Efficient Programming of HPC Systems: A Study on the AMReX Framework

Luis Traffa

June 1, 2023

Contents

1 Introduction

1.1 Area and Problem

HPC systems are used to quickly solve mathematical problems. Often these problems are near impossible to solve in a reasonable amount of time. Therefore, these systems employ techniques from numerical analysis to give approximate solutions, instead of exact ones.

One such technique is called adaptive mesh refinement (AMR). It is used during simulations of various natural phenomena, such as weather, climate, and astrophysics. The idea is to use a coarse mesh to simulate the phenomena, and then refine the mesh in areas of interest. This allows for a more accurate simulation, while keeping the computational cost low.

2 Prerequisites

In order to understand the AMReX framework, it is necessary to have a basic understanding of some concepts. These concepts include meshes, partial differential equations, mesh simulation and adaptive mesh refinement. This section will provide a brief introduction to these topics.

2.1 Meshes

A mesh is a geometrical representation of a physical domain, often a space, which is broken down into small, discrete elements. These elements can be in various shapes such as triangles, rectangles, or hexagons in 2D, and tetrahedra, prisms, or hexahedra in 3D. The elements are often referred to as cells or faces, and the mesh is often referred to as a grid. These small units are interconnected by edges that connect the vertices or nodes, forming a network that covers the entire domain. A Mesh is typically used in numerical simulations to discretize a continuous domain for solving complex mathematical problems.

2.2 Partial Differential Equations (PEDs)

A PDE is a type of differential equation that contains unknown multivariable functions and their partial derivatives. PDEs are used to formulate problems involving functions of several variables, and are used frequently in physics and engineering. They are often used to describe wave propagation, heat diffusion, fluid flow, quantum mechanics, etc.

When you have a PDE, it generally represents a continuum problem, meaning it is defined on a continuous domain that changes with time. The domain can represent various physical spaces or timescales, such as the space inside a heat-conducting bar (1D), weather patterns over a geographical area (2D), smoke rising from a fire (3D).

In real-world applications, the domain can be quite complex (such as the shape of an airplane wing, or the topology of a mountain range), and the PDE might not have a simple analytical solution. That's where numerical methods are useful, which attempt to find approximate solutions by solving the PDE over a discretized version of the domain.

2.3 Meshes and PEDs

The relationship between meshes and PEDs is at the center of many numerical methods in computational physics and engineering. Therefore it is important to understand how the concepts relate to one another.

This is where the concept of a mesh becomes crucial. A mesh breaks down the continuous space into discrete, manageable elements. Each element of the mesh will have nodes (points), edges (lines), or faces (surfaces), depending on whether the mesh is 1D, 2D, or 3D, respectively. When the geometry and the solution are more complex, the higher dimensionality that is required.

The PDE is solved on this mesh. We take the continuous PDE, which has derivatives, and we approximate these derivatives at the nodes, edges, or faces of the mesh. The accuracy of this approximation increases as the size of the mesh elements decreases. This process results in a large system of algebraic equations that we can solve using linear or nonlinear solvers.

For example, let's consider a 2D heat conduction problem described by a PDE (How does the heat flow from the point of the material that is heated to other points). The mesh might be a grid of squares over the domain, and at each grid point, we would approximate the PDE. This approximation, when solved, would give us the temperature at each of these grid points.

In terms of dimensionality, while I used 2D and 3D as examples for simplicity, this concept can be extended to any number of dimensions. However, in practice, 1D, 2D, and 3D are most common due to the physical reality we usually simulate, and the computational cost associated with higher dimensional simulations.

2.4 Mesh Simulation

The goal of a mesh simulation is often to predict physical behavior or solve complex engineering problems. Mesh simulation involves the use of computational methods to solve equations over the domain represented by the mesh. This can involve a variety of different equations depending on the problem at hand, but often involves the solution of PDEs. The domain is discretized into a uniform mesh of cells, and the equations are solved at each node or cell of the mesh. This means that the more cells there are in the mesh the higher the level of detail, but at the cost of a longer simulation time. Simulation steps are performed consecutively, as the next step depends on the results of the previous step. This makes simulations difficult to parallelize.

