A performance portable library in support of future aerothermodynamics simulation software

CExA coffee, March 4, 2023

Benjamin Fovet [1], Séréna Buat [1]

[1]CEA/DAM/CESTA

Atmospheric Reentry Context

Physics and Numerics

- Hypersonic flow around spacecraft
- Quantities of interest: heat flux on the object wall and aerodynamic coefficients
- RANS (Reynolds Averaged Navier-Stokes) simulations with turbulence models
- Real gas chemistry
- Heat transfer for ablation and pyrolysis simulation

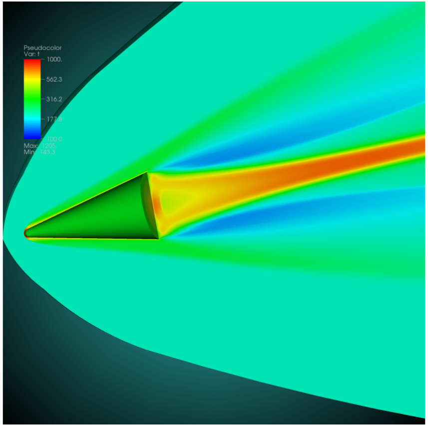


Figure 1: Slice of the flow temperature during reentry [1]

Atmospheric Reentry Context

Software

- In-house Fortran 90 production code at CEA/CESTA, ∼20yo, expected to run for the next 10 years
- Finite Volume scheme
- Capable of 2D, 2D axi, and 3D simulations
- Multiblock Structured Mesh
- Parallelized using MPI and OpenMP
- Some evolutions: neural nets to replace costly chemistry computation

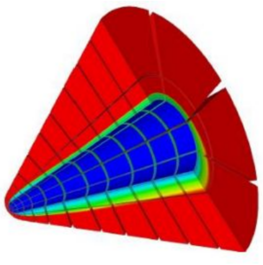


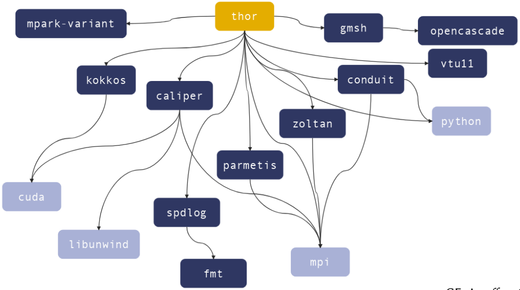
Figure 2: Blocks around a sphere-cone type object

Preparing for future needs

- A next-generation 3D code will be needed
- More complex geometries ⇒ generate unstructured meshes take less engineering time
- Innovative numerical schemes
  - Agnes Chan thesis (2023): *Innovative numerical schemes for 3D supersonic aerodynamics on unstructured mesh*
  - Vincent Delmas thesis (2026): *Design and development of innovative numerical methods to solve the 3D Navier-Stokes equations to model hypersonic flows on hybrid meshes*
- Ensure performance portability on future hardware
  - EXA1 (CEA-HF) with NVIDIA A100 nodes
  - EXA1 with NVIDIA GH200 nodes
  - What about the future systems ?

The Thor library: overview

- C++20 library started from a working group in 2022
- Fork/extension of Axom [3] with additional features, with ideas taken from Parthenon [2] and others
- Relies on Kokkos for performance portability
- Targets unstructured meshes first
- Collection of components: mesh reading, writing, partitioning, dataset parsing, logging, ...



The Thor library: components

Core

- Abstractions for Kokkos and MPI: `thor::ScopeGuard` wrapping init/finalize
- Wraps `Kokkos::parallel_for` if Kokkos enabled

```
template<class ExecutionSpace = DevExecSpace, class UnaryFunction>
void for_each(const std::string& label, int is, int ie, UnaryFunction function,
ExecutionSpace exec_space = ExecutionSpace()) {
  #if defined(THOR_ENABLE_KOKKOS)
  Kokkos::parallel_for(...);
  #else
  #pragma omp parallel for
  for (int i = is; i < ie; ++i) { function(i) };
  #endif
}
```

- Utilities (string, filesystem, etc.)

