GRIDMODELICA - A MODELING AND SIMULATION FRAMEWORK FOR THE GRID

Kaj Nyström, Peter Aronsson and Peter Fritzson Linköping University
Department of Computer and Information Science
Sweden

Abstract

Simulation of complex models is a computationally expensive task. With the advent of grid computing, modelers can gain access to vast amounts of cheap computational power. This has however up until now required quite some effort to be put into specially written simulations in Fortran or C and also in the deployment of the simulation on the grid; a set of interconnected computers behaving as one single computational resource from an end-user perspective. We propose a framework called GridModelica for transparently creating and deploying simulations in the high level modeling language Modelica on computational grids. The first step in this framework is taken with the application GridParamSweep which demonstrates how easily parameter studies can automatically be performed in a grid environment.
Keywords: GridModelica, OpenModelica, Grid Computing, Modelica, parameter sweep

Introduction

One of the greatest problems with modeling and simulation today is the high computational costs associated with simulation of complex models. Many simulations require the computer to solve systems with many hundreds of thousands of equations, which can take quite some time even on powerful workstations. With the application GridParamSweep we take the first step towards putting the power of grid computing in the hands of modelers in order to minimize time- and cost consumption for simulating complex models.

Already today, lots of complex simulations are run on high performance computers and clusters and even on computational grids. To the best of our knowledge though, these models are almost without exception specially written in Fortran, C or similar languages. While these languages can provide high performance, if sufficiently well written, they do not really comply with the needs of today's modelers.

Instead we use the mathematical modeling language Modelica [1, 2] which combines the power of modern solvers written in Fortran or C with a high level of abstraction, object orientation and reuse. This, in combination with the computational power of the grid will provide a cheap high-performance platform for complex simulations.

To achieve the goal of providing the possibility of transparently running simulations on the grid we propose a framework called *GridModelica*. GridModelica will be both a language extension of the Modelica modeling language and a modeling and simulation toolkit, which enables modeling and simulation targeting the grid. As a first example we study how automated parameter studies can be performed in this framework. Such problems can be used for instance in parameter design optimization to find the optimal set of parameters for a given model. These examples are suitable for execution on the grid due to the nature of the problem with many independent simulations. These simulations can be executed on different computers with no communication between them, resulting in little overhead.

Modelica

In this paper the model intended for simulation is specified in the mathematical modeling language Modelica [1], which allows acausal modeling of heterogeneous systems spanning many different domains, such as for example electrical, mechanical and thermo fluid problems. Modelica is object oriented which allows for a high level of reuse and or-

dering of objects into hierarchies.
The modelica model can be either written by hand or specified using a graphical model editor as pictured in Figure 2. Either way, the result is an textual model which can be compiled into an executable which in turn can be run independently of the modeling environment, making it very suitable for grid deployment.

The Grid

A computational grid is an interconnected set of computers which together work as a single computational unit from an end-user perspective. A definition from the book "The Grid" [3] is "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities". The word "grid" is borrowed from the electrical power grids, a much used analogy in grid computing. The idea is that when you plug in for example your toaster to an electrical outlet, you do not care where the electrical power you are using is produced. You just plug in an use it. A computational grid is meant to work in the same way. You do not care about where your job is executed. You just submit your job and then eventually fetch your results. The grid middleware takes care of the rest. A schematic sketch of the grid is pictured in Figure 1.
We would like to point out that using the grid is not yet as simple as using the electrical power grid. However a lot of development is still going on in the area of grid middleware and improvements are constantly being made. In the GridModelica project we use the Nordugrid middleware [4] which is closely related to the de facto standard the Globus toolkit [5]. The Nordugrid middleware has been deployed on both the Nordugrid testbed and the Swegrid [6] computational grid, both of which have been used successfully with the GridParamSweep application. The Nordugrid currently (2004-05-15 13:45 CET) encompasses 2327 processors on 35 different sites, providing from one to 644 processors each. The sites are distributed throughout ten different countries all over the world but concentrated to Scandinavia. Resources are shared with local users as the cluster managers sets the priorities which means that normally, only idle processors are used by the grid, thus exploiting a resource which would otherwise have been wasted.

