EPIS: A Grid Platform to Ease and Optimize Multi-agent Simulators Running

E. Blanchart, C. Cambier, C. Canape, B. Gaudou, T.-N. Ho, T.-V. Ho, C. Lang, F. Michel, N. Marilleau, and L. Philippe

Abstract. This paper presents the work done during the first year of the EPIS project. This project deals with the process of conducting multiple and parallel multi agents-based simulations (MABS) on a cluster or a grid in order to generate sufficient data for scientific use (*e.g.* in the case of a sensibility analysis of a simulation). We provide a new, general and user-friendly approach to marry MABS and High-Performance Computing (HPC). We, thus, propose a workflow and an associated HPC infrastructure. These two permit to easily deploy a lot of simulations on a cluster without any prior parallelizing work. The method wants to be as generic as possible: no particular MABS targeted, no overhead and HPC compliance work has to be done only once. Moreover the user is guided by a web interface that handles the workflow.

Keywords: Multi-agent simulation, distributed simulation, parallelization, High-Performance Computing.

Eric Blanchart
UMR 210 Eco&Sols (INRA, IRD, SupAgro), IRD, Montpellier, France

Christophe Cambier
UMI 209 UMMISCO, IRD, UCAD, Dakar, Sénégal

Clive Canape
Institut de Recherche pour le Développement (IRD), Montpellier, France

Benoit Gaudou
UMR 5505 IRIT, CNRS, Université de Toulouse, Toulouse, France

Ho The Nhan and Ho Tuong Vinh,
UMI 209 UMMISCO, IRD, Institut de la Francophonie pour l'Informatique (IFI),
Hanoi, Vietnam

Christophe Lang and Laurent Philippe
LIFC, Univesité de Franche-Comté, 16 route de Gray, 25030 Besançon cedex, France

Fabien Michel
LIRMM, CNRS, Université Montpellier II, 161 rue Ada 34392 Montpellier Cedex 5, France

Nicolas Marilleau
UMI 209 UMMISCO, Institut de Recherche pour le Développement (IRD), Bondy, France
Contact author: e-mail: nicolas.marilleau@ird.fr

Y. Demazeau et al. (Eds.): Adv. on Prac. Appl. of Agents and Mult. Sys., AISC 88, pp. 129–134.
springerlink.com

1 MABS and HPC

In [12], Shannon identified two steps which success implicitly relies on being able to perform numerous simulation runs: (1) ***Experimentation***. *Executing the simulation to generate the desired data and to perform sensitivity analysis*; (2) ***Analysis and Interpretation***. *Drawing inferences from the data generated by the simulation runs.*

In this respect, whatever the quality of a simulation team, having (1) enough computing resources and (2) the ability of using them to produce a sufficient number of runs is a critical issue considering the success of a simulation study. Indeed, in most cases, numerous runs should be done to study the different aspects of a simulation model (robustness, sensitivity, impact of initial conditions, output statistical analysis, verification and validation, etc.).

This is especially true about the modeling and simulation of Multi-Agent Systems (MAS). Because they describe the trajectory of each agent, MAS models embed dynamics that could lead to gigantic solution spaces, even when only few parameters are used to describe the system [9]. Despite the interest of exploring MABS models through multiple simulation runs, this work is seldom done.

This exploration is even more problematic in the case of heavy simulations. In the SWORM project [2], a simulator aiming at reproducing the earthworms influence on the soil structure and the nutrient availability by simulation has been developped. Even with an optimized source code, 1 week is needed for only one simulation.

Meanwhile, High Performance Computing (HPC) is today a hot topic and many computing resources are actually available through grids or clusters of computers. So, there is obviously a major issue considering the use of HPC by MABS. Until now, MABS using HPC mainly focus on speeding or scaling up MABS relying on implementations designed so that they are already compliant with HPC (*e.g.* [1]). Still, almost all the MABS platforms are not HPC compliant. In such cases, one has to first work on how to deploy his regular MABS on a HPC architecture. This requires HPC programming skills and could thus be a hard task if planned on the fly. So, practitioners often do not even consider this opportunity and translating regular MABS into HPC compliant ones is still a major issue.

