Next Generation Multiphysics Simulation Code Efforts

Melissa Rasmussen, B.S. | CCS-7, Los Alamos National Laboratory

Codes Created

MOYA

Moya is a low-energy density multiphysics production code, the upcoming flagship production code of the Ristra project and its present main development effort. Built on cutting-edge FleCSI 2, Moya will support an unstructured mesh and multiple materials. This Lagrangian hydrodynamics code aims to support ALE (Arbitrary Lagrangian Eulerian) features.

Moya explores features such as task-based parallelism and execution on GPUs, which similar multiphysics simulation codes at LANL do not yet support. It uses staggered-grid hydrodynamics (SGH), which calculates the forces exerted by a cell's sides and moves its vertices accordingly.

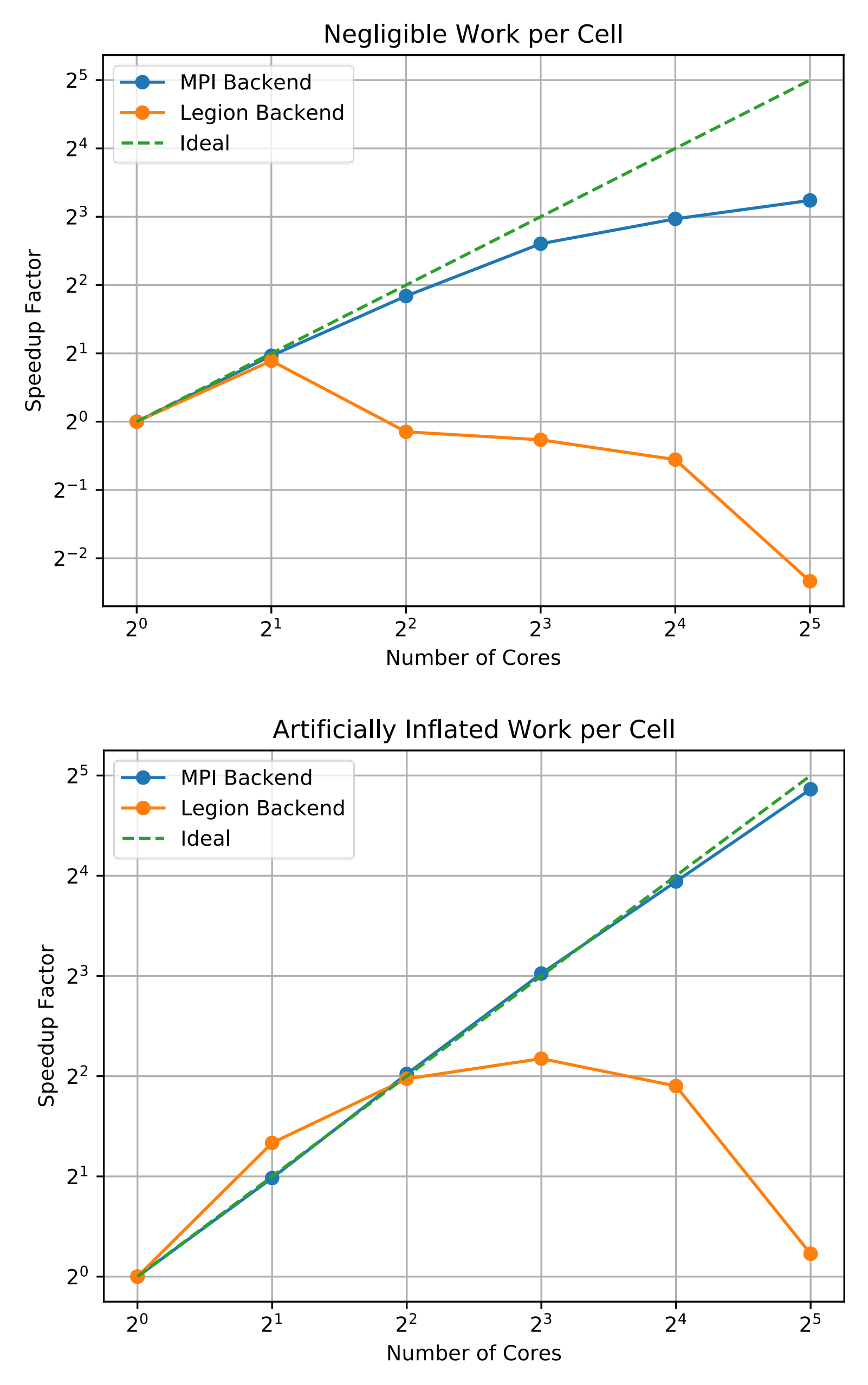
SYMPHONY

Symphony is a radiation hydrodynamics code. Like Moya, it features unstructured mesh and multimaterial capabilities. In contrast, though, it targets high-energy density physics with a coupled radiation hydrodynamics solver and uses FleCSI v1.4. Symphony is coupled to FleCSALE-mm and Puno and provides the combining physics, such as energy exchange between radiation and hydrodynamics fields.

Symphony serves as a base for research in:

- Alternative non-FleCSI frameworks for parallelism
- Radiation hydrodynamics methods
- Techniques targeting GPUs
- Requirements for Moya
- Novel accelerated radiation solve with Capsaicin
- Working on the upcoming 2023 NVIDIA-based LANL machine Venado

Graph 1. Scaling of a Single-Physics Application



On a toy application meant to demonstrate functionality of nuclear reactions with Singe in a FleCSI context, these graphs describe the scaling on multiple cores on a single node. Top, each timestep runs only a forward-Euler integrator with a 6-element nuclear reaction network. The Legion backend of FleCSI scales poorly, as it has not been tuned to the machine nor has the problem been tuned for it. The MPI backend scales well.

Bottom, each timestep has an added 0.1 second sleep. This approximates the work of a full integrator with error limits and sub-cycling and a much larger reaction network to accommodate a range of materials. Legion initially scales comparably, before the overhead costs limit its performance on more cores.