2.5 Adaptive Mesh Refinement

Adaptive mesh refinement (AMR), is a computational technique used to adaptively refine a computational mesh. During traditional mesh simulations we create a uniform grid and simulate every cell with the same level of detail. However, some areas are less affected by the course of the forces being simulated and need less processing power to achive the desired accuracy. The concept behind mesh refinement is to use fine mesh (small cells) in areas where the solution has high gradients or requires a lot of accuracy, and coarse mesh (large cells) in areas where the solution is smooth or less important. This technique allows the efficient use of computational resources, improving the accuracy of the solution while reducing computational cost.

3 AMReX: Approach

3.1 Introduction to AMReX

The AMReX framework is a software library that provides different suites to implement AMR algorithms and manages the grid-datastructures and parallel communications during the computation of mesh simulations. It is developed by the Center for Computational Sciences and Engineering (CCSE) at the Lawrence Berkeley National Laboratory (LBNL). For example, AMReX-Astro is a suite of codes for astrophysical hydrodynamics codes for exascale systems and AMReX- Combustion is a distinct suite of adaptive mesh hydrodynamics simulation codes for reacting flows. The framework is written in C++ and is open source. It is used in a variety of projects, such as the Exascale Computing Project (ECP) and the Energy Exascale Earth System Model (E3SM).

This framework aids in developing block-strucutred AMR algorithms by providing a set of tools to solve systems of partial differential equations and manage data structures and the parallelization of the algorithms. It is designed to be highy performant, easy-to-use, flexible and portable across different architectures. Hence, most of the parallelization was abstracted away, to enable the users to focus on the mathematical and physical aspects of their problems. However, users can still access lower level features if the need arises.

3.2 Key Features

The most important features of AMReX include:

- Supports multiple dimensions (1D, 2D, 3D) and multiple levels of refinement.
- Meshes can be centered by cell, face, edge or node.
- Supports a variety of solutions on the hierarchy of the grid.
- Supports particles and embedded boundary methods (When edge of the cell does not align with the physical object to be simulated, the boudary embedding allows for greater accuracy).
- Block-structures enable the use of different sized cells in different blocks.

3.3 Efficiency Techniques

Several techniques for efficient usage of resources are used in AMReX.

- Adaptive Mesh Refinement: AMReX uses a block-structured AMR, which selectively refines regions of the computational grid that require more resolution. This technique reduces computational cost by allocating resources only where needed.
- Parallelization: AMReX has strong support for parallelization with both MPI (Message Passing Interface) and OpenMP, allowing it to run efficiently on distributed-memory and shared-memory systems. It uses a combination of space-filling curve (SFC) based algorithms and two-level hierarchical algorithms for parallelization. AMReX also supports CUDA for GPUs. When combining MPI, CUDA and OpenMP, AMReX achieves the highest performace.

- Load Balancing: AMReX uses a dynamic load balancing algorithm to ensure that computational work is evenly distributed across all available processors. This minimizes idle time and enhances overall performance. The default algorithm is SFC based, but users can also use their own custom algorithms.
- Vectorization: AMReX supports the use of modern many-core and multi-core architectures and uses vectorization to exploit these architectures to the fullest.
- AMReX has a runtime that can determine which steps can be computed ahead of time, by constructing a dependency graph. This way the limitation of simulation can be slightly bypassed.

3.4 Portability Across Architectures

One of AMReX's strengths is its ability to provide portability across a wide variety of computing architectures. It has been successfully deployed on a variety of high-performance computing systems, including CPUs and GPUs, and is designed to work seamlessly with upcoming exascale architectures.

4 Evaluation

To evaluate the framework, we will look at several papers that use AMReX to solve various problems.

4.1 Publication

This publication is "Meeting the Challenges of Modeling Astrophysical Thermonuclear Explosions: Castro, Maestro, and the AMReX Astrophysics Suite" by M. Zingale.

He discusses the use of the AMReX suite of astrophysics codes for modeling stellar astrophysics phenomena. The authors focus on two AMReX suites. In particular: Maestro and Castro.

