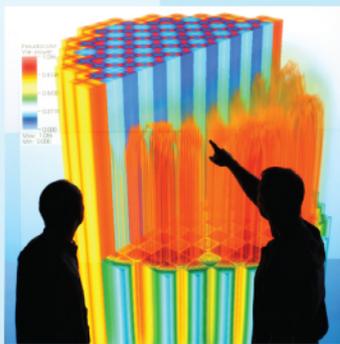


Power uprates
and plant life extension

CASL-U-2014-0013-000



Engineering design
and analysis



Evaluation of the Multiphysics Object-Oriented Software Environment (MOOSE)

Framework

Richard C. Martineau, Ph.D.

Idaho National Laboratory

Science-enabling
high performance
computing



Independent Reviewers

Forrest Brown, Los Alamos National Laboratory

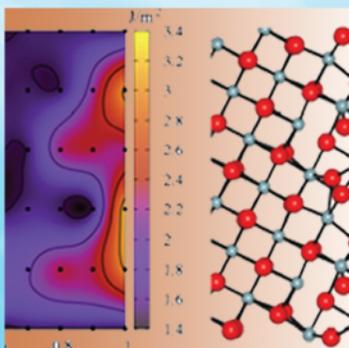
Bill Gropp, University of Illinois

Olle Heinonen, Argonne National Laboratory

Barry Smith (chair), Argonne National Laboratory

Michael Zika, Lawrence Livermore National Laboratory

Fundamental science



December 18, 2013

Plant operational data



U.S. DEPARTMENT OF

ENERGY

Nuclear Energy

Report of the MOOSE Review Committee

Convened to evaluate the Multiphysics Object-Oriented Software Environment

December 16-18, 2013

Forrest Brown, Los Alamos National Laboratory

Bill Gropp, University of Illinois

Olle Heinonen, Argonne National Laboratory

Barry Smith (chair), Argonne National Laboratory

Michael Zika, Lawrence Livermore National Laboratory

Executive Summary

This panel was asked by Kemal O. Pasamehmetoglu, associate laboratory director of Idaho National Laboratories, to provide a “deep dive” technical review of the Multiphysics Object-Oriented Software Environment (MOOSE) software package. He asked us to “assess the qualities and characteristics of MOOSE and, in particular, to determine its suitability as a robust and scalable platform for multiphysics simulation necessary to meet the DOE computational science goals.” In addition he requested an “assessment of MOOSE’s algorithmic capabilities for running well-resolved, fully coupled multiphysics simulations at scale on modern high-performance hardware” and “an assessment of the limitations associated with the MOOSE framework.” The complete charge letter is given as an appendix to this document.

To perform the analysis, we put together a team of mathematicians, computer scientists, and scientists with expertise in some of the areas in which MOOSE may be applied. The panel met with the MOOSE developers, at Argonne National Laboratory, for a two and one-half day review (December 16-18, 2013) which included presentations, discussions, and the installation of the software.

Findings

MOOSE has two major features:

- A mature, well-designed and well-implemented framework for solving multiphysics partial differential equations using finite-element-like methods with a Jacobian-free Newton solver
- A newer system for loosely coupled systems of subsystems each of which may consist of loosely coupled subsystems or may be a strongly coupled system. Although this feature is not yet mature, it appears to have high potential and has been demonstrated at scale on nontrivial coupled multiphysics simulations.

A great deal of effort has been put into the usability, testing, and documentation of the software --- and it shows. The MOOSE framework is well suited for a variety of DOE applications, is of high quality, and has been demonstrated to scale well up to 12,000 cores on nontrivial simulations.

Recommendations

The panel has five recommendations.

- Increase and diversify the MOOSE user base, which could include
 - developing an online community of users,
 - curating Elk with physics and material models, and
 - determining appropriate application areas for focus in the near term (we suggest materials science simulations as an area ripe for additional effort).
- Leverage the unique strength of solving loosely coupled systems of fully coupled subsystems by investigating other applications beyond nuclear reactors for which MOOSE can be applied.
- Increase usability by
 - providing a full users manual with technical background (possibly even a book) with index and online searchability, and
 - developing a plan for production support with regular releases.
- Develop collaborations to examine issues related to
 - the applied mathematical analysis of coupling schemes,
 - techniques for varying the timestep between different physics or scales within the same fully coupled simulation, and
 - enhancements in MOOSE to achieve vectorization and GPU utilization.
- Make the MOOSE/ELK infrastructure open source and freely available to the community.