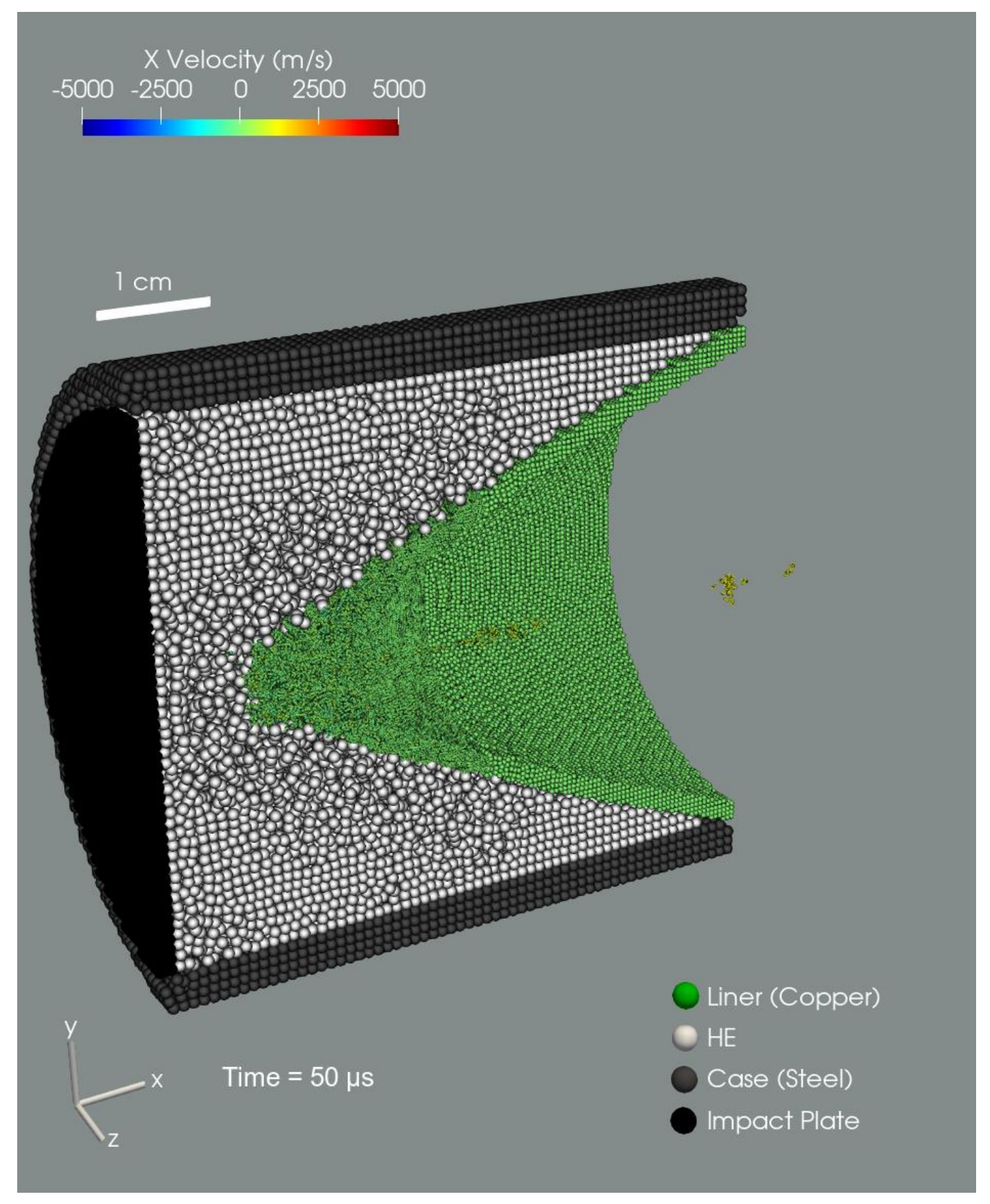


Image 1 & 2. Shaped Charge Test

The test case for an upcoming tri-lab L2 milestone is a shaped charge. Left, see a particle simulation of the case, initiated at the impact plate on the left. This simulation is from the FleCSI-based CartaBlanca++ code, demonstrating functionality that will also be implemented in Moya. This particle functionality helps to model materials breaking into pieces. Below, see a slice through a 3D pure Lagrangian simulation in Moya. This simulation is at an earlier stage, 9.5 microseconds, and shows a shock hitting the outer case and beginning to wrap around the copper liner.

