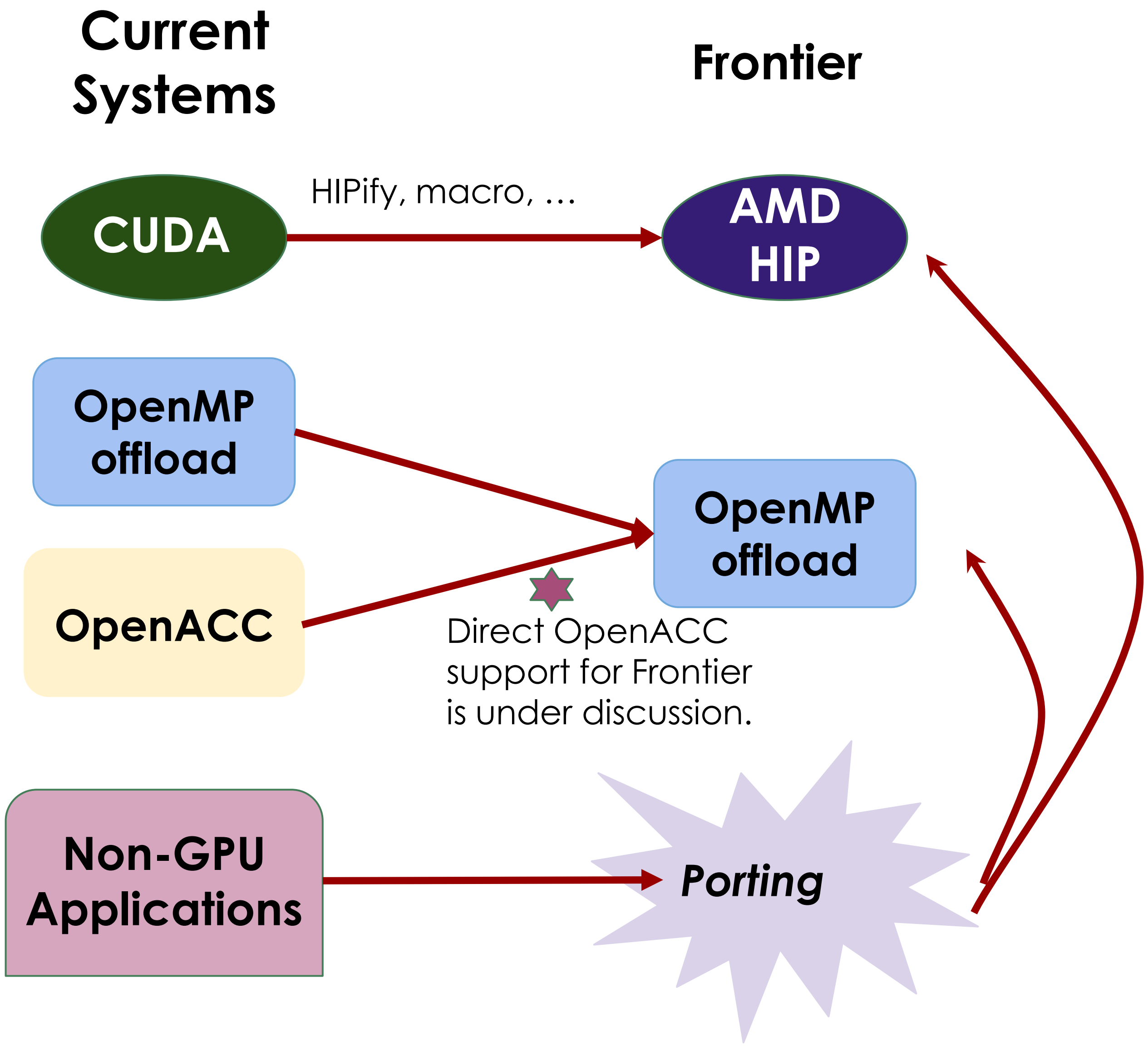
Early Porting Experience on Crusher: Cholla and GenASiS

Reuben D. Budiardja
Computational Scientist
Science Engagement: Advanced Computing for Nuclear, Particles, and Astrophysics
Oak Ridge Leadership Computing Facility (OLCF)
Oak Ridge National Laboratory (ORNL)

Programming for Frontier's GPUs

- The bulk of Frontier's computational power comes from its GPUs.
- Applications must exploit GPUs to fully realize Frontier's capability.
- Several programming models for GPUs are supported.