Introduction --- General Observations

The Multiphysics Object-Oriented Software Environment (MOOSE) software package has been under development for about five years and is now being used in a variety of important nuclear energy-related applications. It is thus important to take a step back, evaluate its current form, and make any adjustments necessary to ensure its continued high quality and forward momentum.

The MOOSE development effort began with a goal of simultaneously solving all components of a multiphysics calculations, using a Jacobian-free Newton-Krylov (JFNK) scheme. The availability of such a general-purpose, easy-to-use tool --- designed to directly solve fully-coupled multiphysics calculations without operator splitting, time-step sequencing of parts of the calculations, and dealing with varied time-step controls --- represents a significant advance in capabilities. The fully coupled JFNK solution scheme often provides more accurate solutions of tightly coupled multiphysics calculations than do traditional approaches. More recently capabilities were added to MOOSE to also support loosely coupled calculations. Such calculations are often better suited to the analysis of slowly changing systems, where steady-state or quasi-static calculations are appropriate for much of the physics. The full range of MOOSE solution techniques --- fully coupled, loosely coupled, or combinations thereof --- provides a rich set of tools for covering the range of solution methods that are needed for nuclear reactor analysis and many other applications.

MOOSE leverages two large-scale mathematical software libraries: PETSc and libMesh. These two libraries are MPI-based object-oriented libraries, PETSc in C, and libMesh in C++. The MOOSE framework uses PETSc for advanced, scalable linear and nonlinear solver algorithms and uses libMesh to enable mesh support of two- and three-dimensional finite element meshes, including mesh adaptivity. As alternatives, MOOSE can use HYPRE for scalable linear solver algorithms and Trilinos for algebraic solvers. Both PETSc and libMesh are software libraries that require programming, --- that is, the writing of source code to complete an application, whereas MOOSE is a framework from which many applications can be completed without writing additional source code beyond straightforward kernel functions.

For efficient utilization of multicore computing systems, MOOSE incorporates the MPI + X programming model by providing its own abstraction of tasks which are assigned to threads. To achieve this, MOOSE provides a uniform wrapper API around OpenMP,

Pthreads, and Intel Thread Building Blocks (TBB). This is a nice design decision preventing MOOSE from being too closely locked into a single thread management model.

The combination of powerful and scalable libraries (PETSc and libMesh) in a framework that is easily expanded or reconfigured gives MOOSE the potential to enable or drive transformative changes in materials science and modeling. The MOOSE design allows for implementation of turnkey systems for analysts with rigorously controlled and validated physics models. Moreover, as a research tool, it enables scientists to try out new models and numerical methods. It also supports incremental development of multiphysics applications by adding slightly more “physics” one small step at a time, with testing and validation taking place before the next piece of “physics” is added. At the same time MOOSE has the ability to control and configure the underlying solver to a considerable extent (e.g., via PETSc options, a custom DM implementation). This affords a sufficiently knowledgeable user an opportunity to achieve high runtime performance by fine-tuning the solver. Thus, MOOSE provides an easy approach to fast implementation of new complex multiphysics simulations as well as powerful production-grade applications.

The MOOSE team has put significant effort into ease of distribution and has simplified the ability to bring the framework up on new platforms. This is a strength of the framework and helps establish a foundation for broader collaboration. The MOOSE team has done excellent work in developing a software installation package for MOOSE and its supporting software packages. The MOOSE Wiki instructions for installing the software are clear and easy to follow. Two of the review panel members independently followed the instructions, downloading and installing MOOSE and all peripheral supporting software packages during the review. The process was remarkably easy (one panel member said it was the only installation in his memory that did not have problems), and the MOOSE package was immediately usable, including full interactive graphics support.

