Large-scale GPU Computational Fluid Dynamics with AMR

Josh Davis[1], *Justin Shafner*[2], *Daniel Nichols*[1], *Nathan Grube*[2], *Pino Martín*[2], *Abhinav Bhatele*[2]

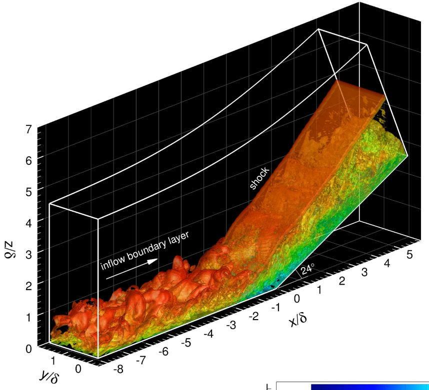
[1]Department of Computer Science, [2]Department of Aerospace Engineering

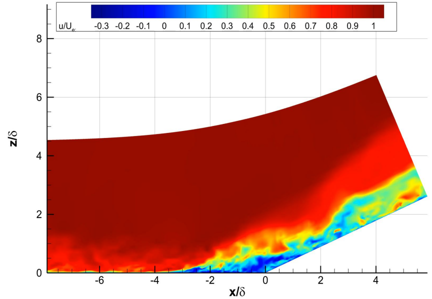
UNIVERSITY OF
MARYLAND

The CRoCCo Code

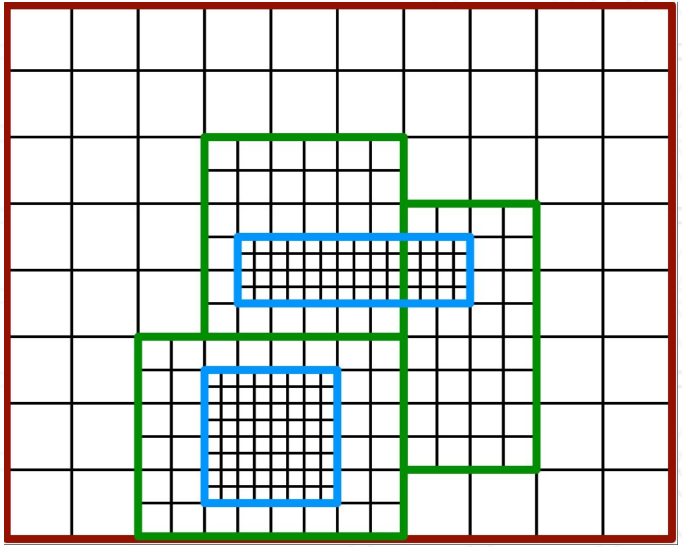
- CRoCCo is an compressible hypersonic flow simulation code validated in prior work [1]
  - CRoCCo was previously entirely Fortran and parallelized with MPI
  - Finite-difference with explicit time integration
- Applications include climate prediction, hypersonic flight vehicle development

Problem: How can we upgrade CRoCCo to take advantage of advances in supercomputing hardware and software?





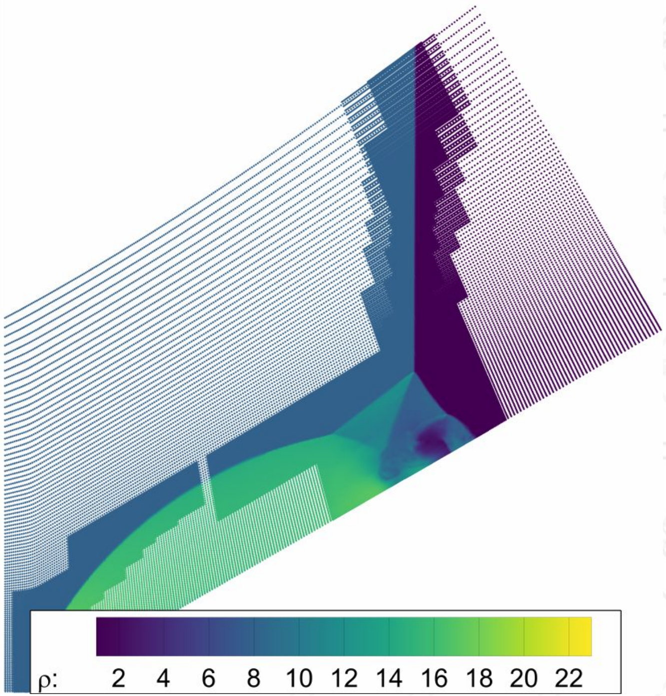
Our Solution: CRoCCo-AMR



1. Add **Adaptive Mesh Refinement (AMR)** to solve the same problem with fewer grid points
    - AMR changes the grid densities adaptively in space and time to match problem characteristics
2. Compute on **GPUs** to take advantage of modern supercomputers