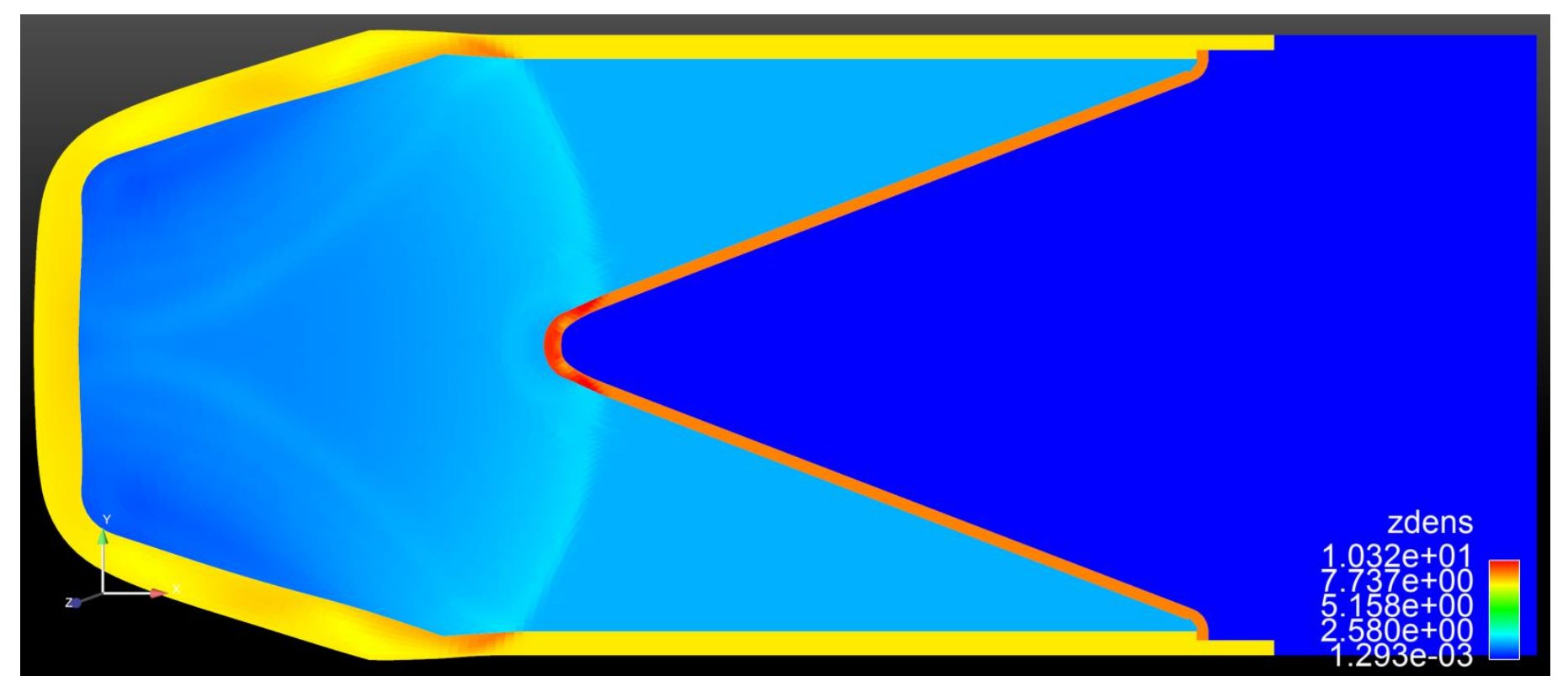
