Journal of Computational Science 14 (2016) 15–27

Contents lists available at ScienceDirect

Journal of Computational Science

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

Alya: Multiphysics engineering simulation toward exascale

Mariano Vázquez[a,b,*], Guillaume Houzeaux[a], Seid Koric[c], Antoni Artigues[a], Jazmin Aguado-Sierra[a], Ruth Arís[a], Daniel Mira[a], Hadrien Calmet[a], Fernando Cucchietti[a], Herbert Owen[a], Ahmed Taha[c], Evan Dering Burness[c], José María Cela[a], Mateo Valero[a]

[a] Barcelona Supercomputing Center BSC-CNS, Campus Nord UPC, Barcelona, Spain
[b] IIIA-CSIC, Bellaterra, Spain
[c] National Center for Supercomputing Applications-NCSA, University of Illinois at Urbana–Champaign, USA

ARTICLE INFO

Article history:
Received 26 May 2015
Received in revised form 3 November 2015
Accepted 12 December 2015
Available online 1 February 2016

Keywords:
Multi-physics coupling
Parallelization
Computational mechanics

ABSTRACT

Alya is a multi-physics simulation code developed at Barcelona Supercomputing Center (BSC). From its inception Alya code is designed using advanced High Performance Computing programming techniques to solve coupled problems on supercomputers efficiently. The target domain is engineering, with all its particular features: complex geometries and unstructured meshes, coupled multi-physics with exotic coupling schemes and physical models, ill-posed problems, flexibility needs for rapidly including new models, etc. Since its beginnings in 2004, Alya has scaled well in an increasing number of processors when solving single-physics problems such as fluid mechanics, solid mechanics, acoustics, etc. Over time, we have made a concerted effort to maintain and even improve scalability for multi-physics problems. This poses challenges on multiple fronts, including: numerical models, parallel implementation, physical coupling models, algorithms and solution schemes, meshing process, etc. In this paper, we introduce Alya's main features and focus particularly on its solvers. We present Alya's performance up to 100.000 processors in Blue Waters, the NCSA supercomputer with selected multi-physics tests that are representative of the engineering world. The tests are incompressible flow in a human respiratory system, low Mach combustion problem in a kiln furnace, and coupled electro-mechanical contraction of the heart. We show scalability plots for all cases and discuss all aspects of such simulations, including solver convergence.

1. Introduction

Across a range of engineering fields, the use of computational models is pervasive in the whole design and manufacturing process. In complex systems, High Performance Computing (HPC) plays an essential role in simulation and modeling. Researchers and manufacturing teams depend on HPC to create safe cars and energy-efficient aircraft as well as effective communication systems and efficient supply chain models. Availability of advanced HPC technologies has also fundamentally altered the investigative paradigm in the field of biomechanics. But paradoxically, for many engineers and researchers, the existing hardware and software cannot be used to solve their problems. There are many reasons why this happens, but we focus here in only two. On one hand, current HPC systems lack the computational power, network bandwidth and data storage needed for solving tomorrow's real-world engineering challenges. On the other hand, while emerging peta-scale computing is already a strategic enabler of large-scale simulations in many scientific areas (such as astronomy, biology and chemistry), even the most powerful hardware will fail to deliver on its full potential unless matched with simulation software designed specifically for such environments.

Several papers describe the effort of performing large-scale simulations on supercomputers, covering key areas: molecular dynamics [26], mantle convection in solid earth dynamics [3], massive N-body simulations [36], seismic wave propagation [25], weather prediction [1] or fundamentals of turbulence on channels using the vortex method [37]. A similar list can be obtained from the 2014 ACM Gordon Bell Prize in High Performance Computing finalists. All these papers address very specific problems using particular algorithms adapted to both the problem and the architecture. The majority of these codes use explicit time schemes, while a few of them use implicit ones with optimal (multigrid) solvers. In most of these papers massive parallelism is achieved for

* Corresponding author at: Barcelona Supercomputing Center BSC-CNS, Campus Nord UPC, Barcelona, Spain.
E-mail addresses: mariano.vazquez@bsc.es (M. Vázquez), koric@illinois.edu (S. Koric).

http://dx.doi.org/10.1016/j.jocs.2015.12.007
1877-7503/

single-physics problems, in relatively simple geometries and on Cartesian meshes.

In this paper, we present three engineering-like cases of very different character: incompressible flow in the respiratory system, the turbulent reactive flow in a rotary kiln and non-linear solid mechanics coupled with electro-physiology of the human heart. In all cases, we use the same simulation tool: Alya, the BSC's in-house software. This paper describes the solution strategy and shows the performance on supercomputers up to 100 K processors.

