The Journal of Open Source Software

AMReX: a framework for block-structured adaptive mesh refinement

Weiqun Zhang[1], **Ann Almgren**[1], **Vince Beckner**[1], **John Bell**[1], **Johannes Blaschke**[1], **Cy Chan**[2], **Marcus Day**[1], **Brian Friesen**[3], **Kevin Gott**[3], **Daniel Graves**[2], **Max P. Katz**[4], **Andrew Myers**[1], **Tan Nguyen**[2], **Andrew Nonaka**[1], **Michele Rosso**[1], **Samuel Williams**[2], **and Michael Zingale**[5]

1 Center for Computational Sciences and Engineering (CCSE), Lawrence Berkeley National Laboratory, Berkeley, CA, USA **2** Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA **3** National Energy Research Scientific Computing Center (NERSC), Berkeley, CA, USA **4** NVIDIA Corporation, Santa Clara, CA, USA **5** Department of Physics and Astronomy, Stony Brook University, Stony Brook, NY, USA

DOI: 10.21105/joss.01370

Software

- Review
- Repository
- Archive

Submitted: 23 March 2019
Published: 12 May 2019

Summary

AMReX is a C++ software framework that supports the development of block-structured adaptive mesh refinement (AMR) algorithms for solving systems of partial differential equations (PDEs) with complex boundary conditions on current and emerging architectures.

Block-structured AMR discretization provides the basis for the temporal and spatial strategy for a large number of applications; see, e.g., (A. S. Almgren, Bell, Colella, Howell, & Welcome, 1998; J. Bell, Berger, Saltzman, & Welcome, 1994; M. J. Berger & Colella, 1989; M. J. Berger & Oliger, 1984; Pember et al., 1998) for some of the earliest block-structured AMR work. There are also a number of block-structured and octree AMR software frameworks publicly available; see ("AMR Resources Web page," n.d.) for links to many of them.

AMR reduces the computational cost and memory footprint compared to a uniform mesh while preserving the local descriptions of different physical processes in complex multiphysics algorithms. Current AMReX-based application codes span a number of areas, including atmospheric modeling, astrophysics, combustion, cosmology, fluctuating hydrodynamics, multiphase flows, and particle accelerators. In particular, the AMReX-Astro GitHub repository holds a number of astrophysical modeling tools based on AMReX (Zingale et al., 2018). The origins of AMReX trace back to the BoxLib (W. Zhang et al., 2016) software framework.

AMReX supports a number of different time-stepping strategies and spatial discretizations. Solution strategies supported by AMReX range from level-by-level approaches (with or without subcycling in time) with multilevel synchronization to full-hierarchy approaches, and any combination thereof. User-defined kernels that operate on patches of data can be written in C++ or Fortran; there is also a Fortran-interface functionality which wraps the core C++ data structures and operations in Fortran wrappers so that an application code based on AMReX can be written entirely in Fortran.

AMReX developers believe that interoperability is an important feature of sustainable software. AMReX has examples of interfaces to other popular software packages such as SUNDIALS (Hindmarsh et al., 2005), PETSc (Balay et al., 2019) and hypre (Balay et al., 2019), and is part of the 2018 xSDK ("xSDK Version 0.4.0 Web page," n.d.) software release thus installable with Spack.

Mesh and Particle Data

AMReX supplies data containers and iterators for mesh-based fields and particle data. The mesh-based data can be defined on cell centers, cell faces, or cell corners (nodes). Coordinate systems include 1D Cartesian or spherical; 2D Cartesian or cylindrical (r-z); and 3D Cartesian.

AMReX provides data structures and iterators for performing data-parallel particle simulations. The approach is particularly suited to particles that interact with data defined on a (possibly adaptive) block-structured hierarchy of meshes. Example applications include those that use Particle-in-Cell (PIC) methods, Lagrangian tracers, or solid particles that exchange momentum with the surrounding fluid through drag forces. AMReX's particle implementation allows users flexibility in specifying how the particle data is laid out in memory and in choosing how to optimize parallel communication of particle data.

Complex Geometries

AMReX provides support for discretizing complex geometries using the cut cell / embedded boundary approach. This requires additional data structures for holding face apertures and normals as well as volume fractions. Support for operations on the mesh hierarchy including cut cells is enabled through the use of specialized discretizations at and near cut cells, and masks to ensure that only values in the valid domain are computed. Examples are provided in the tutorials.

Parallelism

AMReX's GPU strategy focuses on providing performant GPU support with minimal changes to AMReX-based application codes and maximum flexibility. This allows application teams to get running on GPUs quickly while allowing long term performance tuning and programming model selection. AMReX currently uses CUDA for GPUs, but application teams can use CUDA, CUDA Fortran, OpenACC, or OpenMP in their individual codes. AMReX will support non-CUDA strategies as appropriate.

When running on CPUs, AMReX uses an MPI+X strategy where the X threads are used to perform parallelization techniques like tiling. The most common X as of this writing is OpenMP but AMReX is rapidly evolving to work effectively on GPUs. On GPUs, AMReX requires CUDA and can be further combined with other parallel GPU languages, including OpenACC and OpenMP, to control the offloading of subroutines to the GPU. This MPI+CUDA+X GPU strategy has been developed to give users the maximum flexibility to find the best combination of portability, readability and performance for their applications.

Asynchronous Iterators and Fork-Join Support

AMReX includes a runtime system that can execute asynchronous AMReX-based applications efficiently on large-scale systems. The runtime system constructs a task dependency graph for the whole coarse time step and executes it asynchronously to the completion of the step. There is also support for more user-specific algorithms such as asynchronous filling of ghost cells across multiple ranks, including interpolation of data in space and time.

