OAK RIDGE National Laboratory

Using OpenMP to Harness GPUs for Core-Collapse Supernova Simulations with GenASiS

OpenMP ARB Webinar
February 17, 2022

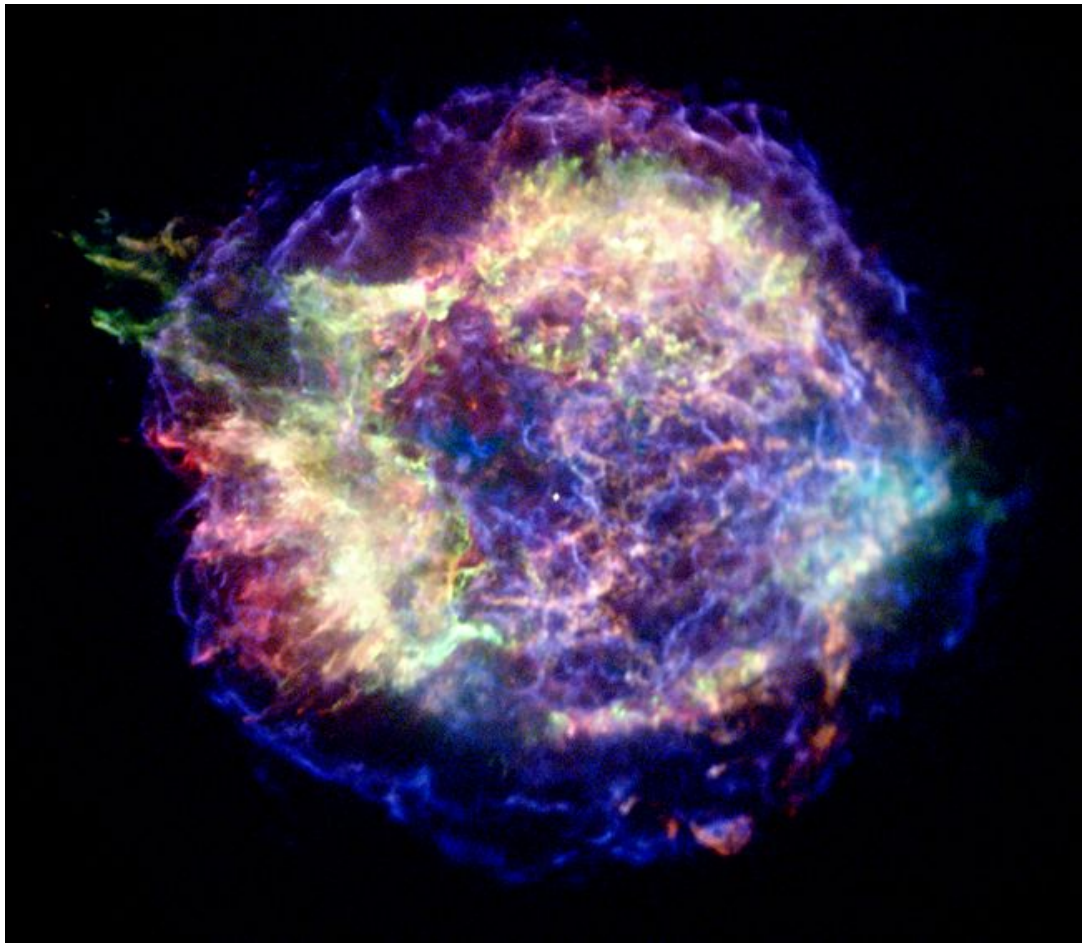
Reuben D. Budiardja
Computational Scientist
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

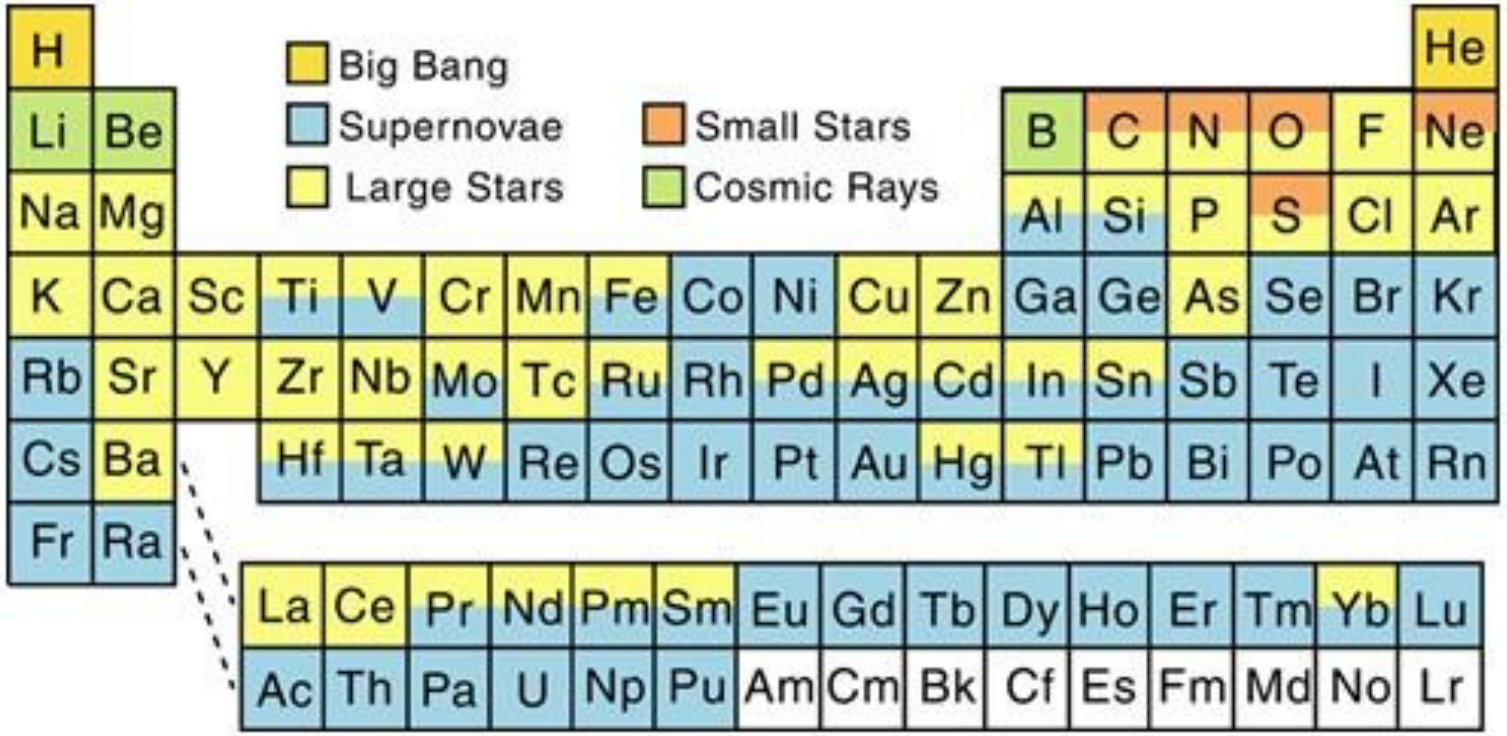
Core-Collapse Supernovae (CCSN)

- The death throes of massive star (M > ~10 Solar M)
  - The birth of neutron stars and black holes
- Among the most powerful explosions in the universe
  - ~10^{53} ergs of energy released as mostly neutrino
  - ~10^{51} ergs visible electromagnetic radiation
    ~ 10^{28} megatonnes of TNT
- Observables:
  - Gamma-ray burst, gravitational wave, neutrino
- Occur about twice per century in our galaxy

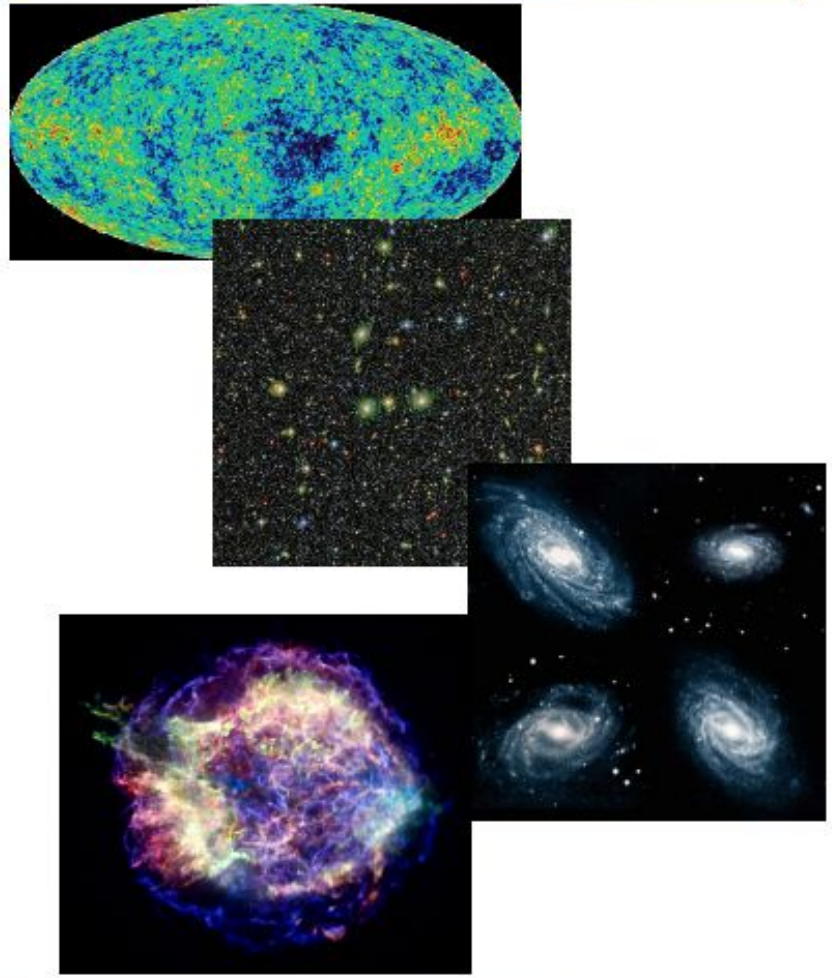


Cassiopeia A Supernova Remnant (Chandra Observatory)

Why Simulate Supernova?

