Annals of Nuclear Energy xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Annals of Nuclear Energy

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

Physics-based multiscale coupling for full core nuclear reactor simulation

Derek R. Gaston [a,*], Cody J. Permann [a], John W. Peterson [a], Andrew E. Slaughter [a], David Andrš [a], Yaqi Wang [d], Michael P. Short [e], Danielle M. Perez [b], Michael R. Tonks [b], Javier Ortensi [d], Ling Zou [c], Richard C. Martineau [a]

[a] Modeling & Simulation, Idaho National Laboratory, P.O. Box 1625, Idaho Falls, ID 83415, United States
[b] Fuel Modeling & Simulation, Idaho National Laboratory, P.O. Box 1625, Idaho Falls, ID 83415, United States
[c] Thermal Science & Safety Analysis, Idaho National Laboratory, P.O. Box 1625, Idaho Falls, ID 83415, United States
[d] Reactor Physics Analysis and Design, Idaho National Laboratory, P.O. Box 1625, Idaho Falls, ID 83415, United States
[e] Dept. of Nuclear Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Room 24-204, Cambridge, MA 02139, United States

ARTICLE INFO

Article history:
Received in revised form 14 February 2014
Accepted 26 September 2014
Available online xxxx

Keywords:
Full core reactor simulation
Multiphysics coupling
Multiphysics

ABSTRACT

Numerical simulation of nuclear reactors is a key technology in the quest for improvements in efficiency, safety, and reliability of both existing and future reactor designs. Historically, simulation of an entire reactor was accomplished by linking together multiple existing codes that each simulated a subset of the relevant multiphysics phenomena. Recent advances in the MOOSE (Multiphysics Object Oriented Simulation Environment) framework have enabled a new approach: multiple domain-specific applications, all built on the same software framework, are efficiently linked to create a cohesive application. This is accomplished with a flexible coupling capability that allows for a variety of different data exchanges to occur simultaneously on high performance parallel computational hardware. Examples based on the KAIST-3A benchmark core, as well as a simplified Westinghouse AP-1000 configuration, demonstrate the power of this new framework for tackling—in a coupled, multiscale manner—crucial reactor phenomena such as CRUD-induced power shift and fuel shuffle.

1. Introduction

Nuclear reactors are archetypal multiscale, multiphysics systems. Atomistic reactions ultimately drive very large-scale energy generation processes through micro- and mesoscale phenomena. Across the range of scales, a myriad of physical processes impact the system: microstructural evolution of materials, nucleate boiling, plasticity, creep, conjugate heat transfer, neutronics, fluid flow, and others. Traditionally, domain-specific software has been developed to tackle individual pieces of this problem (Ransom et al., 1982; Berna et al., 1997; Lyon et al., 2004; Joo et al., 2004) and several development efforts have attempted to utilize such extant software in conjunction with data exchange to simulate entire reactors. Several related US Department of Energy sponsored programs (Weber et al., 2007; NEAMS, 2013; CASL, 2013) have met with varying levels of success in this endeavor.

These nuclear reactor simulation efforts hinge on the ability to efficiently transfer data between different pieces of software. The fact that each constituent program has different data representations, requirements, and parallel partitionings greatly complicates the overall effort. Additionally, the code for exchanging data between n programs scales as $\mathcal{O}(n^2)$, eventually impeding progress and negatively impacting correctness and maintainability. Maintenance of legacy software often occupies more time than the development of new codes and new simulation capability.

Further complications arise because of the need to orchestrate the solution processes of multiple, independently developed codes running at different time and length scales. For instance, software that simulates microstructural grain evolution within fuel and software that simulates engineering-scale fuel performance are frequently designed to run well on timescales of seconds and years, respectively. Furthermore, most existing software is not designed to be controlled or run by an external driver program. Such software may include hard-coded time-stepping routines or may abort when an error condition arises rather than gracefully allowing the error to be handled at a higher level in the software stack. Therefore, modifications to existing codes are frequently

* Corresponding author. Tel.: +1 208 526 6842; fax: +1 208 526 2930.
E-mail address: derek.gaston@inl.gov (D.R. Gaston).

http://dx.doi.org/10.1016/j.anucene.2014.09.060
0306-4549/

necessary: new software must be written to drive the solution process and keep the menagerie of independent codes in sync.

The Multiphysics Object Oriented Simulation Environment (MOOSE) developed at Idaho National Laboratory (Gaston et al., 2009) represents an alternative path toward reactor simulation. MOOSE utilizes a modular approach, allowing scientists and engineers to create new fully coupled, multiphysics applications. A number of different physics simulation capabilities have been developed based on the MOOSE framework, including geomechanics (Podgorney et al., 2010), chemical transport (Guo et al., 2013), and superconductivity (Karpeev et al., 2013) applications. Of particular interest in the present work is a set of nuclear-related applications for simulating fuel performance, neutronics, thermal–hydraulics, fuel microstructure, and the effect of CRUD (Chalk River Unidentified Deposits) on fuel performance.

