A Framework for Coupled Multi-physics Simulations ***Jeff: some charming title?? emphasizing "it can cover various kinds of applications in different requirement, main components can be replaced by other similar softwares, etc."

Soon-Heum Ko[1,2], Nayong Kim[2], Shantenu Jha[3,2]

[1]*National Supercomputing Centre, Linöping University, Linöping, Sweden*

[2]*Center for Computation & Technology, Louisiana State University, USA*

[3]*Dept. of Elec. and Comp. Eng., Rutgers University, Piscataway, New Jersey, USA*

Abstract

we design and develop a multi-physics framework for coupled simulations in which scientific components (codes) are logically separated. Designed framework provides the interface between scientific components, (compilation system for system architectures,) a runtime environment for scheduling of coupled tasks, and data management/code versioning support. The framework is built by developing the data exchange interface between coupled codes, (providing the wrapping script to users' Makefiles,) adopting a BigJob abstraction, and using the PetaShare service. We apply this framework into the hybrid computational fluid dynamics - particle dynamics application to demonstrate its capability.

I. Introduction and Motivation

"Coupled scientific simulations" will refer to the scientific research procedure which involves the data exchange between different components (either in parallel or in tandem). In both cases, whether scientific tools are continuously exchanging the information during a single run or individual tools iteratively give the feedback to their counterparts, scheduling the overall procedure and providing the data stream under the distributed computing infrastructure is a headache to domain scientists. It motivates the development of a framework for coupled multi-component applications, which provides the following capabilities: (1) the framework provides the data interface between users' tools, (2) embedded softwares should be adaptive to users' implementations in diverse formulations, and (3) they should be removable/replaceable by users' preferences.

Two types of formulations can be present on a coupled multi-component framework. One is to modularize all components and bind into a single executable, and the other is to make multiple standalone softwares and provide the interface between individual executable. The former is good for scheduling on most batch queue system and effectively using allocated resources, while some restrictions on data structure or standard grammar may apply on users' codes. Also, the binary can be heavier to contain unused or non-optimized functions for specific targets. The other gives much freedom in writing the individual component but can cause computational headache in scheduling components in the remote production system. We consider that binding into the single binary is recommendable if all software packages are tightly coupled together for a single task or they share most memory allocations. It is preferred to provide the interface and the scheduling function if those packages are sequentially loaded or they have different code structures.

A runtime environment for a coupled multi-physics application [6] has been developed previously. In this work, two coupled yet logically distributed scientific softwares have been effectively scheduled under the single batch queue allocation, by the use of a BigJob [7] with the load balancing capability incorporated. It demonstrated that a Pilot-job formulation can ease the scheduling of a coupled application whose components are hardly packaged into a single binary. However this runtime environment lacks the reusability because it only provides the scheduling functionality between already-coupled distributed application codes.

In this work, we design and develop a coupled multi-component framework. Along with the runtime environment between logically separated components, we also provide the standard interface between these components and take care of the data management/versioning. The structure of a coupled simulation framework is introduced in Sec. II. Specific softwares implemented in this frame-

work are described in Sec. III. A coupled multi-physics application and an experiment-computation integrated research procedure are presented in Sec. IV. Recommendation for future work and conclusions are presented in Sec. V and Sec. VI.

II. Design of a Coupled Simulation Framework

A. Requirements

Multiple components in a coupled application is hard to be packaged into a single binary if (1) each component uses very different computational kernels or (2) a manual procedure is involved in one of components. A hybrid computational fluid dynamics (CFD) - molecular dynamics (MD) simulation is an example whose data structures are completely different so that it is fairly hard to incorporate into a single binary. Another situation is occasionally observed when there is an iterative feedback between a numerical simulation and the experimental measurement. Most probably, the physical experiment involves the human labor of changing specimen/mockup after the numerical investigation, which cannot be digitalized. Figure 1 describes the procedure for these two multi-component simulations.

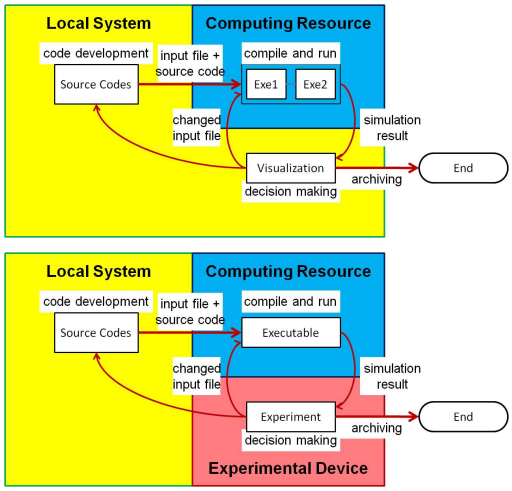

Fig. 1. Research procedure of (1) coupled multi-physics simulations and (2) successive feedback between computations and experiments. In both cases, frequent data transfer takes place between local and remote systems and a manual decision making intervenes (either by visualizing the numerical solution or by performing the experimental measurement). In addition, the framework should support the communication interface and concurrent execution in case of computationally coupling distinct scientific softwares.

A coupled multi-component simulation framework should be capable of scheduling the overall procedure (workflow) and a coupled simulation between multiple tasks (runtime environment). The workflow shall run software components according to the pre-described schedule, which includes scientific softwares and middleware packages. A runtime environment takes the role of running coupled multiple softwares in parallel, in the form of a virtually single executable under the batch queue system.

