Future Generation Computer Systems 53 (2015) 77–87

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Composing, execution and sharing of multiscale applications

Katarzyna Rycerz [a,*], Marian Bubak [a,b], Eryk Ciepiela [b], Daniel Hareżlak [b], Tomasz Gubała [b], Jan Meizner [b], Maciej Pawlik [b], Bartosz Wilk [b]

[a] AGH University of Science and Technology, Department of Computer Science, al. Mickiewicza 30, 30-059 Kraków, Poland

[b] AGH University of Science and Technology, ACC Cyfronet AGH, Nawojki 11, 30-950 Kraków, Poland

HIGHLIGHTS

- Environment supporting composition and sharing of multiscale applications.
- Direct execution of multiscale applications on distributed e-infrastructures.
- Support for hybrid execution on different types of e-infrastructures.
- Support for a variety of realizations of multiscale simulations in a non-invasive way.
- Validation by multiscale application developers.

ARTICLE INFO

Article history:
Received 20 May 2013
Received in revised form
27 September 2013
Accepted 2 June 2015
Available online 16 June 2015

Keywords:
Multiscale simulations
Distributed multiscale computing
Multiscale modeling
e-infrastructures
Tools and environments

ABSTRACT

This paper presents the research which led to elaboration of an environment for composing, executing and sharing multiscale applications. The resulted environment supports ability to connect software modules to form large-scale, multiscale simulations and directly execute them on distributed e-infrastructures suitable for particular application models chosen by users. It also enables hybrid execution, i.e. different parts of the same application can be executed on various types of e-infrastructures i.e. on a grid (e.g. EGI), HPC (e.g. PRACE) or on a cloud. The environment is web based and gives the user a direct access to the distributed resources from a single browser. It supports a variety of possible realizations of multiscale simulations in a unified and non-invasive way and enables storing model metadata such as scale, inputs and outputs.

The presented environment consists of an application composition tool called Multiscale Application Designer (MAD), an application module description registry MAPPER Memory (MaMe) and GridSpace (GS) supporting execution of applications on various infrastructures. We present an architecture of the current implementation along with a detailed description of the tools and their current features. Additionally, we report on validation of our tools by multiscale application developers. We compare the processes of creating and running applications with and without the tools and we present a case study based on a sample multiscale application skeleton.

1. Introduction

Multiscale applications are used in various fields of science such as biomedicine [1], material science [2] or astrophysics [3]. In this paper we focus on multiscale applications that can be described as a set of connected single-scale modules, i.e. modules that implement single-scale process models [4].

Typically, constructing multiscale applications is not trivial and requires a lot of effort on the part of application designers. Although there are many different standards, tools and repositories for describing and executing simulation models in many different fields e.g. system biology,[1] or nanotechnology,[2] they are difficult to be adapted to the requirements of any multiscale application that

* Corresponding author.
E-mail addresses: kzajac@agh.edu.pl (K. Rycerz), e.ciepiela@cyfronet.krakow.pl (E. Ciepiela), d.harezlak@cyfronet.krakow.pl (D. Hareżlak), t.gubala@cyfronet.krakow.pl (T. Gubała), j.meizner@cyfronet.krakow.pl (J. Meizner), m.pawlik@cyfronet.krakow.pl (M. Pawlik).

1 http://systems-biology.org/resources/model-repositories/.
2 https://nanohub.org/.

http://dx.doi.org/10.1016/j.future.2015.06.002
0167-739X/

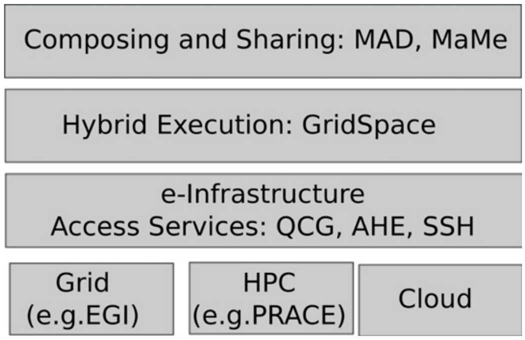

Fig. 1. Layered architectures of the proposed tools that use services to access various e-infrastructures in a hybrid way. Each module of the application can be executed on different types of infrastructures.

comprises of independent single scale models. There is an ongoing effort to systematize approaches in the field of multiscale modeling and computing. In [4] one can find a method of decomposing a multiscale phenomenon into a set of coupled single scale models located on the so-called scale separation map and defining a set of coupling templates between them; this work resulted in the definition of a language called Multiscale Modeling Language (MML) [5].