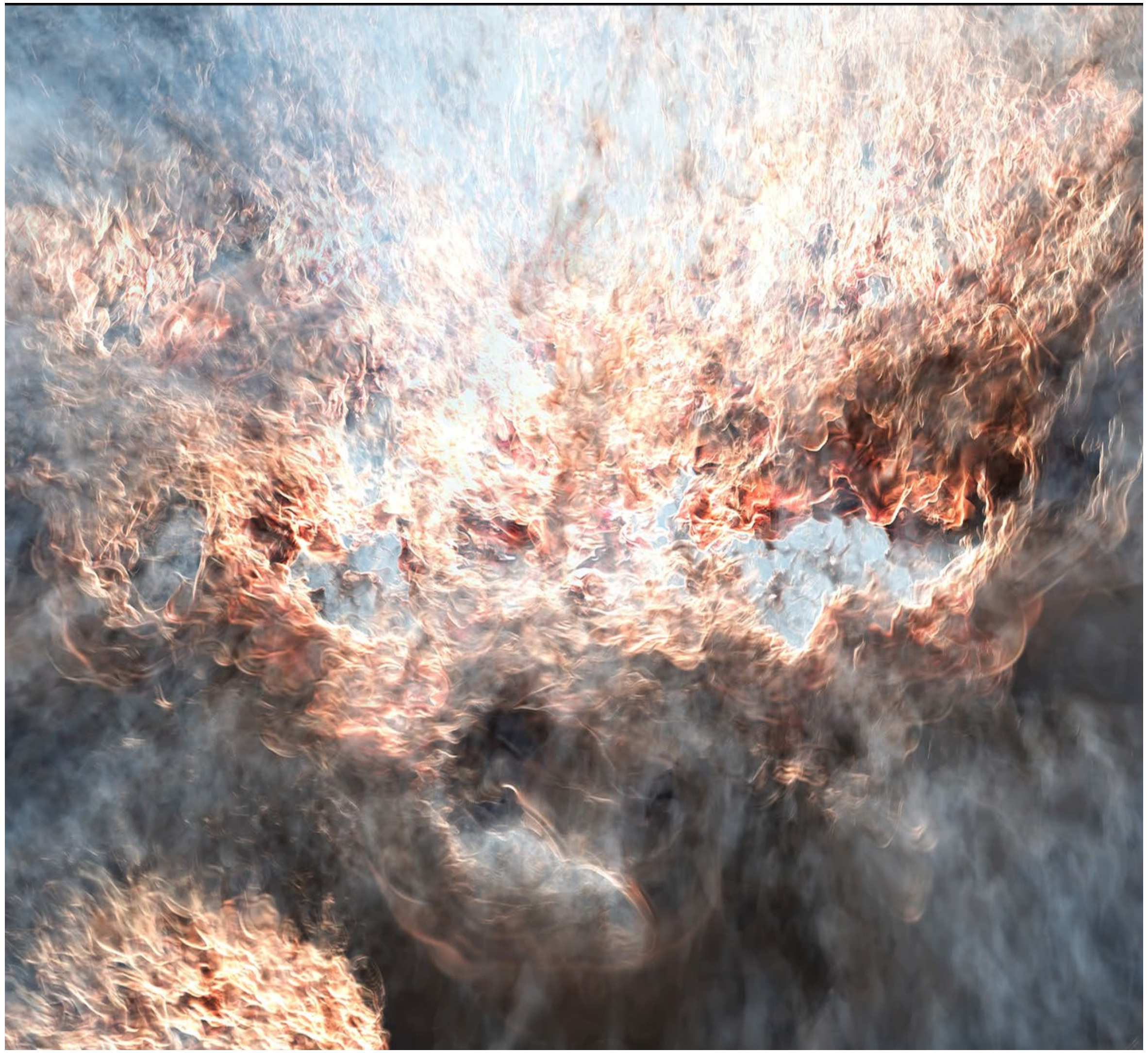


Cholla

Cholla (Computational Hydrodynamics on Parallel (| | |) Architecture) is a hydrodynamics code developed to natively run on GPU using CUDA / HIP.

Features:

- Fully native GPU support
- 1st, 2nd, 3rd order space spatial reconstruction
- Exact, Roe, HLLC Riemann Solvers
- CTU & Van Leer integrator
- Optically-thin radiative heating / cooling
- Additional (non-hydro) features:
  - self-gravity
  - particle-based gravity (for dark matter)
  - magnetic-fields (in development)



CHOLLA Challenge problem on Frontier:

Simulations of a Milky Way-like galaxy at resolutions that allow for self-consistent star formation and feedback within a multiphase interstellar medium.

- Milky Way diameter: ~160,000 light years = ~50,000 parsecs
- Resolutions to resolve star clusters: ~a few parsecs
- **Target resolutions on Frontier: ~10,000^3 cells**

Cholla is a research project within Frontier Center for Accelerated Application Readiness (CAAR) program.

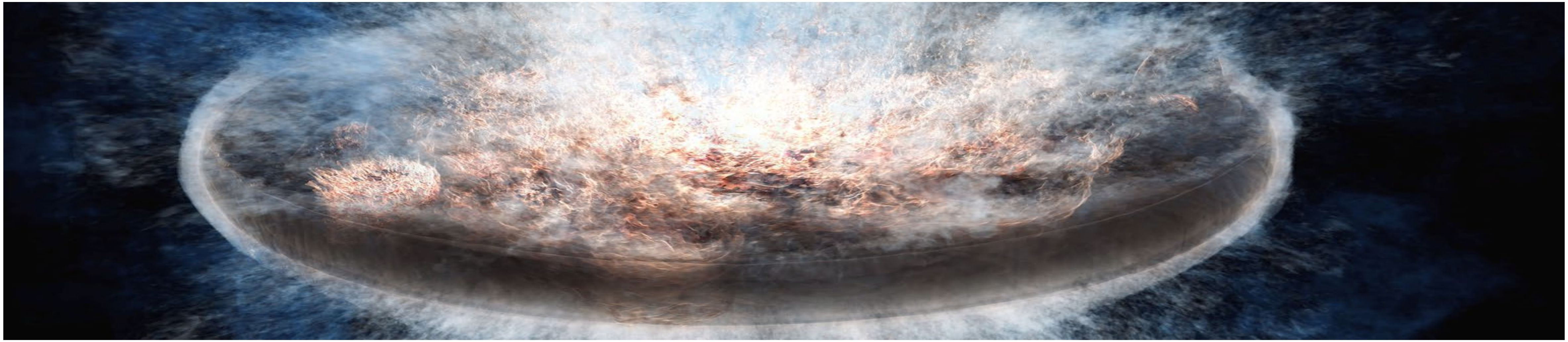


Figure of Merit (FOM) Definition

In general

$$FOM = \sum_{p=1}^{N} \left(C_p \frac{nCells \times nCycles}{Walltime} \right)_p$$

where the sum is over physics modules (e.g. hydrodynamics, self-gravity, particle, ...) involved and C_p is the cost coefficient for each physics module.