Recent MOOSE framework developments have enabled the efficient combination of multiple, independently developed applications with the goal of achieving massive, multiscale calculations. These developments, which include both a flexible execution strategy and a sophisticated data exchange facility, allow MOOSE-based applications to run concurrently while exchanging data, a process we have termed "multicoupling." We believe the multicoupling technique will make an impact on a number of challenging numerical problems, both within the nuclear field and in the broader scientific community. Truly understanding these problems, which in the nuclear field include stress corrosion cracking, radiation void swelling, and irradiation creep, requires data from the atomistic length scale to be efficiently utilized on the engineering scale, as informed by the mesoscale (Short et al., 2014). The MOOSE framework, and others like it, represent a new and computationally efficient pathway to this end.

2. Methodology

The MOOSE framework was developed to simplify the creation of fully coupled, nonlinear, multiphysics applications. Here, "fully coupled" refers to solving all of the coupled partial differential equations (PDEs) simultaneously within one Newton-based solve. More than 30 MOOSE-based applications have been created under the fully coupled paradigm over the last five years. Each MOOSE-based application is made up of physics "modules" that describe the PDEs to be solved, material properties, boundary and initial conditions, postprocessed quantities, etc. Multiple MOOSE-based applications can be "composed" to create applications that comprise the physics from the constituent applications (Zhang et al., 2013a; Tonks et al., 2013a). While fully coupled multiphysics is useful for dealing with problems where the physics are strongly interacting, not all multiphysics problems are (or need to be) fully coupled. Examples include systems with multiple time scales and codes which couple with external software.

These situations can be described as "loosely coupled systems of fully coupled equations." For instance, one MOOSE-based application may be used to compute the microstructural evolution of a material, while another engineering-scale application computes its macroscopic linear elastic response. Each of those two applications is treated as a fully coupled system of nonlinear equations that can be solved independently and later exchange data. In order to enable this solve structure within MOOSE, two new class hierarchies (families of C++ objects) have been developed: `MultiApps` and `Transfers`.

A `MultiApp` object allows multiple MOOSE (or external) applications to run simultaneously in parallel. A single `MultiApp` might represent thousands of individual solves (for example, thousands of individual microstructure calculations in a multiscale simulation). Each subsidiary application (or "sub-app") within a `MultiApp` is considered to be an independent solve. There is always a "master" application at the top level and a hierarchy of `MultiApps` beneath it, as shown in Fig. 1. A sub-app can have its own `MultiApps`; indeed, arbitrarily nested levels of solves are possible. In parallel, all sub-apps are distributed across the available processors and executed simultaneously.

While a `MultiApp` allows for arbitrary levels of hierarchical solves to be computed efficiently in parallel, those solves still require the exchange of data. The `Transfer` system within MOOSE implements this exchange. Although several libraries for mapping solution fields between meshes exist (Mahadevan et al., 2013; Slattery et al., 2013), there are many other types of data that applications must send and receive in order to implement a coupled solve. Thus, there are three main categories of `Transfers` within MOOSE:

1. *Field mapping*: L_2-projection, interpolation, evaluation, etc.
2. *Postprocessed spatial data*: Layered integrals and averages, assembly-averaged data, etc.
3. *Scalar values*: Integrals, averages, point evaluations, etc.

"Field mapping" is simply taking a mesh-supported finite element solution field variable and transferring it (in some way) to another mesh. "Postprocessed spatial data" `Transfers` are designed to move spatially varying data that has been postprocessed (typically from a solution field). For example, transferring the average fuel temperature in axial slices along a fuel rod into a neutronics application (where it could be used to determine fuel temperature feedback in a cross-section calculation). "Scalar values transfers" are primarily useful for transferring data between domains of disparate physical size. An example would be computing the macroscale thermal conductivity from microstructure solves for use within a nuclear fuel performance calculation. The heat conduction solve can receive a "field" comprised of an interpolation of the thermal conductivity sampled from each of the microstructure simulations.

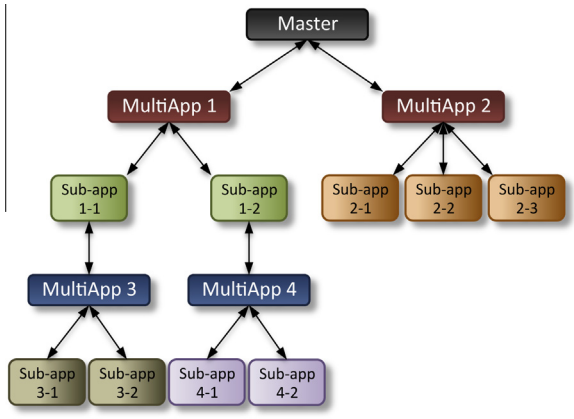

Fig. 1. General MOOSE `MultiApp` hierarchy for multicoupling applications.

Table 1
BISON fuel pellet dimensions.

Pellet quantity	350
Pellet height	9.906×10^{-3} m
Pellet outer radius	4.1275×10^{-3} m
Clad thickness	6.35×10^{-4} m
Clad gap width	8.85×10^{-5} m
Clad bottom gap height	1.0×10^{-3} m
Plenum fuel ratio	0.074541

Please cite this article in press as: Gaston, D.R., et al. Physics-based multiscale coupling for full core nuclear reactor simulation. Ann. Nucl. Energy (2014), http://dx.doi.org/10.1016/j.anucene.2014.09.060