SoftwareX 20 (2022) 101202

Contents lists available at ScienceDirect

SoftwareX

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

Software update

2.0 - MOOSE: Enabling massively parallel multiphysics simulation

Alexander D. Lindsay [a], Derek R. Gaston [a], Cody J. Permann [a], Jason M. Miller [a], David Andrš [a], Andrew E. Slaughter [a], Fande Kong [a], Joshua Hansel [a], Robert W. Carlsen [a], Casey Icenhour [a], Logan Harbour [a], Guillaume L. Giudicelli [a,*], Roy H. Stogner [a], Peter German [a], Jacob Badger [a], Sudipta Biswas [a], Leora Chapuis [a], Christopher Green [f], Jason Hales [a], Tianchen Hu [b], Wen Jiang [a], Yeon Sang Jung [b], Christopher Matthews [c], Yinbin Miao [b], April Novak [b], John W. Peterson [d], Zachary M. Prince [a], Andrea Rovinelli [c], Sebastian Schunert [a], Daniel Schwen [a], Benjamin W. Spencer [a], Swetha Veeraraghavan [g], Antonio Recuero [a], Dewen Yushu [a], Yaqi Wang [a], Andy Wilkins [e], Christopher Wong [a]

[a] Idaho National Laboratory, Idaho Falls, ID, 83415, United States of America
[b] Argonne National Laboratory, Lemont, IL 60439, United States of America
[c] Los Alamos National Laboratory, Los Alamos, NM 87545, United States of America
[d] Akselos Inc., 2101 West Blvd., Houston, TX, 77042, United States of America
[e] CSIRO Mineral Resources, PO Box 883, Kenmore 4069, Australia
[f] CSIRO Energy, Private Bag 10, Clayton South, VIC, 3169, Australia
[g] Department of Civil Engineering, Indian Institute of Science, Bangalore, Karnataka 560012, India

ARTICLE INFO

Article history:
Received 12 August 2022
Accepted 26 August 2022

Keywords:
Multiphysics
Object-oriented
Finite-element
Framework

ABSTRACT

The last 2 years have been a period of unprecedented growth for the MOOSE community and the software itself. The number of monthly visitors to the website has grown from just over 3,000 to now averaging 5,000. In addition, over 1,800 pull requests have been merged since the beginning of 2020, and the new discussions forum has averaged 600 unique visitors per month. The previous publication has been cited over 200 times since it was published 2 years ago. This paper serves as an update on some of the key additions and changes to the code and ecosystem over the last 2 years, as well as recognizing contributions from the community.

Code metadata

Current code version	*V2.0*
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00239
Permanent link to reproducible capsule	https://github.com/idaholab/moose/tree/2022-06-10-release
Legal code license	*GNU LGPL*
Code versioning system used	*git*
Software code languages, tools, and services used	*C++, MPI, OpenMP, python*
Compilation requirements, operating environments and dependencies	*Requirements: GCC/Clang C++17 compliant compiler; 16 GB memory (debug builds); 64-bit x86 + Apple Silicon support; 30 GB disk space* *Operating environments:* Linux, macOS > 10.12 *Dependencies: PETSc, libMesh*
If available, link to developer documentation/manual	https://mooseframework.inl.gov/
Support email for questions	https://github.com/idaholab/moose/discussions

1. Application developer-oriented changes

The automatic differentiation [1] system has moved toward inserting derivatives based off the global degree of freedom indices. This allows construction of residuals that have highly arbitrary

DOI of original article: https://doi.org/10.1016/j.softx.2020.100430.
* Corresponding author.
E-mail address: guillaume.giudicelli@inl.gov (Guillaume L. Giudicelli).

https://doi.org/10.1016/j.softx.2022.101202
2352-7110/

degree of freedom dependence and are useful for large element stencils found in some finite volume discretizations and for coupling disconnected mesh domains in system code calculations and mortar methods.

Parallel communication and reductions of C++ containers are in most cases achievable with one line of code. The templated interfaces to message passing interface (TIMPI) creates new MPI datatypes on the fly as necessary to communicate, automatically dispatching to the appropriate implementation. User classes are supported by specializing a few TIMPI metaclasses, and combinations of user classes, and system datatypes in nested C++ containers are handled recursively. A generic algorithm for global "push" and "pull" (query-response) of arbitrary data is also available, using non-blocking barriers and input/output to support efficient sparse communication between large numbers of processors [2]. This algorithm is widely used within the Multiphysics Object-Oriented Simulation Environment (MOOSE) and libMesh library code, and it is available to application developers as well.

