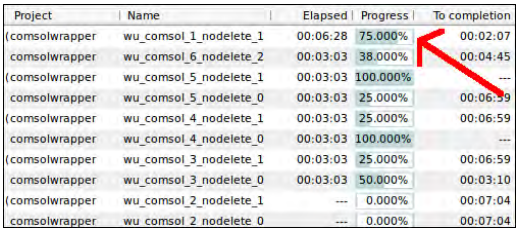
At the bottom of the Fig. 3, the user commands are visualized. Any user of a client computer can stop, resume, and abort the COMSOL simulation on that machine. The stop request is mapped to the Linux signal **stop** *(SIGSTOP)*. The Linux signal **stop** suspends one process. Suspended processes can be resumed by the Linux signal for **continuation** *(SIGCONT)* and **aborted** by the Linux kill signal *(SIGKILL)*. However, in principle one can configure ComsolGrid in a way that the simulations cannot be controlled by the user in order to ensure that a certain amount of computing power of a client machine is always used for ComsolGrid simulations.

We have implemented our own wrapper routine since all available wrapper routines [4, 5, 6] could not handle the COMSOL process tree. In fact, on Linux systems COMSOL internally starts several instances of the Java virtual machine to perform a simulation. Table 1 show the process trees of the COMSOL major versions 3.5x and 4.x. The available wrapper routines can only handle the process identification numbers (PID) of the COMSOL master processes with the PID 30521 and 15895. As a result, these processes can be controlled by operating system signals; however their child processes cannot be manipulated directly by the BOINC Client. As a consequence we have implemented a bridge between the BOINC client and the COMSOL process tree. Fig. 3 describes three processes: **(1)** *BOINC Manager*, **(2)** *Comsolstarter*, and **(3)** *COMSOL & Java VM*. The first one is responsible for the communication between the BOINC user and the BOINC client and starts the *ComsolGridStarter* for performing a simulation based on a given workunit. The second one is the most important and more complex process. *ComsolGridStarter* first checks for a valid COMSOL installation on the client computer and then executes it. It is necessary to start the correct version of COMSOL, because the model files are usually not downward compatibly with other COMSOL major version, i.e. a COMSOL model of the major version 4.x cannot be opened by COMSOL version 3.x. The routine *IntializeComsol()* starts a COMSOL process and passes its PID to the routine *HandleBOINCStuff()*. *HandleBOINCStuff()* polls the state of the internally started Java virtual machines when it becomes valid. The term valid means that the PID of the Java virtual machine is available, e.g. table 1 shows the required PIDs: (1) **comsol_java_pid= 30674** for the process named *java*, and (2) **comsol_java_pid=15919** for the process named *comsollauncher*. If one of them becomes valid, the PID is stored for later use by the handler of the user commands. The wrapper runs until the COMSOL process is finished. With these PIDs, the complete COMSOL process becomes controllable.

During runtime, the wrapper tries to retrieve information about the progress' progress and sends the value to the BOINC Client. This value

is then transferred to the BOINC Manager and displayed as shown in fig. 4.

The third process on the right hand side in fig. 3 shows the native COMSOL process. When it starts, it forks the COMSOL process and the simulation execution is supported by Java virtual machines.



Project	Name	Elapsed	Progress	To completion
(comsolwrapper	wu_comsol_1_nodelete_1	00:06:28	75.000%	00:02:07
comsolwrapper	wu_comsol_6_nodelete_2	00:03:03	38.000%	00:04:45
(comsolwrapper	wu_comsol_5_nodelete_1	00:03:03	100.000%	---
comsolwrapper	wu_comsol_5_nodelete_0	00:03:03	25.000%	00:06:59
(comsolwrapper	wu_comsol_4_nodelete_1	00:03:03	25.000%	00:06:59
comsolwrapper	wu_comsol_4_nodelete_0	00:03:03	100.000%	---
(comsolwrapper	wu_comsol_3_nodelete_1	00:03:03	25.000%	00:06:59
comsolwrapper	wu_comsol_3_nodelete_0	00:03:03	50.000%	00:03:10
(comsolwrapper	wu_comsol_2_nodelete_1	---	0.000%	00:07:04
comsolwrapper	wu_comsol_2_nodelete_0	---	0.000%	00:07:04

Figure 4: Progress in percent of one COMSOL Multiphysics simulation.

Our wrapper routine is fully configurable using an XML configuration file. It is possible to define the current processor architecture or which license should be used. In addition, we can map the output and input channels to files, i.e. for debugging purposes.

2.3 ComsolGridQt – User interface for COMSOL Multiphysics user

We have implemented a GUI *ComsolGridQt* which provides a tool that enables the user to define a new parameter study by means of a high-level description. As shown in fig. 5 the GUI guides the user through the process of creating a complete parameter study. When run for the first time, the user must add the authentication data, i.e. user name, e-mail address, and password. In the next step it is necessary to define the web address of the server interface *ComsolGridFCGI*, e.g. *127.0.0.1/comsolfcgi*. After that basic information about the project server is requested; the user receives a list of the available BOINC projects, data of the server condition, and information of the supported COMSOL versions and plattform.