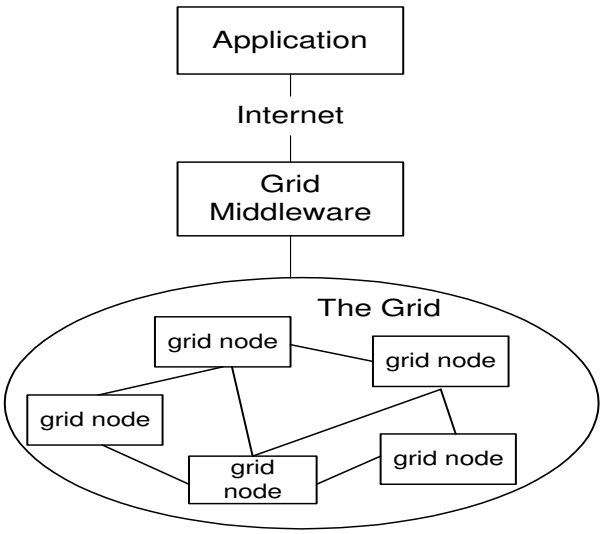

Figure 1: Component sketch: The grid

The grid has a hetrogeneous nature, spanning multiple administrative domains. This means that it is not an entirely stable environment, at least not as stable as a single computer of which the user has total control. There are as yet no guarantees that your job will not be canceled or receive very low priority on behalf of local users or even other grid users. The applications using the Grid should be aware of this so that the appropriate action, for example resubmission or compensation can be taken.

The Application

The parameter studies performed is as stated before a good example of a task which is easily parallelizable on the grid since the different simulations can be made independent of each other. Parameters sweeps can also fairly easily be extended using optimizer routines so that a models parameters can be optimized for a given criterion, increasing the usefulness of parameter studies.
As for the application itself, the user can specify model parameters, the range and the step size use. The job will then be submitted to the grid for execution and the results will be returned to the user at his request when the jobs have finished. The resulting plot can be viewed with the plotting utility of the users choice. Job monitoring is supported automatically through the Nordugrid middleware, both in detailed format about every job and as an overview through the grid monitor depicted in figure 3.