A mortar finite element framework was added to MOOSE and verified. Its usage on contact mechanics and gap heat transfer modeling has improved solver convergence and result quality in practical nuclear reactor simulations [3].

In addition to finite element discretization techniques, MOOSE has added support for cell-centered finite volume discretizations. Finite volume in MOOSE [4] features gradient reconstruction, various first- and second-order advection discretization strategies, and non-orthogonal and skew correction for arbitrary (un)structured meshes. Just as for finite elements, finite volume in MOOSE has at its disposal all the capabilities described elsewhere in this article including adaptivity, distributed and shared-memory parallelism, and coupling.

Understanding the performance of multiscale, multiphysics, and parallel simulation tools can be a complex process. Unexplained pauses in application execution lead to poor user experiences. A capability called the "PerfGraph" was added to MOOSE which creates an execution graph, logs timing and memory information, and automatically prints progress to the screen. The PerfGraph works by manually instrumenting sections of code. If any of the sections are taking a long time to execute, a separate thread will start printing progress information to the screen.

2. MOOSE ecosystem growth

The MOOSE repository hosts modules which tackle physics common to numerous applications. In addition to the previously existing modules for heat transfer or fluid flow for example, new modules were created for fluid–structure interaction [5], ray tracing on unstructured mesh [2], nuclear reactor meshing [6], thermal-hydraulics systems analysis, and geochemistry [7].

MOOSE now supports neural-network-based scientific machine learning through the C++ application programming interface (API) of Pytorch (LibTorch) [8]. Modules from LibTorch can be used to generate and train various neural networks to act as low-order surrogates for MOOSE-based simulations.

3. Application coupling changes

The convergence of multiphysics tight coupling (segregated iterative approach) may be accelerated using fixed-point sequence acceleration methods. The currently implemented methods are the secant and Steffensen methods [9].

MOOSE also has added support for transferring information between applications that may have different orientations in space (rotation), units (scaling), coordinate system types (Cartesian vs. cylindrical), and translations [10]. The coordinate transformation class is essential for performing high-fidelity multiphysics computations such as coupling three-dimensional Cartesian neutronics calculations with two-dimensional axisymmetric nuclear fuel performance computations.

4. End user-oriented changes

Having operated user support through a Google Groups mailing list for many years, the MOOSE development team transitioned to a GitHub discussions forum located alongside the MOOSE repository. This new platform facilitates enhanced user support in numerous ways with subjectively better searchability, easier pinning and highlighting of news and current production issues, more obvious "closing" of a discussion once a solution is reached, and easier ways to seek contributions from internal and external collaborators. Obvious developer quality-of-life improvements include tighter integration with issue triage and subsequent development as users uncover bugs and limitations in the code base.

MOOSE has added the new MeshGenerator system, which allows for mesh generation and modification in MOOSE. Mesh generation and modification may be distributed and multi-threaded.

MOOSE compatibility with various platforms and architectures continues to increase with added support for the ARM64 architecture on MacOS platforms using Apple silicon. This was enabled using architecture-optimized compilers and dependency management within the conda ecosystem and conda-forge community [11].

The MOOSE ecosystem now comes with a flexible binary installation system that works out of the box for installing the necessary libraries, documentation pages, tests, examples, and the binary to a shared location for end users to access. This system leverages the modular Makefile system in MOOSE where complex applications that may build upon several physics modules or other applications naturally roll up to form a single installable package.

Recent improvements to the HIT input file parser have enabled multiple input file support in MOOSE. The user is now allowed to list multiple input files which will subsequently be merged into a single input before the simulation begins. Meaningful errors and information regarding overridden parameters and objects alongside their locations are also produced on the command line, so end users can easily find areas of concern in the merge.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The MOOSE team would like to acknowledge the following contributors to the core of the framework: Lise Charlot, Ruijie Liu, Nathaniel Peat, Jieun Lee, Lynn Munday, Chandrakanth Bolisetti, Vishal Patel, Dmitry Karpeev, Richard C. Martineau, Sterling Harper, Stephen Novascone, Jaron Senecal, Kyle Gamble, Hailong Chen, Alain B. Giorla, Larry Aagesen, Shane Stimpson, CiCi Pham, Yidong Xia, Brandon Langley, Shane Stafford, Daniel Vogler, Adam X. Zabriskie, John Hutchins, Joshua Hanophy, Andrew Parnell, Marie Backman, Congjian Wang, Brycen Wendt, Gavin Ridley, Philip Jagielski, Frederick Gleicher, Stephanie Pitts, Daniel Ruprecht, Daniel VanWasshenova, Malachi Tolman, Mark Messner, Dylan McDowell, Ziyu Zhang, Ling Zou, Mathias Winkel, Michael Tonks, Ju-Yuan Yeh, Matthias Kunick, John-Michael Bradley, John Mangeri, Pritam Chakraborty, Casper Versteeg, Ian Greenquist, Srivatsan Hulikal, Giovanni Pastore, Bradley Fromm,