In addition, AMReX has support for fork-join functionality. During a run of an AMReX-based application, the user can divide the MPI ranks into subgroups (i.e., fork) and assign each subgroup an independent task to compute in parallel with each other. After

all of the forked child tasks complete, they synchronize (i.e., join), and the parent task continues execution as before. The fork-join operation can also be invoked in a nested fashion, creating a hierarchy of fork-join operations, where each fork further subdivides the ranks of a task into child tasks. This approach enables heterogeneous computation and reduces the strong scaling penalty for operations with less inherent parallelism or with large communication overheads.

Linear Solvers

AMReX includes native linear solvers for parabolic and elliptic equations. Solution procedures include geometric multigrid (Briggs, Henson, & McCormick, 2000) and BiCGStab iterative solvers; interfaces to external hypre and PETSc solvers are also provided. The linear solvers operate on regular mesh data as well as data with cut cells.

I/O and Post-processing

AMReX has native I/O for checkpointing and for reading and writing plotfiles for post-processing analysis or visualization. AMReX also supplies interfaces to HDF5. The AMReX plotfile format is supported by VisIt (Childs et al., 2012), Paraview (Ahrens, Geveci, & Law, 2005), and yt (Turk et al., 2011). AMReX also has linkages to external routines through both Conduit ("Conduit Web page," n.d.) and SENSEI ("SENSEI Web page," n.d.).

Documentation, Tutorials and Profiling Tools

Extensive documentation of core AMReX functionality is available online, and many of the application codes based on AMReX are publicly available as well. Smaller examples of using AMReX for building application codes are provided in the AMReX Tutorials section. Examples include a Particle-in-Cell (PIC) code, a compressible Navier–Stokes solver in complex geometry, advection-diffusion solvers, support for spectral deferred corrections time-stepping, and much more.

AMReX-based application codes can be instrumented using AMReX-specific performance profiling tools that take into account the hierarchical nature of the mesh in most AMReX-based applications. These codes can be instrumented for varying levels of profiling detail.

Acknowledgements

The development of AMReX was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under contract number DE-AC02005CH11231, and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

References

Ahrens, J., Geveci, B., & Law, C. (2005). ParaView: An end-user tool for large data visualization. In *Visualization Handbook* (pp. 717–731). Elsevier. doi:10.1016/B978-012387582-2/50038-1

Almgren, A. S., Bell, J. B., Colella, P., Howell, L. H., & Welcome, M. L. (1998). A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. *Journal of Computational Physics*, *142*, 1–46. doi:10.1006/jcph.1998.5890

AMR Resources Web page. (n.d.). https://math.boisestate.edu/~calhoun/www_personal/research/amr_software. Retrieved from https://math.boisestate.edu/~calhoun/www_personal/research/amr_software

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., et al. (2019). PETSc Web page. http://www.mcs.anl.gov/petsc. Retrieved from http://www.mcs.anl.gov/petsc

Bell, J., Berger, M., Saltzman, J., & Welcome, M. (1994). A three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, *15*(1), 127–138. doi:10.1137/0915008

Berger, M. J., & Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, *82*(1), 64–84. doi:10.1016/0021-9991(89)90035-1

Berger, M. J., & Oliger, J. (1984). Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, *53*, 484–512. doi:10.1016/0021-9991(84)90073-1

Briggs, W., Henson, V., & McCormick, S. (2000). *A multigrid tutorial, second edition* (Second.). Society for Industrial; Applied Mathematics. doi:10.1137/1.9780898719505

Childs, H., Brugger, E., Whitlock, B., Meredith, J., Ahern, S., Pugmire, D., Biagas, K., et al. (2012). VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight* (pp. 357–372).

Conduit Web page. (n.d.). https://llnl-conduit.readthedocs.io/en/latest. Retrieved from https://llnl-conduit.readthedocs.io/en/latest

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., & Woodward, C. S. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, *31*(3), 363–396. doi:10.1145/1089014.1089020

Pember, R. B., Howell, L. H., Bell, J. B., Colella, P., Crutchfield, W. Y., Fiveland, W. A., & Jessee, J. P. (1998). An adaptive projection method for unsteady low-Mach number combustion. *Combustion Science and Technology*, *140*, 123–168. doi:10.1080/00102209808915770

SENSEI Web page. (n.d.). https://sensei-insitu.org. Retrieved from https://sensei-insitu.org

Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., & Norman, M. L. (2011). yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *The Astrophysical Journal Supplement Series*, *192*, 9. doi:10.1088/0067-0049/192/1/9

xSDK Version 0.4.0 Web page. (n.d.). https://xsdk.info/packages/. Retrieved from https://xsdk.info/packages/

Zhang, W., Almgren, A. S., Day, M., Nguyen, T., Shalf, J., & Unat, D. (2016). BoxLib with tiling: An AMR software framework. *SIAM Journal on Scientific Computing*, *38*(5). doi:10.1137/15M102616X

Zingale, M., Almgren, A. S., Sazo, M. G. B., Beckner, V. E., Bell, J. B., Friesen, B., Jacobs, A. M., et al. (2018). Meeting the challenges of modeling astrophysical thermonuclear explosions: Castro, maestro, and the AMReX astrophysics suite. *Journal of Physics: Conference Series*, *1031*, 012024. doi:10.1088/1742-6596/1031/1/012024