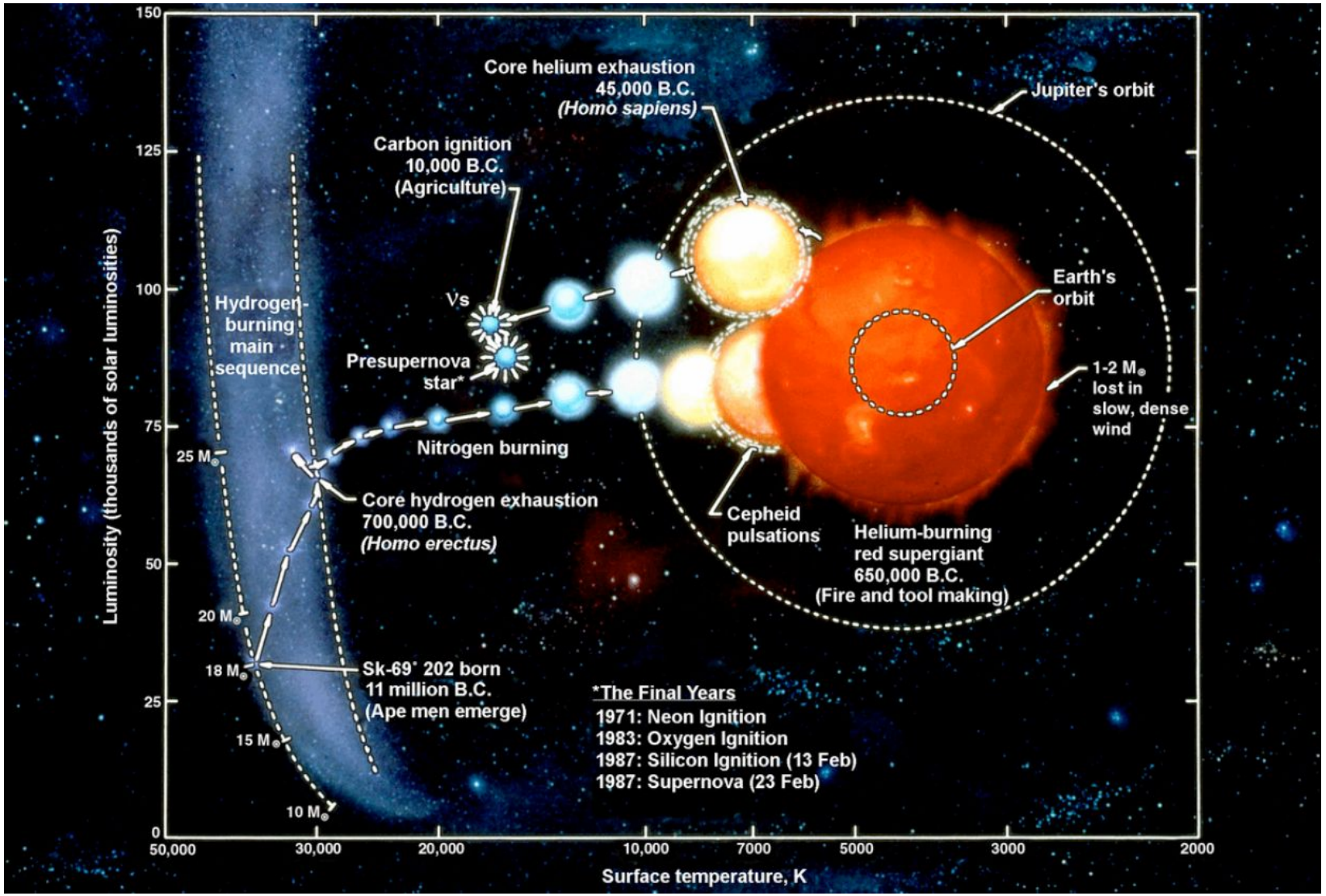


Understanding our universe and our place in it will require an understanding of phenomena on all scales.



Answers to questions relating to our origins in the universe

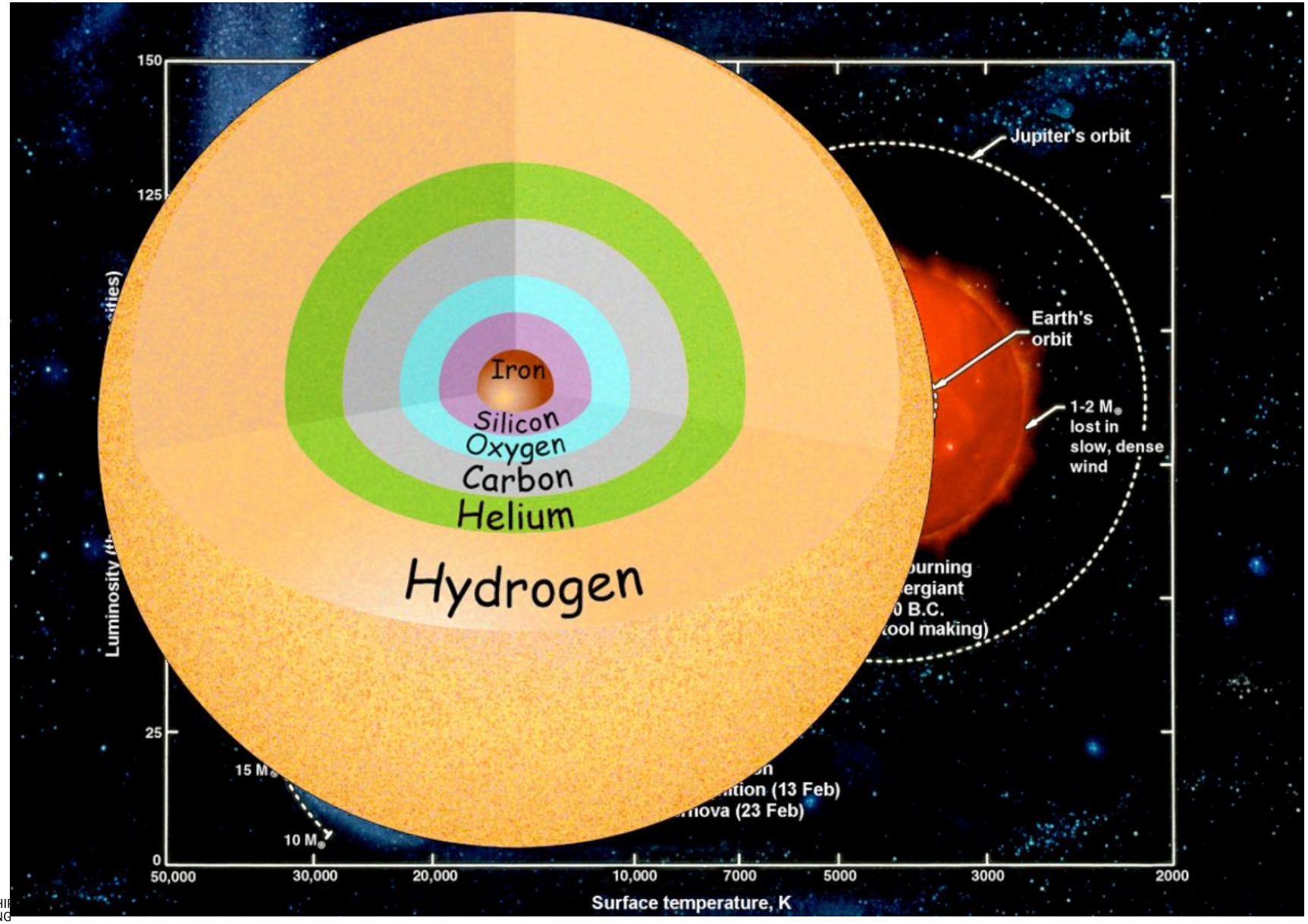
The Path to Explosion



Supernova 1987A

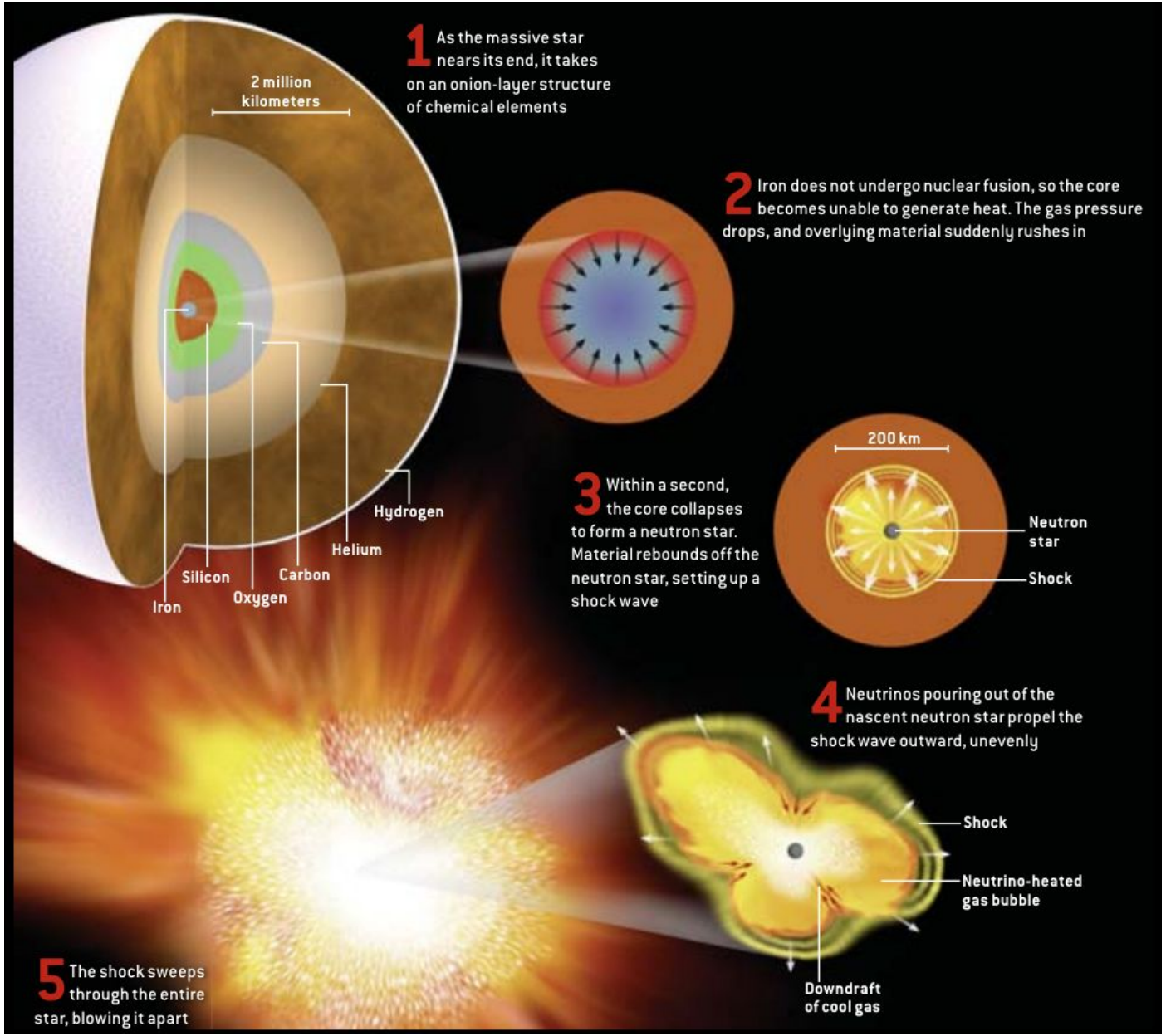
OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

