Future Generation Computer Systems 29 (2013) 1981–1990

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

OpenMOLE, a workflow engine specifically tailored for the distributed exploration of simulation models

Romain Reuillon *, Mathieu Leclaire, Sebastien Rey-Coyrehourcq

Géographie-cités - UMR 8504, 13 rue du Four, 75006 Paris, France
Institut des Systémes Complexes, 57-59 rue Lhomond, 75005 Paris, France

HIGHLIGHTS

- Experiments on complex-system models imply numerous model executions.
- This difficult task has been automated through a Domain Specific Language (DSL).
- It maps design of experiments to High Performance Computing Environments (HPCE).
- User-supplied models are executed in the Cloud (HPCE are exposed as services).
- The DSL is explained through a toy example and a real-life experiment.

ARTICLE INFO

Article history:
Received 15 February 2013
Received in revised form
6 May 2013
Accepted 17 May 2013
Available online 7 June 2013

Keywords:
Model exploration
Distributed computing
Workflow
Cloud computing
Complex-systems

ABSTRACT

Complex-systems describe multiple levels of collective structure and organization. In such systems, the emergence of global behaviour from local interactions is generally studied through large scale experiments on numerical models. This analysis generates important computation loads which require the use of multi-core servers, clusters or grid computing. Dealing with such large scale executions is especially challenging for modellers who do not possess the theoretical and methodological skills required to take advantage of high performance computing environments. That is why we have designed a cloud approach for model experimentation. This approach has been implemented in OpenMOLE (Open MOdeL Experiment) as a Domain Specific Language (DSL) that leverages the naturally parallel aspect of model experiments. The OpenMOLE DSL has been designed to explore user-supplied models. It delegates transparently their numerous executions to remote execution environment. From a user perspective, those environments are viewed as services providing computing power, therefore no technical detail is ever exposed. This paper presents the OpenMOLE DSL through the example of a toy model exploration and through the automated calibration of a real-world complex-system model in the field of geography.

0. Introduction

A complex-system can be defined as a "system comprised of a great number of heterogeneous entities, among which local interactions create multiple levels of collective structure and organization" [1]. The emergence of the collective structure from numerous local interactions is generally unpredictable analytically. Therefore scientists use simulation models as a medium to study complex-systems. Like the physical system they represent, the behaviour of such models are unpredictable and counter intuitive. That is why large scale numerical experimentation is required in order to understand how patterns emerge from one scale to another.

Complex-system models are often multiscale, stochastic and individual centred. Therefore, their execution is generally computationally intensive. Furthermore, the numerical experimentation on such models might imply millions of executions [2,3]. This huge computational load can only be carried out by high performance computing environments.

Dealing with such broad computational loads is brain consuming, technically tricky, error prone and far from the specific field of expertise of modellers. Hopefully, most model exploration algorithms expose a naturally parallel aspect: the large number of independent executions of the model is with no doubt the most computationally intensive part. In this paper we describe how we leveraged this natural parallelism to design a generic formalism for distributed experimentation on complex-system models. This formalism has been implemented in a platform called OpenMOLE

* Corresponding author at: Institut des Systémes Complexes, 57-59 rue Lhomond, 75005 Paris, France. Tel.: +33 142174033.
E-mail address: romain.reuillon@iscpif.fr (R. Reuillon).

0167-739X/$ – see front matter
http://dx.doi.org/10.1016/j.future.2013.05.003

(Open MOdeL Experiment),[1] which provides a convenient way to explore home-brewed models with quickly evolving implementations using advanced design of experiments. The contributions of OpenMOLE are twofold: it exposes a language for describing reusable design of experiments for simulation models and it provides an execution platform which distributes these experiments on high performance computing environments in a transparent manner.

This paper demonstrates the central concepts of the OpenMOLE formalism. This platform is mature and used daily to explore real-life complex-system models. However for the sake of the comprehension of this paper, OpenMOLE's concepts are illustrated here by the exploration of a toy complex-system model. The first section presents the goal of the platform. Then the test model is exposed. After that, this model is explored through several numerical experiments of increasing complexity. Finally, the last section describes a real case experiment on a multi-agent geographical model.

1. Distributed model exploration

1.1. The naturally parallel aspect of model experimentation

In the physical world, experimenting on complex systems (such as: human societies, neural networks, insect swarms...) is generally impossible, unethical or very costly. That is why scientists design numerical models in order to facilitate the study of such systems. The numerical modelling of complex phenomena especially eases the experimentation required to understand how general patterns emerge from local interactions to global behaviour. The experiments are thus achieved in-silico, according to methods designed for this purpose.