The results presented in this paper are directed towards developers of multiscale applications, in which each of the conceptual single scale models can be implemented as a software module connected to other modules in various ways to form the whole multiscale simulation. The aim of our work is to support composing and running such applications without any requirement on their realization (i.e. requirements on software used to implement modules of the application). Our approach supports storing metadata of different realizations of similar single and multiscale models, simplifies building different applications from the same modules and enables reusing and sharing of modules as well as their validation against different implementations. Moreover, in contrast to local run on scientist's computer, we aim at supporting distributed execution on several available distributed e-infrastructures as depicted in Fig. 1. We offer hybrid execution, i.e. each module of the application can be executed on different types of infrastructures. To achieve that, we offer the framework of cooperative tools for metadata storage, application composition and execution.

We evaluate and verify the presented framework by assessment of users' effort to create multiscale application manually and by using the tools. We also describe examples of tool exploitation by different applications.

This paper is organized as follows: Section 2 describes the features and requirements of multiscale applications, Section 3 overviews related work in that topic. Then, Section 4 presents analysis of support for such applications on various levels of abstraction, from description languages to distributed execution. Next Section 5 describes the process of building multiscale applications and Section 6 outlines tools supporting MML transformation into executable form, the MAPPER Memory (MaMe) registry of model metadata, the Multiscale Application Designer (MAD) and the GridSpace. Section 7 describes a sample test application, compares processes of creating multiscale applications with and without the tools and summarizes results of exploitation of existing tools. We conclude the paper in Section 8.

2. Characteristics and requirements of multiscale applications

Regarding the properties of single scale models coupling, a multiscale application can be classified as **loosely** or **tightly** coupled. In a loosely-coupled simulation there is no loop in the coupling topology of single scale models (i.e. the topology is a directed acyclic graph), whereas in a tightly-coupled simulation the coupling topology contains loops. There is also an ongoing effort to transform this information into a task graph that contains detailed information about module coupling and could, in the future, be a direct input to advanced brokering services [5]. However, currently, from a practical point of view, to avoid waiting in a queue system for each step of the coupling loop, modules implementing tightly coupled models are usually designed to be executed concurrently and exchange data among themselves during execution. Examples include in-stent restenosis which simulates complications in treatment [6], irrigation canal modeling which aggregates multiscale water models [7] and modeling of fusion processes [8] with focus on the description of core plasma in a Tokamak device.[3] For loosely coupled applications, communication modules are usually executed one after another; an example is the modeling clay–polymer interactions [2].

There is a variety of possibilities of realization of multiscale simulations. For example, tightly coupled applications, the in-stent restenosis application and hydrological irrigation canal simulation use Multiscale Coupling Library and Environment (MUSCLE) environment [9] and direct network communication for coupling their modules, while loosely coupled modeling clay–polymer interactions use specialized LAMMPS[4] and CPMD[5] scientific packages which are run one after another passing files with input and output data. These two cases are shown in Fig. 2.

Our goal was to find a way to support the variety of possible realizations of multiscale simulations in a unified way. In other words, no matter how the application was actually realized, it should be possible to describe and execute it in the same way, on various distributed resources. The method should be not-invasive i.e. it should not require changes in realization of the application (e.g. does not require to use any special communicating library etc.).

The approach towards unified way of description and execution of multiscale simulations presented in this paper allows for composability and reusability of modules implementing simulation models by meeting the following requirements:

- support for different possible application realizations,
- support for sharing of simulation modules,
- support for building different applications from a single set of modules, (needed by e.g. the irrigation canal and the fusion applications),
- the ability to switch between different versions of modules with the same functionality (needed by e.g. the application for modeling of fusion),
- support for different distributed resources, from HPC to Grids,
- support for module interactivity (i.e. user interaction during application execution).

The need for distributed execution comes from the fact that modules composing multiscale applications often have different computational requirements—some of them may be MPI parallel programs requiring computational cluster, others may perform best on symmetric multiprocessor machines or GPU. Additionally, some models may have different software requirements not available on a single site. The presented approach enables usage of different middleware services such as AHE [10] or QCG [11] to access e-infrastructures. Instead of creating our own implementations, we allow users to conveniently use middleware services with various features. QCG supports brokering, resource reservation and synchronization of execution in heterogeneous infrastructures.

3 http://www.iter.org/.
4 http://lammps.sandia.gov/.
5 http://www.cpmd.org.