The Path to Explosion



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

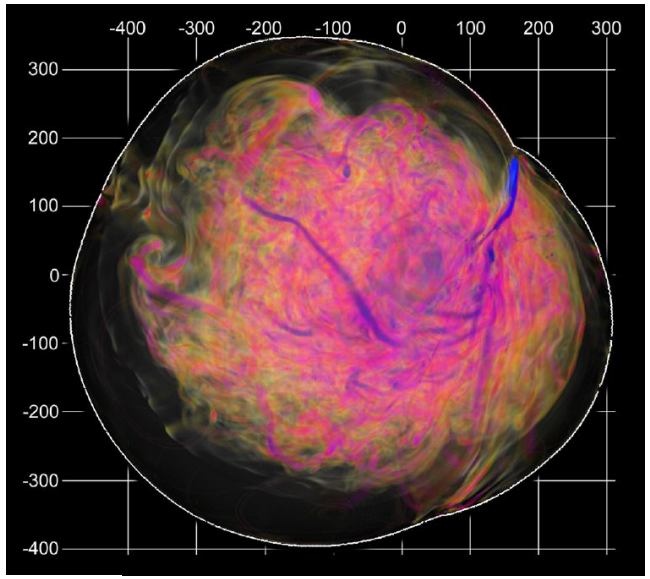
Textbook Supernova



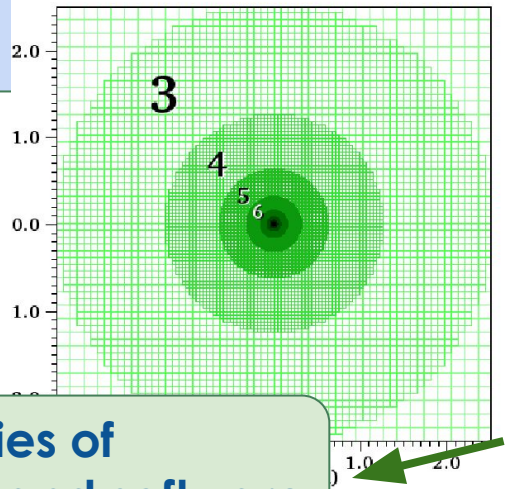
Understanding detail explosion mechanism requires high-fidelity simulations.

Supernova: Multi-Physics & Multi-Scale Problem

- General relativistic gravity
- Compressible Fluid
- Magnetic fields
- Convection
- Nuclear kinetic
- Dense equation of state
- Radiation transport



Daunting software integration tasks



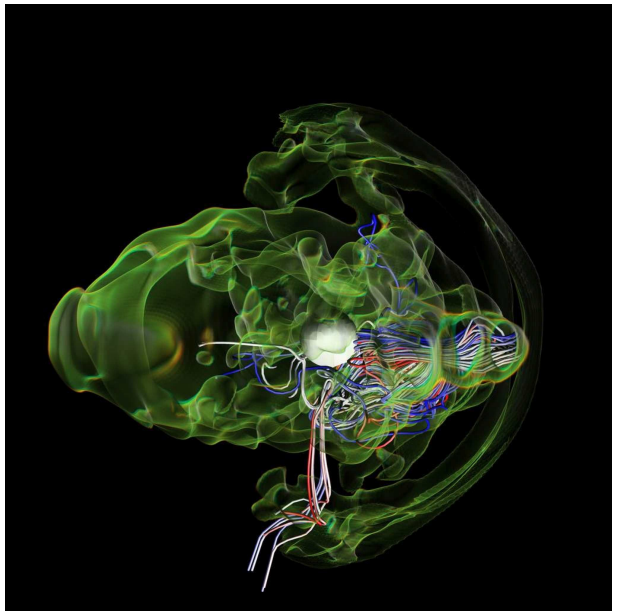
Pushing boundaries of computational hardware and software

- Collapse: ~ 10^6 density increase
- Proto-neutron star: < ~1 km scale
- Outer layer: ~ 10^3 km
- 3D → > 10^9 cells
- 10^{-6}s timestep to 1s total time
- Turbulent cascade
- Magnetorotational instability

Compute power requirements

GenASiS

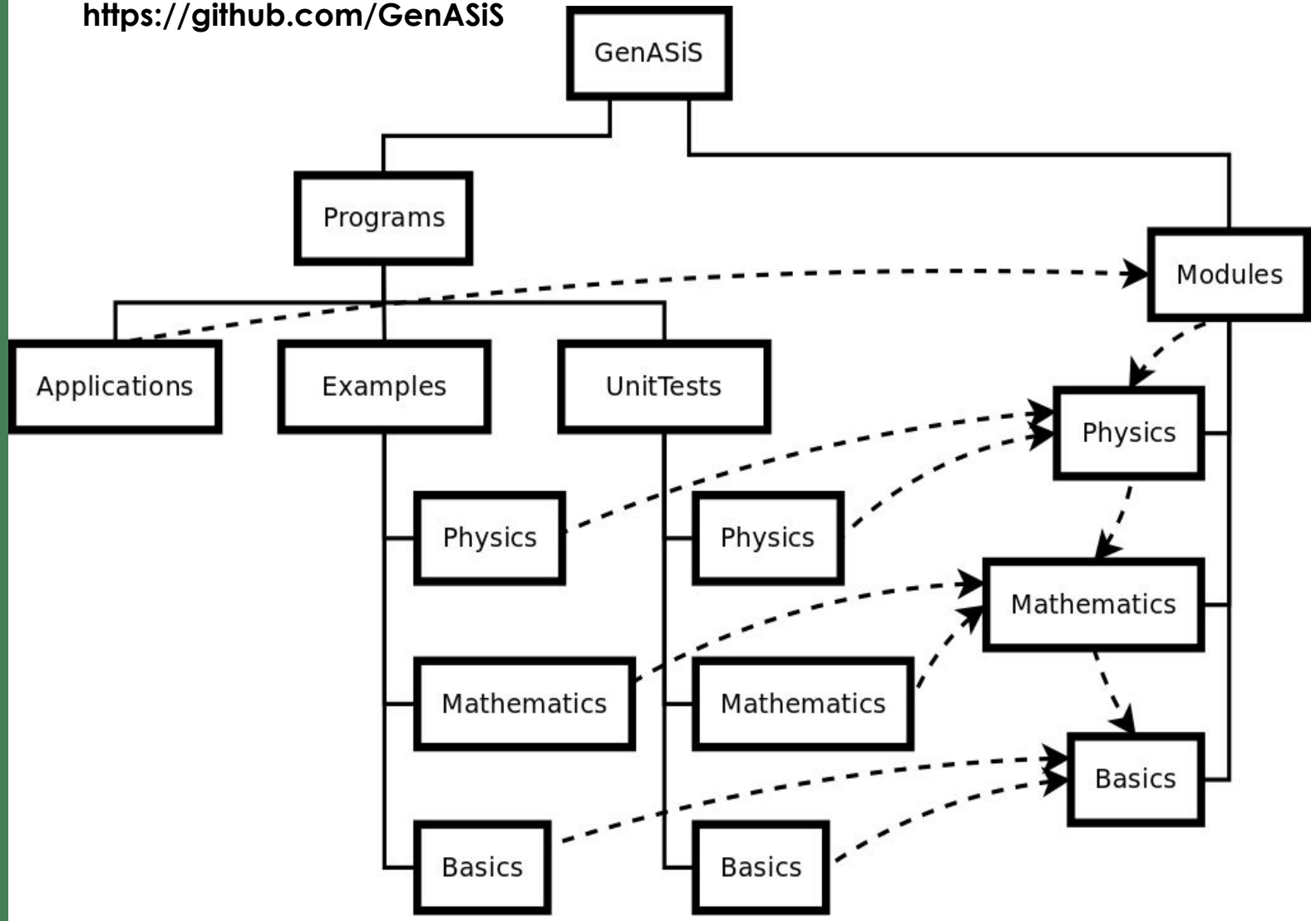
- ***Gen****eral* ***A****strophysics* ***Si****mulation* ***S****ystem*
  - Current target: 3D position space + 1D momentum space the simulations of core-collapse supernovae; Towards 3D + 3D (sustained exascale)
  - Earlier versions have been used to study of fluid instabilities in supernova dynamics, 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

GenASiS Structure

https://github.com/GenASiS



Basics: Utilitarian infrastructure, data management, I/O, Units, Devices, MPI facades

Mathematics: manifolds / meshes (cartesian, curvilinear), solvers (time-dependent ODE, PDE, elliptic)

Physics: fluid type, equation of states, stress-energy tensors, space-type (newtonian, relativistic, …)

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

GenASiS Storage Functionality

- StorageForm :
  - a class for data and metadata; the 'heart' of data storage facility in GenASiS
  - metadata includes units, variable names (for I/O, visualization)
  - used to group together a set of related physical variables (e.g. Fluid)
  - render more generic and simplified code for I/O, ghost exchange, prolongation & restriction (AMR mesh)

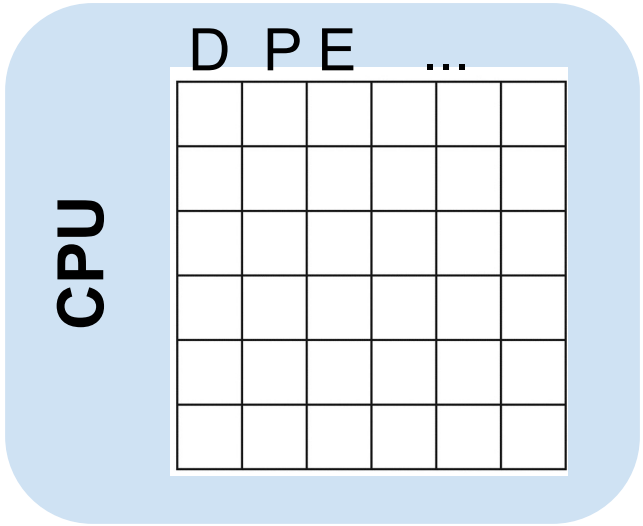
- Data :
  - `StorageForm % Value ( nCells, nVariables )`
  - use as, e.g.
```
Pressure => StorageForm % Value ( :, 1 ),
Density  => StorageForm % Value ( :, 2 )
```

- Methods:
  - `call S % Initialize ( )` ← **allocate** data on **host**
  - `call S % AllocateDevice ( )` ← **allocate** and **associate** data **on GPU**
  - `call S % Update{Device,Host} ( )` ← **transfer** data