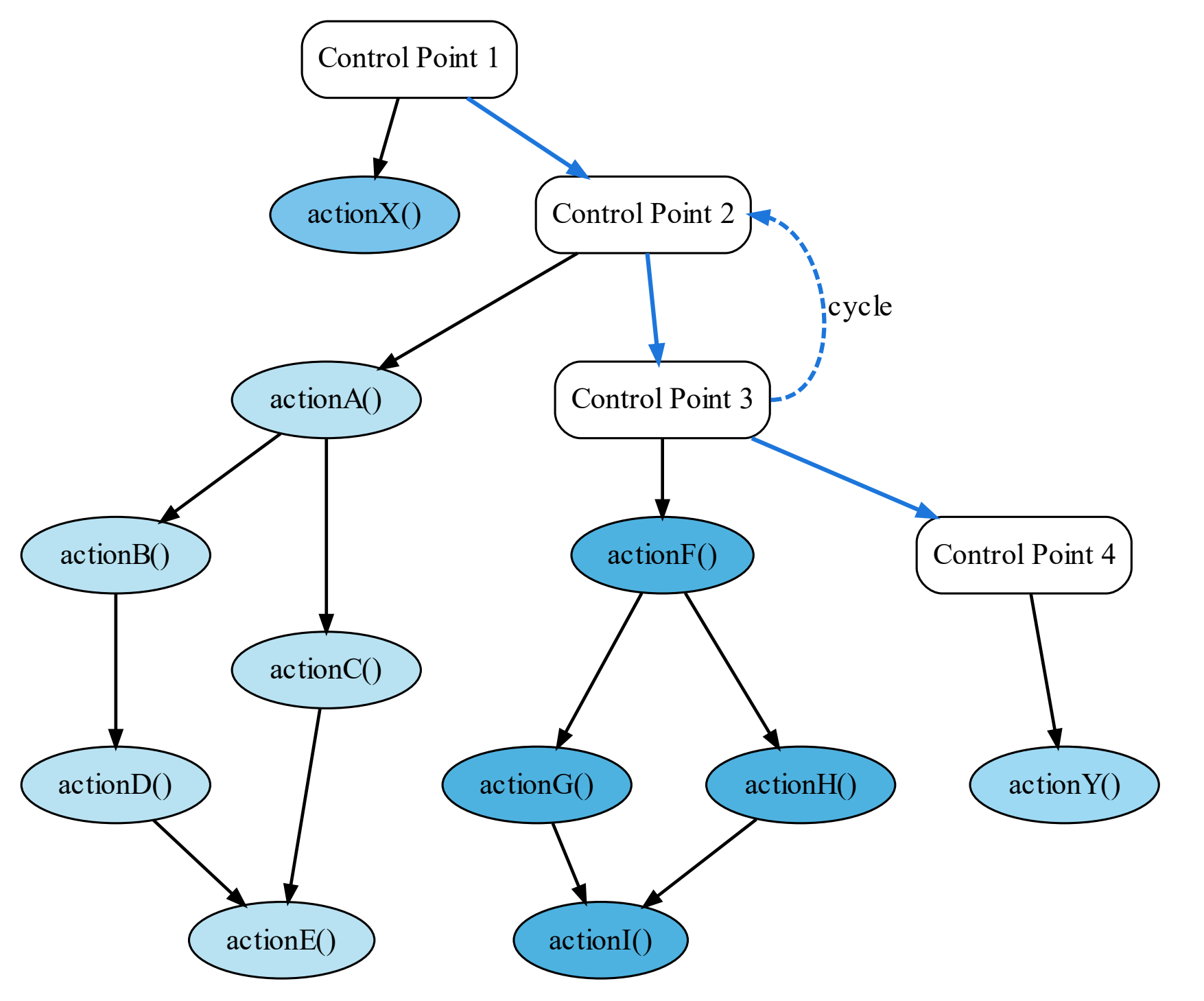