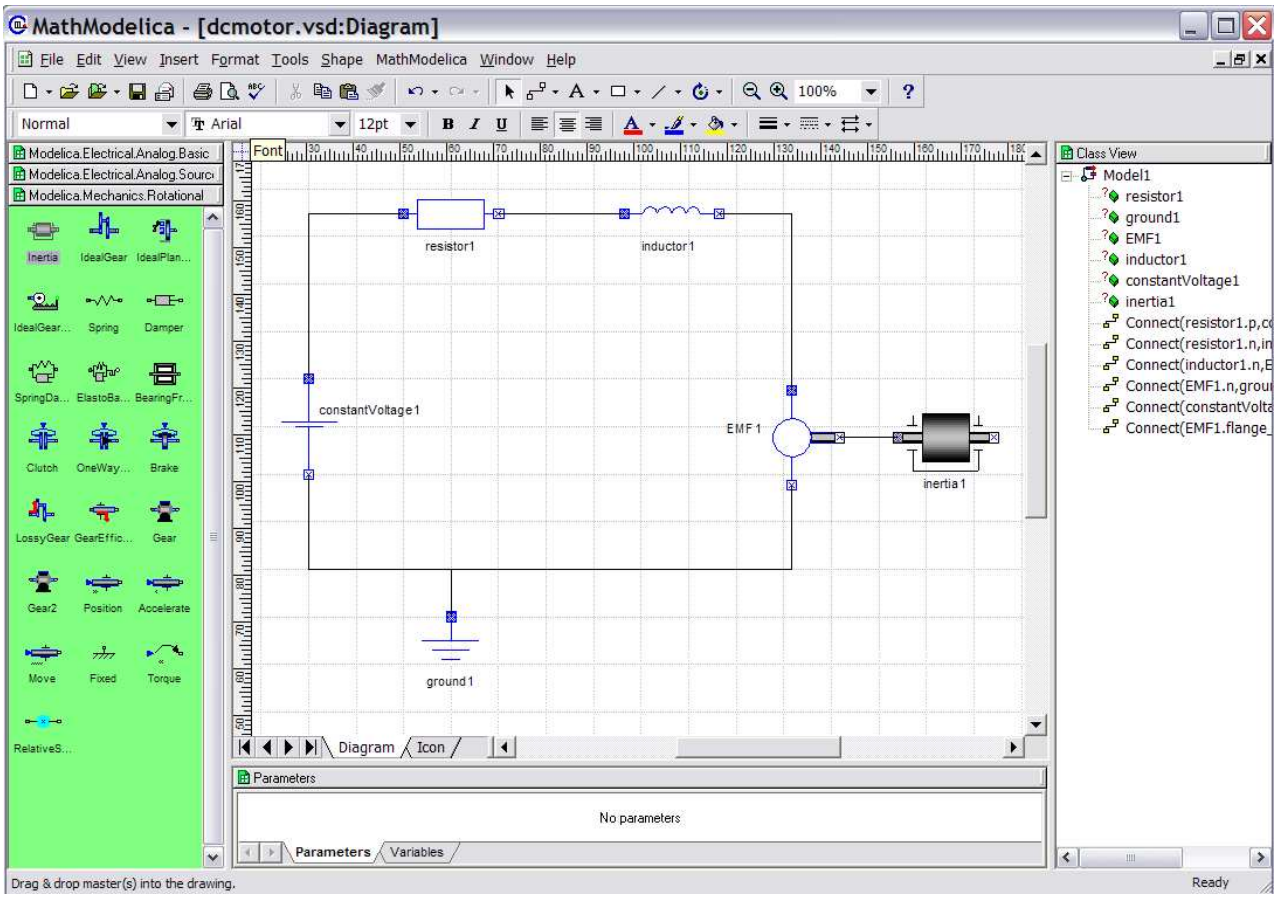

Figure 2: The MathModelica graphical model editor from MathCore Engineering [9]

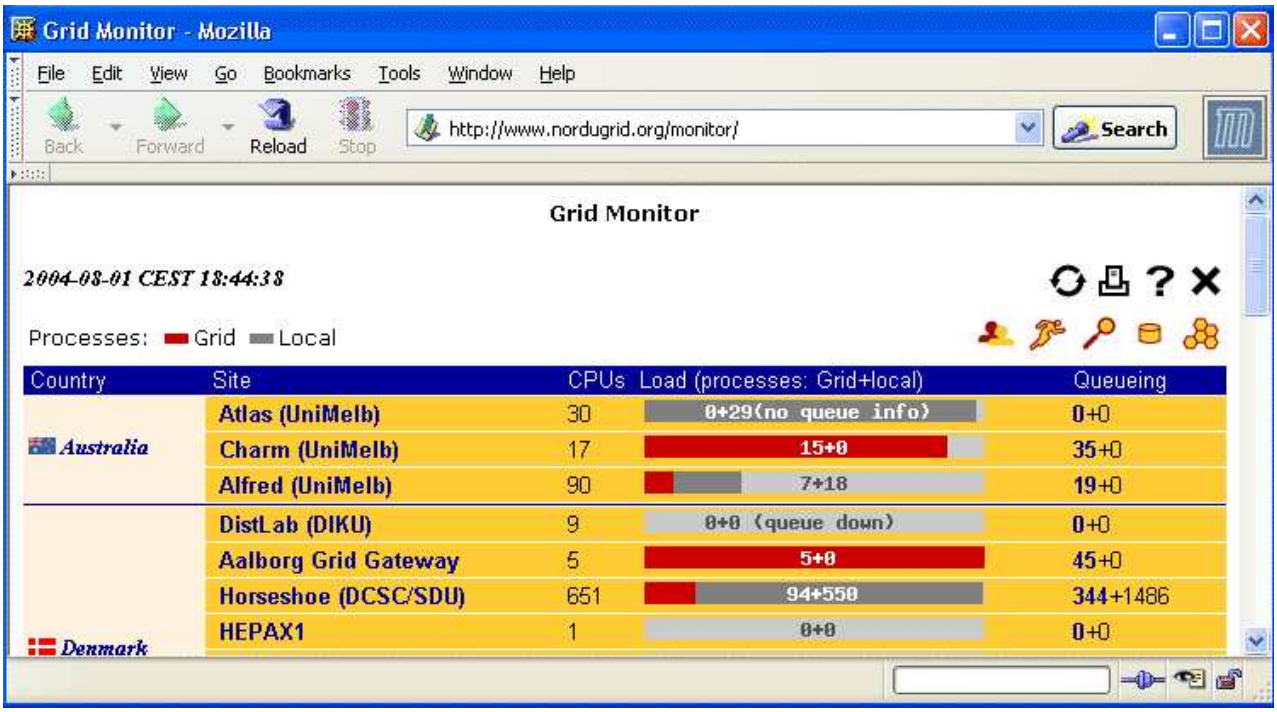

Figure 3: The Nordugrid Grid Monitor

Example	Timesteps	N.of parameter values	Exec. time, single	Exec. time, Nordugrid.
DCMotor, 40 eqns	10000	10	0.5m	10m
DCMotor, 40 eqns	10000	100	4m	175m
DCMotor, 40 eqns	1000000	10	7m	29m
DCMotor, 40 eqns	1000000	100	68m	200m
Twoloops, 1016 eqns	1000	10	35m	24m
Twoloops, 1016 eqns	1000	100	330m	75m
Twoloops, 1016 eqns	10000	10	423m	59m
Twoloops, 1016 eqns	10000	100	4229m	560m

Table 1: Time to execute different parameter studies on the Nordugrid compared to a single machine.