Addressing this issue, efforts have been done in the scope of the RePast platform [8]. [7] proposes a middleware that allows the distribution of RePast sequential models. [4] also uses a middleware approach together with an Aspect Oriented Programming (AOP), again in order to minimize *code intrusions* in the original model.

In this paper, we address these drawbacks by using a more general approach. Linking MABS and HPC, the idea is to work at the platform level rather than the model level. So, our proposal relies on two main features: A workflow and a HPC infrastructure supporting it. The workflow is intended to guide MABS platform developers so that they easily make their platform compliant with the proposed HPC infrastructure. The main idea is that the infrastructure will only be a means to easily deploy a lot of simulation runs over a cluster of nodes, without any prior parallelizing work. So, (1) our approach does not target a particular MABS platform, (2)

there is no overhead as it is only about deploying multiple model instances and (3) the HPC compliance work has to be done only once as it works at the platform level.

The outlines of this paper are as follows. Section 2 presents the context of our work. Then an overview of the framework is proposed in Sections 3 and 4. We illustrate our framework with the example of the SWORM simulator [2] coming from soil sciences.

2 Overview of Grid Computing Framework Dedicated to Simulation Domains

Two main approaches can be used to reduce experiment computation time to get results by taking advantage of the power of a grid or a cluster. The first one intends to *parallelize experiment plans* on a grid. The second one intends to *distribute a unique simulation* on a grid. We develop below the first approach which is up to now the only one used in the EPIS project. For a description of the second one, readers can refer to [11].

An experiment plan is composed of a set of simulations qualified by a set of valuated parameters. Parallelizing a plan aims at deploying and running its simulations on several nodes (processors). For example, if a Sworm experiment plan defines thousand simulations to be run on a grid (composed of 5 nodes), two hundred simulations could be affected to each node. Each simulation is independent of each other (*i.e.* simulations does not exchange any data). Note that, this approach is rather simple to apply. A real gain can be observed if many simulations are needed: this approach does not reduce the compute duration of one simulation, it permits only to take advantage of a grid architecture to run simultaneously several simulations.

Several works try to provide frameworks that allow to parallelize one simulation. In this context OpenMole [10] decomposes a simulator into few independent modules. The schedule of these modules is organized according to a workflow defined by a script. This script written in Groovy language is interpreted by the OpenMole framework which starts and manages the simulation. This work focuses on simulator optimization and forget user aspect. On contrary, projects such as SimExplorer [3] (an extension of OpenMole) or GPGCloud [6] take a particular attention to the user interface. These works try to develop a user-friendly graphical user interface permitting to define experiment plans and to execute them on a Grid. Nevertheless, these tools suffer of either a lack of genericity or a lack of simplicity. For example, the use of SimExplorer needs to setup the software and to have skills in Groovy. Moreover GPGCloud does not allowed to run simulators coming from another platform.

The aim of our work is to provide a platform associating genericity with simplicity. It is a user friendly portal (dedicated to non-computer scientists) allowing grid computing, especially parallel simulation running without the complexity of the grid using. The grid is also hidden behind a web access.

3 Epis Workflow

Let consider a modeler faced to a complex problem, such as the one tackled by the Sworm model. He will thus implement his own simulator or reuse the existing Sworm, depending on his needs, on his own computer. To have significant results, he wants to play many and many experiments of the model. Due to the complexity of the studied system, the simulator needs a huge computer power to give intended results (*e.g.* 1 week is needed to play the simple Sworm experiment).

The modeler thus wants to get benefits from the computational power of the cluster to explore the influence of the simulation parameters. He connects to the EPIS portal through its preferred web browser. Then, he selects a simulator (*e.g.* Sworm) in a list or upload a new one, and creates a new experiment.

