Simulation Modelling Practice and Theory 58 (2015) 65–78

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

Sustainability through flexibility: Building complex simulation programs for distributed computing systems

Michael Hofmann *, Gudula Rünger

Department of Computer Science, Chemnitz University of Technology, 09111 Chemnitz, Germany

ARTICLE INFO

Article history:
Available online 11 June 2015

Keywords:
Scientific computing
Distributed simulations
Data coupling
Parallel computing

ABSTRACT

Complex simulation programs in science and engineering are often built up from a diverse set of existing applications. The large variety of application codes and their high computational demands lead to an increasing utilization of distributed computing systems. Furthermore, the need for developing sustainable simulation programs, especially with regard to ever increasing data sizes, requires a profound flexibility such that application codes and hardware resources can be easily replaced or extended. In this article, we propose a methodology for building complex simulation programs for distributed computing systems. A software library specifically designed to support a client–server-based development of simulation program components is presented. An application example for the simulation and optimization of lightweight structures in mechanical engineering is used to demonstrate the approach.

1. Introduction

Today's simulation programs in science and engineering are increasingly complex and are often built up from already existing application codes which might come from commercial or scientific sources. Since the existing application codes are diverse in many aspects, such as the mathematical simulation method, the internally and externally available data structures, the programming language and environment as well as the sequential or parallel hardware addressed, their combination includes several challenges. In addition, the cooperation of the application codes should usually be flexible in the sense that each specific scientific or engineering problem might require a different combination of these codes. Naturally, building a specific simulation program that combines different application codes leads to a programming effort which is too large a burden for a quick and productive use by the scientist interested in the specific scientific question. Furthermore, the continuing trend towards multi-simulations as well as the need for an efficient utilization of distributed computing resources and future ultrascale systems leads to great challenges for application programmers.

The goal of our work is to support the building of complex simulation programs with great flexibility to ensure a sustainable development process. We propose a component-based client–server programming model with which a coarse-grained program structure can be specified. As an application example in mechanical engineering, we consider the optimization of lightweight structures within the project MERGE.[1] This complex simulation program includes application codes for computational fluid dynamics (CFD) and finite element methods (FEM), optimization codes, and customized programs to prepare the input data for the simulations and to evaluate their results.

* Corresponding author. Tel.: +49 371 531 35571.
E-mail addresses: mhofma@cs.tu-chemnitz.de (M. Hofmann), ruenger@cs.tu-chemnitz.de (G. Rünger).

[1] MERGE Technologies for Multifunctional Lightweight Structures, http://www.tu-chemnitz.de/merge.

http://dx.doi.org/10.1016/j.simpat.2015.05.007
1569-190X/

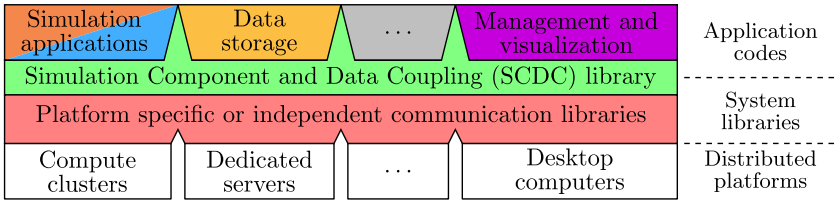

Fig. 1. Overview of the software and hardware environment for complex simulations with the SCDC library.

For each application code, there might exist different variants which should be interchangeable in the complex simulation program. An interchange might be desirable when a specific functionality is required; for example, an FEM code might be interchanged by an adaptive FEM code. The execution of the simulation program should be flexibly distributed in a hardware setting with replaceable components. That means it should be possible to utilize both single computers and distributed computing systems with sequential or parallel hardware. To support these demands, we have designed a programming model in which each application code is a component which provides its functionalities as service and accesses other components as client.

The proposed programming model requires a transformation of already existing application codes into codes that are able to act as clients and/or servers. Such a transformation can be a tedious and time-consuming work. To support the application programmer, we developed a library for Simulation Component and Data Coupling (SCDC). Fig. 1 gives an overview of the software and hardware environment with the SCDC library. The SCDC library provides functionalities to set up and access application codes as clients and servers. To support a flexibly distributed execution of the simulation components, the SCDC library encapsulates all data exchange operations. Depending on the distributed execution, these operations are mapped to appropriate data access methods, for example, through direct function calls or network communication. Since the data sizes of simulation programs can be very large, the SCDC library especially allows data exchanges without a limitation of size. Thus, our general contribution is twofold. We propose a programming model for large modular scientific simulation programs and we provide the SCDC library for the actual programming in such a model. This approach will lead to a more sustainable development process for simulation programs by reducing their need for ad hoc interfaces and solutions that are only applicable for single use cases or platforms.

The rest of this article is organized as follows. The programming model is described in Section 2 and the SCDC library in Section 3. Section 4 demonstrates the proposed development process for an application example in mechanical engineering. Section 5 discusses related work and Section 6 concludes the article.

2. Programming model for complex simulation programs

Advanced scientific or technical simulations, such as weather or atmospheric simulations or material design simulations, require complex simulation programs comprising several specific simulations as sub-simulations. Usually, these sub-simulations are developed in isolation by specialists with the appropriate knowledge about the specific mathematics or algorithmic properties. From these basic application codes, complex simulation programs are built.

2.1. Non-functional requirements for complex simulation programs

Complex simulations in science and engineering are usually very specific and assembled for a specific problem to be solved. Thus, a standard approach is that the scientist designs the entire task, identifies subtasks to be performed by existing application codes, e.g. an FEM code, plans the interactions of existing and newly programmed parts, and finds strategies for storing and exchanging the data used or created. Furthermore, the execution of the simulation program is adapted to a specific hardware platform to be used while keeping in mind the hardware requirements of the different application codes.

The development of complex simulation programs is usually done by the application programmer in several steps in a very individual and interactive way. Although, the experienced scientist can handle such a task, there are several drawbacks with the approach. First, this is a tedious and time-consuming work, which leads to a software that is only useful for a single scientist or research group. The development process usually has to be repeated for each new simulation program and if the problem to be solved or the hardware platform to be used changes significantly. Second, the development is strongly focused on functional properties to get a single simulation program work. Non-functional requirements, such as a reasonable flexibility of the simulation program, are usually neglected.

The intention of our work is to provide a methodology for a sustainable development of complex simulation programs. To achieve this goal, the simulation code created has to be flexible in the sense that the scientist gets all the variations he is otherwise used to implement in a manual and time-consuming way. These variations include flexible combinations of basic application codes, exchanges of basic application codes with similar functionality, diverse ways of storing data, and adaptations to and utilizations of different hardware platforms. Thus, it will be necessary to support a flexible combination of simulation components and a flexibly distributed execution on distributed platforms. The development should be based on a