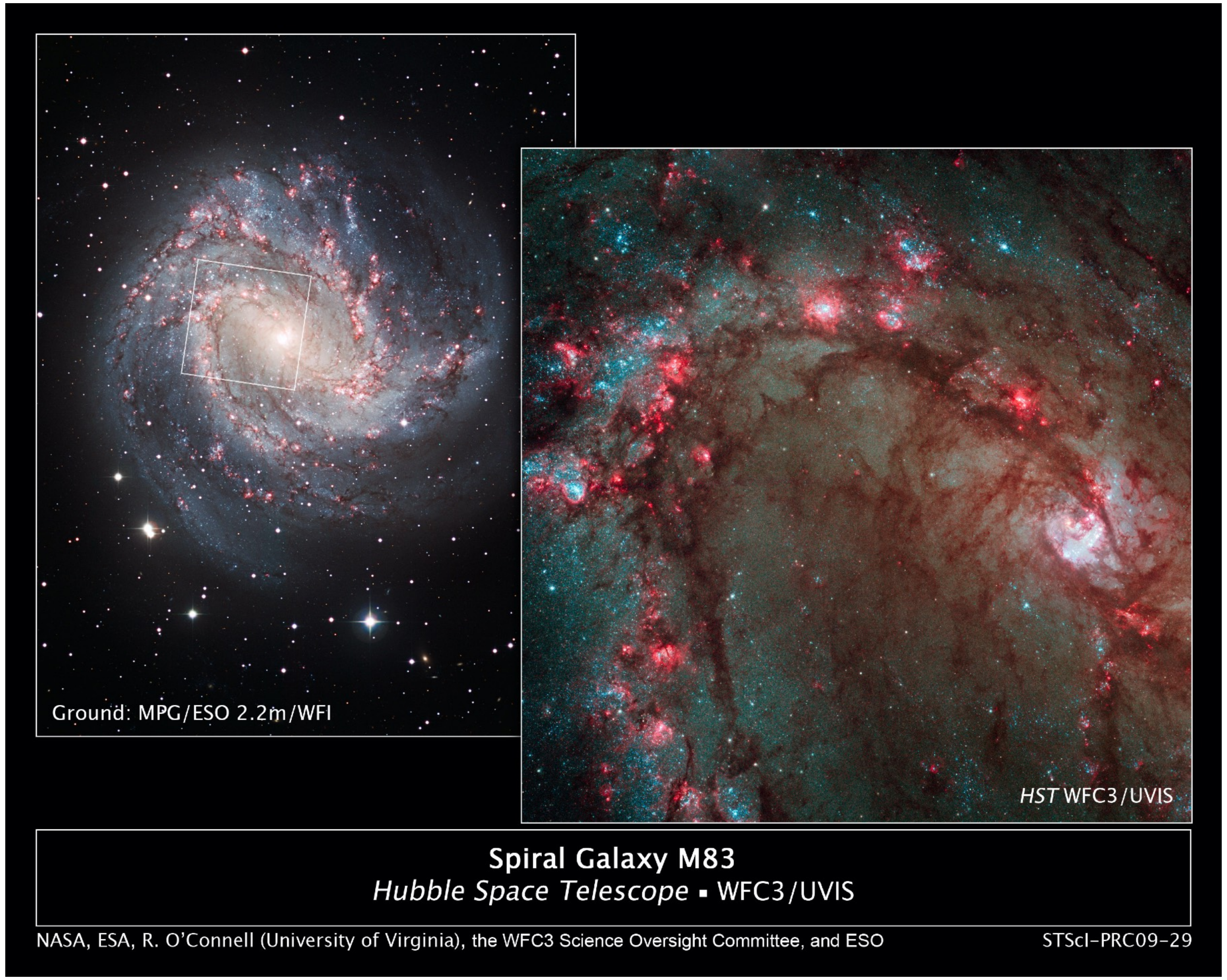
Astrophysics at Exascale

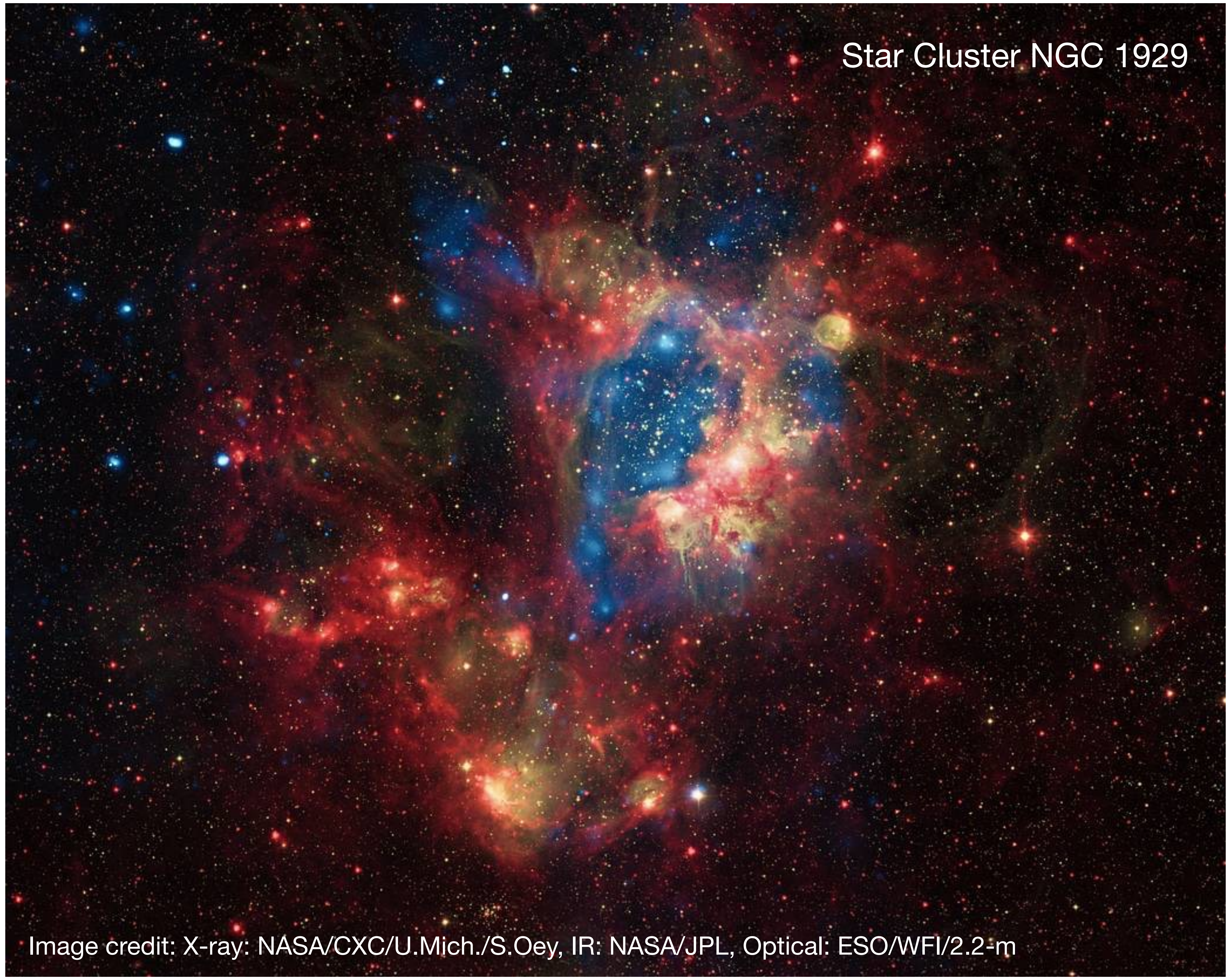
Preparing Cholla for Frontier

Evan Schneider,
University of Pittsburgh

OLCF User Meeting, June 24th 2021