Among the available methods for numerical experiment on models, one of the classics is the Design of Experiments (DoE) based on statistics [4]. DoE have been widely studied to produce sensitivity analysis on deterministic and stochastic simulations [5]. Most DoE generate a set of sampling values for the input of the model. Each of them is evaluated through one or several model executions depending on the stochastic nature of the model. In DoE, each evaluation is independent and constitutes a naturally parallel aspect.

Apart from DoE other methods have been designed to automate the model calibration phase. Indeed, during the modelling process some parameters might lack an empirical value. They are thus fixed during a calibration phase. This task is time consuming, that is why some methods have been designed to automate it. They are generally based on an optimization algorithm (like in [6]) that aims at minimizing the difference between the model behaviour and an expected behaviour. Automated calibration algorithms repeatedly evaluate the model for various sets of parameters in order to find a global minima. Evaluating several configurations in parallel speeds up the optimization process, therefore automated calibration processes can also be considered as naturally parallel.

More recently, novel methods have been developed to explore the fitness landscape, based on particle swarm optimization [7] and on genetic algorithms [8]. These methods are particularly well suited to map model dynamics. Swarm optimizations and genetic algorithms both consider a population of solutions. The evaluation of the fitness of individuals of the population can be processed in a concurrent manner. Once again, that kind of method exposes a naturally parallel aspect.

Another example of this naturally parallel aspect of model exploration methods arises from a recent work of ours. We have applied viability theory [9] to explore all possible dynamics of a model under a given set of constraints (this work is presented in [2]). In this experiment the model is evaluated millions of times. Most of these evaluations are independent of each other, exposing once again a naturally parallel aspect.

The list of methods in this section is not exhaustive, yet it shows that when dealing with model exploration many methods expose naturally parallel aspects. This natural parallelism concerns the numerous executions of the model which are required in order to understand its dynamics. This aspect is however rarely leveraged to enhance computation speed. It is even called embarrassingly parallel by some authors.

1.2. A generic platform for distributed model experimentation

Using distributed execution environment efficiently requires methodological and technical skills. People possessing those skills are generally not present in the community where model experimentation on a large scale would be required. That is why we have designed a platform that completely hides the burden of distributed computing for model exploration.

The platform called OpenMOLE is based on a workflow formalism. This formalism is very suitable for representing parallel processes. Several platforms already propose workflow engines for scientific computing (for a review on scientific workflow platforms see [10]). The most popular free and open-source ones are Kepler [11],[2] Taverna [12],[3] Triana [13],[4] Pegasus[5] [14] and P-Grade [15].[6] Unlike those platforms, the design of OpenMOLE has been focused on automating distributed experiments on complex-systems simulation models. This requirement has led to a significantly different design.

The design of OpenMOLE has been driven by the practices of complex-system modellers. For instance, quickly evolving implementations of complex-system models, in heterogeneous languages, and the need to experiment all along the modelling process made it crucial to easily embed continuously changing user software components in the platform. Other workflow platforms can generally use external calls to users-provided programs but they are not able to delegate them transparently to a remote execution environment. In OpenMOLE, we have made it easy to embed models based on diverse languages (such as all JVM languages (java, scala, groovy, clojure), NetLogo,[7] native executables (C / C + + / Python / Fortran / Scilab / Octave...), etc.) and to delegate their executions to remote execution environments.

To delegate model executions, OpenMOLE enforces a cloud approach. Following cloud concepts, it exposes remote execution environments as if they were services that provide computing power. It accesses them without exposing any technical specificities to the user. To do so, the workload is delegated transparently directly from the user computer to remote execution environments [16] following a zero-deployment approach: required software components are installed transparently and on-the-fly.

Finally, the necessity of carrying advanced design of experiments, such as genetic algorithms or iterative refinement of the exploration space, has pushed the design of the OpenMOLE towards a very flexible workflow formalism. This formalism provides processing structures which are not yet available in many other workflow platforms: cycles (loops), conditional branching, nested workflows and implicit representation of massively parallel workflows.

[1] http://www.openmole.org.

[2] http://kepler-project.org.

[3] http://www.taverna.org.uk.

[4] http://www.trianacode.org.

[5] http://pegasus.isi.edu.

[6] http://portal.p-grade.hu.

[7] http://ccl.northwestern.edu/netlogo.