With all the requisite physics included, the FOM simplifies to

$$FOM = \frac{nCells \times nCycles}{Walltime}$$

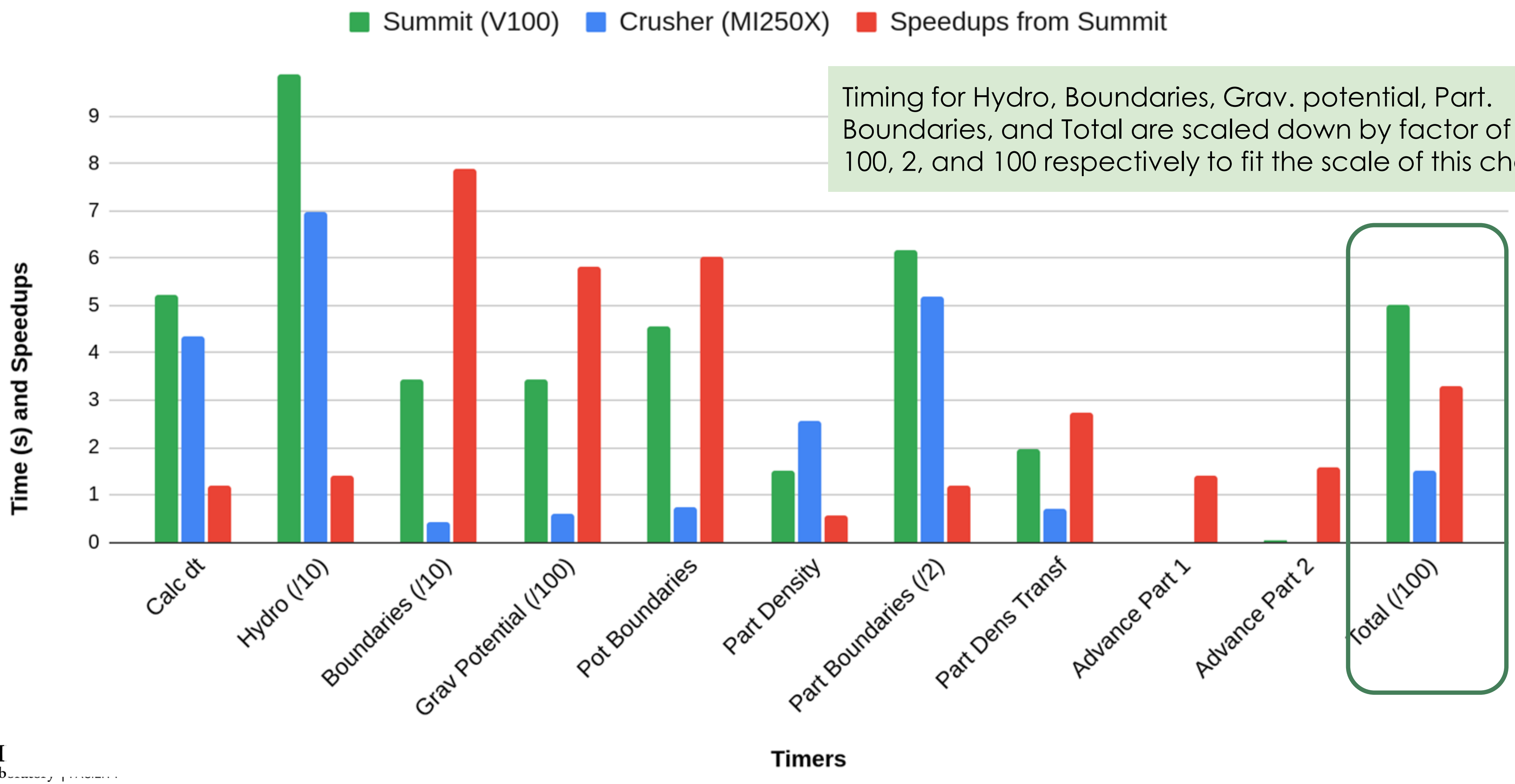
Porting Cholla to Frontier

```
#pragma once
...
#ifdef O_HIP
...
#define cudaDeviceSynchronize hipDeviceSynchronize
#define cudaError hipError_t
#define cudaError_t hipError_t
#define cudaErrorInsufficientDriver hipErrorInsufficientDriver
...
```

- "HIP-i-fly" (HIP on-the-fly) code to maintain CUDA portability
  - a small preprocessor file to translate CUDA calls to HIP calls
- Leverage GPU-attached NICs and GPU-aware MPI:
  - Keep grid (hydro) data GPU-resident → avoid more costly host-device data movement
  - Communicate boundary data directly from GPU memory
- Developed new FFT-based Poisson Solver ("Paris"):
  - replaced the original CPU PFFT based solver
  - completely on GPU with direct GPU-MPI communication, utilizing rocFFT library
- Ported particle evolution to GPU (with HIP)

CHOLLA Results with 64 GPUs (GCDs)

Crusher is a system for test and development that contains identical hardware and similar software as the upcoming Frontier system.