Scientific Motivation:
Galaxy Evolution at Parsec* Scale

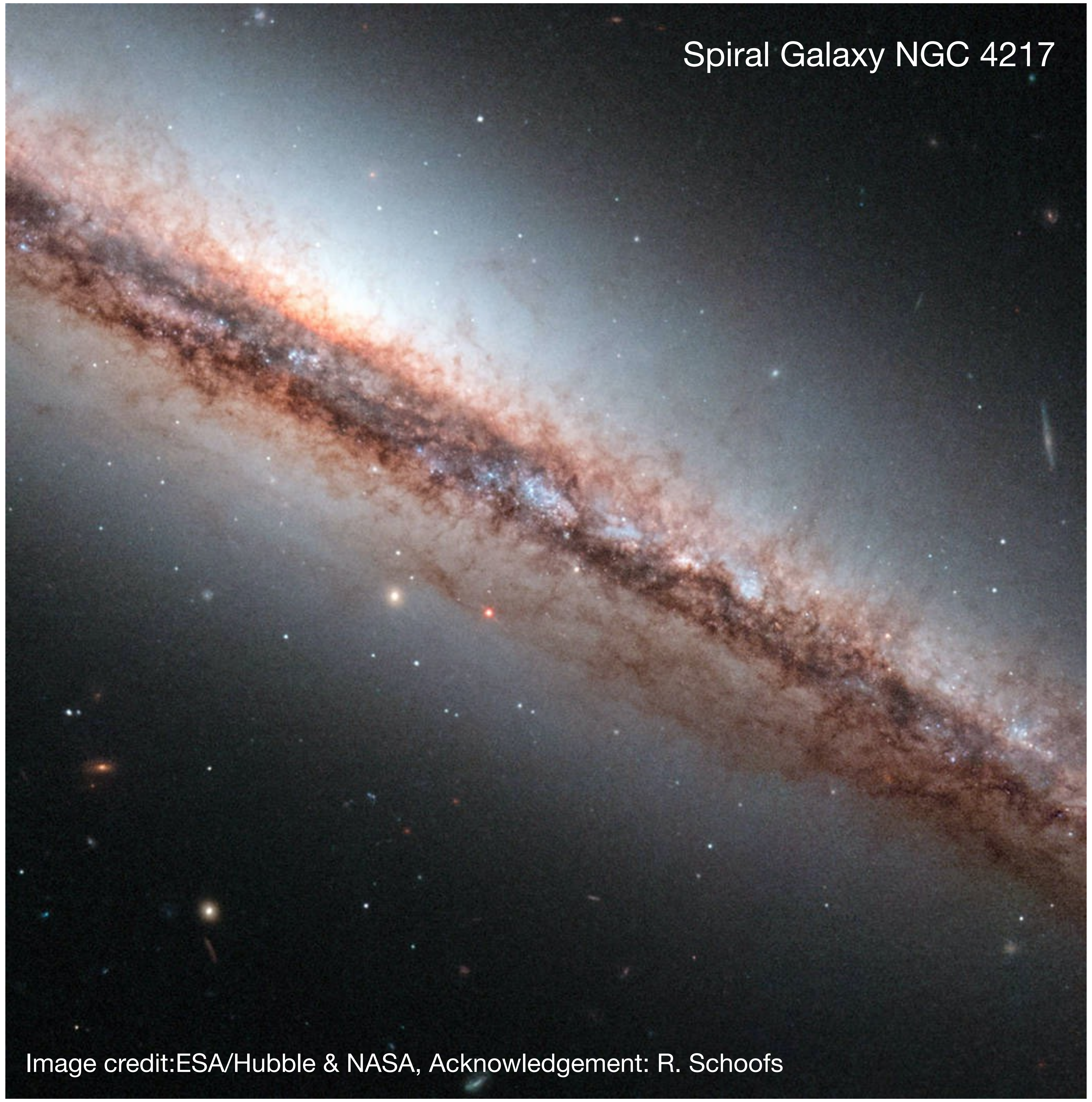


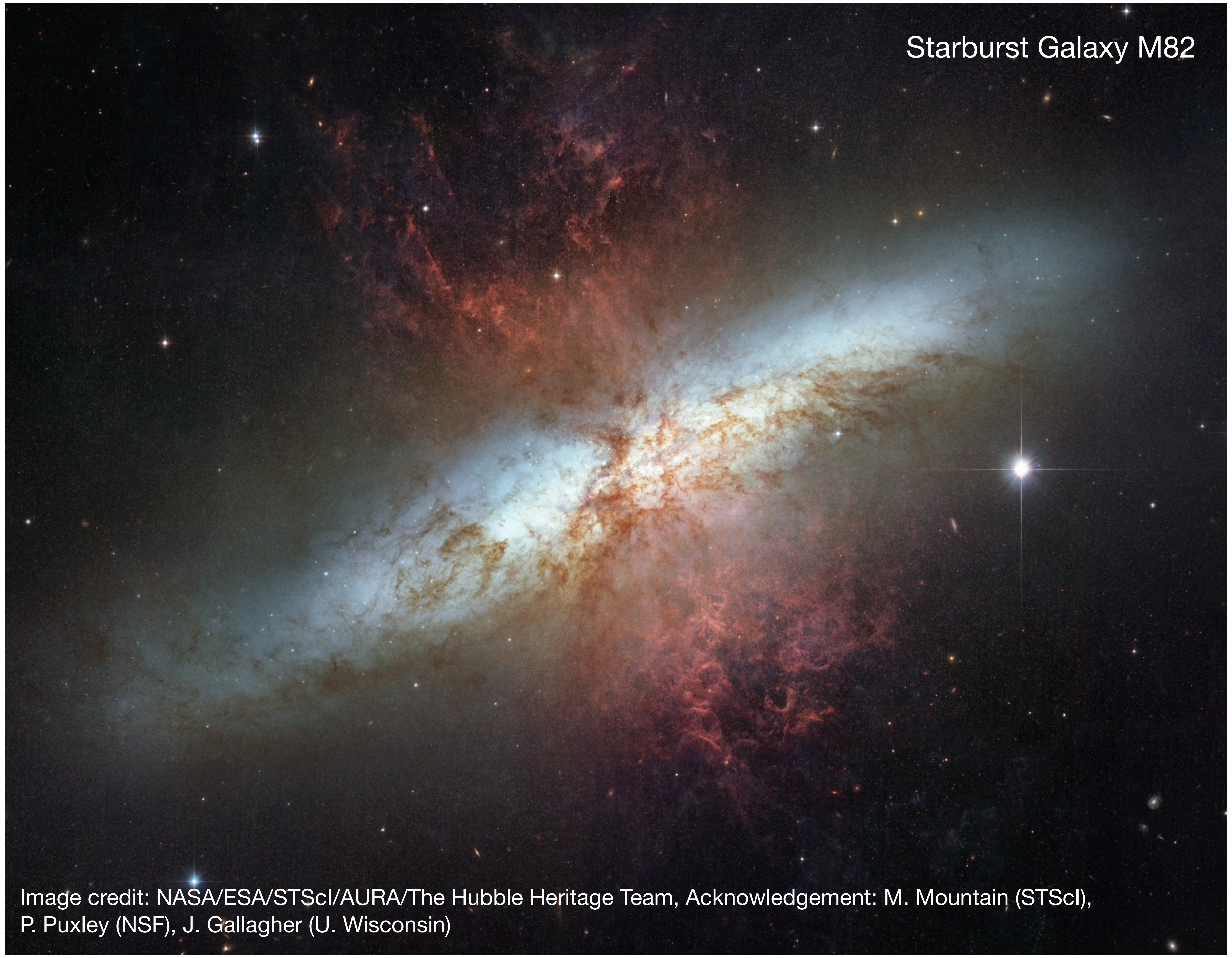


*1 Parsec = a few light years

Scientific Motivation:

Galaxy Evolution at Parsec Scale





Scientific Motivation:
CAAR project grand challenge problem

- Goal is to simulate a Milky Way-like galaxy at a resolution that allows for self-consistent star formation and supernovae within a multiphase interstellar medium
- Milky Way diameter: ~30 kpc
- Resolution required to resolve star clusters: ~few pc
- Target resolution for challenge problem on Frontier ~$10{,}000^3$ cells

