MPhyScas-P - Multi-Physics and Multi-Scales Solver Framework: Parallel Simulators

Félix Christian Guimarães Santos, Eduardo Roberto Rodrigues de Brito Junior and José Maria Bezerra da Silva

Federal University of Pernambuco

Abstract. MPhyScas (Multi-Physics and Multi-Scale Solver Environment) is a computational system aimed at supporting the automatic development of simulators for coupled problems. In despite of its completeness in what regards all stages of a multi-physics simulation, the current version of MPhyScas supports the development of sequential simulators only. Thus, The simulators it develops do not support any kind of communication between their computational entities besides those defined by direct references (pointers). In this work we present an improvement of MPhyScas, which is able of developing parallel simulators as well as sequential simulators. The basic modifications were placed on the framework of the simulator and in the pre-processor level. In this development we took an advantage of the architecture in layers of MPhyScas-S (the sequential framework) in order to define a hierarchical parallel computation scheme in such a way that the need for communication between processes is automatically identified and objects are built in order to fulfill that need. Also, the hierarchy provides a natural way of defining data structures and their access dynamics for all memory levels, providing simpler ways of dealing with non-uniform memory access patterns. This ideas are presented in the form of a new simulator framework called MPhyScas-P.

Keywords: Finite Element Method, Coupled Phenomena, Development of Simulators
PACS: 46.15.x, 07.05.Tp

INTRODUCTION

MPhyScas uses the definition of a simulator framework as guidance to instantiate an entire simulator. The original MPhyScas [1] provides support for the automatic building of sequential simulators only (also affected by limitations of the framework MPhyScas-S). The new version of MPhyScas should satisfy a number of new requirements, including the support for the development of simulators for execution in clusters of PC's. Therefore, it needs a new simulator framework. The old simulator framework, MPhyScas-S, is a framework with the support of an extended finite element library and a knowledge management system. The new framework, MPhyScas-P, uses the same extended library and knowledge management system from MPhyScas-S (with minor differences). It also makes use of the concept of layers already used in MPhyScas-S [1] in order to define a hierarchical parallel structure. Such a hierarchy is useful for the automatic definition of synchronization schemes; data partition and distribution procedures; inter-process communication patterns and data management across several levels of memory. Procedures are automatically specialized for the pre-processing; the simulation and the post-processing phases, depending on the hierarchical distribution and on the characteristics of the hardware being used. Also, two types of communication between processes during a simulation are identified and patterns are defined for their representation.

RELATED WORK

Definition and building of computational frameworks that support programming of simulators for multi-physics problems have been a very active area in the last decade. For the sake of providing a context for the present work, we classify current research efforts into only two classes: (i) libraries, which, besides providing important abstractions for supporting data, procedures and relationships for coupled physics simulation, do not provide a deep structural guidance (framework) for the building of simulators, and (ii) frameworks, which provide a configurable deep structure of abstractions and patterns for the definition of simulators and their data and procedures. Important works of class (i) are the Component Template Library (CTL) [2] and the Comsol [3]. CTL provide abstractions that support the building of solutions algorithms for loose and tight coupling. Comsol is a commercial package and not much of its internal behavior is exposed, but, as CTL, it does not provide a framework.

CP1148, Vol. 2, *Computational Methods in Science and Engineering, Advances in Computational Science*
edited by T. E. Simos and G. Maroulis

In the class of frameworks (class (ii)), one can find powerful packages, such as Uintah [4], Cactus [5] and Sierra [6]. Uintah Computational Framework and Cactus Framework consist of a set of software components and libraries that facilitate the solution of Partial Differential Equations (PDEs) on Structured AMR (SAMR) grids using hundreds to thousands of processors. Although they do not provide a structure for simulators as done by Sierra and MPhyScas, they are in this class due to how they bind components together; define and use coupling information and provide access to high performance (e.g. parallel) processing through a shared service infrastructure. That characterize them as having a configurable deep interoperability system. We detail the structure of Sierra a little bit more, because its structure resembles MPhyScas' structure.

Sierra framework provides a structural guidance in layers composed of (from top to botom) **Application**, **Procedure**, **Region** and **Mechanics**. Application articulates user-provided algorithms in order to establish high-level activities. It uses services from a set of Procedures, which can freely articulate Regions using a set of user-provided algorithms. A Region defines activities, which are related to a fixed geometric region, for which a mesh is provided. Those activities are defined by user-provided algorithms. It uses services provided by a set of Mechanics. In order to perform the desired work, a Mechanics uses a set of MechanicsInstances and a set of algorithms. A Mechanics implements procedures related to a specific physics - defined on a subset of its Region's mesh - and its MechanicsInstances are responsible for an atomistic operation defined on a subset of its Mechanics' mesh. A Mechanics may use another set of Mechanics, building one more layer. This may be used in multiscale computations, where a lower level Mechanics is used to compute constitutive data for a MechanicsInstance of a its parent [6]. If a Mechanics A needs data from another Mechanics B (provably in another Region), the core services of Sierra provides means to transfer mesh-dependent data from B's mesh to A's mesh. The result is then stored in the Region of Mechanics A. The SIERRA Framework core services also manage the parallel distribution of mesh objects for an application.