The Thor library: components

Mesh

- Data structures for 3D unstructured meshes with multiple cell types supported
- Reduced mesh connectivity: store the cell-to-node connectivity only, others can be computed if needed

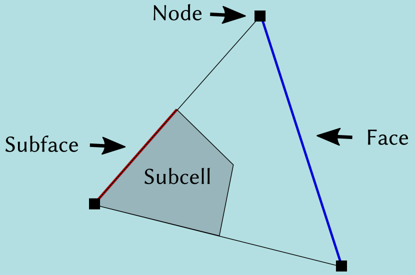


Figure 3: Cell with its connected entities

The Thor library: components

Mesh

- Variables attached to the mesh, using `Kokkos::DynRankView` as the underlying container

```
auto md_cell = Metadata({Metadata::Cell, Metadata::Output, Metadata::SyncGhosts});
std::vector<std::string> prim_labels = {"PrimDensity", "VelX", "VelY", "VelZ", "Pressure"};
auto prim = mesh->create_field<double>("Primitive", md_cell, {NUM_PRIM}, prim_labels);
auto some_matrix = mesh->create_field<double>("Mat", Metadata({Metadata::Face}), {3, 3});
```

- Metadata flags:
  - `None`,`Cell`,`Face`,`Subface`,`Node`: associate a variable with a topological element
  - `Output`: Mark field to be written to output files
  - `SyncGhosts`: Mark field ghost cells to be synchronized between MPI partitions

The Thor library: components

Mesh

- Parallel execution interface on mesh elements

```
// Loop over all cells, default execution space is DevExecSpace
for_all_cells("InitSod", mesh, THOR_LAMBDA(int cell_i) {...});
// Loop over all faces, with cell neighbors info
for_all_faces<SomeOtherExecSpace, xargs::cellids>("ComputeFluxInnerFaces", mesh,
  THOR_LAMBDA(int face_i, int left_cell_i, int right_cell_i) {...});
// Loop over all faces, with face nodes coordinates
for_all_faces<xargs::coords>(mesh,
  THOR_LAMBDA(int face_i, const FaceNodeCoords& nodes, const FaceNodeIds& node_ids) {...});
```

Loops on a subset of faces: `for_internal_faces`, `for_external_boundary_faces`

The Thor library: components

Mesh

- Partitioning using ParMetis or Zoltan, handling multiple ghost cell layers

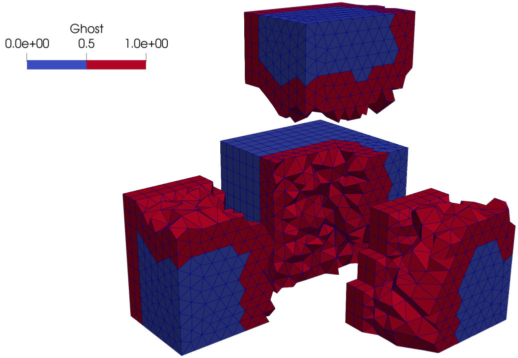


Figure 4: 2 layers of ghost cells on a 3D hybrid mesh

The Thor library: components

Mesh

- IOs: Gmsh reader, legacy VTK and (P)VTU writers
- Interested in ADIOS2, and PDI, especially for in transit analysis

Linalg

- Backport some C++26 `std::linalg` functions, e.g. `matrix_vector_product`

Input

- Read and verify input datasets in YAML, JSON, or Python (wip)

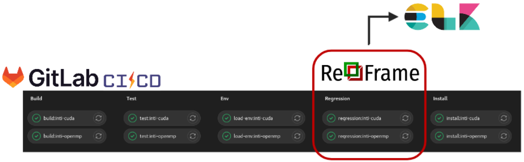
Profiling and Performance Monitoring

Profiling

- Score-P (OpenMP backend) and Nsight Systems and Compute (CUDA backend)

Automated Performance Monitoring

- Caliper (`https://github.com/llnl/caliper`) annotations in the code
- GitLab CI pipelines submitted nightly on EXA
- These nightly tests are automated with ReFrame
- Performance metrics are parsed from the tests output and visualized in a Kibana dashboard



MiniEuler

- CFD Miniapp using Thor
- Based on A. Chan and V. Delmas developments
- Classical and multi-point finite volume schemes
- Used for Thor integration and performance regression testing