After creating the experiment, the modeler has to specify his experiment plan by defining the ranges of parameters, constraints, outputs and so on. In order to make the configuration easier for any researcher, we have developed a user-friendly web interface dedicated to drive the modeler through the definition of an experiment plan. Figure 1 summarizes the data exchanges. The interface allows the modeler to determine the variation domain of each parameter of the simulator and which simulation outputs he wants to observe. An illustration of that is, for the Sworm simulator, to identify a fixed population of agents representing earthworms and the range of soil parameter values that determines different kinds of soil. This kind of experiment plans can exhibit the impact of soil structure on earthworm dynamics and behavior.

Once the user has clicked on the "send" button, the interface produces, from the modeler's experiment plan, an XML file describing all the simulations that must be launched. Then the web server calls a script (in an EJB component on the application server) to transform this XML file into an SGE file (file that can be executed on the cluster). These two files are then sent via an SSH tunnel to the cluster and the jobs are launched. Once the jobs are performed, the output XML files are stored on the application server and the web server allows the user to download the results.

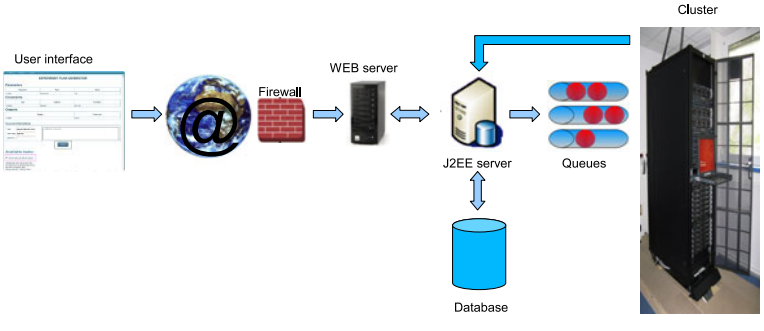

Fig. 1 Workflow of the EPIS project

4 Framework Architecture

As shown on Figure 1, the proposed framework is based on standard components: web server, application server, database, and queue manager for the cluster.

As usual, the web server (a Tomcat server[1]) is in charge of the presentation part of the framework. The presentation part covers all the web pages needed to download the model, to give the set of studied parameters, to start the simulation and to return the results to the user. The download pages are generic to all simulators.

The application server (a Jonas server[2]) is the core of the framework. It is in charge of uploading the data from the web server, of recording them in the database, of starting the simulations on the cluster, of gathering the results from the simulator runs and of presenting them to the user. Aside from the web server, the application also provides an access for heavy clients, *i.e.* with a web service interface.

The database contains all the data needed for the simulation runs: the model (or the simulator if the platform is not supported by the portal), the description files, the parameter files and the resulting data. Conceptually a model is stored with its name, description, identifying number, a number identifying its simulation platform, the set of its parameters and the set of its outputs (practically these two sets are stored in their own tables).

The cluster part is in charge of submitting the runs to the queue manager. Most of the exchanges between the application server and the cluster part are done remotely, from the application server, by using remote commands as *scp* or *ssh*. As several runs will be issued from one parameter set, a global management of the job submissions is done by the SGE *Sun Grid Engine* management system. SGE prevents the cluster overhead by limiting ressource using

5 Conclusion

The EPIS framework is an interesting solution giving a simple access to high performance computing. It allow scientists to get earlier their results or to play bigger experiments.

Without prior knowledge in computer sciences, users are able to determine experiment plans, and to play it on a grid or a cluster. They are guided by a web interface that handles a simple workflow (from the simulator upload to the parallel running of simulations).

The presented platform is based on technologies coming from Web and distributed systems. The framework is based on a J2EE application server and a SGE queue manager for the cluster. The J2EE application server aims at managing web portal and controlling the cluster and jobs (simulation) pushed in cluster queue.

