Globus-Based Grid Computing Simulations of Action Potential Propagation on Cardiac Tissues

José M. Alonso, Vicente Hernández, and Germán Moltó[⋆]

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia. Camino de Vera s/n 46022 Valencia, Spain
Tel. +34963877356, Fax +34963877359
{jmalonso,vhernand,gmolto}@dsic.upv.es

Abstract. With the advent of Grid technologies, the study of the electrical activity of the heart, by means of concurrent parametric simulations of the action potential propagation on cardiac tissues, can be greatly benefited. Studies of the electrical behaviour, such as late ischemia require the execution of multiple computational and memory intensive parametric simulations. This paper describes the integration, into a Grid infrastructure, of a parallel MPI-based system for the simulation of action potential propagation on a three-dimensional parallelepiped-modelled cardiac tissue. Developed upon the Globus Toolkit, it features state-of-the-art capabilities such as data compression, simulation failure recovery, and the combination of parallel execution on distributed resources, what has enabled an outstanding increase in research productivity.

1 Introduction

The simulation of action potential propagation on cardiac tissues represents a major computational challenge. The fine spatial and time discretization steps required to solve the equation (1) that governs this phenomenon on a monodomain cardiac model makes this problem only affordable with High Performance Computing techniques. This is particularly important for three-dimensional executions, where a simulation of action potential propagation during few milliseconds on a medium-sized tissue may last for several days on a sequential platform.

$$\nabla \cdot \sigma \nabla Vm = C_m \cdot \frac{dVm}{dt} + I_{ion} + I_{st}. \tag{1}$$

The previous equation relates the membrane potential of the cells, Vm, the ionic currents that traverse the membrane, I_{ion}, the membrane capacitance, C_m, the anisotropy tensor, σ, and the electrical stimulus, I_{st}. The comprehensive Luo-Rudy Phase II [1] cellular model has been employed to calculate the I_{ion} term.

[⋆] The authors wish to thank the financial support received from The Spanish Ministry of Science and Technology to develop the project GRID-IT (TIC2003-0131). This work has been partially supported by the Structural Funds of the European Regional Development Fund (ERDF).

In addition to the inherent computational cost of a single simulation, there are many research studies that require the execution of a huge amount of parametric simulations. Studies of vulnerable window in ischemia require to vary the time interval between two consecutive stimulus in order to detect the range of values which provokes a reentry, a phenomenon that can derive into heart fibrillation. Besides, to study the effects of late ischemia it is necessary to vary the coupling resistances in all the dimensions of the tissue and analyze the evolution of the electrical activity for different anisotropy ratios. Moreover, to evaluate the influence of certain medicines, it is crucial to alter the concentration of these drugs, over a determined range, to study how it affects to the action potential propagation.

Even though there have been several parallel approaches to this computational problem [2], the good efficiency results achieved on a beowulf cluster, together with appearing to be the first cardiac simulation system to combine both a parallel and a Grid Computing approach, represent a step forward in the study of the electrical activity of the heart.

Therefore, both parallel computing techniques, that speedup a single simulation, and Grid Computing technology, that enhances the efficiency of multiple simulations, will be combined in order to achieve a global simulation system that increases the productivity for these computational demanding cardiac case studies.

The article is structured as follows: Section 2 describes the main functionality of the simulation system. Then, section 3 details the characteristics of the Grid Computing system designed. Next, in section 4, a case study is presented to expose the functionality in a production testbed. Finally, section 5 concludes the paper, exposing the relevant achievements.

2 Characteristics of the Simulation System

A cardiac tissue simulation consists of an iterative process that allows to calculate the membrane potential of the cells along a time period.

First, a parallel simulation system was developed for two-dimensional tissues [3] in order to reduce the simulation time on beowulf architectures with outstanding efficiency results (94% of efficiency with 32 processors). Next, the simulation system has been extended for three-dimensional anisotropic tissues, achieving good scalability results. For example, Fig. 1 shows the speedup and efficiency when simulating, on a cluster of PCs, an action potential propagation during 250 ms in a 100x100x100 cell cardiac tissue, with a timestep of 10 μs. Such a simulation lasts 177.97 hours on a sequential platform, but only 6.45 hours when using 32 processors.

The simulation system periodically generates a set of checkpoint files, using the MPI-2 parallel I/O routines, what allows a simulation to be restarted from the point that was stopped, even with a different number of processors. The checkpoint data consist of a snapshot of the tissue, that is, a double precision binary dump of all its cells, along with other ones of the membrane potential and ionic vectors.