→ GPU port yields **19-38x speedup** on Summit

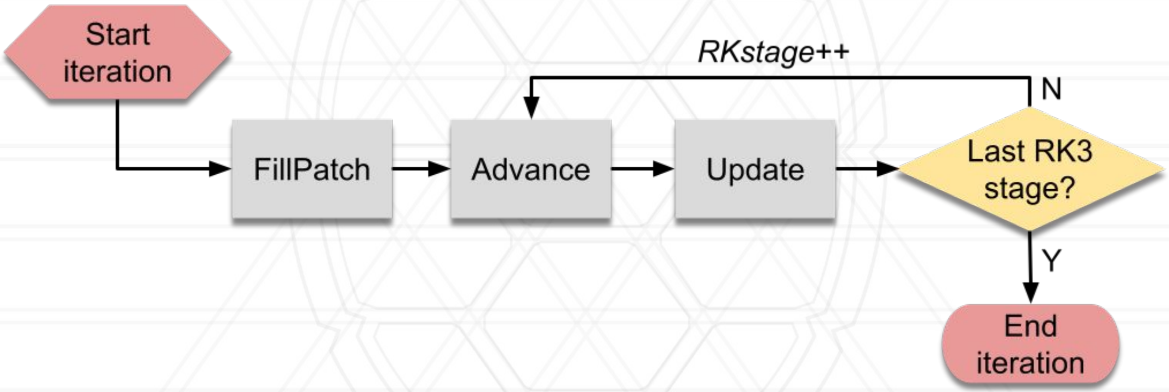
AMReX Framework



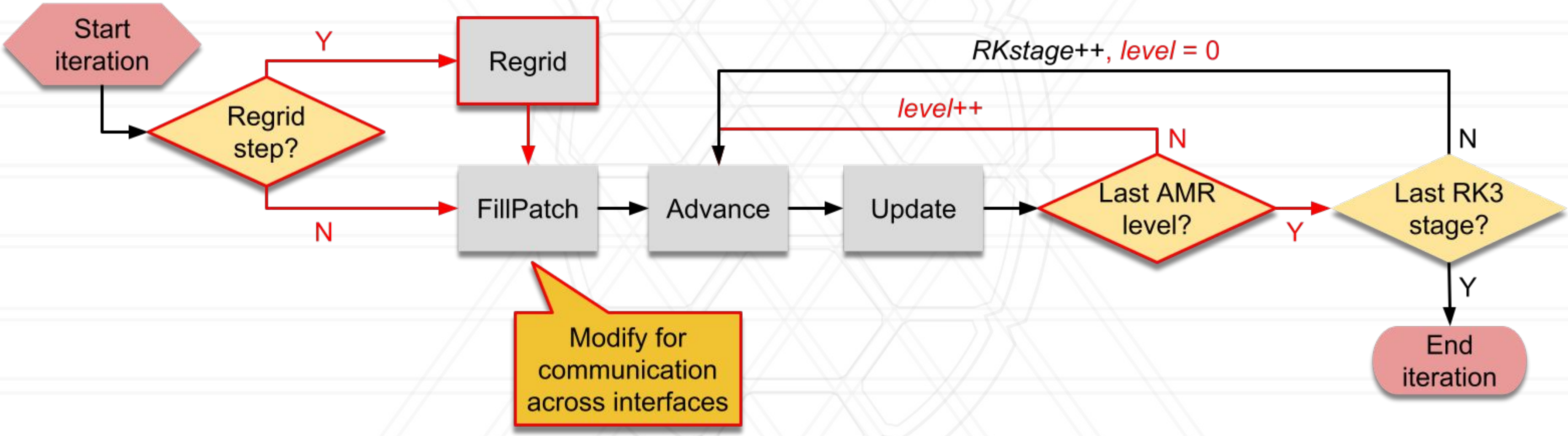
- AMReX framework provides both block-structured AMR and GPU capabilities [4]
  - Plus handling of MPI communication and load balancing
- However, AMReX is a C++ framework and supports Cartesian grids only
  - CRoCCo is a curvilinear solver in Fortran

We need to convert nearly all our Fortran to C++ and adapt AMReX for curvilinear grids

CRoCCo Before AMR



Adding AMR to CRoCCo



Modify for communication across interfaces

Working with AMReX in a Curvilinear Code

- Using the AMReX framework in a curvilinear grid code required two major changes:
    1. **Grid metrics and regridding**: store entire grid in memory, to avoid I/O operations in regrid and computing 4th-order mapping metrics on-the-fly
    2. **Interpolation**: replace the default AMReX trilinear interpolator with our custom interpolator accounting for non-uniform spacing of grid points
        - Computing intermediate points when a fine grid needs to get ghost points from a coarse neighbor