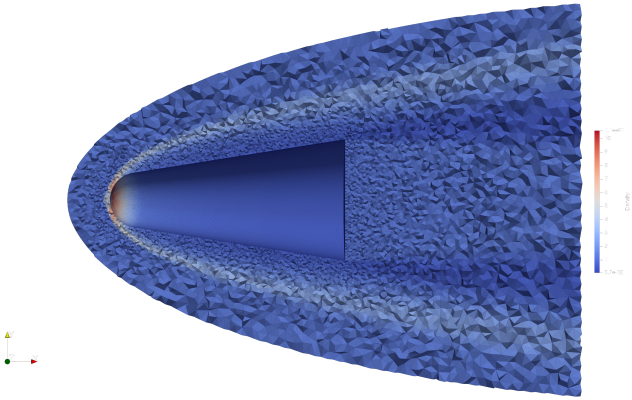
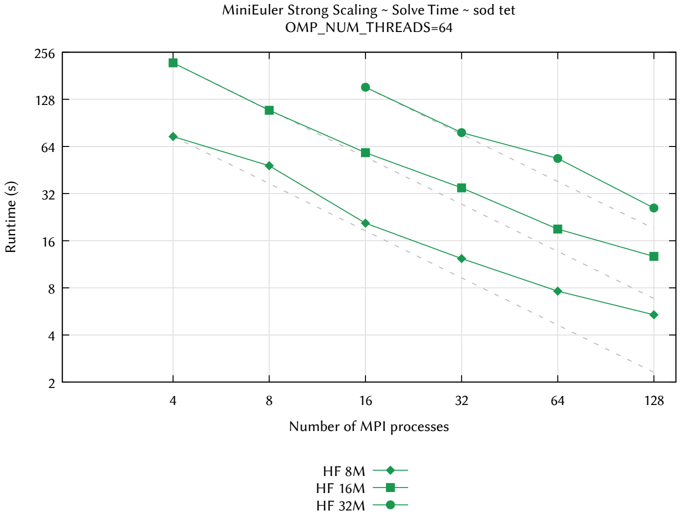


Figure 5: Density around RAM-C II vehicle

Strong Scaling benchmarks



Sod shock tube on tet mesh

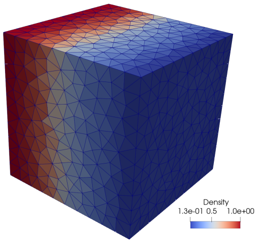
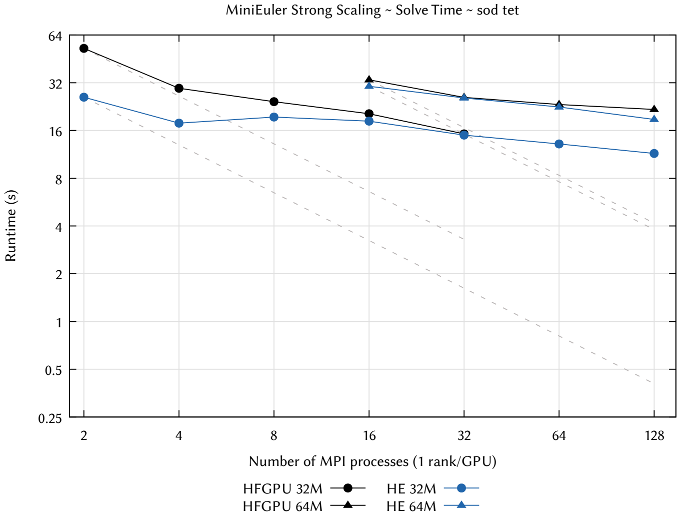


Figure 6: Density at t=0.2s

Strong Scaling benchmarks



Sod shock tube on tet mesh

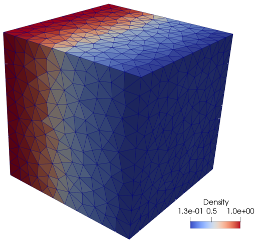
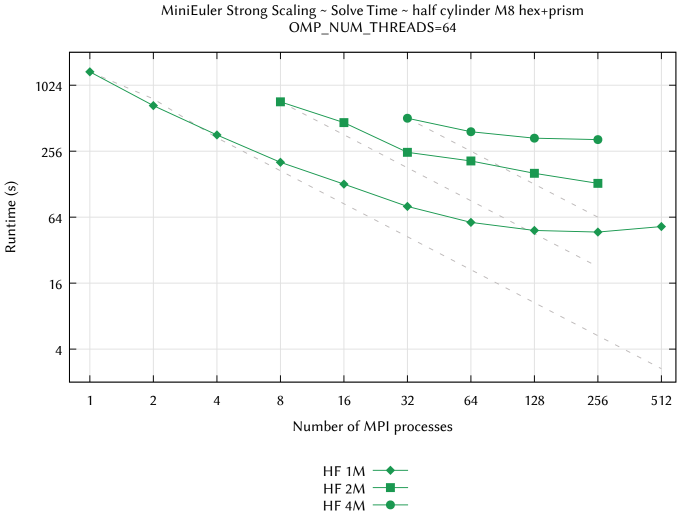
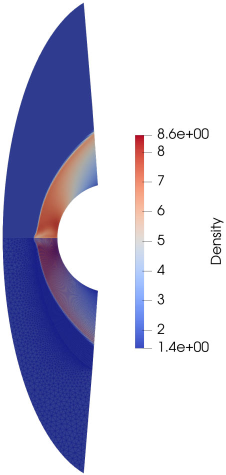