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

GPU Data Allocation and Mapping

- call StorageForm % Initialize &
  (Shape = [6, 6], &
  VariableOption = [“Pressure”, “Density”, &
  “Energy”, …])



```
call S % AllocateDevice()
```

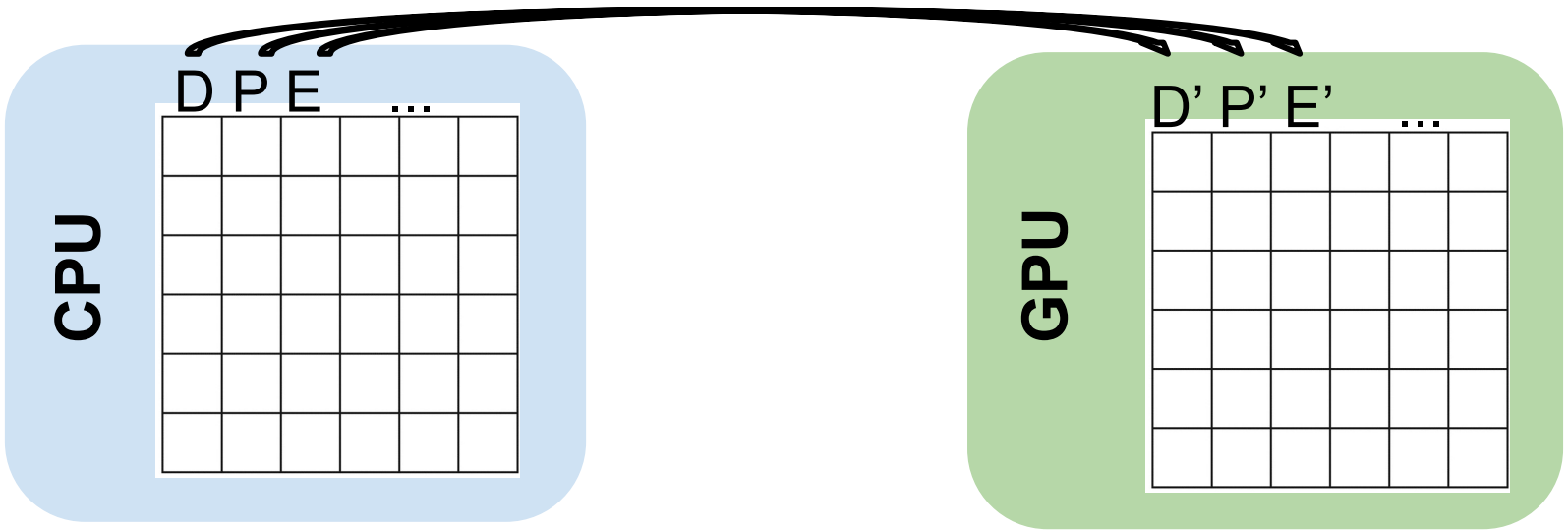
Two ways to *associate* ("***map***" in OpenMP) the host and device memory:

1. One mapping for each *variable* (column-wise) → *nVariables* mapping
2. One mapping for the whole (contiguous) block

Which mapping to use to use depends on how a kernel is written (more on this later).

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

```
call S % AllocateDevice ( AssociateVariables = .true. )
```

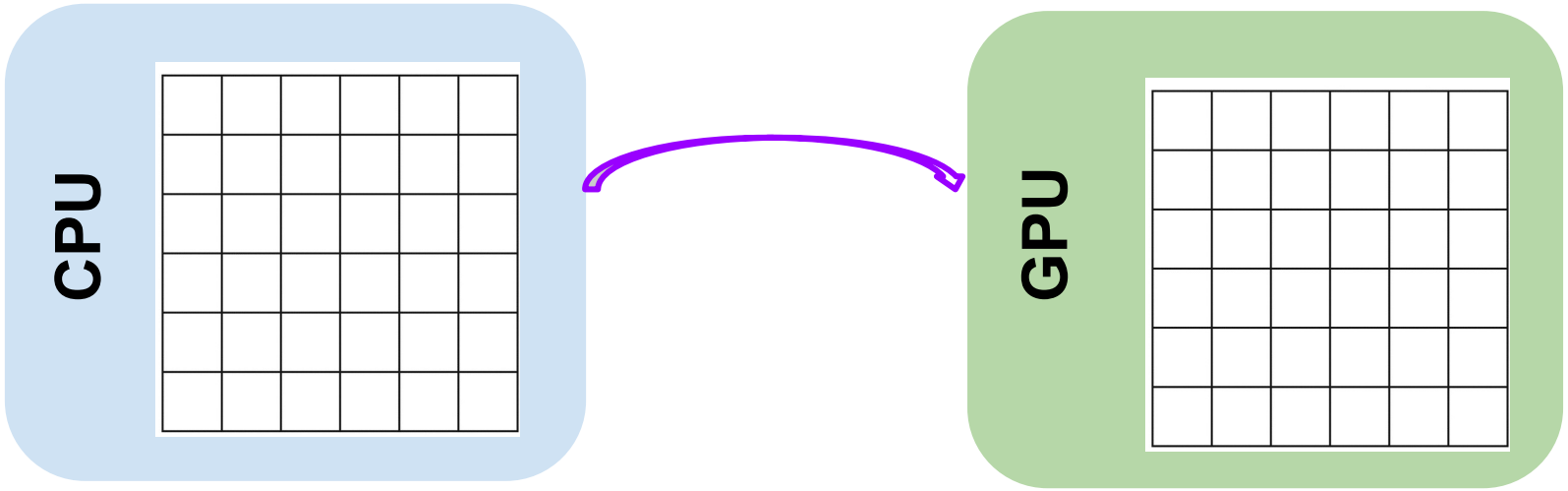



```
real ( KDR ), dimension ( :, : ), pointer :: Scratch
type ( c_ptr ) :: D_Value

call AllocateDevice ( S % nValues * S % nVariables, D_Value )

call c_f_pointer ( D_Value, Scratch, [ S % nValues, S % nVariables ] )

do iV = 1, S % nVariables
  D_Value = c_loc ( Scratch ( :, iV ) )
  Variable => S % Value ( :, iV )
  call AssociateHost ( D_Value, Variable )
end do
```

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

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

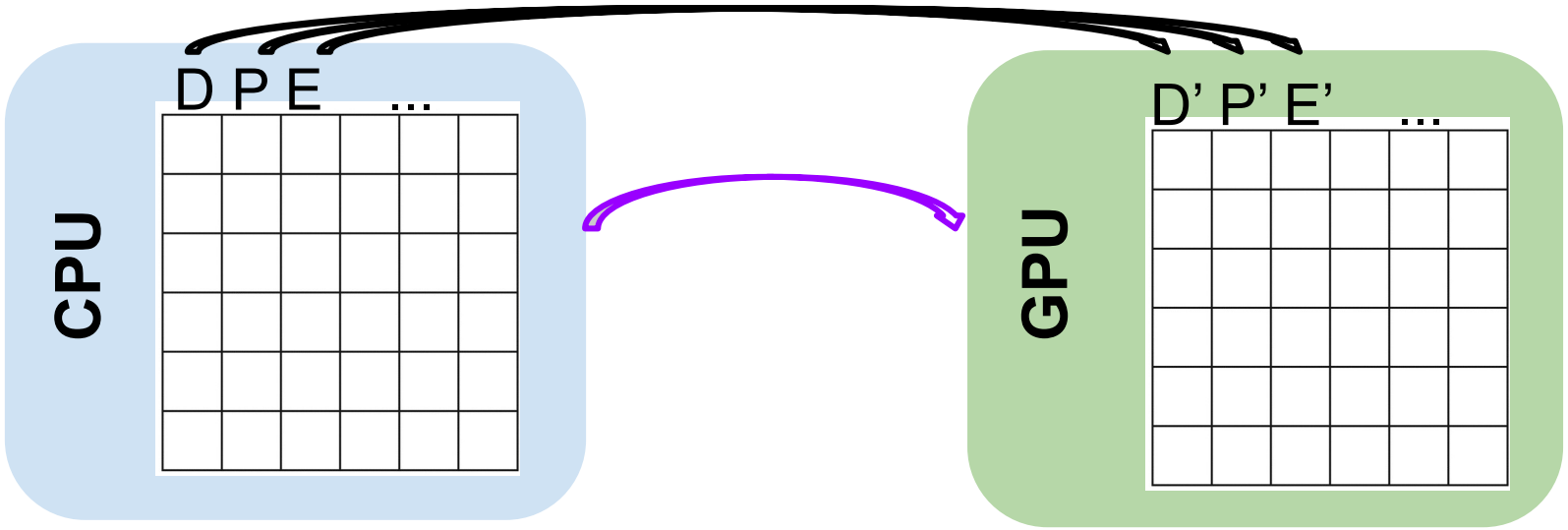
```
call S % AllocateDevice ( AssociateVariables = .false. )
```




```
real ( KDR ), dimension ( :, : ), pointer :: Scratch
type ( c_ptr ) :: D_Value

call AllocateDevice ( S % nValues * S % nVariables, D_Value )

call AssociateHost ( D_Value, S % Value )
```

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

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY



One can go back-and-forth between the two ways of mapping with:

```
call S % ReassociateHost ( AssociateVariables = [.true., .false.]  )
```


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Lower-Level GenASiS Functionality

- Fortran wrappers to OpenMP APIs
  - call AllocateDevice(Value, D_Value)
    → omp_target_alloc()

    call AssociateHost(D_Value, Value)
    → omp_target_associate_ptr()

    call UpdateDevice(Value, D_Value),
    call UpdateHost(Value, D_Value)
    → omp_target_memcpy()

    call ReassociateHost (…)
    → omp_target_disassociate_ptr (...), …
    omp_target_associate_ptr (h_ptr, d_ptr, totalSize, …)

Value is a Fortran array

D_Value is a **type(c_ptr)** object holding GPU address

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Offloading 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 ) )
```

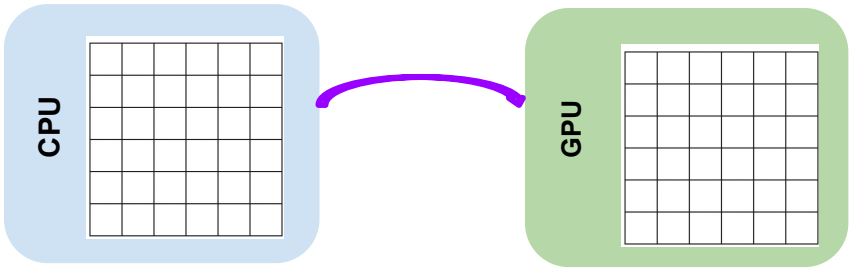
```
1  subroutine AddKernel ( A, B, C )
2
3    real ( KDR ), dimension ( : ), intent ( in ) :: A, B
4    real ( KDR ), dimension ( : ), intent ( out ) :: C
5
6    integer ( KDI ) :: i
7
8    !$OMP target teams distribute parallel do
9    do i = 1, size ( C )
10     C ( i ) = A ( i ) + B ( i )
11   end do
12   !$OMP end target teams distribute parallel do
13
14 end subroutine AddKernel
```

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

Offloading Computational Kernel