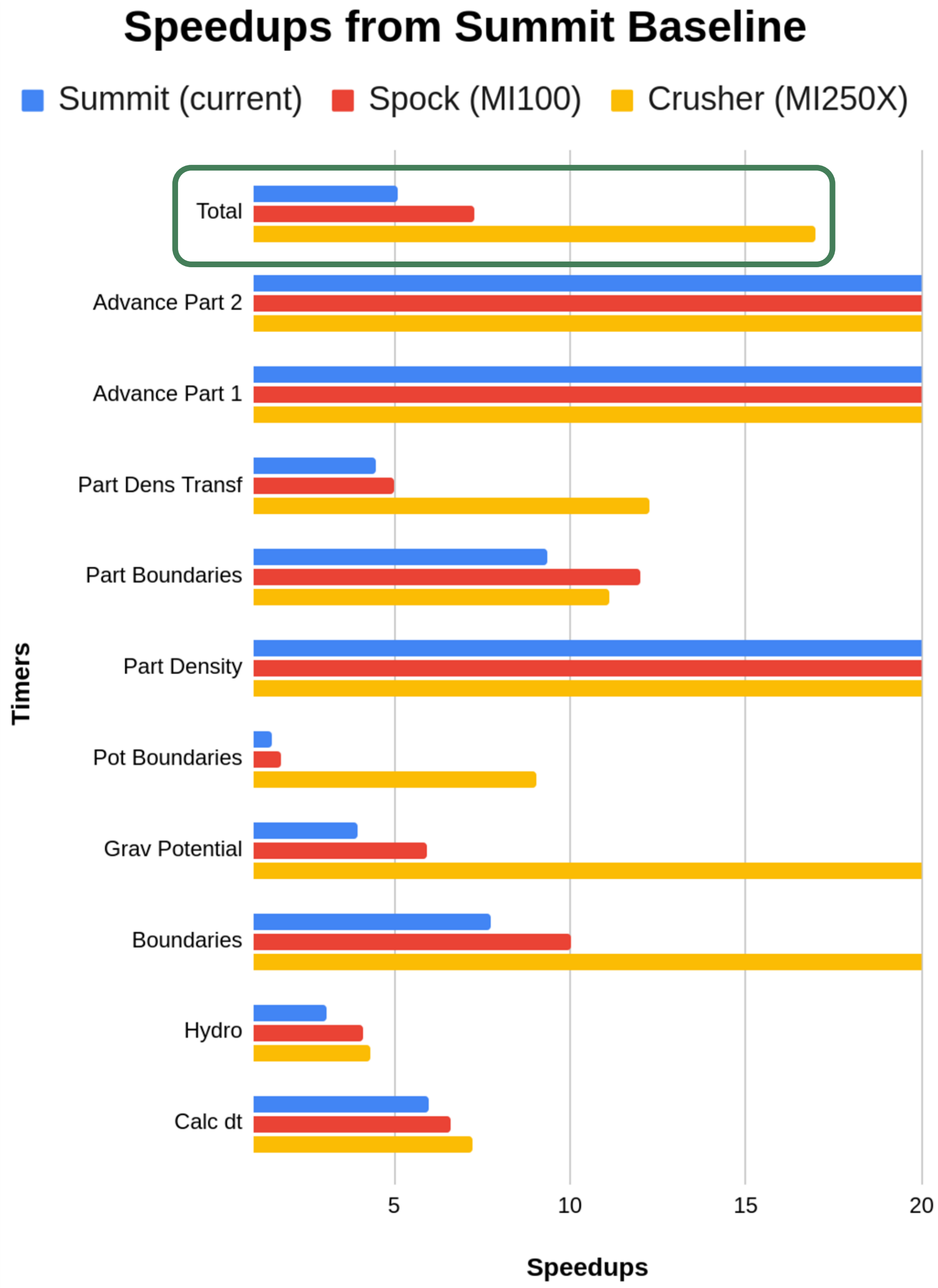


Timing for Hydro, Boundaries, Grav. potential, Part. Boundaries, and Total are scaled down by factor of 10, 10, 100, 2, and 100 respectively to fit the scale of this chart.

OAK RI National Lab

Cholla Status

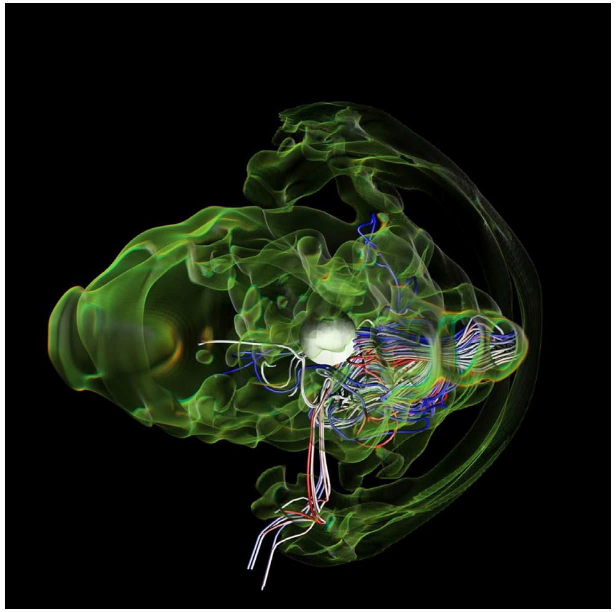
- Total speedups: **~16X** on 64 GPUs on Frontier hardware (Crusher) from baseline (see plot).
- Software development contributed to ~5X speedups on Summit (blue bars on the plot). Major highlights:
  - Made hydro grid fully GPU resident
  - Exploited GPU-aware MPI
  - Ported gravity solver to GPU
  - Ported particle solver to GPU
- Hardware improvements from Summit to Crusher: ~3X speedups
- Pending: Scaling up to the full Frontier