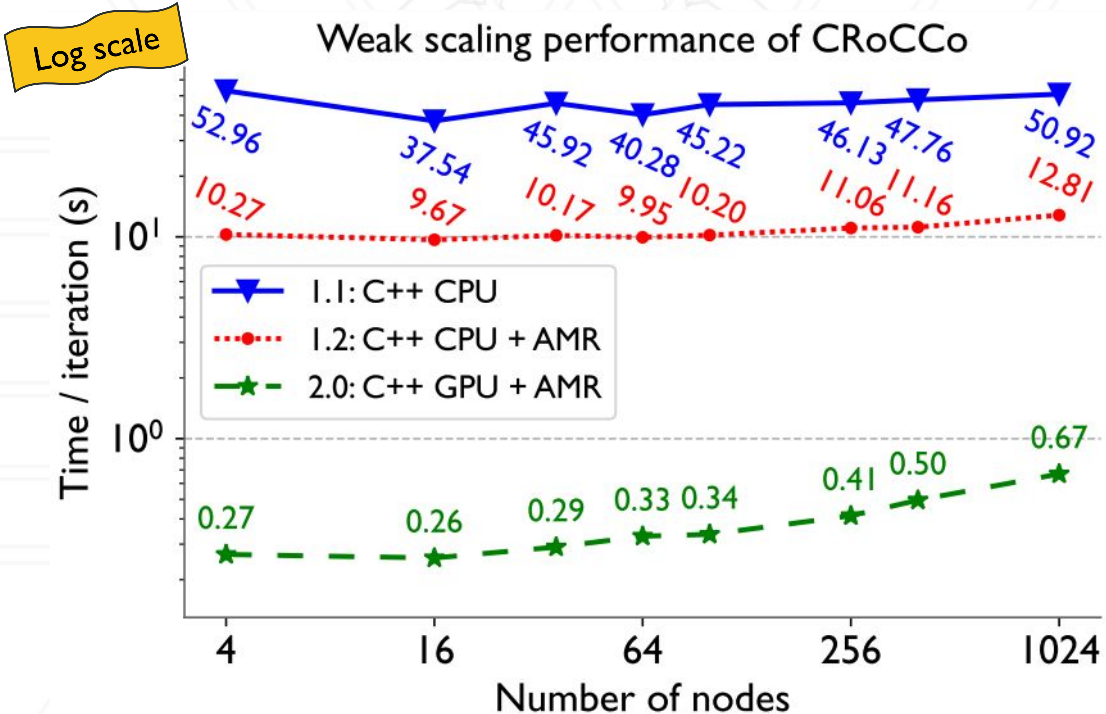
Porting CRoCCo Kernels to GPU

- Two step process: Fortran to C++, adding AMReX GPU support to C++
- Needed to divide kernels up by loop bounds to match the AMReX paradigm
  - Stencil loops extracted into ParallelFor, while regular loops stayed in Launch
  - Needed to increase dimensionality of scratch arrays reused between outer loop iterations to prevent data races
- Regular validation runs to ensure correctness

```
WenoFx(...)
```

```
launch(gbx,
    [=] AMREX_GPU_DEVICE ( Box const& tbx) {
        WenoFx(...);
    });
```

Weak Scaling CRoCCo with DMR (1/2)