```
!-- F1, F2, F3 are StorageForm object

call [F1, F2, F3] &
       % ReassociateHost ( AssociateVariablesOption = .false. )
call SumVariableKernel (F1, F2, F3 )
```

```
2  subroutine SumVariableKernel ( V_1, V_2, V_3 )
3
4    real ( KDR ), dimension ( :, : ), intent ( inout ) :: V_1
5    real ( KDR ), dimension ( :, : ), intent ( in ) :: V_2, V_3
6
7    integer ( KDI ) :: iV, iS
8
9  !$OMP target teams distribute parallel do collapse ( 2 )
10 do iS = 1, size ( V_1, dim = 1 )
11   do iV = 1, size ( V_1, dim = 2 )
12     V_1 ( iS, iV ) = V_2 ( iS, iV ) + V_3 ( iS, iV )
13   end do
14 end do
15 !$OMP end target teams distribute parallel do
16
17 end subroutine SumVariableKernel
```



Example of Kernel with Pointer Remapping

```
1  subroutine ComputeDifference_X ( V, dV )
2
3    real ( KDR ), dimension ( -1:, -1:, -1: ), &
4      intent ( in ) :: &
5        V
6    real ( KDR ), dimension ( -1:, -1:, -1: ), &
7      intent ( out ) :: &
8        dV
9
10   integer ( KDI ) :: i, j, k
11
12   !$OMP target teams distribute parallel do collapse ( 3 ) schedule ( static, 1 )
13   do k = 1, nZ
14     do j = 1, nY
15       do i = 0, nX + 2
16         dV ( i, j, k ) &
17           = V ( i, j, k ) - V ( i - 1, j, k )
18       end do
19     end do
20   end do
21   !$OMP end target teams distribute parallel do
22
23 end subroutine ComputeDifferences_X
```

Caller, using *pointer remapping* feature:

```
real ( KDR ), dimension ( :, :, : ), &
  pointer :: V, dV

V ( -1:nX+2, -1:nY+2, -1:nZ+2 ) &
  => F % Value ( : , iV )

dV ( -1:nX+2 , -1:nY+2 , -1:nZ+2 ) &
  => dF % Value ( : , iV )

call ComputeDifferences_X ( V, dV )
```

Pointer Remapping + ReassociateHost ( )

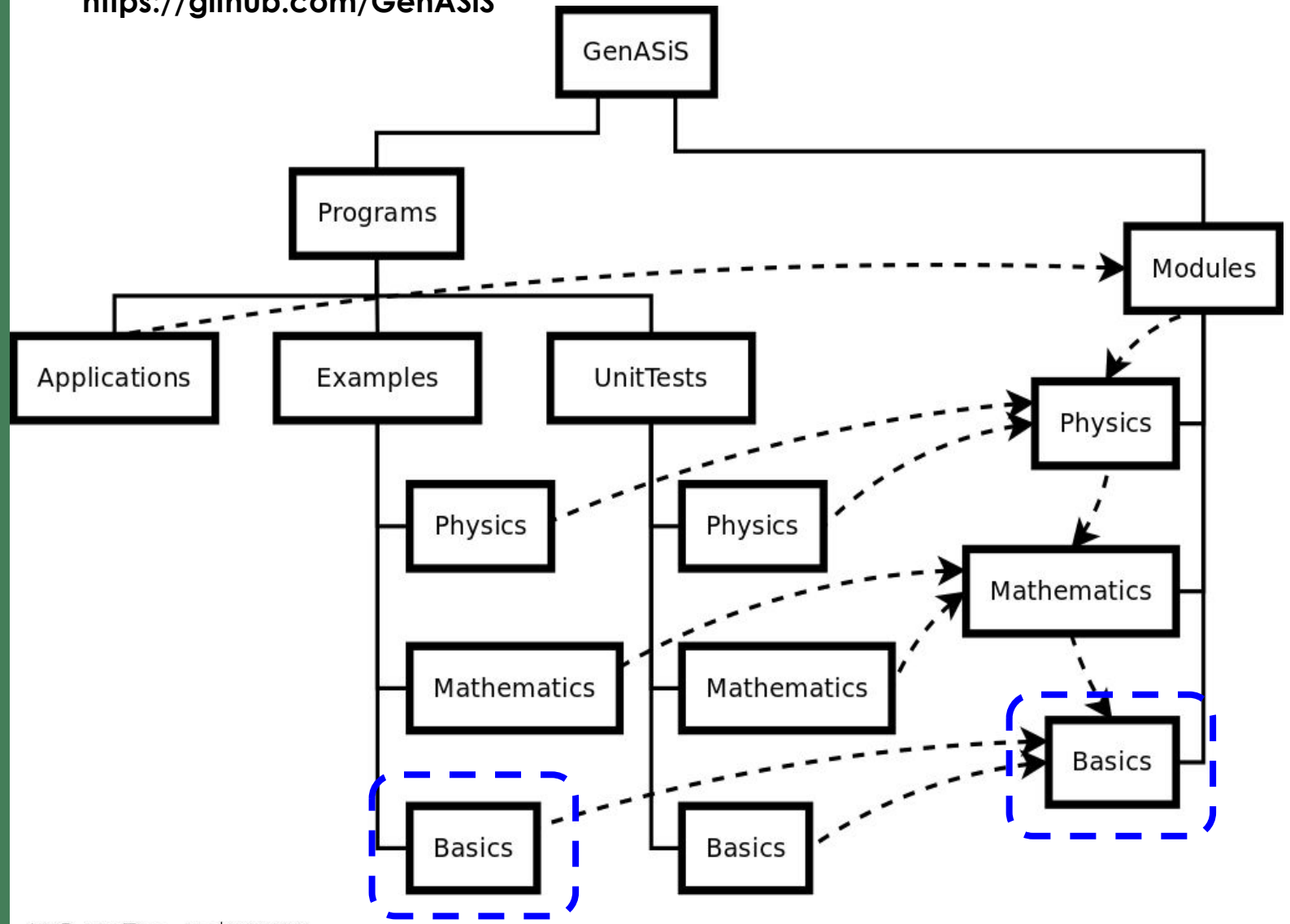
```
!$OMP target teams distribute parallel do collapse ( 4 ) &
!$OMP private ( iaVP )
do iS  =  1,  nS
  do kV  =  lV ( 3 ),  uV ( 3 )
    do jV  =  lV ( 2 ),  uV ( 2 )
      do iV  =  lV ( 1 ),  uV ( 1 )

        iaVP  =  [ iV, jV, kV ]  +  iaS

        S ( iV, jV, kV, iS )  &
          =  S ( iV, jV, kV, iS )  &
             -  (    A_I ( iaVP ( 1 ), iaVP ( 2 ), iaVP ( 3 ) )  &
                  *  F_I ( iaVP ( 1 ), iaVP ( 2 ), iaVP ( 3 ), iS )  &
                -  A_I ( iV, jV, kV )  &
                  *  F_I ( iV, jV, kV, iS ) ) &
               /  V ( iV, jV, kV )

      end do !-- iV
    end do !-- jV
  end do !-- kV
end do !-- iS
```

Caller, *pointer remapping* + ReassociateHost ()

```
call S % ReassociateHost ( .false. )
call Flux_I % ReassociateHost ( .false. )