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

GenASiS

- ***Gen***eral ***A***strophysics ***Si***mulation ***S***ystem
  - Challenge problem: 3D position space + 3D momentum space the simulations of core-collapse supernovae (sustained exascale)
  - Earlier versions have been used to study of fluid instabilities in supernova dynamics, discover exponential magnetic field amplification in progenitor star

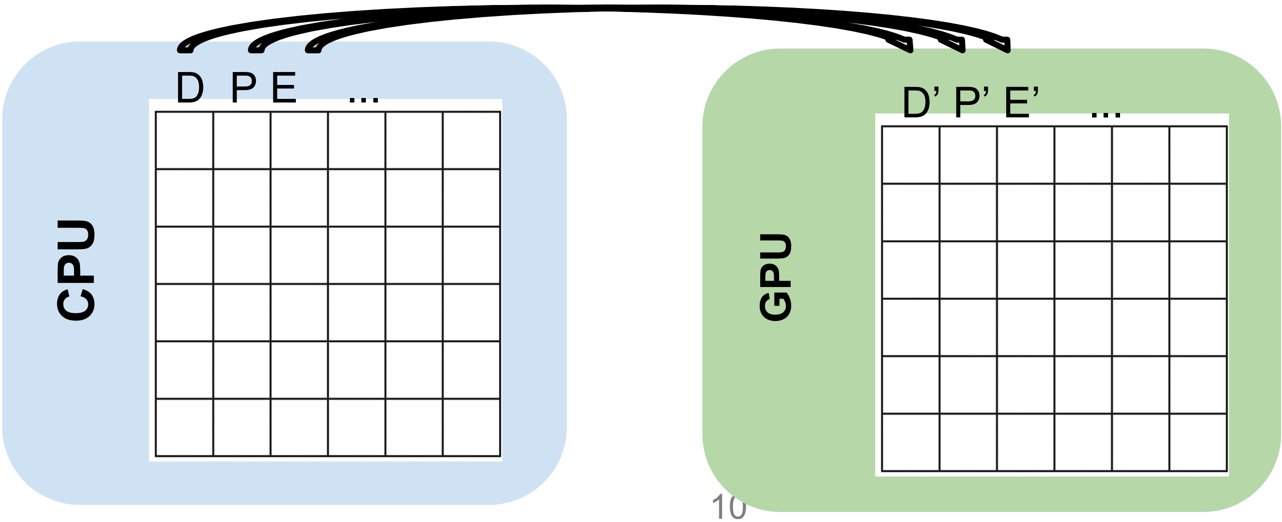


- **Code characteristics:**
  - Modern Fortran (mostly F2008, some F2018)
  - Modular, object-oriented design, extensible
  - OpenMP for threading + offloading

StorageForm Class

A class for **data** and **metadata**; the ‘**workhorse’** of data storage facility in GenASiS. Simplified tasks like I/O, ghost cell exchange, prolongation & restrictions, etc.

```
StorageForm % Value ( nCells, nVariables )
e.g. Pressure => StorageForm % Value ( :, 1 ),
     Density  => StorageForm % Value ( :, 2 ), ...
```




```
StorageForm % AllocateDevice ( )
```

- allocate on GPU
- map CPU and GPU variables

Informs OpenMP data location on GPU
→avoid (implicit) allocation & transfer

Offloading a Computational Kernel

Persistent **allocation** and **association**

```
call F % Initialize &
  ([nCells, nVariables])
call F % AllocateDevice ( )
call F % UpdateDevice ( )
call AddKernel &
  ( F % Value ( :, 1 ),
    F % Value ( :, 2 ), &
    F % Value ( :, 3 ) )
```