We propose a modular and extensible architecture permitting, up to now, to: (i) select a simulator; (ii) create an experiment plans; (iii) start a huge number of simulations on a cluster, (iv) upload and set up new simulators. Tomorrow, this architecture will be improved by adding new modules dedicated to, for example,

[1] `http://tomcat.apache.org/`

[2] `http://wiki.jonas.ow2.org`

experiments result analysis. In addition, users are able to deploy and try themselves simulators they have developed. So, more simulators will be plugged in the portal and offered to scientific communities.

But the major contribution of the EPIS project (maybe done during its second year) will be in the domain of agent-based distributed simulation. Up to now, we focus on the experiment plan parallelizing. Another interesting way intends to distribute a single simulation over several nodes of a cluster. A such simulation needs to propose specific simulator architecture and algorithms ensuring the clock synchronizing, the MAS environment coherency, and so on. Some results have been proposed in for example [5] and [11]. But, this way must be investigated further.

References

1. Aaby, B.G., Perumalla, K.S., Seal, S.K.: Efficient simulation of agent-based models on multi-gpu and multi-core clusters. In: Simutools 2010: Proceedings of the 3nd International Conference on Simulation Tools and Techniques/ OMNeT++ 2010 Workshop (2010)
2. Blanchart, E., Marilleau, N., Chotte, J., Drogoul, A., Perrier, E., Cambier, C.: SWORM: an agent-based model to simulate the effect of earthworms on soil structure. European Journal of Soil Science 60(1), 13–21 (2009)
3. Chuffart, F., Dumoulin, N., Faure, T., Deffuant, G.: Simexplorer: Programming experimental designs on models and managing quality of modelling process. International Journal of Agricultural and Environmental Information Systems (IJAEIS) 1, 55–68 (2010)
4. Cicirelli, F., Furfaro, A., Giordano, A., Nigro, L.: Distributed simulation of repast models over hla/actors. In: Turner, S.J., Roberts, D., Cai, W., El-Saddik, A. (eds.) 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, Singapore, October 25-28, pp. 184–191. IEEE Computer Society Press, Los Alamitos (2009)
5. Hassoumi, I., Marilleau, N., Lang, C.: Mise en place et évaluation d'un algorithme de répartition de charge pour les plateformes de simulations distribuées basées sur les systèmes multi-agents. In: JFSMA: Défis Sociétaux, Madhia, Tunisie, pp. 85–94 (2010)
6. Kato, Y., Yamaki, H., Asai, Y.: GPGCloud: Model sharing and execution environment service for simulation of international politics and economics. In: Yang, J.-J., Yokoo, M., Ito, T., Jin, Z., Scerri, P. (eds.) PRIMA 2009. LNCS, vol. 5925, pp. 616–623. Springer, Heidelberg (2009)
7. Minson, R., Theodoropoulos, G.K.: Distributing repast agent-based simulations with hla. Concurrency and Computation: Practice and Experience 20(10), 1225–1256 (2008)
8. North, M.J., Tatara, E., Collier, N., Ozik, J.: Visual agent-based model development with Repast Simphony. In: Agent 2007 Conference on Complex Interaction and Social Emergence, pp. 173–192. Argonne National Laboratory, Argonne, IL, USA (2007)
9. Parunak, H.V.D.: Pheromones, probabilities, and multiple futures. In: Bosse, T., Jonker, C., Geller, A. (eds.) MABS 2010. LNCS, vol. 6532, pp. 44–60. Springer, Heidelberg (2011)
10. Reuillon, R., Chuffart, F., Leclaire, M., Faure, T., Dumoulin, N., Hill, D.: Declarative task delegation in OpenMOLE. In: HPCS, Caen, France, pp. 55–62 (2010)
11. Sébastien, N.: Distribution et parallelisation de simulations orientées agents. Ph.D. thesis, University of La Réunion (2009)
12. Shannon, R.E.: Introduction to the art and science of simulation. In: WSC 1998: Proc. of the 30th conference on Winter simulation, pp. 7–14. IEEE Computer Society Press, USA (1998)