THE FRAMEWORK MPHYSCAS-P

In modern clusters of PC's one can identify at least four hierarchical levels of different procedures and/or memory usage: (i) **Cluster Level**: it is composed of all processes running in all machines being used in a simulation; (ii) **Machine Level**: it is composed of all processes running in one individual machine among all those used in a simulation; (iii) **Processor Level**: it is composed of all processes running in one individual processor among all those running in one individual machine; (iv) **Process Level**: it is composed of one single process running in one individual processor among all other processes in this same processor. This last level can be further divided into two sub-levels: (iv.i) **Core Sub-level**: it is composed of all parts of the code from one individual process, which is not strongly hardware specific; (iv.ii) **Software-Hardware (SH) Sub-level**: it is composed of all parts of the code from one individual process, which is strongly hardware specific (cache management, fpga or GPU acceleration, etc.)

Now, we explain the structure and behavior of MPhyScas-P framework. There are two views of the MPhyScas-P: (i) **Logical View**: the logic of MPhyScas-P's workflow is the same as the MPhyScas-S', that is, it has the same levels of procedures (Kernel, Blocks, Groups and Phenomena), all relationships between them are preserved and all procedures within each layer are technically the same (besides the fact that data are now distributed). Therefore the relationships among entities in the different levels of MPhyScas-P is also a DAG (direct acyclic graph). Thus, we are able of cloning a suitable modification of MPhyScas-S to all processes in a SPMD scheme. In this sense, one can imagine that MPhyScas-P is MPhyScas-S with distributed data and a hierarchical synchronization scheme (see Topological View below); (ii) **Topological View**: the topology of the procedures in the workflow of MPhyScas-S is implemented in MPhyScas-P in a hierarchical form with the aid of a set of processes, which are responsible for the synchronization of some processes. We identify three types of processes (see Figure 1): (ii.i) **ClusterRank Process**: it is responsible for the execution of the **Kernel** and to synchronize the beginning and the end of each one of its level's tasks, which requires demands to lower level processes. In a simulation there is only one ClusterRank process (for instance, the process with rank equal to zero in an MPI based system). Figure 2 depicts the relationship between a ClusterRank process and the simulator layers; (ii.ii) **MachineRank Processes**: one process is chosen among all processes running in an individual machine (a cluster node) to be its leader. Thus, there is only one MachineRank process per machine. It is responsible for the execution of procedures in the **Block** level and to synchronize the beginning and the end of each one of its level's tasks, which requires demands to lower level processes. ClusterRank is also the MachineRank in its own machine. Figure 3 depicts the relationship between a MachineRank process and the simulator layers; (ii.iii) **ProcessRank Processes**: it is responsible for the execution of the procedures in the **Group** level. The ClusterRank and all MachineRank processes are also ProcessRank processes. Figure 4 depicts the relationship between a ProcessRank process and the simulator layers.