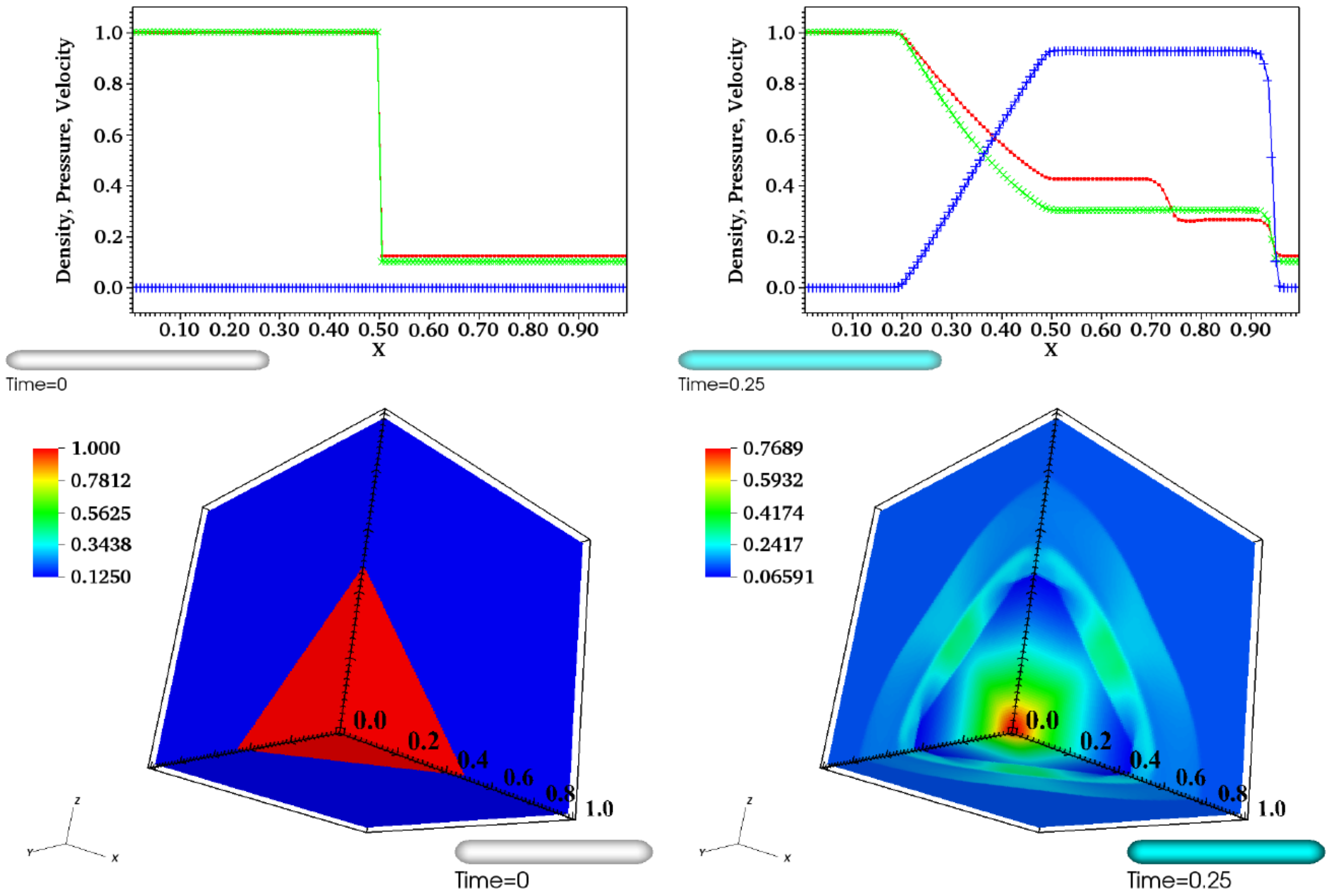
```
subroutine AddKernel ( A, B, C )

  real ( KDR ), dimension ( : ), intent ( in ) :: A, B
  real ( KDR ), dimension ( : ), intent ( out ) :: C

  integer ( KDI ) :: i

  !$OMP target teams distribute parallel do
  do i = 1, size ( C )
    C ( i ) = A ( i ) + B ( i )
  end do
  !$OMP end target teams distribute parallel do

end subroutine AddKernel
```

No implicit data transfer,
no explicit **map()**

Porting GenASiS to Frontier (Crusher)