The functionality to handle the data transfer/exchange provides more convenience for performing the coupled simulation and building coupled software packages. Automated data transferring eases the successive feedback between distributed tools, and also provides the basic-level versioning service of source codes into the remote computing machines. The coupling interface can take care of the synchronous communication between concurrently running softwares.

B. Design

We design the concrete structure of a coupled multi-component simulation framework as presented in Fig. 2. Looking at the default operation flow, the updated source code is copied to the remote computing resource and the individual source code is linked with the coupling interface during the compilation. These distinct yet coupled executables run concurrently by the help of a runtime environment. The solution is referenced by a third-party and the decision is made to accept/decline the numerical solution (either through the visualization or the experimental measurement). The acceptable solution is archived in the local server, or the new computation is performed using different parameter/modified source codes. In case the computation and the experimental measurement are coupled in sequence, compilation and job submission processes become simpler.

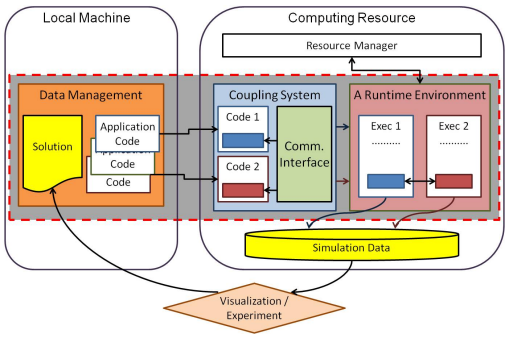

Fig. 2. Structure of a coupled multi-component application framework. The framework is composed of three main packages: a data archiving/versioning service, a coupler and a runtime environment. A workflow script is running these packages according to the coupled simulation procedure.

Data management service provides the data sharing

among distributed computing facilities, updating the source codes between local and remote machines, and archiving final solutions. These functions can be implemented by running a couple of secure FTP commands; Installing a versioning service such as CVS (Concurrent Versioning System: [4]), SVN (Apache Subversion: [2]), or GIT (The fast version control system: [9]) on a central repository server is another way; Using a data Grid service such as PetaShare [1] can be helpful if sharing large-scale data on a distributed environment is of particular interest.

Coupling system provides the data exchange between distinct application codes and helps the compiling procedure. The coupling interface can be supported either in the form of a library which opens a file-based channel between distributed executables, or as a separate binary which binds all MPI communicators under a single MPI runtime environment. In either case, the interface function calls should be deployed in coupled source codes. Compilation system handles all compilations including individual source codes and the coupler. It wraps up compiling options on users' makefiles to link the coupling library with the executable and to apply optimal options for specific systems. A single wrap-up script can provide these capabilities up to some extent; Some compilation systems such as the Simulation Factory [10] can provide more generality independent of different system architectures and configurations.

A runtime environment is designed to co-schedule and load-balance between coupled distinct tasks. The above issue is easily resolved by allocating a pilot-job in which coupled subtasks are effectively scheduled. A number of pilot-job implementations such as a Condor Glidein Job [5], a BigJob [7], or a Pilot Factory [3], are present. Also, some supercomputing centers provide the capability of concurrently starting multiple jobs in the basic level.

III. Implementation

A. Data Management and Archiving

B. Coupling System

- Library form (lighter), none-resource usage

C. Runtime Environment

A BigJob is a pilot-job implementation under the simple API for grid applications (SAGA) [8]. SAGA provides a simple, POSIX-style API to the most common grid functions at a sufficiently high-level of abstraction in order to be independent of diverse and dynamic grid environments. The SAGA specication defines interfaces for the most common grid-programming functions grouped as a set of functional packages. Thanks to various adaptors in SAGA, the BigJob application is infrastructure-neutral, which is the strength over other Pilot-job implementations.

We apply the previously-established BigJob runtime environment for coupled multi-physics applications [6]. The load balancing function is turned off since it requires relevant changes (time checking function and checkpointing capability) in the application code. We directly submit a job to the remote batch queue system in case the sequential feedback process between a computation and an experiment is of interest.

D. Workflow Engine

IV. Numerical Experiments

V. Further Achievements

VI. Conclusions

Acknowledgment

References

[1] A Distributed Data Archival, Analysis and Visualization Cyberinfrastructure for Data-intensive Collaborative Research. http://www.petashare.org/.

[2] Apache Subversion. http://subversion.apache.org/.

[3] P.-H. Chiu and M. Potekhin. Pilot factory a condor-based system for scalable pilot job generation in the panda wms framework. *Journal fo Physics: Conference Series*, 219:062041:1–7, 2010.

[4] Concurrent Versioning System. http://www.cvshome.org/eng/.

[5] Condor Project. http://www.cs.wisc.edu/condor/.

[6] S.-H. Ko, N. Kim, J. Kim, A. Thota, and S. Jha. Efficient runtime environment for coupled multi-physics simulations: Dynamic resource allocation and load-balancing. In *The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010)*, 2010.

[7] A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor. Adaptive distributed replica-exchange simulations. *Philosophical Transactions of the Royal Society A: Crossing Boundaries: Computational Science, E-Science and Global E-Infrastructure Proceedings of the UK e-Science All Hands Meeting 2008*, 367:2595–2606, 2009.

[8] SAGA - A Simple API for Grid Applications. http://saga.cct.lsu.edu/.

[9] The fast version control system. http://git-scm.com/.

[10] The Simulation Factory. http://simfactory.org/.