The GUI includes some well-defined elements for adding COMSOL specific simulation model files, changing the order of these files, and adding the BOINC input/result template files. These files contain XML structures. When a new model is added, the GUI automatically determines the parameter names which are defined in the model and adds these names as column descriptions to a table in the *Parameter* tab. After that the user can define parameter values for each parameter name. After every change of these values, it is automatically checked whether the values are valid or not. Four values are required for a valid state. The format string of these values is:

"START:STOP:STEP:DEFAULT"

This format is modified with regard to the COMSOL *range(…)* function. Each of these values can keep one data item as a floating-point value. The user can add arbitrarily many new rows of data items.

The input template and result template files are not generated but must be created manually. In our cases, we are using only one COMSOL simulation model for each parameter study with different parameter ranges. As a result of this, we have only two files and reuse them for each new parameter study.

Table 1: Example of a process tree of the two COMSOL Multiphysics major versions: (1) 3.5x, and (2) 4.x.

(1) COMSOL Multiphysics 3.5x process tree:

su(30443) --- bash(30452) --- **comsol(30521)** --- ***java(30674)*** -+- {java}(30675}
|- {java}(30676)
`- …

(2) COMSOL Multiphysics 4.x process tree:

su(13564) --- bash(13576) -+- **comsol(15895)** --- ***comsollauncher(15919)*** -+- {comsollauncher}(15020)
| |- {comsollauncher}(15021)
| `- …

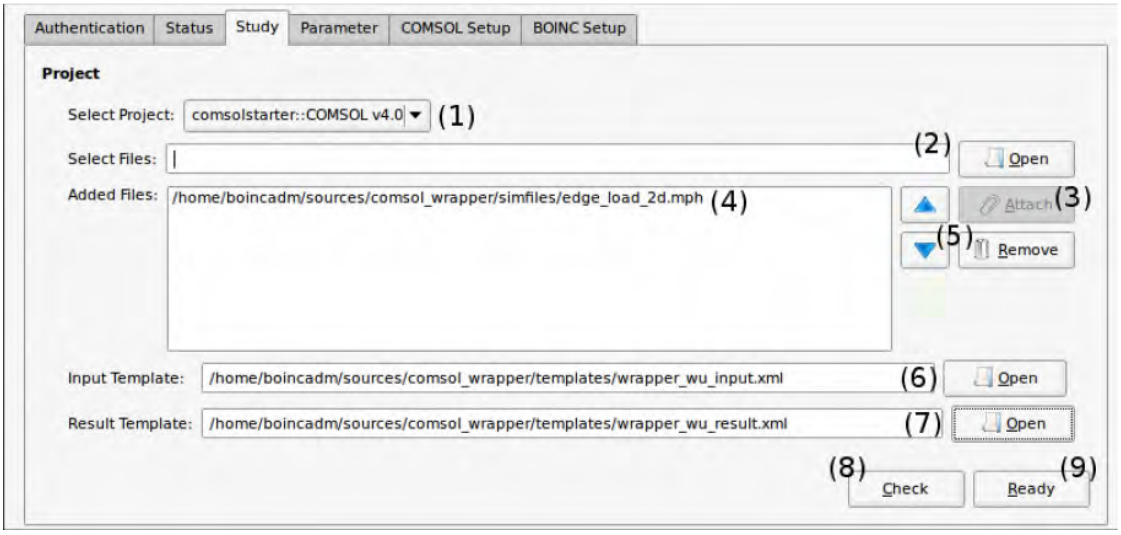


Figure 5: The GUI *ComsolGridQt* provides a high-level description of a parameter study: **(1)** select the available BOINC project, **(2, 3, 4, 5)** select, add, and arrange COMSOL Multiphysics files and define the according parameter data ranges using the *Parameter* tab, **(6, 7)** add the input template and result template file, and **(8, 9)** check your configuration and upload the files and parameter data ranges to the BOINC project server.

Fig. 6 shows an example for a complete specification of a workunit. As one can see, the workunit uses three external components: **(1)** the *COMSOL model file (*.mph)*, **(2)** the *Input Template* file, and **(3)** the *Result Template* file. The *Input Template* file defines the set of the COMSOL simulation files, the according file of the parameter values and the BOINC logical names of these files, e.g. *comsol.mph*, and *comsol.txt* (described later). It does not matter which name a COMSOL simulation file has been assigned, it is always mapped to the name *comsol.mph*. The *comsol.txt* file contains the parameter values.