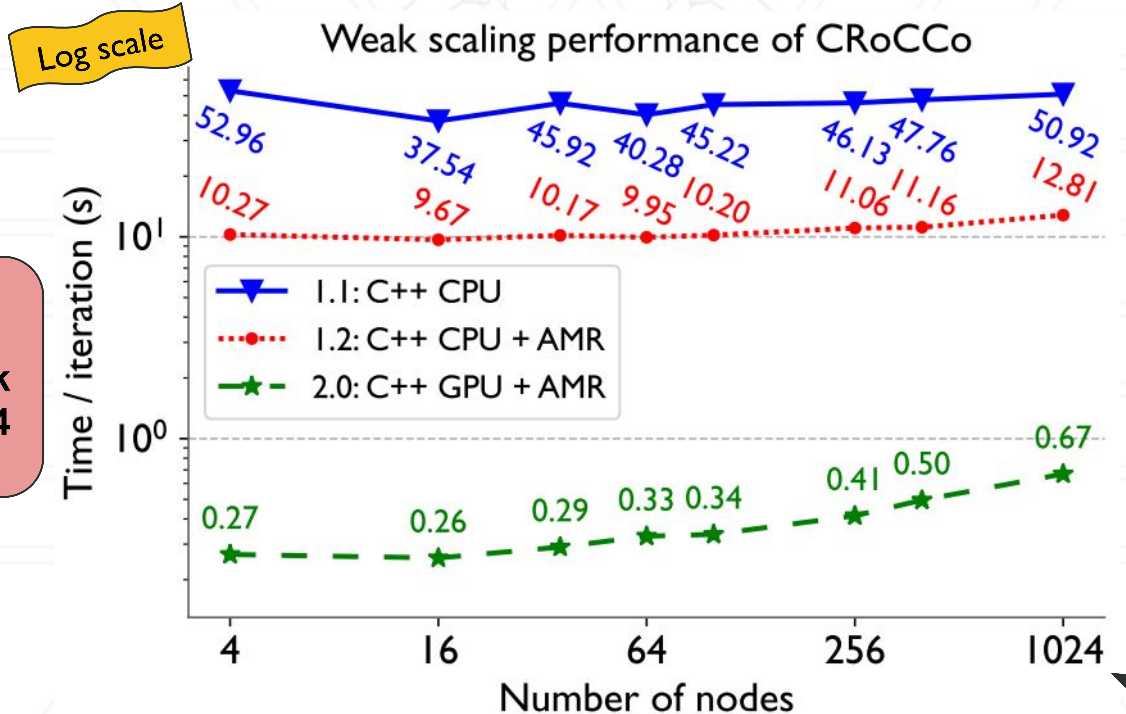
Lower is better

6144 GPUs

Data collected on OLCF Summit with DMR problem

Weak Scaling CRoCCo with DMR (1/2)

GPU acceleration has exposed a scaling bottleneck starting around 64 nodes

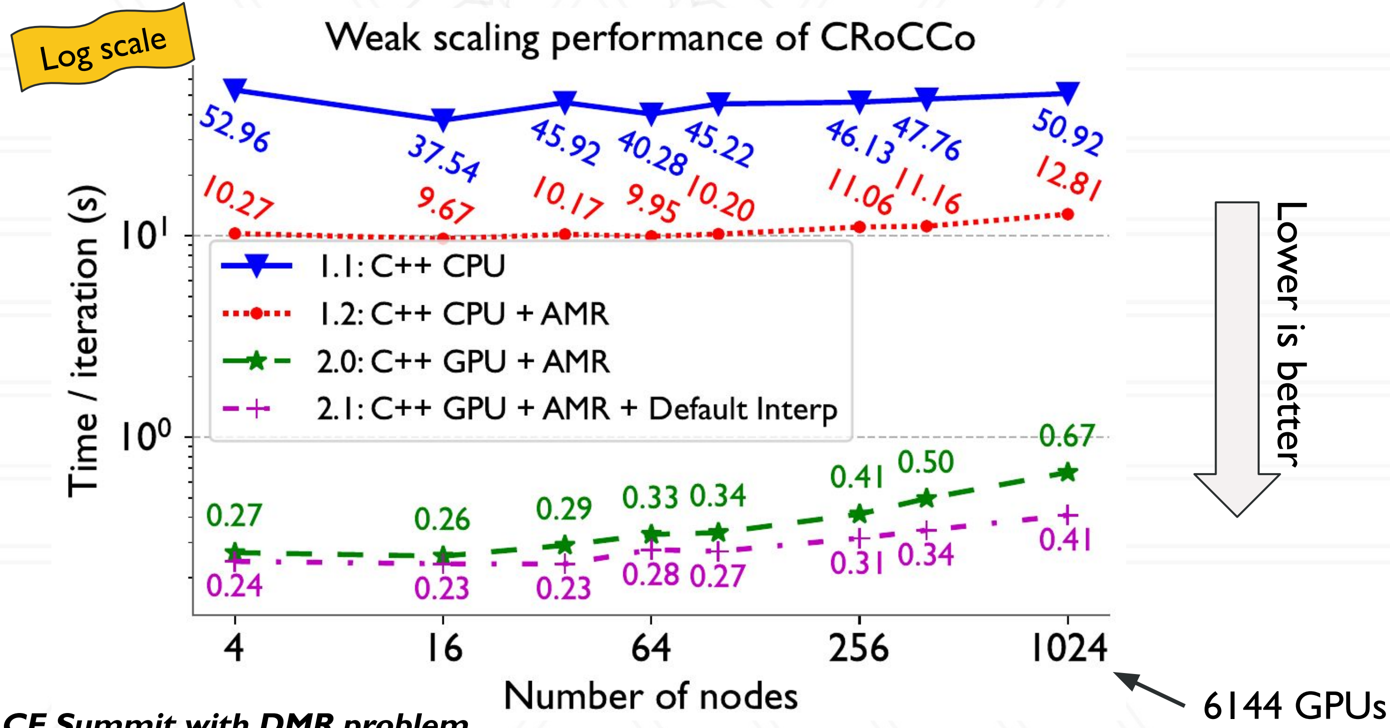


Lower is better

6144 GPUs

Data collected on OLCF Summit with DMR problem

Weak Scaling CRoCCo with DMR (2/2)