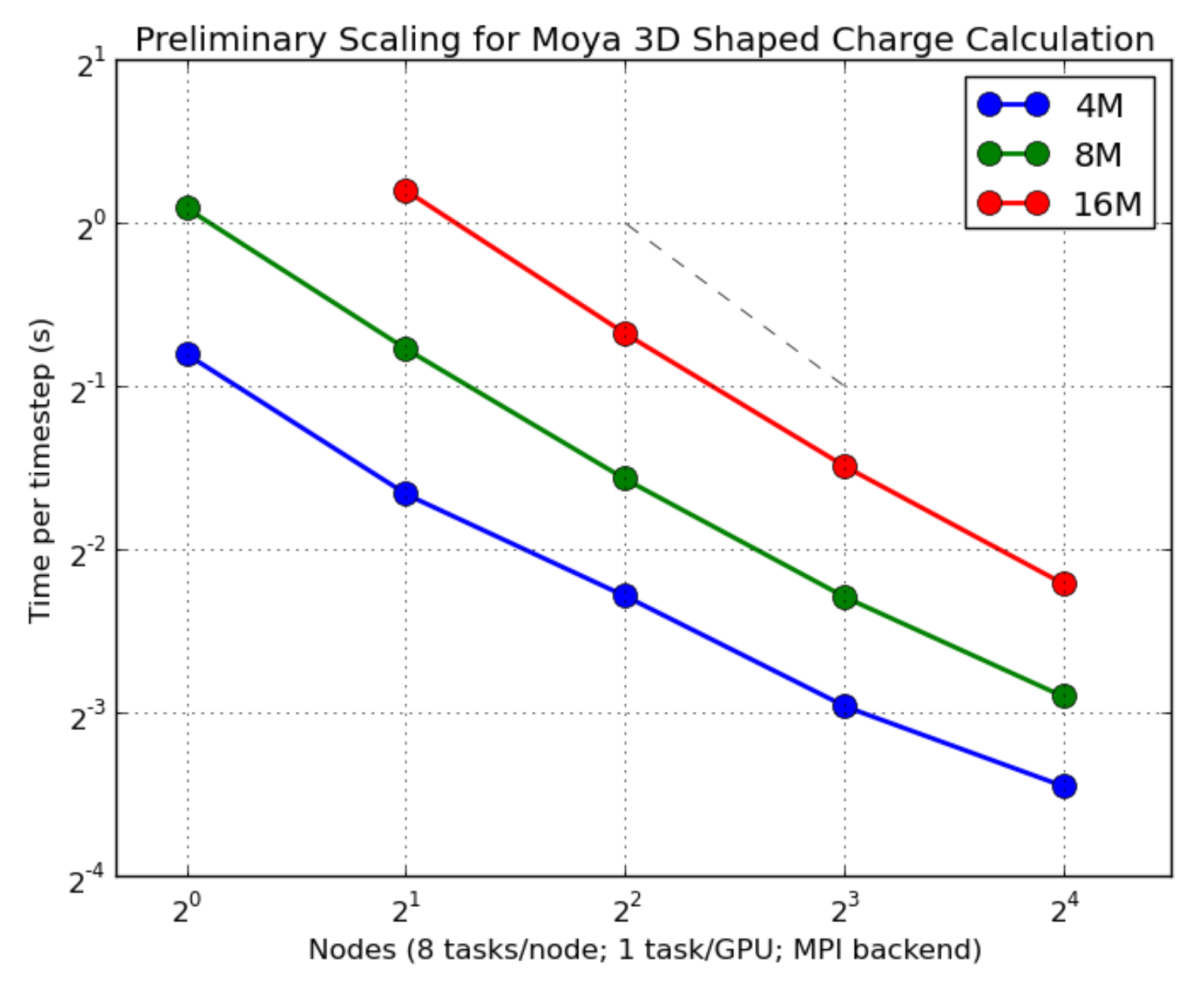