What is done in the application is in short this:

1. Initialization of the environment, including setting up of the Nordugrid proxy and credentials for accessing the grid.

2. Generation of submission and result retrieving scripts.

3. Generation of Extendend Resource Job Specification, XRSL-file. The XRSL-file specifies the nature of the job, such as files which should be included, command line arguments and the estimated resources required.

4. Compilation of the Modelica model

5. Merging of results into comprehensible format.

What the user essentially has to do is first to write the model in question, using a text editor or a visual tool such as MathModelica depicted in Figure 2. The user also has to specify the parameters, the range to investigate and the step size in a simple and straightforward text file.
Some additional software is required for usage:

1. A modelica compiler, such as the OpenModelica compiler [10, 11] or Dymola [12].

2. A C-compiler, for example gcc.

3. The Nordugrid toolkit [4]

4. Certificates authorizing usage of the appropriate grid resources.

GridParamSweep can be downloaded from the GridModelica Project homepage [7].

Results and Conclusions

Not surprisingly the grid proves very well suited for this type of applications, especially if the job itself is fairly complex. As we are dealing with independent tasks, the speedup compared to doing the same job on one workstation is almost proportional to the number of steps in the parameter sweep. We have successfully tested different models generating systems with up to a thousand equations. The number of processors used are roughly equivalent to the number of steps in the parameter sweep. Exact results are found in table 1. The first example is the circuit depicted in figure 2 and the second example is from the Modelica Multibody library [8].
Standalone simulations in table 1 were conducted on a Pentium III 1.8 Ghz with 512 Mb of RAM. Grid simulations in table 1 are averages from five simulation runs per test case, conducted on the Nordugrid in five consecutive days, 2004-07-14 2004-07-19 between 10.00 and 14.00. It is important to note that execution time may vary greatly with the current load of the grid. Some jobs may be placed in queue, some may receive low priority and some may be scheduled for execution on a comparatively slow cluster node. Even so, results show that all but the smallest parameter studies of simulations benefits from grid deployment. As the overhead for job submission is fairly large the greatest speedup compared to single machine execution are found when the job is fairly complex. The DCmotor model proves too small for efficient grid execution.
It should be noted that the Twoloops model with its 1016 equations is by no means a large model. Tests of the GridParamSweep application have been carried out on models consisting of way beyond 20000 equations, producing over 3GB of data with no scal-

ing problem whatsoever.

Discussion

The most serious problem encountered in this study was the large overhead for submitting jobs to the grid, limiting its usefulness to fairly complex simulations. This is not really that surprising since most grid jobs today are extremely computationally expensive physics simulations and data analysis jobs. Their normal execution time may very well exceed 12 hours and if that is the case, an overhead of half a minute per job submission is really not a problem.

The average overhead for each job submission varies with the grid load and the connection speed of the submitting machine but never exceeded 60 seconds in our measurements. This overhead is however not quite cumulative with the number of jobs since the major part of this delay normally is due to the grid manager program and on the assigned grid node rather than on the submitting workstation, leaving the user free to submit the next job rather than wait for the previous one to start it's execution. Recent grid middleware development has reduced this overhead and further improvement can also be expected as the number of smaller grid jobs increases. It is also possible to reduce the overhead significantly by manual tuning of the grid submission procedure, for example avoid checking some grids that are known to be slow or not operational.

This overhead still means though that grid execution is not so useful for small simulations spanning few parameter steps. It is difficult to give estimations on when a job should be submitted to the grid and not since so many factors are involved which affect the execution time and the overhead. What can be clearly stated is that simulation of complex models no longer requires a top-of-the-line workstation closely at hand. Instead, the huge amount of spare cycles the grid has to offer can be used.

Another problem which is far more difficult to solve is the problem of moving data. The Twoloops example with 10000 time steps produces approximately 257 MB of data. When executing on a single machine, moving the data is of course not an issue since the data is produced directly on the workstation in question. When executing the job on a grid node however, the data is produced somewhere else. A high speed network is thus quite useful for retrieving the data efficiently. In our test runs, we used a mere 5 Mbit non dedicated ethernet connection to the internet. This means that a significant decrease in data retrieval time should be obtained with a better internet connection.