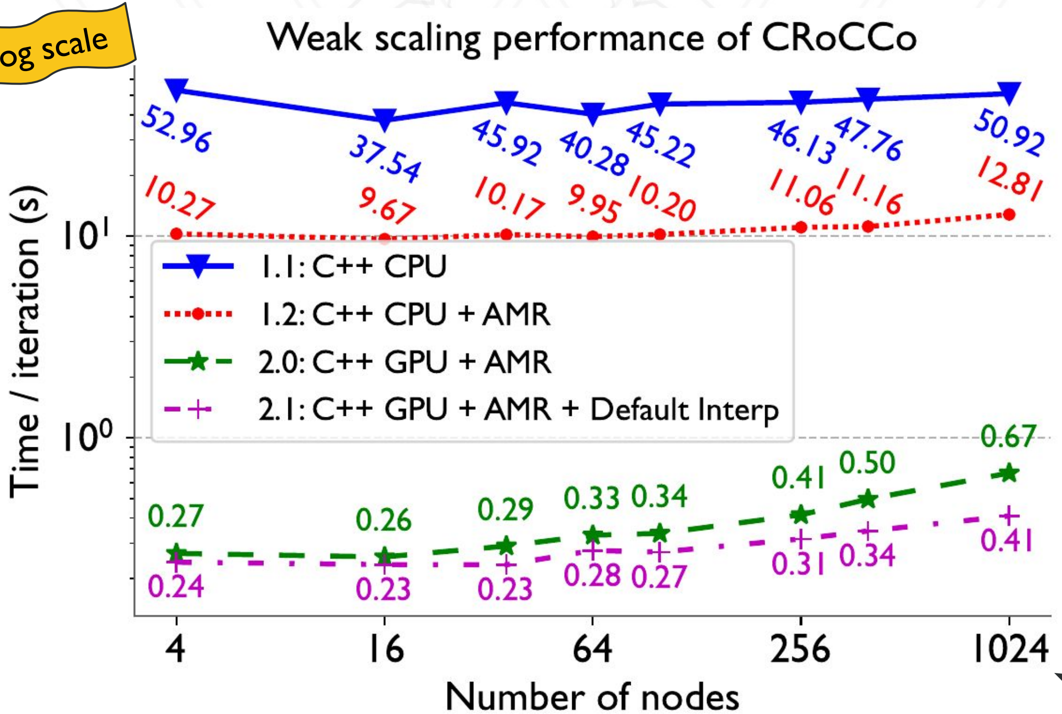


Data collected on OLCF Summit with DMR problem

Weak Scaling CRoCCo with DMR (2/2)