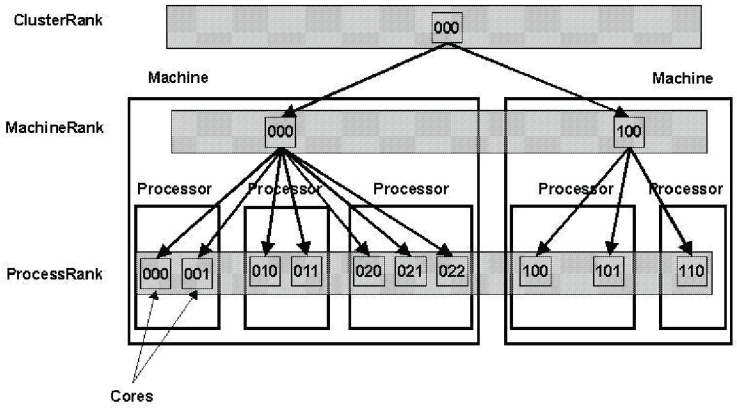

FIGURE 1. Hierarchy of the simulator in MPhyScas-P

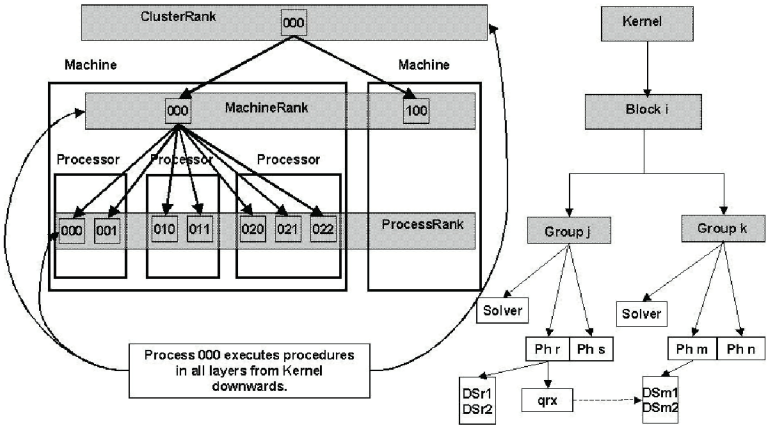

FIGURE 2. Layers with procedures executed by clusterRank in MPhyScas-P

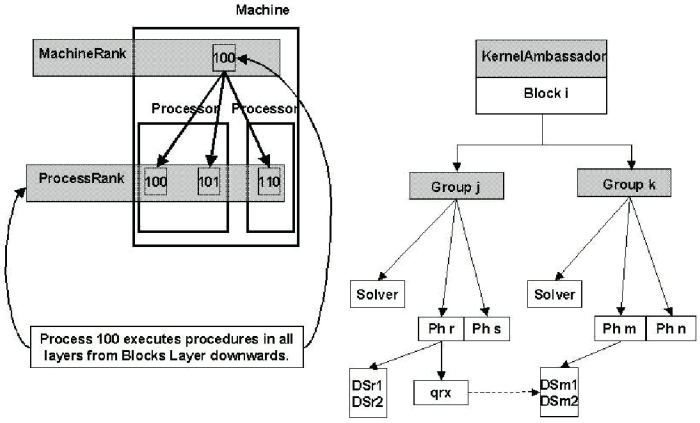

FIGURE 3. Layers with procedures executed by machnineRank processes in MPhyScas-P

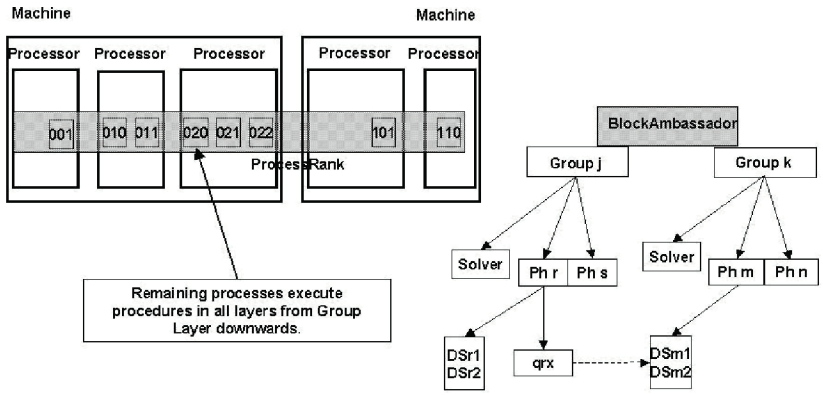

FIGURE 4. Layers with procedures executed by processRank processes in MPhyScas-P

The framework MPhyScas-S [1] has satisfactorily solved the main problems related to data dependence and sharing between phenomena for the sequential processing. However, an essential difference, when considering parallel processing, is the need for interprocess communication and synchronization. Communication can be of three generic types: (a) **Communication during linear algebra operations**: Code parallelization implies that interprocess communication is needed during several linear algebra operations (e. g., matrix-vector multiplication); (b) **Communication along process hierarchy**: The establishment of the hierarchy of procedures will require some processes to assume some kind of leadership depending on the layer where they are located. This will require interprocess communication throughout the hierarchy; (c) **Communication for coupled information**: MPhyScas-S has dealt with coupling dependences between different phenomena already, but in parallel processing this type of dependence can become more complex. Whenever a coupled quantity had to be computed by one given Phenomenon object and the coupled information (information from other phenomenon) is not available in the current process, interprocess communication should take place. Such an information frequently depends on the place where the quantity is being computed.

Those three types of communication are implemented (using groups of processes and threads) by two patterns: (i) Control_Comm pattern: one group of processes involving the ClusterRank - as many groups of processes as the number of MachineRanks - relating them with their respective ProcessRank processes; and (ii) DataTransfer_Comm pattern: takes care of any geometry dependent information between two processes (types (a) and (c)). Those patterns are automatically instantiated and localized in the pre-processing phase.

CONCLUSIONS

We presented the framework MPhyScas-P aimed at supporting the automatic development of high performance simulators for multi-physics problems. This framework inherits from MPhyScas-S (its sequential version) all its workflow representation, with the obvious difference that MPhyScas-P is distributed in a hierarchical way. Although MPhyScas-S has already a fully functional prototype, MPhyScas-P has a prototype (using MPI) currently being tested.

REFERENCES

1. F. Santos, E. J. Brito, J. M. Barbosa, *Coping with data dependence and sharing in the simulation of coupled phenomena*, International Congress on Computational and Applied Mathematics, Leuven, Belgium, 2006.
2. R. Niekamp, *Software component architecture*, http://congress.cimne.upc.es/cfsi/frontal/doc/ppt/11.pdf.
3. http://www.comsol.com/.
4. S. G. Parker, *A component-based architecture for parallel multi-physics pde simulation*, Future Generation Computer Systems, (22): 204216, 2006.
5. A. G. Lanfermann, et al. *The Cactus framework and toolkit: Design and applications*, Vector and Parallel Processing, 2002, 5th International Conference, Springer, 2003.
6. H. C. Edwards, *Sierra framework version 3: Core services theory an design*, Sandia National Laboratory, report SAND2002-3616, November 2002.