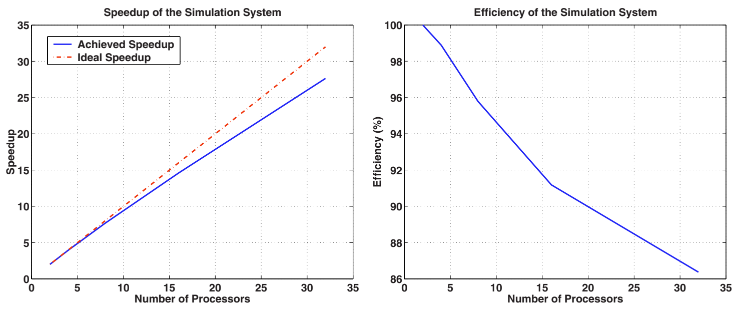

Fig. 1. Speedup and efficiency of the simulation system. Running on a 20 Pentium Xeon 2.0 Ghz biprocessor cluster, with 1 GByte of RAM and a interconnected by a SCI network.

Provided that the checkpoint data for this application can result in a very large data set, we have analyzed the effectivity of data compression on realistic checkpoint files, where a 100x100x100 cell cardiac tissue is stimulated to provoke an action potential that depolarizes all the tissue.

Table 1 shows the compression ratio that is achieved, for the best and the worst tissue state, using a Lempel-Ziv coding provided by the standard *gzip* Unix command. The best case corresponds to a tissue in rest state, i.e. at the beginning of the simulation, where similar values may be found for all the cells of the tissue. On the other hand, the worst case corresponds to a propagating wavefront (once applied the supra-threshold stimulus) on an anisotropic tissue, where changes between the cells are very frequent.

Table 1. Effectivity of checkpoint data compression for the best and the worst case of a 100x100x100 cell tissue state. Size is expressed in MBytes.

	Uncompressed Data	Compressed Data	Compression Ratio
Best Case	515.2	4.9	105.14
Worst Case	515.2	142.4	3.62

In both cases, data compression required an average 60 seconds, while decompression lasted for an average 14 seconds. Therefore, compression offers a significant reduction of the binary data generated by the simulator, as in the worst case the result files can be reduced to less than a third part.

3 Grid Computing System Developed

3.1 Portability and Interoperability

Enabling portability requires that all the platform-dependent optimizations, such as the compiler flags *-march* or *-mcpu*, are avoided. Besides, architecturedependent optimized numerical libraries, such as the BLAS [4] and LAPACK [5] implementation by the Intel Math Kernel Library, should not be used, as they may result in executing illegal instructions on the remote host if both architectures do not match. Fortunately, traditional compiler optimization flags, i.e. -*O3*, can be used with no risk.

Our application has been statically linked to generate a self-contained simulation system. Even the MPI communication library has been introduced into the executable, using a MPICH [6] implementation, configured to disable shared-memory communication between processes on the same node of a cluster, which can potentially introduce memory-leak problems because of relying on the System V IPC facilities [7]. This procedure enables to perform a parallel execution without depending on the MPI implementation of the execution host.

This parallel self-contained simulation system can be executed on a wide range of Linux machines, thus isolating the application from the runtime environment, something that may, a priori, be unknown in a Grid. This has been ensured by executing parallel simulations in a variety of different architectures such as Pentium III, Pentium IV, Pentium Xeon and even Intel Itanium 2 running different Linux flavours such as Red Hat Linux Advanced Server, Red Hat 8.0, Fedora Core 1 and Debian GNU/Linux.

It should be pointed out that such a simulator, compiled on an Intel Pentium Xeon PC (32 bit), runs on compatibility mode on an Intel Itanium 2 (64 bit) platform, but it is up to 8 times slower than on the original architecture. Therefore, we have natively compiled on the Intel Itanium 2 platform in order to achieve comparable execution times on both architectures, and to be able to exploit Itanium Grid execution nodes. This results on two self-contained simulation systems, one for IA-32 and other one for IA-64, an strategy that could be refined to target more architectures.

3.2 Modules Developed

Figure 2 shows a conceptual view of the Grid Computing system developed based on the Globus Toolkit [8]. The *JobScheduler* is the module responsible for the allocation of simulations to computational resources. This module delegates, for each simulation, into a *JobSubmitter*, which is in charge of the proper execution of the task in the resource.