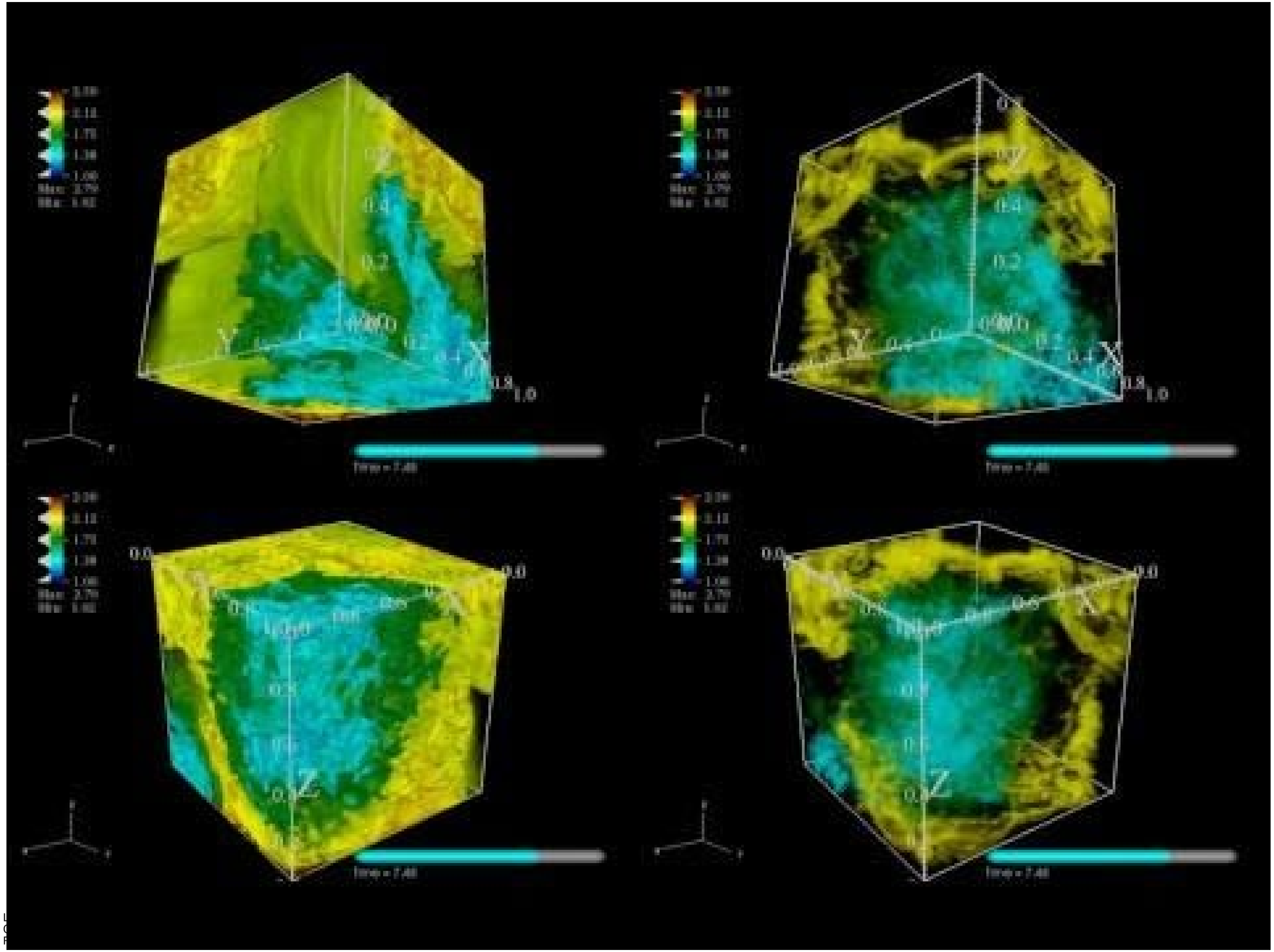
- A new Makefile for Crusher. The End. Well, almost …
- Differences in compiler mapping of parallelism to hardware:
  “`!$omp target teams distribute parallel do simd`”
- Initially needed explicit map for reduction variable
  - OpenMP 5 spec addressed this issue but was not implemented
  - Subsequently this was fixed in compiler (CCE)