Chart 1. FleCSI control point model



FleCSI provides a structure of control points to organize the execution of an application. A developer specifies the control point or action each action is dependent upon.

Graph 2. Moya scaling



This strong scaling study shows Moya's performance on the shaped charge test case. With 8 AMD MI250 GPUs per node, the large meshes scale well down to 16 nodes. A mesh smaller than 8 million zones struggles to scale down as well due to lack of work per timestep.

Challenges on FleCSI

POINTERS

These multiphysics codes depend on the task-based structure on FleCSI. FleCSI handles the movement of data across tasks via FleCSI fields. Outside of the scope of a single task instance, pointers are not guaranteed to point to their expected data. Thus, any data structures that contain a pointer are unreliable across tasks.

MPI-BASED LIBRARIES

Libraries that use MPI for data management are unable to be used in a non-MPI FleCSI task, where most work is done. Instead, for instance, remapping using the library Portage must be done with reduced parallelization or special configuration.

TEMPLATED PROGRAMMING

FleCSI is written in C++17, and takes advantage of modern C++ features. Notably, this includes heavily templated programming. Syntactically, coding on FleCSI is nearly a different language than traditional C++ and entails a steep learning curve.

SPECIALIZATION

Physics capabilities built on top of FleCSI need a topology with additional connectivities between cells specified. This allows for values to be stored on a cell face, corner, or any custom area and accessed in association with each other.

Codes Incorporated

FLECSI

FleCSI is an open source C++17 library available at flecsi.org. It is a framework designed to support development of multiphysics applications, like Moya and Symphony. It provides different mesh topologies, a control model and data model, and can switch between backends of MPI, Legion, or HPX. It supports GPUs through Kokkos.

SINGE

Singe is a C++14 library in XCAP, a project developing a set of portable microphysics codes. Singe evolves thermonuclear reaction networks and aims to standardize the way multiphysics codes at LANL treat thermonuclear burning.