Figure 7: Density at t=0.2s

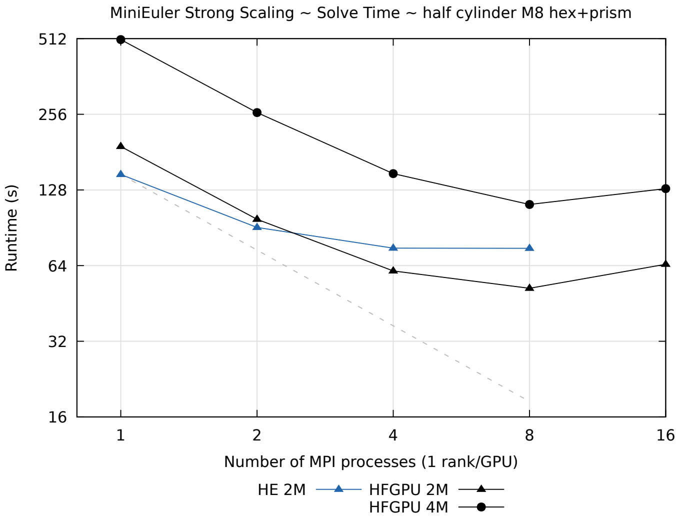
Strong Scaling benchmarks



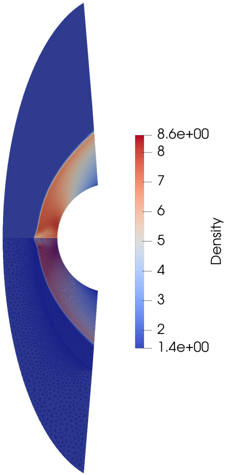
Half cylinder @Mach 8 (hex+prism mesh) using the “two-point” solver



Strong Scaling benchmarks



Half cylinder @Mach 8 (hex+prism mesh) using the “two-point” solver



Conclusion and future work

- Working library prototype, successfully used for creating the C++ version of the Fortran code by V. Delmas
- Runs with the OpenMP, CUDA and HIP backends
- Nice to have: Kokkos interop with MPI, to pass Views directly and pack/unpack behind the scenes

- Working on improving acceptance from Fortran developers
- No magic to hide complexity - training on C++, Kokkos, and GPU stuff is essential

Thanks! Any questions?

References I

Céline Baranger, Nicolas Hérouard, Julien Mathiaud, and Luc Mieussens.
Models and numerical method for the simulation of rarefied flows in atmospheric reentry applications.
http://julienmathiaud.perso.math.cnrs.fr/presentation_baranger.pdf, 2014.
Accessed: 2024–01-05.

Philipp Grete, Joshua C. Dolence, Jonah M. Miller, Joshua Brown, Ben Ryan, Andrew Gaspar, Forrest Glines, Sriram Swaminarayan, Jonas Lippuner, Clell J. Solomon, Galen Shipman, Christoph Junghans, Daniel Holladay, James M. Stone, and Luke F. Roberts.
Parthenon—a performance portable block-structured adaptive mesh refinement framework.

References II

International Journal of High Performance Computing Applications, 37(5), 12 2022.

George Zagaris, Brian M. Han, Kenneth Weiss, Lee A. Taylor, Robert A. Carson, Christopher A. White, Arlie Capps, Esteban T. Pauli, Benjamin C. Hornung, Josh B. Essman, Richard D. Hornung, Randolph R. Settgast, Matthew Larsen, Cyrus D. Harrison, Max Yang, Noah S. Elliott, Adam T. Moody, and USDOE National Nuclear Security Administration.
Axom, version 0.6.0, 6 2021.