William Hoffman, Stephen Thomas, Mike Rose, Philipp Schaedle, Heather Sheldon, Chandana Jayasundara, Jed Brown, Al Casagranda, Steve Prescott, Michael Short, Jacob Peterson, Alex McCaskey, Yuxiang Wang, Jacob Bair, Weixiong Zheng, Danielle Perez, Andrea Jokisaari, Paolo Balestra, Xueyang Wu, Parikshit Bajpai, Yingjie Liu, Spencer Gehin, Andrea Alfonsi, Kevin J. Dugan, Robert Kinoshita, moosebuild, Matt Ellis, Axel Seoane, Mark L. Baird, Tami Grimmett, Som L. Dhulipala, Nick Thompson, Weiqian Zhuo, Katie Wilsdon, and Andrew Hermosillo. This research was partially funded by the Office of Nuclear Energy of the U.S. Department of Energy, NEAMS project, under Contract No. DE-NE0008983. This research work was partially prepared for DOE through Idaho National Laboratory (INL)'s LDRD Program under the DOE Idaho Operations Office. This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy under Contract No. DE-AC07-05ID14517.

References

[1] Lindsay A, Stogner R, Gaston D, Schwen D, Matthews C, Jiang W, et al. Automatic differentiation in MetaPhysicL and its applications in MOOSE. Nucl Technol 2021;207:905–22. http://dx.doi.org/10.1080/00295450.2020.1838877.

[2] Gaston DR. Parallel, asynchronous ray-tracing for scalable, 3d, full-core method of characteristics neutron transport on unstructured mesh. (Doctoral dissertation), Massachusetts Institute of Technology; 2020.

[3] Recuero A, Lindsay A, Yushu D, Peterson JW, Spencer B. A mortar thermo-mechanical contact computational framework for nuclear fuel performance simulation. Nucl Eng Des 2022;394:111808. http://dx.doi.org/10.1016/j.nucengdes.2022.111808.

[4] Giudicelli GL, Lindsay AD, Freile R, Lee J. NEAMS-TH-CRAB. Idaho National Laboratory; 2021, http://dx.doi.org/10.2172/1847108.

[5] Dhulipala SLN, Bolisetti C, Munday LB, Hoffman WM, Yu CC, Mir FUH, et al. Development, verification, and validation of comprehensive acoustic fluid-structure interaction capabilities in an open-source computational platform. Earthq Eng Struct Dyn 2022. http://dx.doi.org/10.1002/eqe.3659.

[6] Shemon E, Jung YS, Kumar S, Miao Y, Mo K, Oaks A, et al. MOOSE framework meshing enhancements to support reactor analysis. Argonne National Laboratory; 2021, http://dx.doi.org/10.2172/1821454.

[7] Wilkins A, Christopher PG, Harbour L, Podgorney R. The MOOSE geochemistry module. J Open Source Softw 2021;6(68):3314. http://dx.doi.org/10.21105/joss.03314.

[8] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. NeurIPS 2019. http://dx.doi.org/10.48550/arXiv.1912.01703.

[9] Hu R, Nunez D, Hu G, Zou L, Giudicelli G, Andrs D, et al. Development of an integrated system- and engineering-scale thermal fluids analysis capability based on SAM and pronghorn. In: ANL/NSE-21/36. Argonne National Laboratory; 2021.

[10] Lindsay AD, Harbour L, Giudicelli GL, Icenhour C, Kong F, Stogner R, et al. User-oriented improvements in the MOOSE framework in support of multiphysics simulation. Idaho National Laboratory; 2022, INL/RPT-22-67144.

[11] Conda-Forge Community. The conda-forge project: community-based software distribution built on the conda package format and ecosystem. Zenodo; 2015, http://dx.doi.org/10.5281/zenodo.4774216.