HPX

HPX is a community-based open-source project available at github.com/STEllAR-GROUP/hpx. It provides concurrency and parallelism as specified by the C++ standard.

LEGION

Legion is a parallel programming system that provides programmers with the ability to increase parallelism by performing tasks asynchronously. When tailored to a machine, it can understand the structure of program data and extract parallelism without needing a programmer to specify it explicitly.

Legion is an open-source library and is accessible at legion.stanford.edu.

PUNO

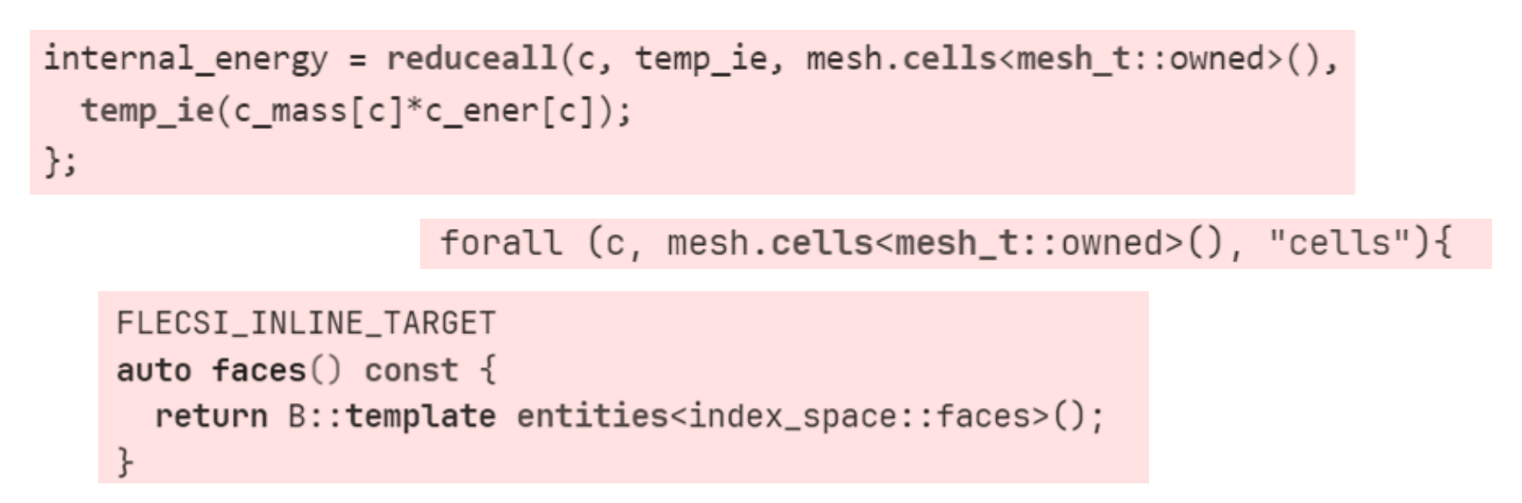
Puno is a P1 radiation diffusion solver. As such, it approximates the angular component of radiation transport with a single Legendre polynomial.

FLECSALE - MM

FleCSALE–mm is an ALE code built to model the dynamics of continuous systems, tailored to large distributed memory system architectures. The "mm" denotes multimaterial capabilities. This code models fluid flow problems and uses FleCSI for the mesh structure.

On GPUs

GPU functionality on Moya is underway with a low adoption cost, due to FleCSI's support of Kokkos and the early development stage of Moya. Current efforts are testing on additional architectures.



```
internal_energy = reduceall(c, temp_ie, mesh.cells<mesh_t::owned>(),
  temp_ie(c_mass[c]*c_ener[c]);
};
```

```
forall (c, mesh.cells<mesh_t::owned>(), "cells"){
```

```
FLECSI_INLINE_TARGET
auto faces() const {
  return B::template entities<index_space::faces>();
}
```

Comparable codes to Moya at LANL do not run on GPUs. The possibility here for expanding the computational capability of LANL multimaterial multiphysics simulation codes is high.