S ( -1:nX+2, -1:nY+2, -1:nZ+2, 1:nF ) &
  => S % Value ( :, 1:nF )
F_I ( -1:nX+2, -1:nY+2, -1:nZ+2, 1:nF ) &
  => Flux_I % Value ( :, 1:nF )
A_I ( -1:nX+2, -1:nY+2, -1:nZ+2 ) &
  => G % Value ( :, AREA_INNER )
V ( -1:nX+2, -1:nY+2, -1:nZ+2 ) &
  => G % Value ( :, VOLUME )
```

GenASiS Structure

https://github.com/GenASiS

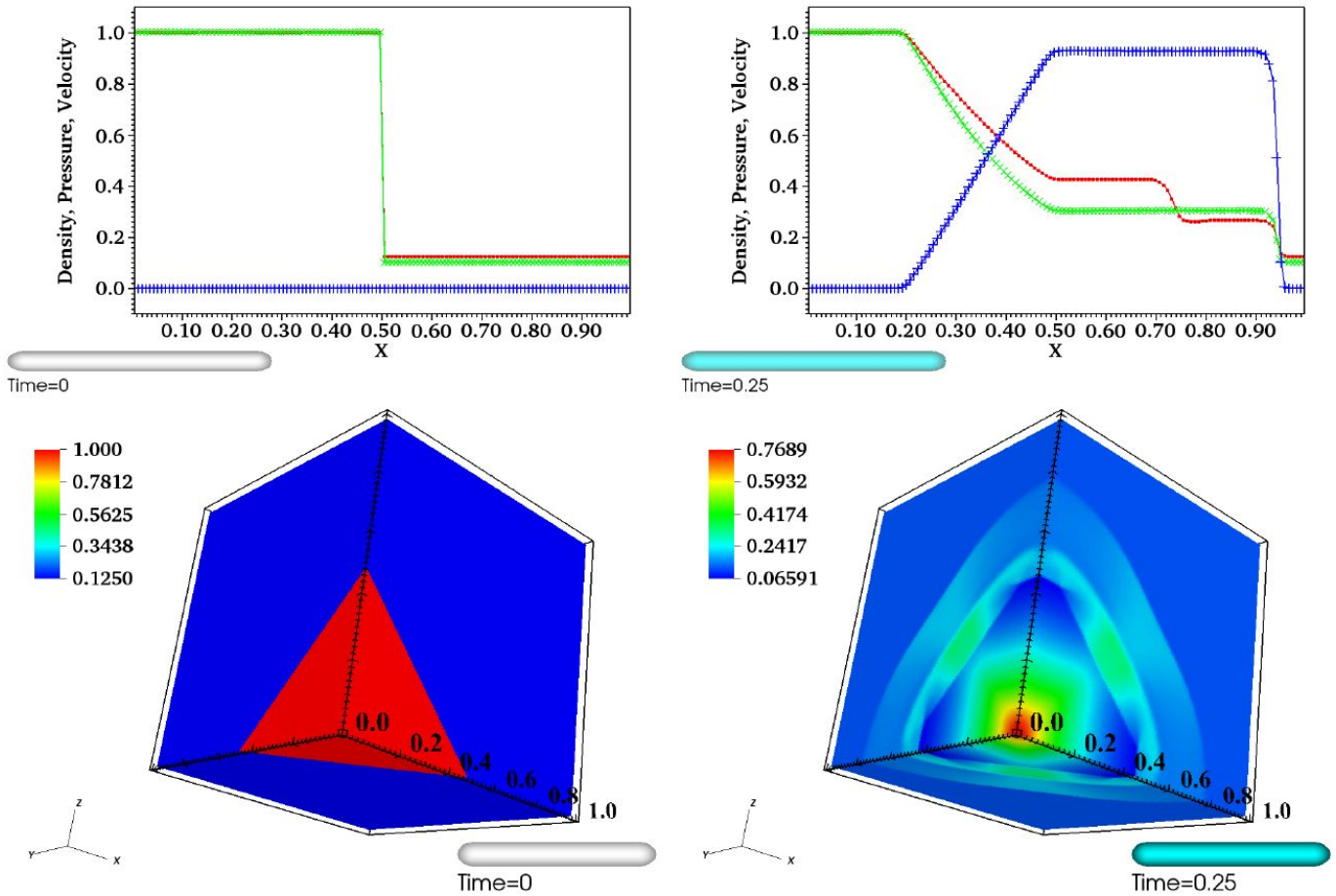


Basics RiemannProblem:

- An example problem using only *Basics classes* with simplified fluid solvers representative of higher-level solvers in (Mathematics, Physics)
- Useful for **testings & experimentations** with only a ***handful*** of computational kernels and simplified mesh (distributed cartesian)

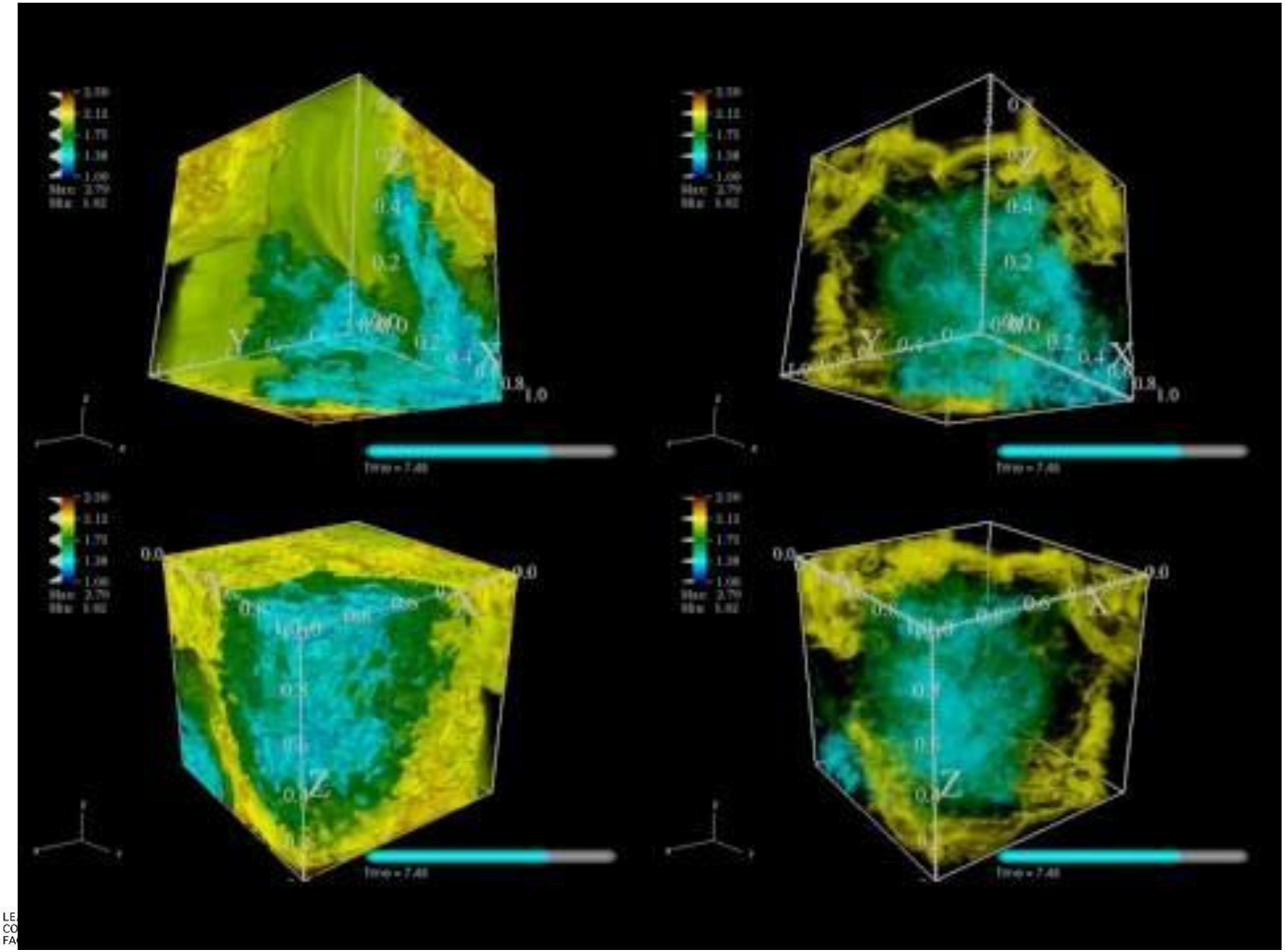
OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Basics Riemann Problem

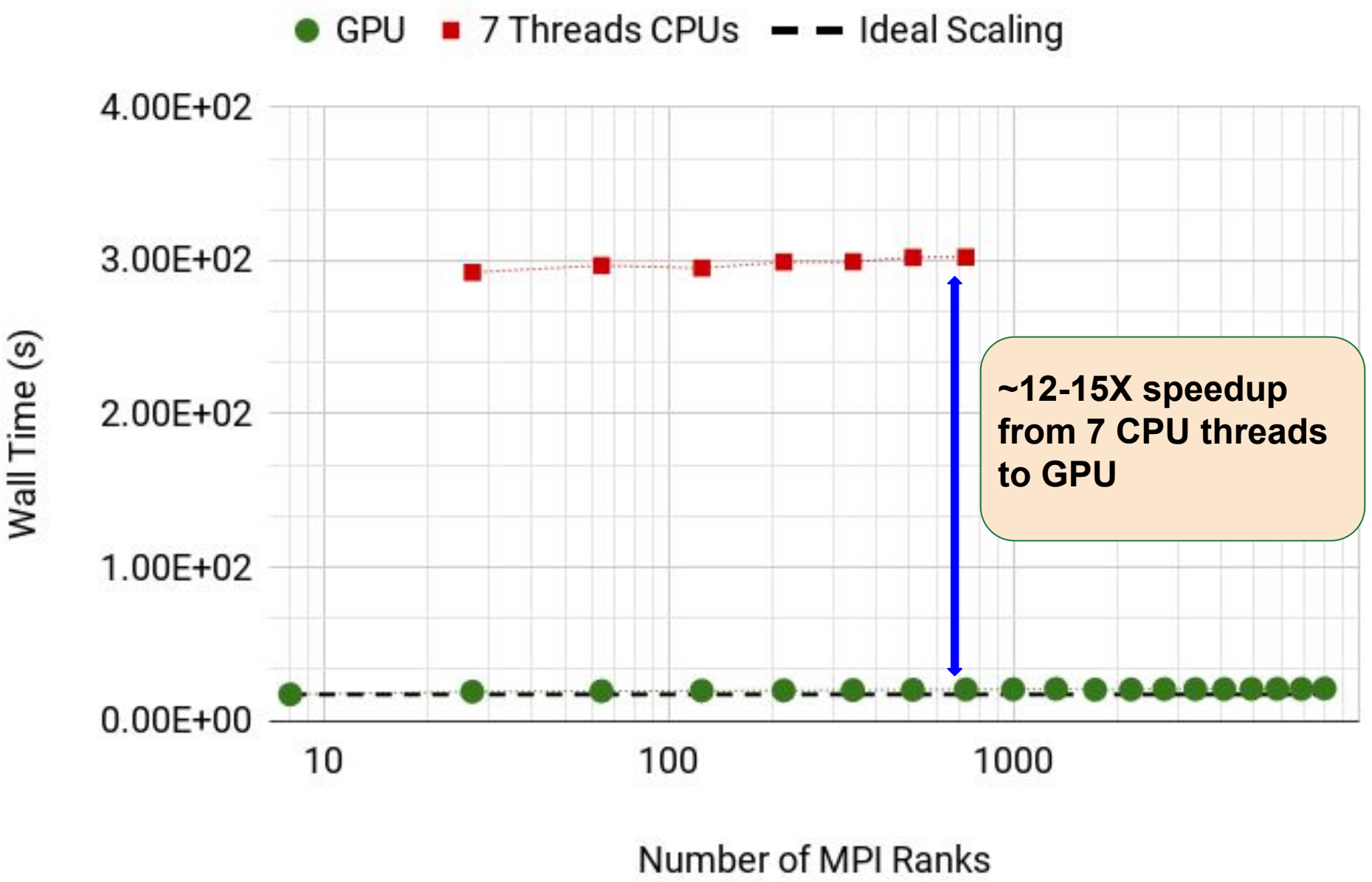


Initial (left) and final (right) density of 1D and 3D RiemannProblem