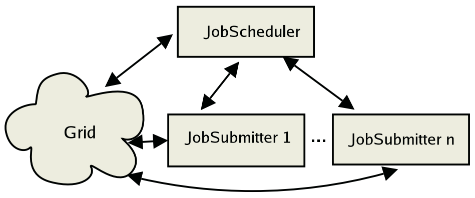

Fig. 2. Scheme of the Grid Computing system developed.

The JobScheduler Module. This module reads an input file with a parametric description of the multiple simulations that form the case study. For each simulation, it computes the best available resource, from a predefined list of machines, by consulting its number of available nodes, via the Monitoring and Discovery Service (MDS). Clusters with the Globus Resource Allocation Manager (GRAM) Reporter installed, report the number of free computing nodes, delegating in the local queue manager (LoadLeveler, PBS, etc). For workstations, an estimation of the CPU usage during the last minute serves as an indicator of the availability of the resource. This strategy allows to customize a parallel execution to the number of available nodes in the host. Then, this module selects an appropiate executable based on the architecture of the remote machine.

The JobScheduler is also responsible for submitting the unassigned simulations and restarting the failed executions, delegating, for each of them, on an instance of the JobSubmitter module. If no available resources exist, it periodically checks their availability to continue submitting pending tasks.

The JobSubmitter Module. This module is in charge of the proper execution of a single simulation. First of all, the input files that the simulation system needs are staged in, via the GridFTP service, to the execution host. Through the Globus native interface, the remote machine is queried about its availability to run MPI jobs, so the parallel or serial execution can be selected. The execution of the simulation is integrated, if configured, with the queue manager of the remote node (PBS, LoadLeveler, etc), thus respecting the execution policies of that organization.

While the simulation is running on the remote resource, a checkpoint job is periodically submitted by this module, which transfers, if not already done, a compressed image of the generated checkpoint data to the local machine. Thus, the latest checkpoint data always resides at the submission machine and a failed simulation can be automatically resumed on a new computational resource. A message digest mechanism ensures that no old checkpoint data is transferred twice, wasting bandwidth.

Once the execution has finished, all the result data are compressed, transferred back to the submission node and saved on the appropriate local folder created for this simulation. All the temporary created files in the execution node are deleted, and finally, the JobSubmitter module anotates whether the simulation has finished correctly or not. This information will be used by the JobScheduler module to be able to restart the failed simulations.

4 Case Study

4.1 Description

Myochardial ischemia is a condition caused by oxygen deprivation to the heart that can result in an angina. It is known that ischemia increases the extracellular potassium concentration in the affected area, what shortens the action potential duration. Therefore, it is possible to study the effects of several degrees of ischemia, by means of multiple parametric simulations, varying the extracellular potassium concentration for a group of cells in the tissue.

For a 50x50x50 cells cardiac tissue, a range from 5 to 12.9 milliMolar (mM.) potassium extracellular concentration will be studied, with an increment of 0.2 mM. between each simulation. Only 2 ms will be simulated with a timestep of 0.01 ms. This results in 40 independent parametric simulations that can be executed in the computational resources that a Grid testbed offers.

4.2 Testbed

The available testbed is composed of local resources, belonging to our research group, the High Performance Networking and Computing Group (GRyCAP-UPV), and remote resources from the Distributed Systems Architecture & Security group (ASDS-UCM), at Madrid Complutense University. Table 2 summarizes the main features of the machines.

The Globus Toolkit version 2.4 [9] has been installed on the testbed. Ramses cluster is the Certication Authority of GRyCAP-UPV and its credentials have been installed on ASDS-UCM machines to allow remote job submission.

Table 2. Detailed machine characteristics of the testbed.

Machine	Processors	Memory
Kefren (grycap)	20 (2 x Intel Xeon 2.0 Ghz)	1 GByte
Ramses (grycap)	12 (2 x Intel Pentium III 866 Mhz)	512 MBytes
Bastet (grycap)	2 x Itanium 2 900 Mhz	4 GBytes
Hydrus,Cygnus (asds)	1 x Pentium IV 2.53 Ghz	512 MBytes
Aquila (asds)	1 x Pentium III 666 Mhz	128 MBytes
Cepheus (asds)	1 x Pentium III 666 Mhz	256 MBytes

4.3 Execution Results

Table 3 summarizes the tasks distribution in the Grid. The maximum number of processors in a parallel execution has been limited to eight, a polite policy with the rest of the users that allows multiple concurrent simulations. In the table, an entry like 7 (8 p.) indicates that seven simulations were performed with eight processors each one. Machine Bastet does not appear on the table because it was heavily loaded and the scheduler never chose it for job submission. Each simulation generates 64 MBytes of data, that can be compressed to 1.7 MBytes.