The MOOSE framework employs a continuous integration model. The team members have put in place many excellent software quality practices and have set team expectations for high-quality software. Their development environment includes (and depends on) software configuration management and a wiki-based issue-tracking system. Testing is integral to their software quality approach. Each commit is tested across multiple platforms with multiple user applications, including standardized expectations for line coverage. The team also collects metrics that reflect their software quality expectations. Currently MOOSE and its applications are all stored in a single repository. This approach has several advantages. Tests are run on all applications on every commit; and when any interface changes are made, it is trivial to change all applications that use that interface. Thus, all applications are

always up to date and tested with the latest changes to the framework. Overall, their software quality approach supports responsive distribution of a high-quality product.

The MOOSE team members have given a good deal of thought to how to extend this approach to a distributed development environment with independent development teams for different applications. The plan is to host the MOOSE framework on github and have application developers also host their application codes at github. The test harness would then still be able to test the application codes whenever updates to the framework are made. In addition, updates to the MOOSE API would result in pull requests being generated to the application development teams. The teams would then be able to examine the changes (and modify them if needed) before incorporating these changes into their repositories. If successful, this model will be a valuable contribution to the open source development community, demonstrating scaling of continuous integration techniques across several development teams.

The MOOSE team members have focused, as was appropriate for DOE engineering-oriented simulations, on medium sized scalability in their application runs, up to about 12,000 cores. At this scale they demonstrate very good performance for both weak and strong scaling. Nothing is inherent in their approach to prevent additional scaling as needed, though it will require modifications to their work flow, specifically with regard to mesh generation. The choices of MPI + X for utilizing multicore processing nodes are very appropriate and well done.

In order to provide some perspective on the significance of the MOOSE development effort, it is useful to consider the nearly 60-year history of nuclear reactor design calculations. The principal focus of reactor core design has been the calculation of core power distributions and the associated heat transfer/fluid flow calculations that govern overall reactor behavior. Sophisticated methods were developed first to perform 2D calculations, with synthesis methods to construct 3D power distributions, and beginning in the 1990s to perform explicit 3D calculations. These calculations were run on the largest computing systems. During this 60-year history, the principal focus for reactor design was the core power and temperature distributions; calculational methods for reactor shielding, mechanical analysis, and materials analysis were fragmented and lagged far behind. The MOOSE effort (basic MOOSE framework plus applications such as Bison and Marmot) successfully integrates a variety of computational materials models with neutronics and heat-transfer/fluid flow. MOOSE provides the enabling technology to bring together a number of materials calculational methods into a single, fully coupled calculational package. This new capability is a significant advance in the state of the art in reactor core design. Since many of the practical, operational problems with existing nuclear reactor

systems are related to materials issues (crud, rod fretting), the ability to include materials calculations in the core design process should have significant benefits to the nuclear industry and future reactor designs.

Future Directions and Recommendations

The MOOSE developers have made some high-quality technical design decisions and provided a high-quality implementation with good testing and quality control. The MOOSE framework would benefit from having a clearly enunciated strategy to achieve broad uptake and, as a result, applications-driven funding opportunities. The team may need to make choices about appropriate “limiting” of scope. The MOOSE team may also benefit from the interactive process of developing a vision statement and a mission statement.

The main focus of our recommendations is how MOOSE can be made sustainable for the future, from both a usage perspective and a funding perspective. This process requires several enhancements, both to MOOSE and to the strategy related to MOOSE’s future.

Increase and diversify the MOOSE user base

Without doubt, the versatility and flexibility of MOOSE make a powerful tool for many different communities, ranging from engineering and analysis to basic physics and materials science. The MOOSE team are doing a commendable job in disseminating information to potential user communities through the tutorials that the team gives several times per year at different locations. We encourage growing the user communities, supported by making the code open source and by establishing and maintaining a github for the community. Also, a good strategy would be to select and nurture a specific user community where MOOSE is currently not well established but where MOOSE can have a transformative impact. The MOOSE team could establish direct personal contacts and collaborations with select individuals in that community.