As a matter of fact, most references mentioned above show code performance in a larger core counts. So in order to fairly measure the novelty of this work, we must consider the following:

- The three examples presented here show many of the potentially complex aspects of a real-world engineering problem.
- Two of these problems are real multi-physics one, such as combustion or cardiac electro-mechanics.
- All of them involve both implicit and explicit schemes and we analyze the solver convergence properties all the way up to 100.000 processors.
- Two of them have complex geometries: the respiratory system and a complete cardiac bi-ventricular geometry.
- Meshes are unstructured, including different element types: prisms in boundary layers, tetrahedral, etc.
- Meshes are very large, up to more than four billion elements (4×10^9). We also describe our strategy for producing such meshes.
- Alya's algebraic solvers are programmed in-house, with no external libraries, seamlessly integrated to the solution strategies.
- The problems involve fluid mechanics, non-linear solid mechanics, excitable media, species and chemical reactions.
- All problems are run on the same supercomputer, which is made of general purpose hardware.

As a matter of fact and until now, **engineering**, **one-hundred thousand cores** and **supercomputer** were three concepts hardly found in the same sentence, unless in a negative connotation. This paper represents an effort to turn the negative sense into a positive one, leading the way toward Exascale computing in multiphysics engineering simulations.

This paper addresses the performance of Alya on supercomputers, running on up to 100.000 processors in Blue Waters, the sustained peta-scale system [31] hosted at the University of Illinois' National Center for Supercomputing Applications (NCSA). Blue Waters consists of traditional Cray XE6 compute nodes and accelerated XK7 compute nodes in a single "Gemini" interconnect fabric. Only XE6 nodes where used for this work, with each node containing two AMD "Interlagos" processors and a total of 16 floating point cores (NCSA, USA). Performance is measured through scalability when simulating coupled multi-physics problems in complex geometries coming from different domains.

2. Alya general view

Alya (see for instance [5,12,15,17,30]) is a simulation code developed at Barcelona Supercomputing Center (BSC-CNS) since 2004, whose main architects are authors GH and MV. Alya is not a born-sequential simulation code which was parallelized afterwards. Instead, it was designed from scratch as a multi-physics parallel code. It's main features are the following:

- It solves discretized partial differential equations (PDEs), preferring variational methods (particularly Finite Elements).
- Space discretisation is based on unstructured meshes, with several types of elements, such as hexaedra, tetraedra, prisms, pyramids... linear, quadratic. etc.
- Both explicit and implicit time advance schemes are programmed.
- Depending on the case, staggered or monolithic schemes are programmed. However, staggered schemes with coupling iterations are preferred for large multi-physics problems.
- Parallelization is based on mesh partitioning (for instance using Metis [24]) and MPI tasks, which is specially well-suited for distributed memory machines. On top of that, some heavy weight loops are parallelized using OpenMP threads. Both layers can be used at the same time in a hybrid scheme.
- Alya sparse linear algebra solvers are specifically developed, with a tight integration with the overall parallelization scheme. There are no third-parties solver libraries required.
- Alya includes some geometrical tools which operate on the meshes for smoothing, domain decomposition or mesh subdivision. In particularly, the latter is a key tool for large-scale simulations [13].

Alya is organized in a modular way: *kernel*, *services* and *modules*, which can be separately compiled and linked. Each *module* represents a single set of Partial Differential Equations (PDE) for a given physical model. To solve a coupled multi-physics problem, all the required modules must be active and interacting following a well-defined workflow. Alya's *kernel* controls the run: it contains the solvers, the input-output workflow and everything related to the mesh and geometry. The kernel and the modules enable a given Physical problem to be completely solved The *services* are supplementary tools, notably the parallelization service or the HDF5 writer. Kernel, modules and services have well-defined interfaces and connection points.

2.1. Computational mechanics equations

Let us establish the theoretical setup and very briefly summarize the transition from continuous problem formulation to a discrete one in Alya. Generally speaking, Alya deals with Computational Mechanics problems that can be modeled through conservation laws expressed as a set of partial differential equations:

$$\partial_t^* \Phi^\alpha = \partial_{x_i} F_i^\alpha = \partial_{x_i} C_i^\alpha + \partial_{x_i} K_i^\alpha$$

where Φ^α labels the unknown for the equation α of the set. F_i^α is the compact notation of the fluxes for each of the equations, being divided in two terms, C_i^α and K_i^α, for convenience. Note that the temporal derivative $\partial_t^* \Phi^\alpha$ is starred to note that it can be of first (like in fluid flows or excitable media) or second order (like in solid mechanics or unsteady turbulent flows). Latin subindices label the space dimensions of domain Ω. To these equations, boundary and initial conditions must be added depending on the problem under study.

The variational form is obtained by projection on a space W with its usual properties, where $\Psi \in W$ is the test function or the characteristic function depending on the context (finite elements, finite volumes, collocation, etc.). After integrating-by-parts some of the fluxes (say the K ones), we obtain

$$\partial_t^* \int \Psi^\alpha \Phi^\alpha d\Omega = \int \Psi^\alpha \partial_{x_i} C_i^\alpha d\Omega - \int \partial_{x_i} \Psi^\alpha K_i^\alpha d\Omega + \int \Psi^\alpha K_i^\alpha n_i d\partial\Omega. \tag{2.1}$$

Boundary conditions on the fluxes K are imposed through the last term, being $\partial\Omega$ the domain boundary and n_i its exterior normal vector. The interpolation space where the variational form solution