Our curvilinear interpolator is partially at fault for scaling issues

Log scale

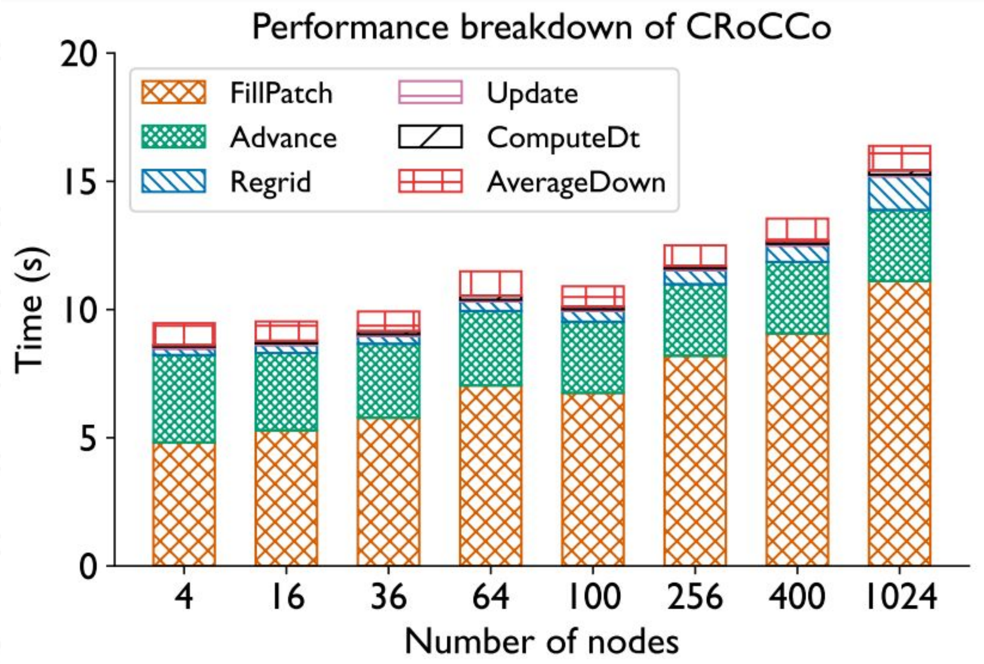


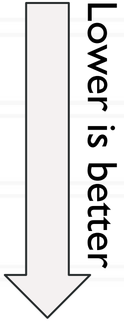
Lower is better

6144 GPUs

Data collected on OLCF Summit with DMR problem

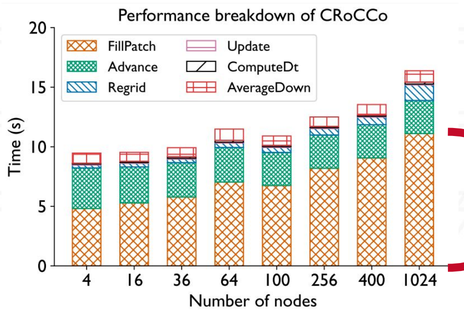
Profiling the Final Implementation

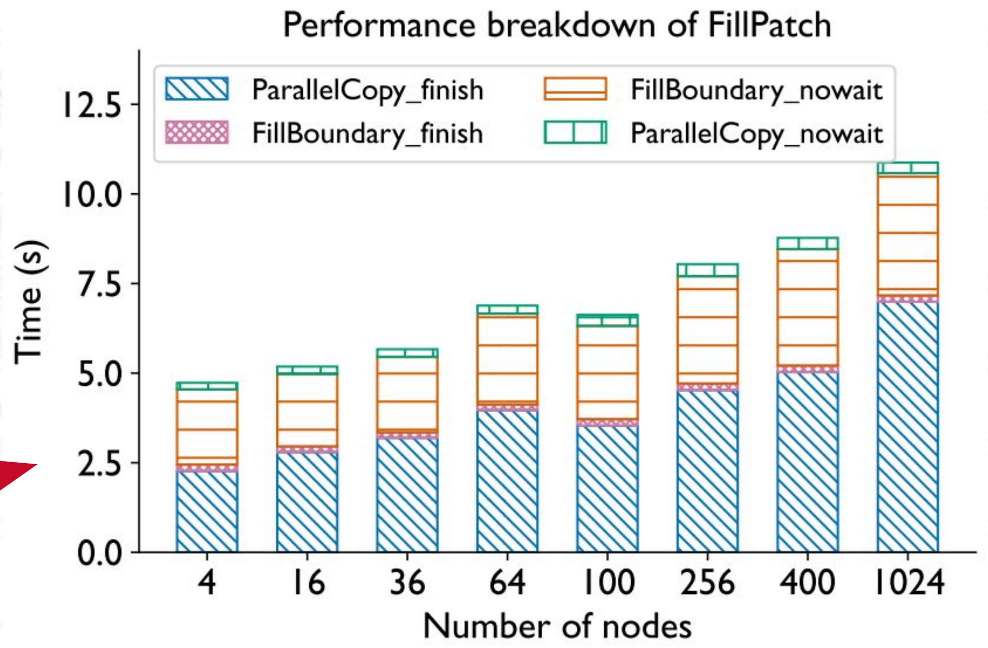




Data collected on OLCF Summit with DMR problem

Profiling the Final Implementation





Lower is better

Data collected on OLCF Summit with DMR problem

Insights From Our Experiences

- ParallelCopy operations are a significant bottleneck in adapting AMReX to curvilinear grids
- Carefully matching refinement of AMR and non-AMR cases to can ensure a comparison of the "same" science
- Scaling trend is likely to degrade when accelerating compute regions on the GPU
- Likely to achieve low GPU utilization in numerics kernels with high register usage in a direct port from CPU

See our paper for more results

Conclusion and Future Work

- We have described our efforts porting CRoCCo from MPI-only to support AMR and GPUs using AMReX, with previously-unavailable curvilinear grid support
- 19x to 38x overall speedup from our improvements
- Future work:
  - Better understand and address observed communication bottleneck
  - Determine impact of load imbalance, if any
  - Improve GPU theoretical occupancy in kernels by lowering register usage

Contact: Josh Davis — jhdavis@umd.edu

Paper: http://www.cs.umd.edu/~bhatele/pubs/pdf/2023/ipdps2023b.pdf

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.

NSF

OAK RIDGE
National Laboratory

Lawrence Livermore
National Laboratory

References

[1] I. Beekman, S. Priebe, Y.-C. Kan, and M. P. Martin, "DNS of a large-domain, Mach 3 turbulent boundary layer: turbulence structure," *AIAA 2011-0753*, 2011.

[2] M. Martín et al., "Bandwidth-optimized weno scheme for the direct numerical simulation of compressible turbulence," *J. Comp. Phys.*, vol. 220, pp. 270–289, 2006.

[3] E.M. Taylor, M. Wu, and M.P. Martín, "Optimization of Nonlinear Error Sources for Weighted Non-Oscillatory Methods in Direct Numerical Simulations of Compressible Turbulence," *J. of Com. Phys.*, 223, 384-397, 2007.

[4] 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.

[5] J. Shafner and M.P. Martín, "Predictive Tools for Supersonic Retropropulsion Flows", 2nd International Conference on Flight Vehicles, *Aerothermodynamics and Re-entry Missions & Engineering*, Jun 22, Heilbronn, Germany.