The presentation of the raw results presents a bit of a problem. In general, the user is interested in how the behavior of a variable x varies with time depending on how a parameter p is chosen. In the dcmotor in Figure 2, one would perhaps be interested in the rotation angle of the inertia in relation to what value is chosen for R on the resistor. This means that ordinary 2d plots are not always sufficient to present the whole result, at least not if the number of parameter values studied is large. The plot will have to be extended to three dimensions using some simple Matlab commands.

In conclusion, grid computing indeed has a lot to offer the modeling community in conjunction with high level modeling languages such as Modelica. Many of the typical jobs submitted to the grid today are different types of simulations but they are to a great extent specially written in C or Fortran and adapted for the grid by hand which is a difficult and time consuming task. High level modeling in GridModelica should considerably facilitate the process of cheaply simulating computationally expensive tasks in the future.

Future work in GridModelica

We would like to emphasize that this is by no means a complete grid simulation environment yet but merely an example of how the grid can transparently be used by modelers even though the grid is still comparatively young.

There are quite a number of other improvements to the GridParamSweep application that could be conceived. A few of them we have considered are:

- Integration with the OpenModelica compiler in order to make the application completely open source.
- Multiple submissions of the same job in order to reduce chances of a total job failure. At least one instance of the same job should always execute successfully.
- Automatic adaptive job submission, which could greatly reduce overhead.

- Clustering of smaller jobs into one job, thus increasing the applications usefulness for smaller jobs by reducing the number of submissions that has to be done.

- Better checking of job termination and acting accordingly. If a job seems to have failed, an immediate resubmission should be done.

Some of these may eventually make their way into the GridModelica framework but it is actually not likely that they will be implemented in the GridParamSweep application.
Future work on the GridModelica framework will further facilitate simulation deployment on the grid and will also include features such as language extensions for high level parallelization, internal automatic parallelization and job scheduling for the grid.

Related Work

There are a number of other tools for conducting parameter studies, some even in parallel though none has to our knowledge yet been adapted for computational grids. Two applications that treat Modelica models are the MOPS framework [13] and the Distributed Parameter Study application [14]. They have in common that they enable distributed parameter studies but they also require direct access to the computational nodes, for example via rsh, which the grid for security reasons does not allow. They also require considerably more setup on the remote computing nodes. This makes them perhaps more reliable to use but also less powerful and more expensive due to the necessary investment in hardware compared to parameter sweeps performed on the grid.
Automatic parallelization of Modelica models can also be performed on a per-model basis on a much more fine grained level as done by Peter Aronsson [15]. This approach will in the future be merged into GridModelica, adding low level parallelization to the framework.

References

[1] Fritzson P. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2004.

[2] Elmqvist H, Mattson S-E, Otter M. *Modelica-A Language for Physical System Modeling, Visualization and Interaction*. In Proceedings of the IEEE Symposium on Computer Aided Cibtrol System Design, Hawaii, USA, August 22-27 1999.

[3] Foster I, Kesselman C, editors. *The Grid - Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.

[4] *The Nordugrid Virtual Organization*. http://www.nordugrid.org.

[5] *The Globus Alliance*. http://www.globus.org.

[6] *The Swegrid Computational Resource*. http://www.swegrid.se.

[7] *The GridModelica homepage*. http://www.ida.liu.se/labs/pelab/modelica/GridModelica.html.

[8] Otter M, Elmqvist H, Mattson S-E. *The new Modelica Multibody Library* proceedings of the Modelica 2003 conference, November 3-4 2003, Linköping Sweden.

[9] *MathCore Engineering*. http://www.mathcore.com.

[10] *The OpenModelica project*. http://www.ida.liu.se/~pelab/modelica/OpenModelica.html.

[11] Fritzson P, Aronsson O, Bunus P, Engelson V, Johansson H, Karström A, Saldamli L. *The Open Source Modelica Project*. In Proceedings of the 2nd International Modelica Conference, Munich Germany, 18-19 2002. http://www.ida.liu.se/~pelab/modelica/OpenModelica.html.

[12] *Dynasim*. http://www.dynasim.com.

[13] Joos H-D, Looye G, Moormann D. *Design of Robust Inversion Control Laws using Multi-Objective Optimization* AIAA Guidance and Control Conference, Montreal, Canada, 2001.

[14] Engelson V, Fritzson P. *A Distributed Simulation Environment* Proceedings of SIMS 2002 conference.

[15] Aronsson P. *Automatic parallelization if Simulation Code from Equation Based Languages*. Licenciate thesis No. 933, Department of Computer and Information Science, Linköping University, Sweden