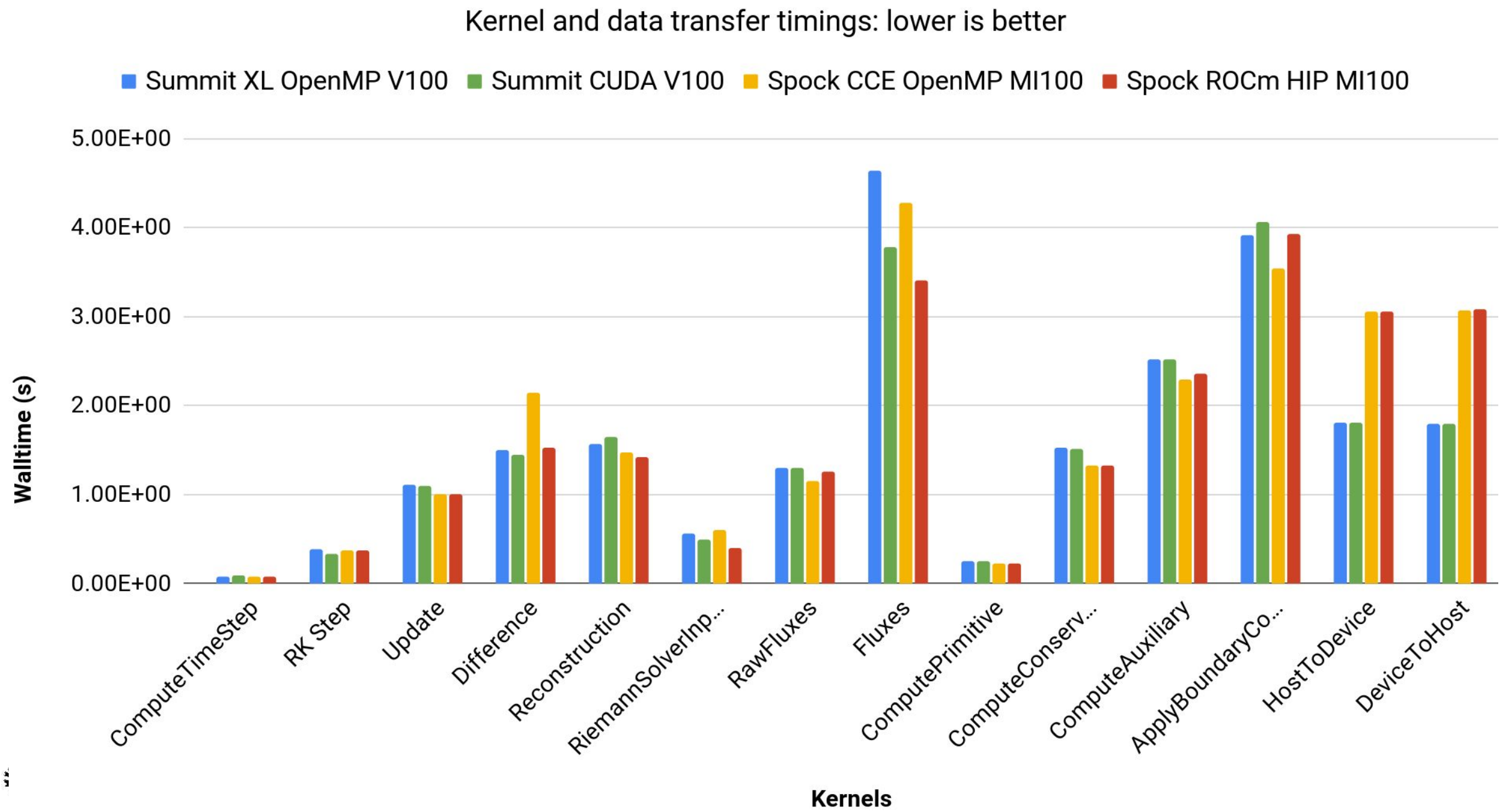
- A benchmark problem for hydrodynamics solvers.
- Similar solvers are used for the explicit part of radiation transport in an IMEX scheme
- Workhorse *proxy-application* for experimentation and testing, including (currently):
  - metadirectives
  - requires unified_shared_memory
  - new platforms / compilers
- Kernels for this problem have also been ported to CUDA / HIP for comparisons.


Y
X
Z
0.0
0.2
0.4
0.6
0.8
1.0

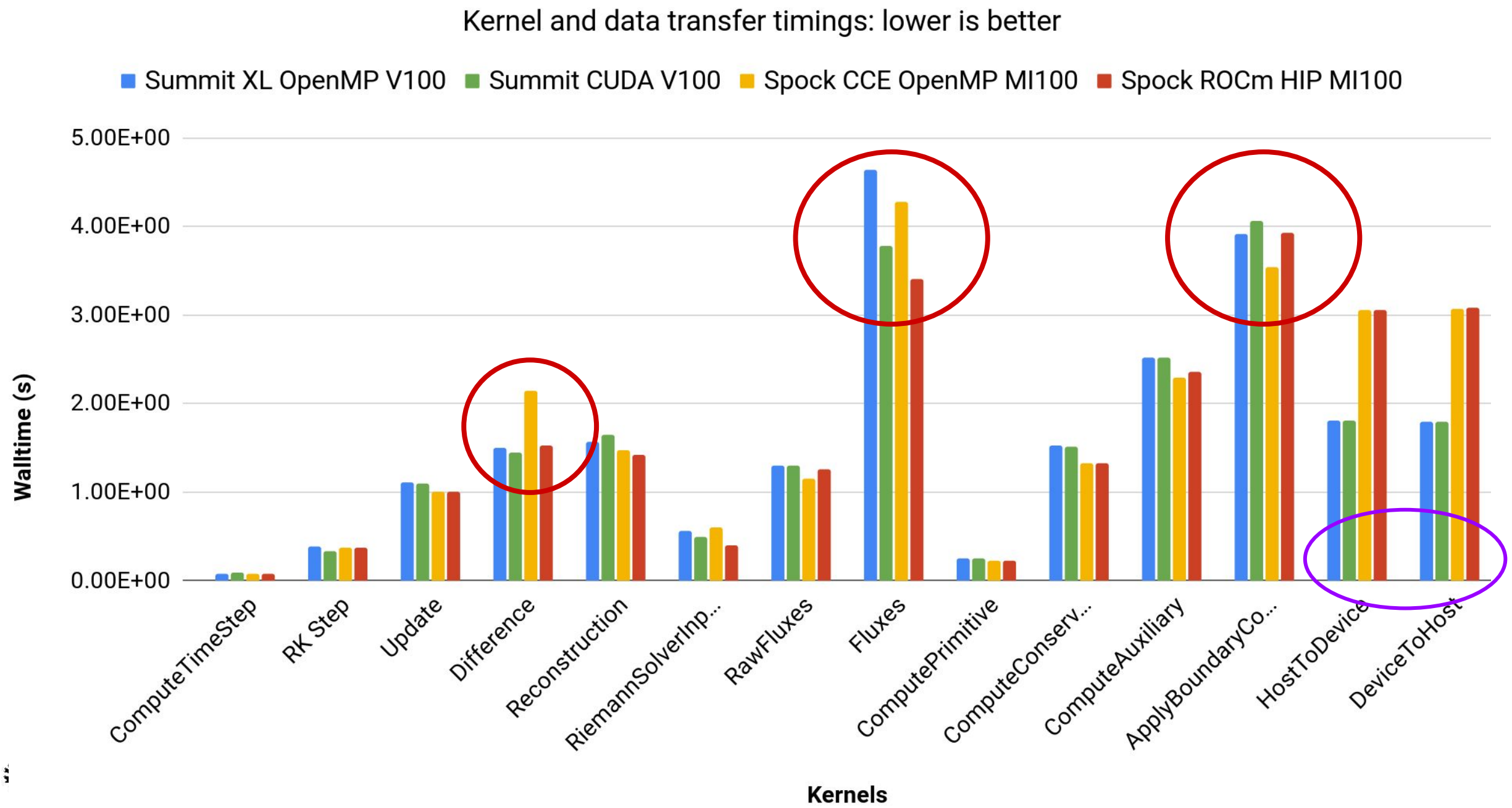
Weak Scaling & Speedups



GenASiS Basics: RiemannProblem 3D, 256^3 per GPU, 1 process, 50 cycles



GenASiS Basics: RiemannProblem 3D, 256^3 per GPU, 1 process, 50 cycles



Ongoing Work: Metadirective

- Example use case: evolve fluid on host + radiation on GPU + load balancing
- Currently code duplication needed for compiler to generate both device and host versions.
- Better: metadirective with user-selector and dynamic condition