[6] "Summit, Oak Ridge National Laboratory" https://www.olcf.ornl.gov/summit/

[7] "Quartz, Lawrence Livermore National Laboratory" https://hpc.llnl.gov/hardware/compute-platforms/quartz

[8] P. Woodward and P. Colella, "The numerical simulation of two-dimensional fluid flow with strong shocks," *J. of Computational Physics*, vol. 54, no. 1, pp. 115–173, 1984.

UNIVERSITY OF MARYLAND

Josh Hoke Davis
jhdavis@umd.edu
https://jhdavis8.github.io/
8125 Paint Branch Dr
College Park, MD 20742

The Team

- Dept. of Computer Science
    - Josh Hoke Davis
    - Daniel Nichols
    - **Prof. Abhinav Bhatele**
- Dept. of Aerospace Engineering
    - Justin Shafner
    - Nathan Grube
    - **Prof. Pino Martín**

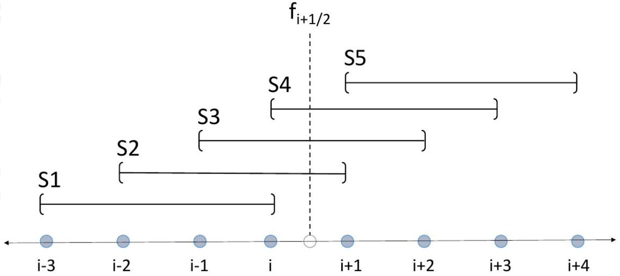
PSSG | PARALLEL SOFTWARE AND SYSTEMS GROUP

A. JAMES CLARK SCHOOL OF ENGINEERING

CROCCO LABORATORY

CRoCCo Numerical Scheme

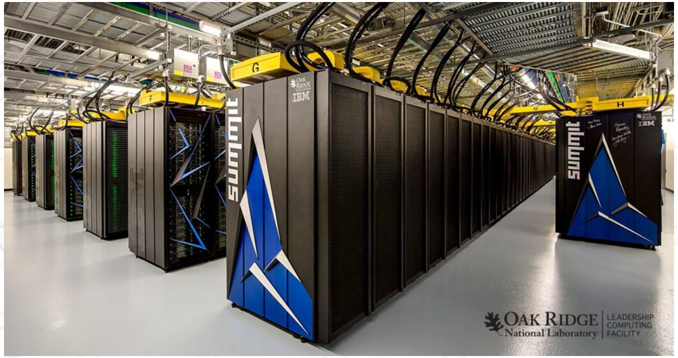
- CRoCCo solves the conservative form of the Navier-Stokes equations using a finite-difference, weighted essentially non-oscillatory (WENO) method [2,3]
  - Flux at interface is reconstructed by choosing from multiple candidate stencils based on a relative smoothness coefficient
  - The WENO method is bandwidth- and nonlinearly-optimized (WENO-SYMBO)
  - 4th-order inviscid flux splitting with 4th-order central-difference viscous fluxes
  - Explicit time integration with 3rd-order Runge-Kutta



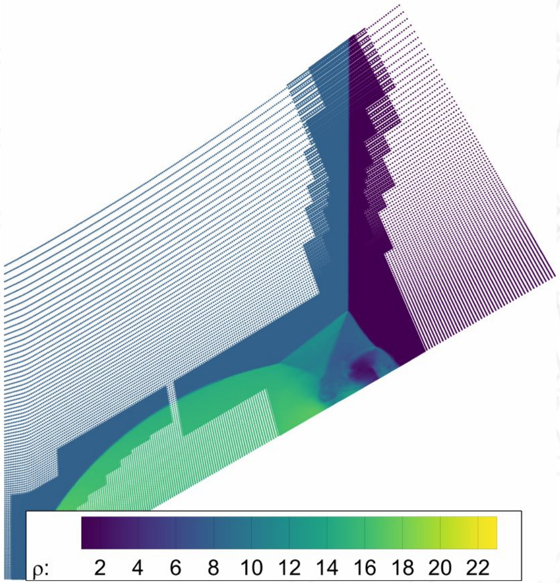
Benchmarking Platforms Used

- All runs collected on Summit at Oak Ridge National Laboratory [4]
  - Two 22-core IBM POWER9 CPUs with six NVIDIA V100 GPUs per node
  - Non-blocking fat tree network topology, dual-band InfiniBand interconnect

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY



Benchmarking Problem



- Double Mach Reflection (DMR) problem used for benchmarking [6]
  - Extensively studied in the literature and easy to set up and validate
- Includes regions of turbulent and freestream flow with moving shockwave
- Solved in three dimensions

Scaling Problem Sizes

- **Weak scaling**: 1.2x10^5 grid points per GPU in non-AMR
  - Weak scaling node counts break from perfect doubling to respect AMR blocking factor and DMR problem aspect ratio while ensuring fixed number of grid points per GPU
- Number of grid points in AMR-enabled cases is dynamic
  - We set the refinement at the finest AMR level to equal the overall refinement of the non-AMR case
  - In practice, the AMR case uses 89-94% fewer grid points than the non-AMR case for the same problem
- All scaling runs are run out to 40 iterations, and time per iteration is averaged for latter 20 iterations