We suggest making ELK a curated repository for physics models and material models. As a potential future use of the MOOSE framework, domain scientists are likely to want to “plug and play” models developed by others (for example, to test new transient scenarios or to perform side-by-side comparisons with a new model). We suggest setting standards for implementations such as style of implementation and naming conventions. The computer science and software standards being applied to MOOSE are appropriate and are evidence of a high degree of careful, forward-looking thought. Comparable high standards, but with a different organizing principle, may be necessary for ELK. With proper support and management, ELK could be viewed by the user community as the de facto

place for contributing and collecting models, a repository of the community work to be shared. In order to accomplish this, ELK needs a curator, likely a computational physicist.

A strength of the MOOSE framework is its support of exploration and rapid development, particularly of materials models and their behavior in increasingly sophisticated scenarios. This strength suggests an opportunity to play a key role in the growing community of computational physics support in materials science research.

Modern materials science modeling is inherently multiscale, involving length scales from atomic to macroscopic ones, and time scales from femtoseconds to seconds. In addition, drivers for accelerated materials discovery for a wide range of application focus on novel functionalities that arise through strong coupling of many order parameters and degrees of freedom. Much progress has been made in areas such as first-principle modeling (for example, density functional theory methods or quantum chemistry methods) and force-field molecular dynamics modeling that cover atomistic processes. However, as emphasized by the recent BESAC report *From Quanta to the Continuum: Opportunities for Mesoscale Science*, there is a need --- and an opportunity --- to develop mesoscale materials codes, and this is an area where MOOSE can make a huge impact. Some of the necessary infrastructure is there, such as unstructured irregular meshes (with dynamical refinement), the inherent design of MOOSE that allows for multiphysics strong coupling, and the materials science pieces in the ELK library.

In order to position itself as a premier tool in mesoscale materials modeling, some issues need to be addressed. First, since many mesoscale materials systems involve strong coupling between thermomechanical degrees of freedom and free or bound charges, an efficient solver is needed for the Poisson equation with correct boundary conditions at infinity. Modern ways to address this within a finite-element framework usually employ a boundary matrix method. The current difficulty in implementing this in MOOSE is that the boundary matrix is dense; hence, there need to be some separate hooks to an efficient (perhaps a GPU-accelerated) routine for calculating the contribution to the electrostatic potential from the boundary matrix. Second, many mesoscale methods need as input coupling constants and parameters that can, in many cases, be obtained from lower-scale modeling, such as first-principle or molecular dynamics models. While the MultiApp capabilities within MOOSE have huge potentials for coupling different MOOSE applications, it would be extremely valuable to have an interface that allows for analogous concurrent multi-scale simulations that couple MOOSE applications with, for example, density functional or molecular dynamics simulations.

Leverage unique strength of loosely-coupled system of fully-coupled subsystems

The MultiApp and loosely coupled capability is a powerful complement to the fully coupled time integration approach. This is a novel capability that may open promising research directions and provide more highly integrated tools for analysis and design. The MultiApp approach is ideal for coupling several systems, each of which may require PDE simulations with millions, or even billions, of degrees of freedom. The MOOSE team has already begun to demonstrate this capability with full reactor simulations, including thermohydraulics, neutronics, and materials modeling. Another such example of system simulations is that of batteries, which have several materials science subsystems each of which needs cutting-edge, large-scale PDE simulations. Other systems include buildings and their energy use, which account for 39% of all U.S. energy consumption. We urge the MOOSE developers to consider possible areas to which they could apply their technology of loosely coupled systems of systems for future funding opportunities.

We also have two technical suggestions for the team: that they reconsider the requirement of a “master” application in modelling systems of subsystems and that they may need to refactor the data transfer design as increasingly sophisticated modeling becomes desired. For example, they may need to provide a method for registering data transfer between applications (without requiring the transfer through the master application).

Increase usability

The MOOSE framework contains many features that enhance usability for new and advanced developers in multiple scientific domains. As MOOSE grows a wider user community, we have recommendations for consideration to increase usability and help meet anticipated demands on the MOOSE framework.