Maestro is designed to efficiently model subsonic convective flows, while Castro models the highly compressible flows occuring in stellar explosions. Both codes are built on AMReX.

The authors discuss the application of these codes to model phenomena such as Type Ia supernovae and X-ray bursts. They also discuss the challenges of making these codes performant on current and future many-core and GPU-based architectures.

The paper also discusses the science applications of these codes. For example, Maestro has been applied to convection in the Chandrasekhar-mass model and the sub-Chandra model for Type Ia supernovae, X-ray bursts, and convection in massive stars. Castro, on the other hand, has been applied to core-collapse supernovae, radiative shock breakout in supernovae and white dwarf mergers.

The authors also discuss their efforts to port their microphysics, reaction networks, to GPUs. They have created a small proxy app from Castro with just the hydrodynamics and stellar equation of the current state. They have seen significant performance gains with the CUDA version of their reaction networks, even for moderate-sized networks.

The paper concludes by discussing future development efforts for Maestro and Castro, including higher-order hydrodynamics and time-integration, rotation, stronger coupling between hydrodynamics and reactions, new solvers, and finishing the GPU port.

In the context of this paper, AMReX was vital for a number reasons:

Efficient Handling of Complex Grid Structures: Astrophysical phenomena like stellar explosions involve a wide range of length and time scales, which makes them challenging to model. AMReX's adaptive mesh refinement allows for high resolution where it's needed (e.g., at the center of an explosion), while using coarser resolution elsewhere (e.g. where the explosion has not yet reached). This makes simulations more computationally efficient.

Parallel Computing Capabilities: The simulations discussed in the paper are computationally intensive and require parallel computing to be feasible. AMReX handles the complexities of parallel communications and data distribution, which allows the scientists to focus on the physics of their problem.

GPU Support: The authors discuss the importance of porting their codes to GPUs for performance reasons. AMReX has support for GPU computing, which is crucial for achieving the performance gains discussed in the paper.

Modularity: AMReX allows for the integration of different physics modules (like hydrodynamics, reactions, etc.) in a modular way. This is important for the authors as they discuss the need for stronger coupling between hydrodynamics and reactions in their future work. Integrating these modules is curently rather challenging.

As for why another framework might not have been as good, it would depend on the specific features and capabilities of that framework. However, a framework lacking in any of the above areas (adaptive mesh refinement, parallel computing capabilities, GPU support, modularity) would likely have made the authors' work more difficult and potentially less successful.

5 Discussion

5.1 Advantages of AMReX

One of the primary advantages of AMReX is its flexibility and efficiency. Its adaptive mesh refinement allows computational resources to be concentrated where they are needed most, thereby increasing accuracy and reducing computational costs. Moreover, AMReX's high level of parallelization, load balancing, and asynchronous I/O capabilities ensure efficient utilization of resources.

5.2 Drawbacks of AMReX

Despite its numerous advantages, AMReX also has its challenges. For instance, there could be a steep learning curve for beginners, especially for those not familiar with C++. Furthermore, while AMReX provides support for a variety of architectures, optimizing it for a specific hardware or a problem that requires many different AMReX suites could require a deep understanding of that architecture and could be time-consuming.

5.3 Comparison with Other Approaches

Compared to other software frameworks for AMR and high-performance computing, AMReX stands out with its combination of computing power and flexibility. While other frameworks may offer similar capabilities, few can match the high number of features and level of support provided by AMReX.

6 Summary and Outlook

In conclusion, AMReX proves to be a powerful and flexible software framework that is well-suited to a range of applications requiring adaptive mesh refinement and high-performance computing. Despite some potential challenges in learning and optimization, it offers considerable advantages in terms of efficiency and accuracy.

6.1 Future Research Directions

Future work with AMReX will likely focus on further enhancing its capabilities, improving user-friendliness, and expanding its application in emerging fields. With ongoing developments in hardware, particularly in quantum computing, AMReX's flexibility and adaptability in regards to the integration of its modules will be crucial.