The Grid execution of this short case study lasted for 1232 seconds (20.53 minutes). On the other hand, a traditional sequential execution in only one node of cluster Kefren required 154.05 minutes, what represents a speedup of 7.5 in the cardiac case study execution in the Grid. A parallel computing approach, performing 8-processors parallel executions sequentially in cluster Kefren, required 32.74 minutes.

Table 3. Distribution of the simulations in the testbed, for each machine. The number in parentheses indicates the number of processors involved in the execution.

Machine	Simulations	Machine	Simulations
Kefren	7 (8 p.), 5 (5 p.), 3 (1 p.), 2 (4 p.), 2 (7 p.)	Hydrus	3 (1 p.)
Ramses	7 (8 p.), 3 (1.p)	Cygnus	3 (1 p.)
Cepheus	2 (1 p.)	Aquila	3 (1 p.)

It can be seen that the scheduler has distributed the tasks proportional to the computational power of each machine, what represents a proper balance loading scheme. Had the machine Bastet been available, it would have received a task load adequate to its computational power. Besides, as the state of the Grid is investigated before each task submission, the job allocation is dynamically adjusted to the computational load of the resources during the scheduling process.

It is important to point out that a Grid execution is ideal for resource-starved cardiac case studies, as it broadens the computing resources available, no longer confined to those belonging to a single organization.

5 Conclusions

This paper has presented the integration of a system for the simulation of action potential propagation on three-dimensional monodomain modelled cardiac tissues, into a Globus-based Grid infrastructure.

The Grid Computing system developed features state-of-the-art capabilities such as data compression, self-contained executable and dependencies migration, cross-linux portability and parallel execution of simulations on cluster nodes of the Grid.

With the execution of cardiac case studies in a Grid environment, productivity has been largely enhanced compared to traditional sequential execution approaches. It is clear that the advent of new Grid technologies is getting possible to increase the research productivity by performing multiple concurrent, geographically distributed, parallel or sequential simulations of action potential propagation on cardiac tissues.

Therefore, having available a parallel simulation system that can be integrated with a Grid infrastructure enables to focus both on speedup, running on a cluster of PCs, and productivity, taking full advantage of the computational power of a Grid.

Acknowledgements

We would like to thank the Distributed Systems Architecture & Security group, belonging to Madrid Complutense University (Spain), for sharing both its computational resources, enlarging our testbed, and its knowledge through the results of the GridWay project [10].

References

1. Luo, C.H., Rudy, Y.: A Dynamic Model of the Cardiac Ventricular Action Potential. I Simulations of Ionic Currents and Concentration Changes. Circulation Research **74** (1994) 1071–1096
2. Vigmond, E.J., Aguel, F., Trayanova, N.A.: Computational Techniques for Solving the Bidomain Equations in Three Dimensions. IEEE Transactions on Biomedical Engineering **49** (2002) 1260–1269
3. Alonso, J.M., Ferrero (Jr.), J.M., Hernández, V., Moltó, G., Monserrat, M., Saiz, J.: High Performance Cardiac Tissue Electrical Activity Simulation on a Parallel Environment. Proceedings of the First European HealthGrid Conference (2003) 84–91
4. Lawson, C.L., Hanson, R.J., Kincaid, D., Krogh, F.T.: Basic Linear Algebra Subprograms for FORTRAN Usage. ACM Trans. Math. Soft. **5** (1979) 308–323
5. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide. Third edn. Society for Industrial and Applied Mathematics, Philadelphia, PA (1999)
6. Gropp, W., Lusk, E., Doss, N., Skjellum, A.: A High-Performance, Portable, Implementation of the MPI Message Passing Interface Standard. Parallel Computing **22** (1996) 789–828
7. Gropp, W.D., Lusk, E.: User's Guide for MPICH, a Portable Implementation of MPI. Mathematics and Computer Science Division, Argonne National Laboratory. (1996)
8. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for the Distributed Systems Integration. Infrastructure WG, Global Grid Forum (2002)
9. Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. Intl. J. Supercomputer Applications **11** (1997) 115–128
10. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Software Practice and Experience (2004) To appear.