The MOOSE team has done a commendable job in developing for MOOSE, a sophisticated graphical user interface, Peacock. Peacock is a versatile, well-organized, convenient, and powerful tool to accelerate code development under MOOSE. A few modifications however could be made to Peacock to improve it. First, while the MOOSE installation process has been carefully designed to be portable across virtually any computer platform or network, the installation process of Peacock is not quite as streamlined. For example, dependences on huge libraries could be awkward to install on some arbitrary platform. It would be desirable to have a Peacock installation process that is largely self-contained and streamlined, as is the MOOSE installation. Second, it would be nice if Peacock had a more advanced interface to the kernels. For example, Peacock could have some way of visualizing the organization of kernels, and perhaps also have some interface to edit kernels.

Currently, the documentation for the MOOSE framework is contained in two forms: (1) the reference manual, available through the wiki and (2) the training presentation, available in through the training workshop. We recommend writing a users manual to complement these documents. Such a manual should contain the technical background of the algorithms and their uses (not just a description of the API) and include a comprehensive, searchable index. This may take the form of a book, technical report, or manual covering the capabilities in MOOSE and ELK (either combined or separately) as the team members find appropriate.

The MOOSE framework is developed by using a continuous integration approach. Consequently, every commit is viewed as being appropriate and available for release and distribution. We understand that this approach has met the requirements to release a version of the BISON application. As more applications are developed that establish production release practices, we anticipate the need for production release support of both MOOSE and ELK. The demands of production releases will require extensions to existing practice and may require regression testing of user problems at scale, detailed performance testing, and generation of release notes. The MOOSE team members have been diligent in developing software quality practices. We recommend that the team anticipate the needs of production release support and steadily put in place the needed practices.

Generation of sophisticated meshes (e.g., a body-fitted 3D mesh of a fuel pellet) for simulations in MOOSE is performed outside MOOSE by using a separate toolkit. The process typically requires that the problem setup phase be performed on a big-memory machine. Requiring this resource imposes two impediments: access to such a machine may not be commonplace, and mesh generation becomes inherently unscalable. We recognize that this approach is state of the art practice for many engineering analyses. As scalability becomes an increasingly important goal, however, the MOOSE team may need to consider scalable approaches to problem setup and mesh generation.

The MOOSE framework does not appear to have software support for expressing units or a standard that establishes expected practices. The application developer is expected to keep units internally consistent. Again, we recognize that this is state of the art practice in many areas of the computational sciences. Unlike many other areas, MOOSE creates the potential to compose simulations of systems built on subsystem models (that were developed by disparate research and development teams). We recommend establishing standards (or some other novel technical approach) to help ensure consistency as the

MOOSE community moves toward composing existing models of systems built of subsystems.

Currently the MOOSE installation uses non-name spaced environmental variables to affect the operation of MOOSE. This approach can have unintended consequences in new computer development environments. As an alternative, we suggest using a standardized “name space” for such variables (e.g., MOOSE_JOBS).

Develop collaborations to examine specific technical issues

The MOOSE team should seek out collaborations where MOOSE could be used to design and analyze experiments. The ultimate validation of a computer code system is direct comparison of computed and experimental results. Active involvement with experimenters provides unique and valuable opportunities for validating the MOOSE methodology.

An important consideration in any multiphysics package is the mathematical foundations underlying the transfer of information between components. MOOSE implements reasonable choices, but a more comprehensive theoretical analysis and framework would increase the confidence in the accuracy and robustness of the solutions. An increased user base and the use of MOOSE in new application areas are likely to necessitate further review and improvements. We encourage the MOOSE developers to seek out collaborations with applied mathematicians and numerical analysts in order to put the transfer functions on a firmer theoretical foundation. MOOSE’s demonstrated ability to handle complex multiphysics problems should be attractive to practical-minded numerical analysts.

MOOSE has a well-thought-out approach to using threads to make effective use of multicore processors. We recommend that the developers consider two other important architectural directions. First is the use of vectorization in current processors; this involves usually very short (2 or 4 element) vectors, which may require special care, such as attention to alignment or to organizing processing in appropriate units to enable either compilers or other tools to generate vectorized code. Second, and potentially more important, is to develop a strategy for taking advantage of GPU-style processing, including streaming processing. This will be important in the long term, and having a good strategy now will ensure that MOOSE is well positioned for future processors. In addition, the developers should consider whether investing in the current GPUs (including dealing with the additional overheads of memory copies and immature programming systems) will be important enough for targeted user communities to justify the development cost. Even the

current software systems, such as OpenACC, may offer enough benefit at relatively low development cost to justify the effort.