```
if ( UseDevice ) then
  !$OMP target teams distribute parallel do &
  !$OMP private ( KE )
  do iV = 1, size ( E )
    [kernel code ...]
  end do
  !$OMP end target teams distribute parallel do
else
  !$OMP parallel do &
  !$OMP private ( KE )
  do iV = 1, size ( E )
    [duplicated kernel code ...]
  end do
  !$OMP end parallel do
end if
```

```
!$OMP metadirective &
!$OMP when ( user = { condition ( UseDevice ) } &
              : target teams distribute parallel do )&
!$OMP default ( parallel do ) &
!$OMP private ( KE )
do iV = 1, size ( E )
    [kernel code ...]
end do
```

Awaiting full compiler support for metadirective.

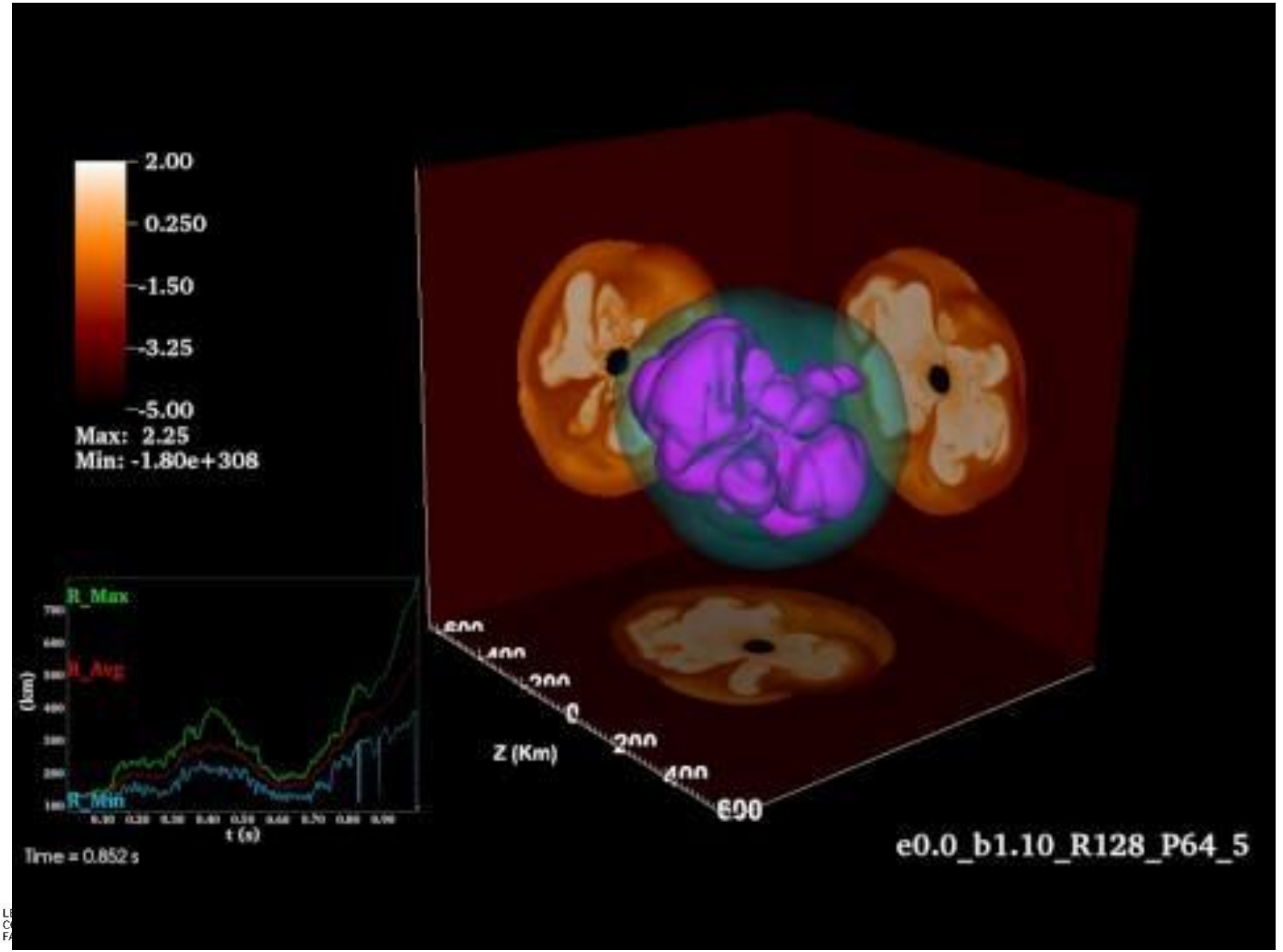
Ongoing Work: Remove Compatibility Interfaces

- Currently uses [cuda/hip]HostMalloc() to allocate page-locked (pinned) memory
  - Replacing to use OpenMP 5 allocator + allocate directive
  - Pre-defined (extension) allocator **OMPX_PINNED_MEM_ALLOC** is available in some compilers. Needs consensus & standardization.

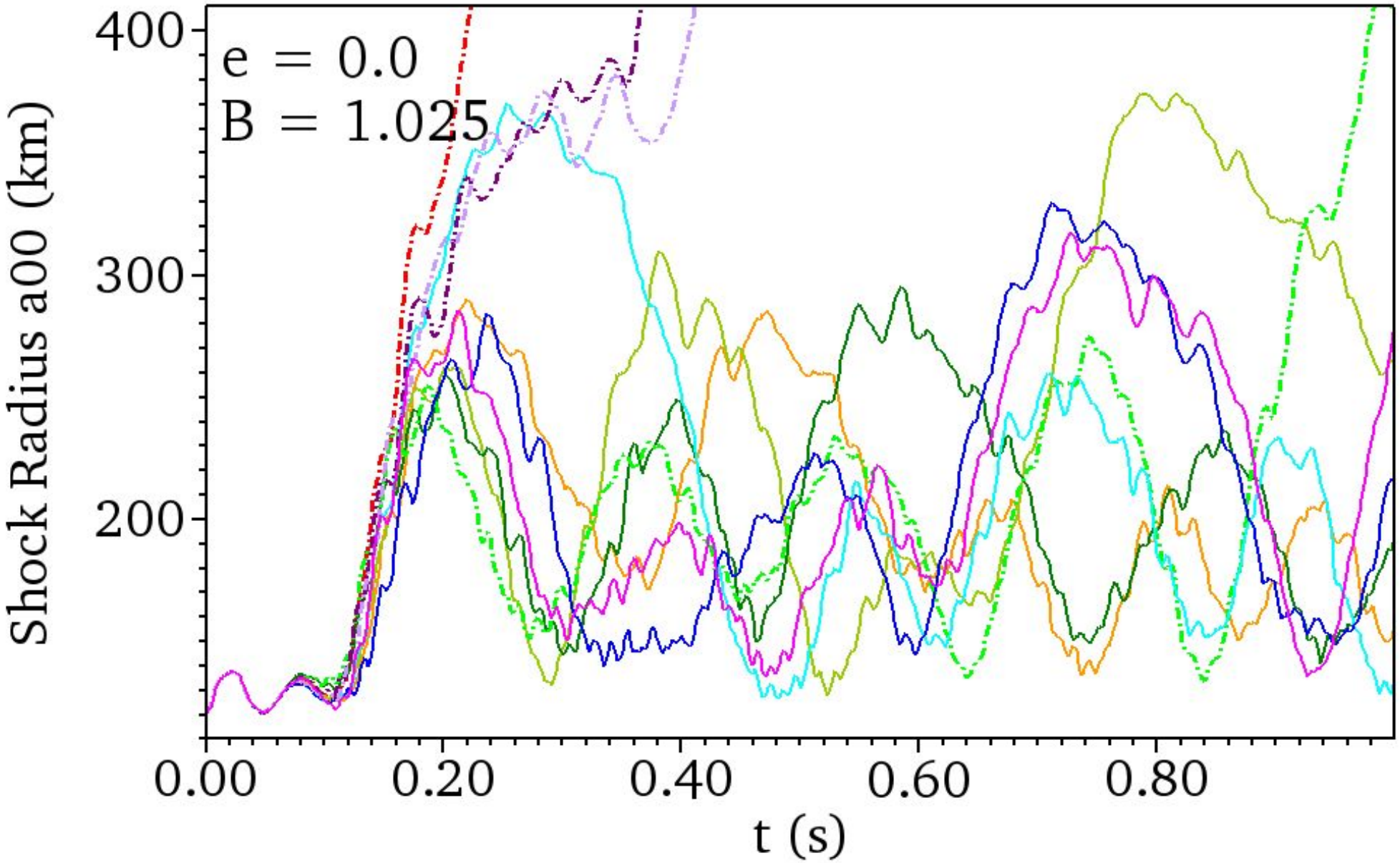
- In OpenMP 4.5 some library routines only have C binding. Replacing these as they become available from compilers supporting OpenMP 5.x
  - For e.g. omp_target_alloc(), omp_target_[associate/disassociate]_ptr()

Ongoing Work: Optimizations

- What is the best scheduling policy (`(static, X)`, `guided`, `auto`, `runtime`)? This could be compiler dependent.
- Best mapping to hardware (`teams`, `parallel do`, `simd`). Compilers map to hardware differently.
- More work per target region vs. multiple (concurrent) target launch (with `nowait`).
  - could be compiler dependent
  - could be kernel dependent
  - could be hardware dependent
- Exploiting GPU "shared memory" via `allocate` clause in `target`


2.00
0.250
-1.50
-3.25
-5.00
Max: 2.25
Min: -1.80e+308
R_Max
R_Avg
R_Min
(km)
t (s)
Time = 0.852 s
Z (Km)
600
e0.0_b1.10_R128_P64_5

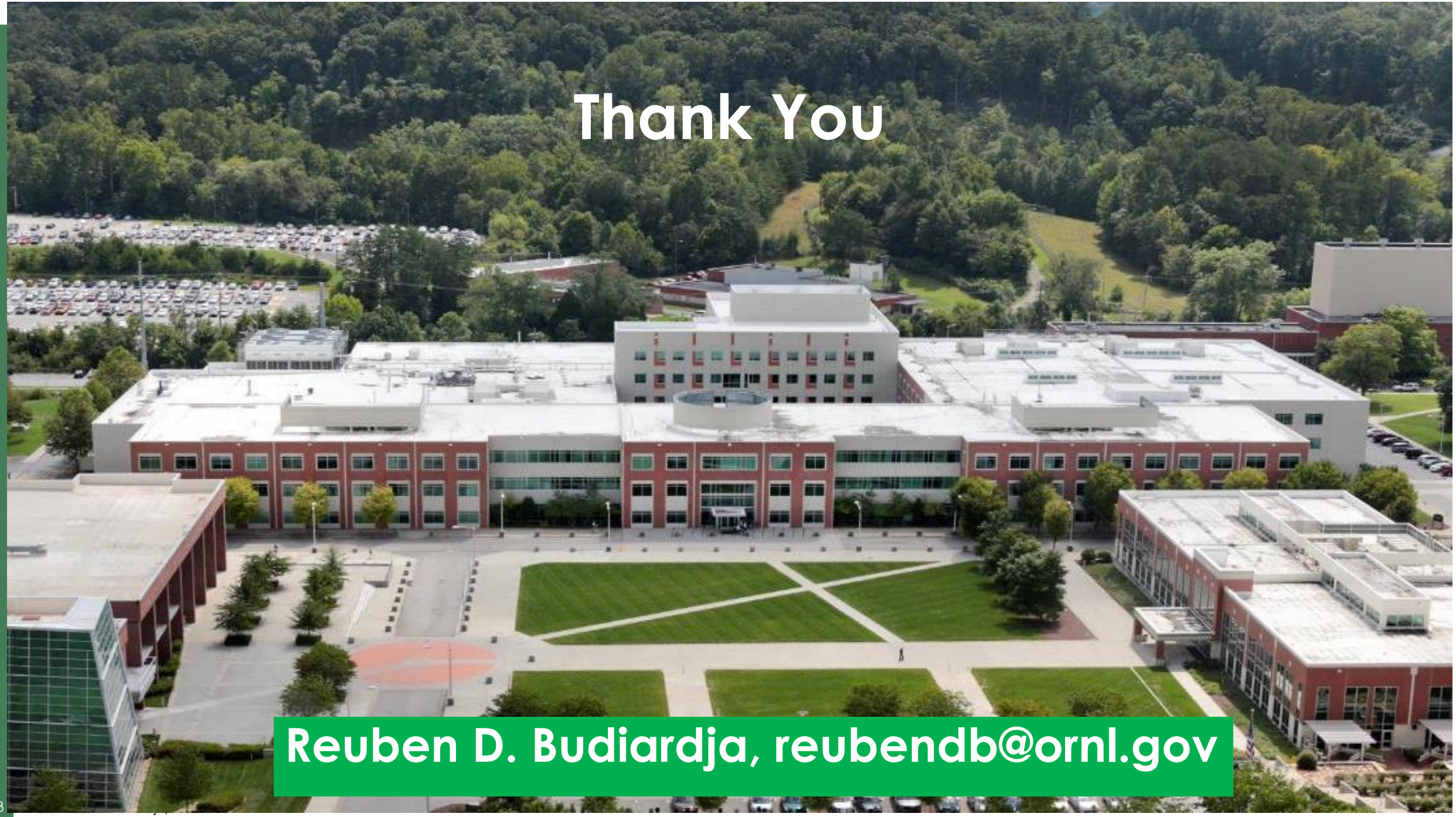
Stochastic Behavior of SASI-dominated Models



Ongoing Science Investigation

- Using OpenMP offload to exploit GPU, we now have the computational capability to perform ensemble CC-SN simulations with more realistic setups, including with radiation transport.

- Bridging the gap of two traditional classes of CC-SN sims:
  - A handful of cutting edge sophistication, vs.
  - Large ensembles of simplified phenomenological simulations
  - **Broad survey with intermediate sophistication to target scrutiny of selected region with increased sophistication.**


Thank You
Reuben D. Budiardja, reubendb@ornl.gov