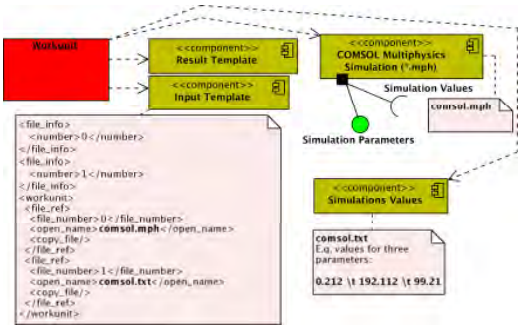


Figure 6: Components of a ComsolGrid workunit

Each row of this file describes a set of parameter values separated by tabulators. The file *comsol.txt* is created for each workunit by *ComsolGridFCGI* whenever *ComsolGridQt* uploads a new parameter study from the server.

2.4. ComsolGrid Validator and Assimilator

Two more applications are needed for the complete simulation process, namely the *comsol_bitwise_validator* and *comsol_copydir_assimilator*. Both applications are basically copies of the original BOINC validator and assimilator functions. The validator is responsible for checking whether or not a maximum processing time is reached or whether two results for the same workunit parameters are equal; if so, the result is valid, otherwise it is not valid and will be discarded. This procedure is a standard check used for public projects but can be dropped completely for projects using trustworthy computer networks like company intranets. The assimilator copies all valid results to a specified target directory that is accessible by users assigned the role *Scientist*.

3. Proof of Concept

We have set up a test installation and successfully performed a parameter study using a number of workunits. For this installation, we exported one COMSOL installation by using the Network File System (NFS) to other computers. The computers are equipped with the BOINC Client and BOINC Manager software only. Each machine is registered to the above mentioned BOINC project *"Comsolwrapper"*. Whenever a BOINC client requests a new workunit, the BOINC project searches for parameter values that have not been processed already, and sends these in form of a workunit to the requesting BOINC client.

Our test installation uses the model library example *falling_sand.mph*, which is provided by the COMSOL default installation *(Model Library / COMSOL Multiphysics / Fluid Dynamics / falling_sand)*. We have added two parameter names for the test study. Fig. 7 shows the parameter names: **(1)** *objWidth*, and **(2)** *objHeight*.

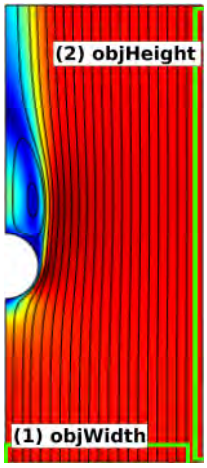


Figure 7: COMSOL Simulation model with the two parameters **(1)** *objWidth*, and **(2)** *objHeight* used for a parameter study with ComsolGrid.

The parameters have values ranging between 0.001 to 0.015 with a step size of 0.0005 for *objWidth* and 0.001 to 0.025 with a step size of 0.001 for *objHeight*. Currently there is no multiple variation scheme implemented; i.e. when one parameter is varied, the other parameters are fixed. We have created 54 workunits that have been validated using the bitwise validator which needs two results. Therefore, a total of 108 workunits have been processed by the client computers.

The parameter study took approximately 209 minutes with a success rate of 95 percent per run. It is feasible to reach 100 percent, but in our case five percent have been lost due to a malfunction of one computer where an X11 process has crashed.

4. Conclusions

In this article we have presented a new approach to perform large-scale parameter studies with COMSOL Multiphysics on a heterogeneous computer network using a tool chain based on the public resource computing framework BOINC. Our *ComsolGrid* approach enables one user to create parameter studies with an easy to use GUI. The implementation of our wrapper routine is very generic and therefore usable with other legacy applications as well. Further work is focused on the realization of a version for Mac OS X and Windows.

8. References

1. D. P. Anderson, C. Christensen, and B. Allen, *Designing a Runtime System for Volunteer Computing*, UC Berkeley Space Sciences Laboratory, Dept. of Physics, University of Oxford, and Physics Dept., University of Wisconsin - Milwaukee (2006)

2. O. Baskova, O. Gatsenko, G. Fedak, O. Lodygensky, and Y. Gordienko, *Porting Multiparametric MATLAB Application for Image and Video Processing to Desktop Grid for High-Performance Distributed Computing*, International Supercomputing Conference (ISC), Hamburg, Germany (2010)

3. M. R. Brown, *FastCGI Specification*, Open Market, Inc., 245 First Street, Cambridge, MA 02142 U.S.A, April, 1996, URL: http://www.fastcgi.com

4. D. Gonzales, F. Vega, L. Trujillo, G. Olague, M. Cardenas, L. Araujo, P. Castillo, K. Sharman, and A. Silva, *Interpreted Applications within BOINC Infrastructure*, May (2008)

5. A. C. Marosi, Z. Balaton, and P. Kacsuk, *GenWrapper: A Generic Wrapper for Running Legacy Applications on Desktop Grids*, 3rd Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid 2009), Rome, Italy (2009)

6. BOINC – Open-source software for volunteer computing and grid computing, URL: http://boinc.berkeley.edu

7. COMSOL Installations and Operations Guide, Version 3.5a, 4.0, 4.0a, included in COMSOL Multiphysics installations

8. MathWorks, Accelerating the pace of engineering and science, http://www.mathworks.com