GenASiS Basics RiemannProblem

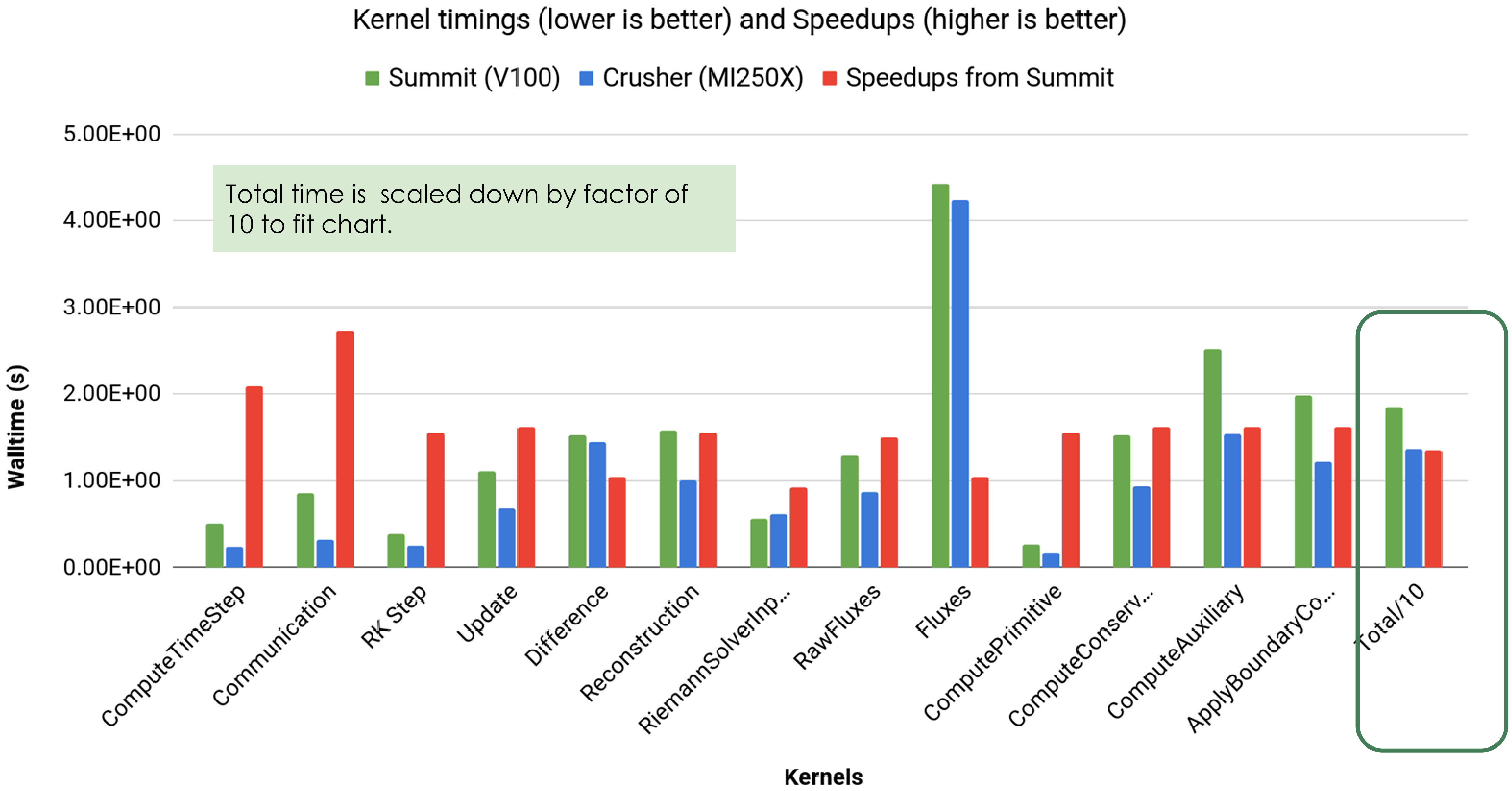


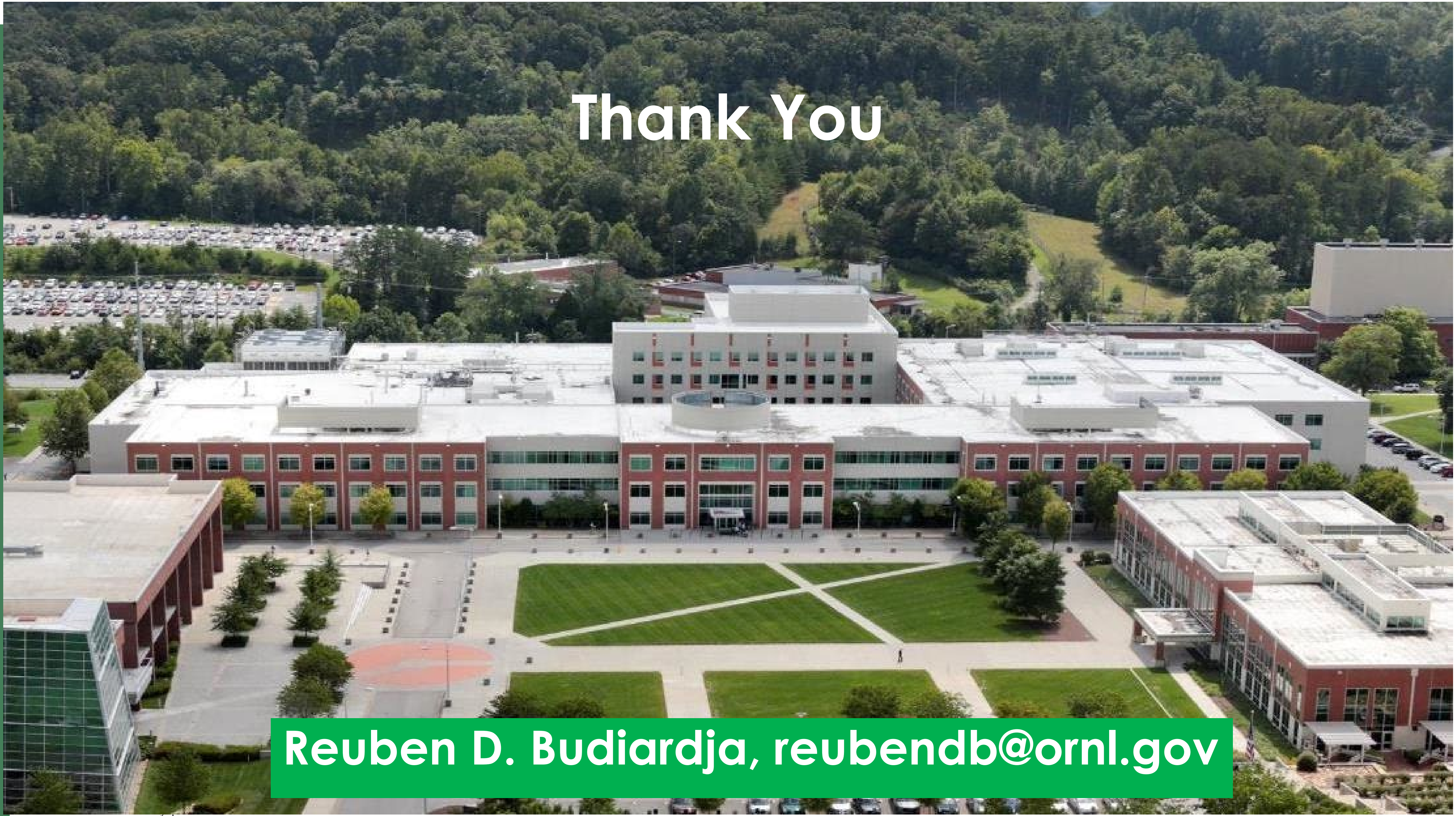
Initial (left) and final (right) density of 1D and 3D RiemannProblem

- A simplified version of divergence solvers without the sophistication of multi-patches meshes and other physics modules (self-gravity, radiations, nuclear EoS, …)
- HLL / HLLC Riemann solver with 2nd order RK time integration
- A standard benchmark problem with shocks in fluid dynamics, extended to 3D
- Our workhorse *proxy-application* for experimentation and testing (new platforms, compilers, …)


Y
X
Z
0.2
0.4
1.0
0.0

GenASiS Basics: RiemannProblem 3D, 256^3 per GPU, 64 MPI Tasks, 50 cycles





Thank You

Reuben D. Budiardja, reubendb@ornl.gov