Open Source the Software

It is imperative that MOOSE be released as open source software with a generous license to ensure its widespread usage. The MOOSE framework and many of the basic kernels and models are fundamental research support tools that do not fall under the auspices of the U.S. export control laws and regulations. While some of the applications developed by using MOOSE may be export controlled (and should be handled accordingly), the MOOSE framework itself should be treated in the same manner as most compilers and operating system software; that is, freely available without restriction. Distributing the MOOSE framework as open source software enables its use by students, collaborators, and other researchers, providing significant leverage of the government's investment, potentially improving and expanding the product at little to no expense. The benefits of open source distribution overwhelm any small licensing fees possible from corporate distribution.

The open source distribution of MOOSE would provide a number of benefits:

- Access by a broad user community provides an easier route for collaboration on science research.
- A wider user community generates and drives innovations, a wider range of requests, and new thinking about products in ways that do not arise otherwise.
- Patches and code enhancements are added by external users.
- Wider adoption leads to more funding opportunities.

Drawbacks from open source distribution could include the following:

- Need for increased user support, to assist new and lower quality users.
- Need for a change in development model (the build, test, automerge systems, would become more "complicated").
- Possibility that with a diversity of users the MOOSE team may not get proper credit for its work (other possibilities could include misuse, unfair comparisons and the quality of contributions may also be variable).

The drawbacks are largely those common to any successful project and can be readily dealt with. The advantages of open source distribution provide significant technical benefits, foster an open scientific approach, and potentially leverage the DOE research investment to provide a large payback.

Acknowledgments: We thank Angie Herman and Dmitry Karpeyev for managing the logistics for the review.

September 3, 2013

CCN 231298

Dr. Barry Smith
Senior Computational Mathematician
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

SUBJECT: Evaluation of the Multiphysics Object-Oriented Software Environment (MOOSE) Framework

Dear Dr. Smith:

I am writing to ask that you consider serving as the lead of a review panel that I propose to convene to evaluate the merits of the MOOSE software framework. MOOSE -- the Multiphysics Object-Oriented Software Environment -- is a computational development and runtime framework developed at Idaho National Laboratory and described here <http://www.inl.gov/research/moose-introduction>.

MOOSE itself and numerous MOOSE-based applications are used in the DOE complex, in the academia, and in the industry to develop numerical simulations of various physical models. The purpose of this review is to assess the qualities and characteristics of MOOSE and, in particular, to determine its suitability as a robust and scalable platform for multiphysics simulation necessary to meet the DOE computational science goals. Specifically, I am looking for an assessment of MOOSE's algorithmic capabilities for running well-resolved, fully-coupled multiphysics simulations at scale on modern high-performance hardware. In addition, I am interested in your team's assessment of the limitations associated with the MOOSE framework for scientific and engineering applications.

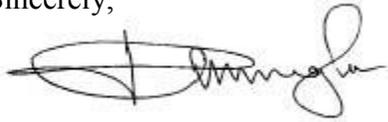
Based on your reputation as a recognized leader in the area of computational science and applied mathematics, as well as the architect of some of the most widely-used scientific software libraries, I am convinced that you are a most qualified chair person to lead this review.

Within the confines of DOE's and Argonne's policy, Idaho National Laboratory will gladly cover transportation costs, local expenses, and an honorarium for you and up to five other reviewers. I leave the total number of reviewers and the composition of the review panel, as well as the precise agenda, up to your professional judgment. I request that at the conclusion of the review the panel issue a report containing conclusions as to the appropriateness of MOOSE as a basis for high-performance computational science applications.

September 3, 2013
CCN 231298
Page 2

I propose that the review be held at Argonne National Laboratory and suggest October 1 - 3, 2013, as the tentative review dates. I sincerely hope that we can find a suitable arrangement that fits your schedule and look forward to your reply.

Sincerely,

A handwritten signature in black ink, appearing to read 'Kemal O. Pasamehmetoglu', written in a cursive style.

Kemal O. Pasamehmetoglu, Associate Laboratory Director
Nuclear Science and Technology

KOP:CH