Weak Scaling Problem Size Table

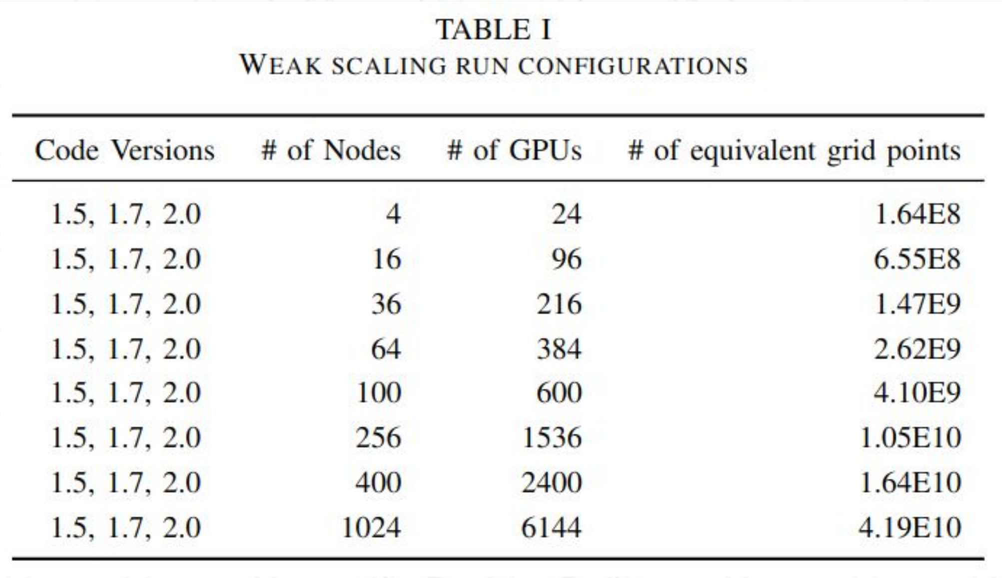
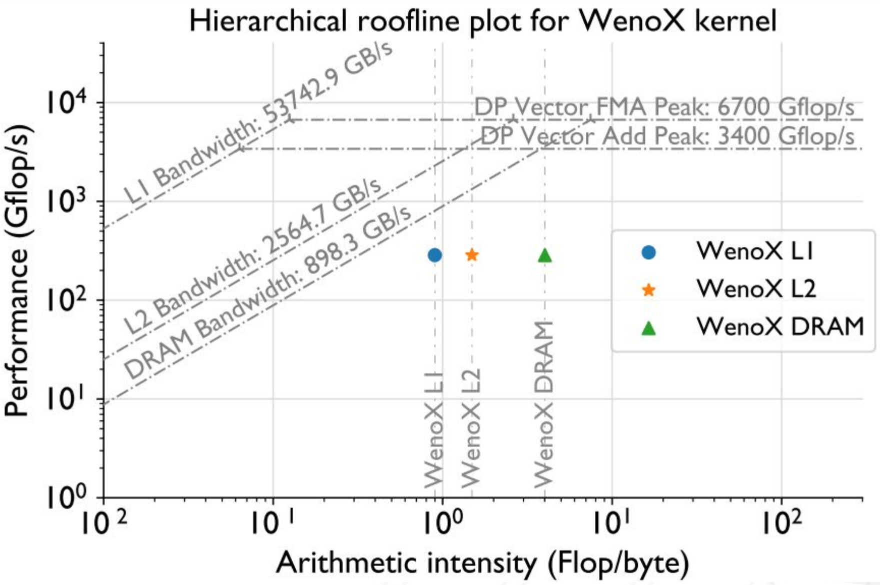


TABLE I
WEAK SCALING RUN CONFIGURATIONS

Code Versions	# of Nodes	# of GPUs	# of equivalent grid points
1.5, 1.7, 2.0	4	24	1.64E8
1.5, 1.7, 2.0	16	96	6.55E8
1.5, 1.7, 2.0	36	216	1.47E9
1.5, 1.7, 2.0	64	384	2.62E9
1.5, 1.7, 2.0	100	600	4.10E9
1.5, 1.7, 2.0	256	1536	1.05E10
1.5, 1.7, 2.0	400	2400	1.64E10
1.5, 1.7, 2.0	1024	6144	4.19E10

Kernel Roofline Analysis



- Double-precision roofline plot of representative numerics kernel, WenoX, on a V100
- ~4% of peak DP performance achieved for all kernels
  - Low theoretical occupancy (12.5%), due to high register usage
  - Bandwidth-bound
- We are exploring improvement with mixed-precision, removing division operations