Extreme-scale Computational Fluid Dynamics with AMR on GPUs

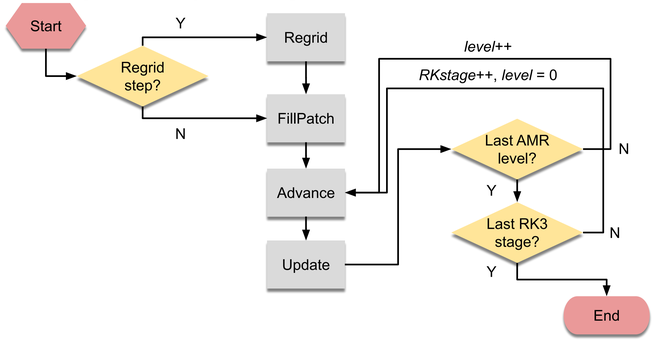
Joshua Hoke Davis[1], Justin Shafner[2], Daniel Nichols[1], Nathan Grube[2], Pino Martin[2], Abhinav Bhatele[1]
[1]Dept. of Computer Science, University of Maryland [2]Dept. of Aerospace Engineering, University of Maryland

Abstract

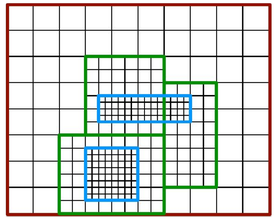
Accurate modeling of turbulent hypersonic flows has tremendous scientific and commercial value, and applies to atmospheric flight, supersonic combustion, materials discovery and climate prediction. In this poster, we describe our experiences in extending the capabilities of and modernizing CRoCCo, an MPI-based, CPU-only compressible computational fluid dynamics code. We extend CRoCCo to support block-structured adaptive mesh refinement using a highly-scalable AMR library, AMReX, and add support for a fully curvilinear solver. We also port the computational kernels in CRoCCo to NVIDIA GPUs to enable scaling on modern exascale systems. We present our techniques for overcoming performance challenges and evaluate the updated code, CRoCCo-AMR, on the Summit system, demonstrating a 5× to 24× speedup over the CPU-only version.

CRoCCo Application

- Compressible CFD solver using a weighted essentially non-oscillatory (WENO) bandwidth-optimized finite difference method [1].
  - 4th-order inviscid flux splitting with 4th-order central viscous fluxes
  - Explicit time integration with 3rd-order Runga-Kutta (RK3)
- In below flowchart, Advance contains main numerics kernels, FillPatch does ghost exchange between patches with MPI.

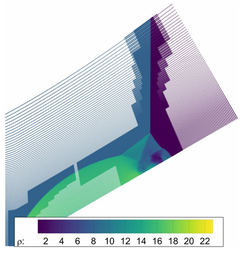


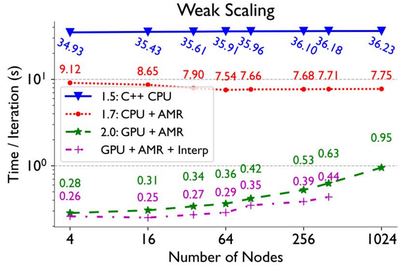
Curvilinear Support and GPU Porting Method

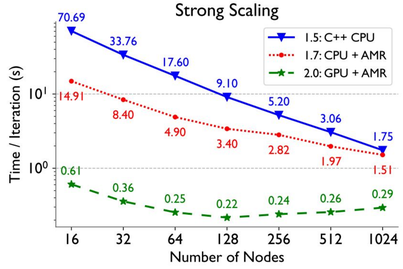


- CRoCCo was an MPI-only code in C++ with FORTRAN numerical kernels
- AMReX framework is used for block-structured adaptive mesh refinement (AMR) [2].
  - AMReX also provides functions for communication/ghost cell exchange.
- We extend the AMReX framework to support curvilinear grids:
  1. **Grid metrics**: store grid in memory, instead of expensively computing 4th-order mapping metrics.
  2. **Interpolation:** replace the default AMReX trilinear interpolator with custom interpolator accounting for non-uniform grid points.
  3. **Regridding**: store the entire grid in memory to avoid expensive I/O when creating new patches in regrid
- We use the AMReX GPU API to provide support for NVIDIA GPUs through CUDA.
  - Requires additional loop layers in many of the major numerics loops to avoid data races in scratch arrays.

Benchmarking CRoCCo-AMR



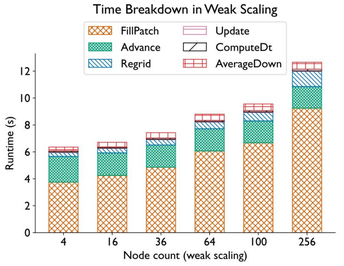




- Validated and benchmarked with double mach reflection (DMR) problem, which has been extensively studied [3], using all curvilinear code features.
- We ran strong/weak scaling of curvilinear DMR on OLCF Summit [4] for GPU runs (NVIDIA V100, 6 per node) and LLNL Quartz [5] for CPU (Intel Xeon E5, 36 cores/node).
  - "Interp" version shows performance benefit of removing custom curvilinear interpolator
- 5x to 24x speedup over the original CPU version for our new CRoCCo-AMR with GPU.
- The scaling trend of the GPU version worsens due to a communication bottleneck.

Performance Profiling

- FillPatch routine, which does ghost cell exchange, dominates performance at scale.
- Within FillPatch, non-blocking ParallelCopy dominates
  - Global communication to obtain curvilinear grid points in interpolation
- ParallelCopy also happens in Regrid, as seen in scaling



Conclusion and Future Work

- Ported a state of the art CFD code with high-fidelity numerics in FORTRAN to modern C++ using a well-supported AMR framework.
- Demonstrated a 5x to 24x speedup on the Summit GPU platform
- Employed a novel methodology for handling curvilinear grids in AMReX.
- Recommendations to developers on similar projects:
  - Expect degraded scaling performance due to shift from compute-bound to communication-bound characteristics
  - Avoid expensive ParallelCopy operations wherever possible.
- Future work will focus on improving scaling performance
  - Tackle the communication bottleneck from ParallelCopy
  - Optimizing GPU kernels to reduce register usage
  - Further benchmarking on more advanced use cases, and performance portability analysis across other GPU vendors (AMD, Intel)

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1840340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This research used resources of the Livermore Computing Division at the Lawrence Livermore National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.

References

[1] M. Martin et al., "Bandwidth-optimized weno scheme for the direct numerical simulation of compressible turbulence," *J. Comp. Phys.* , vol. 220, pp. 270–289, 2006.
[2] W. Zhang, et al., "AMReX: Block-structured adaptive mesh refinement for multiphysics applications," *The International Journal of High Performance Computing Applications*, vol. 35, no. 6, pp. 508–526, 2021. [Online]. Available: https://doi.org/10.1177/10943420211022811
[3] F. Kemm, "On the proper setup of the double Mach reflection as a test case for the resolution of gas dynamics codes," *Computers & Fluids*, vol. 132, pp. 72–75, 2016.
[4] "Summit, Oak Ridge National Laboratory" https://www.olcf.ornl.gov/summit/
[5] "Quartz, Lawrence Livermore National Laboratory" https://hpc.llnl.gov/hardware/compute-platforms/quartz