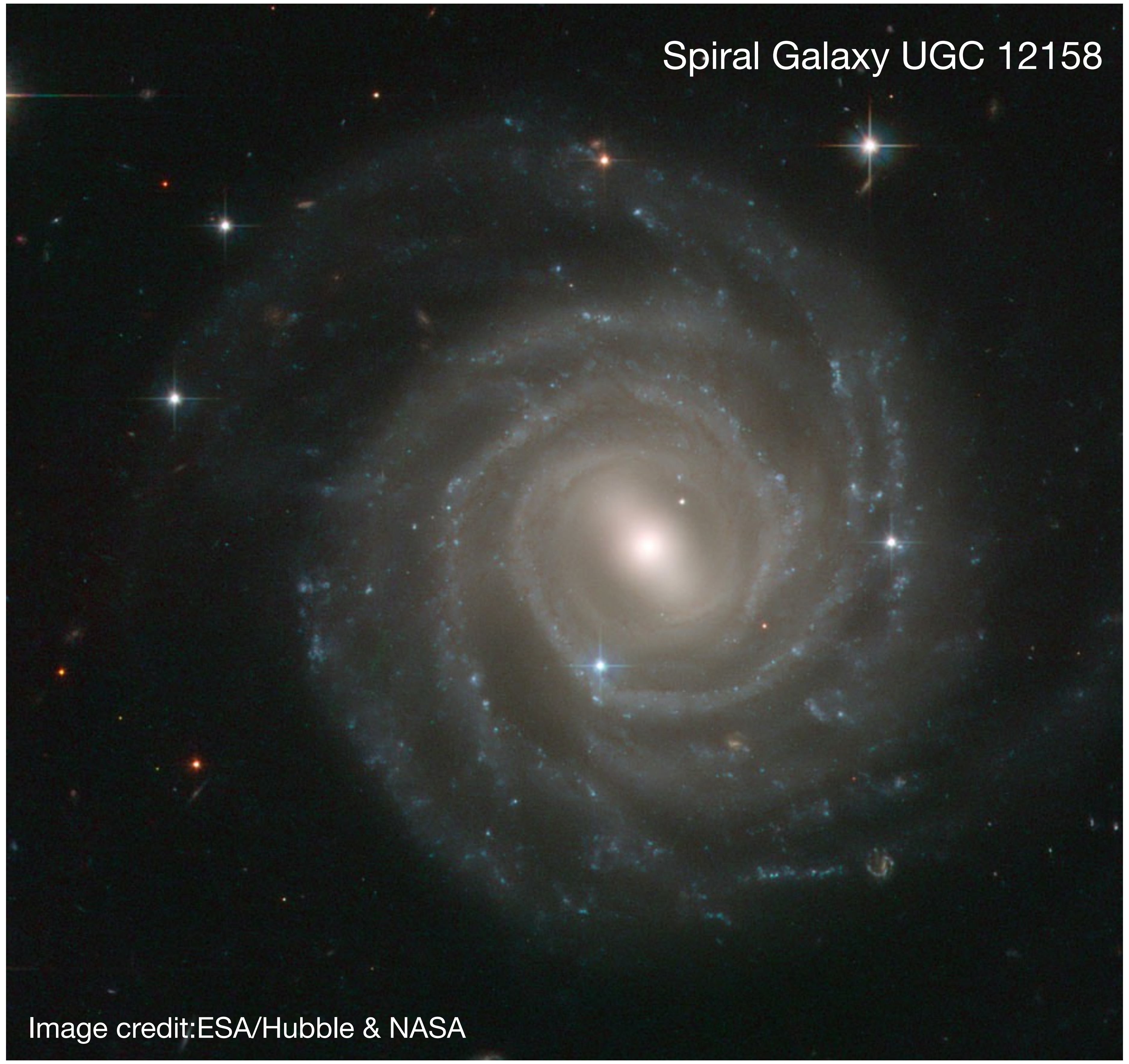


Image credit:ESA/Hubble & NASA

Cholla: **C**omputational **H**ydrodynamics **o**n **ll** **A**rchitectures

A tool for astrophysical simulations

- Cholla is a GPU-native, massively-parallel, finite-volume hydrodynamics code developed for astrophysics simulations

Cholla: **C**omputational **H**ydrodynamics **on ll A**rchitectures

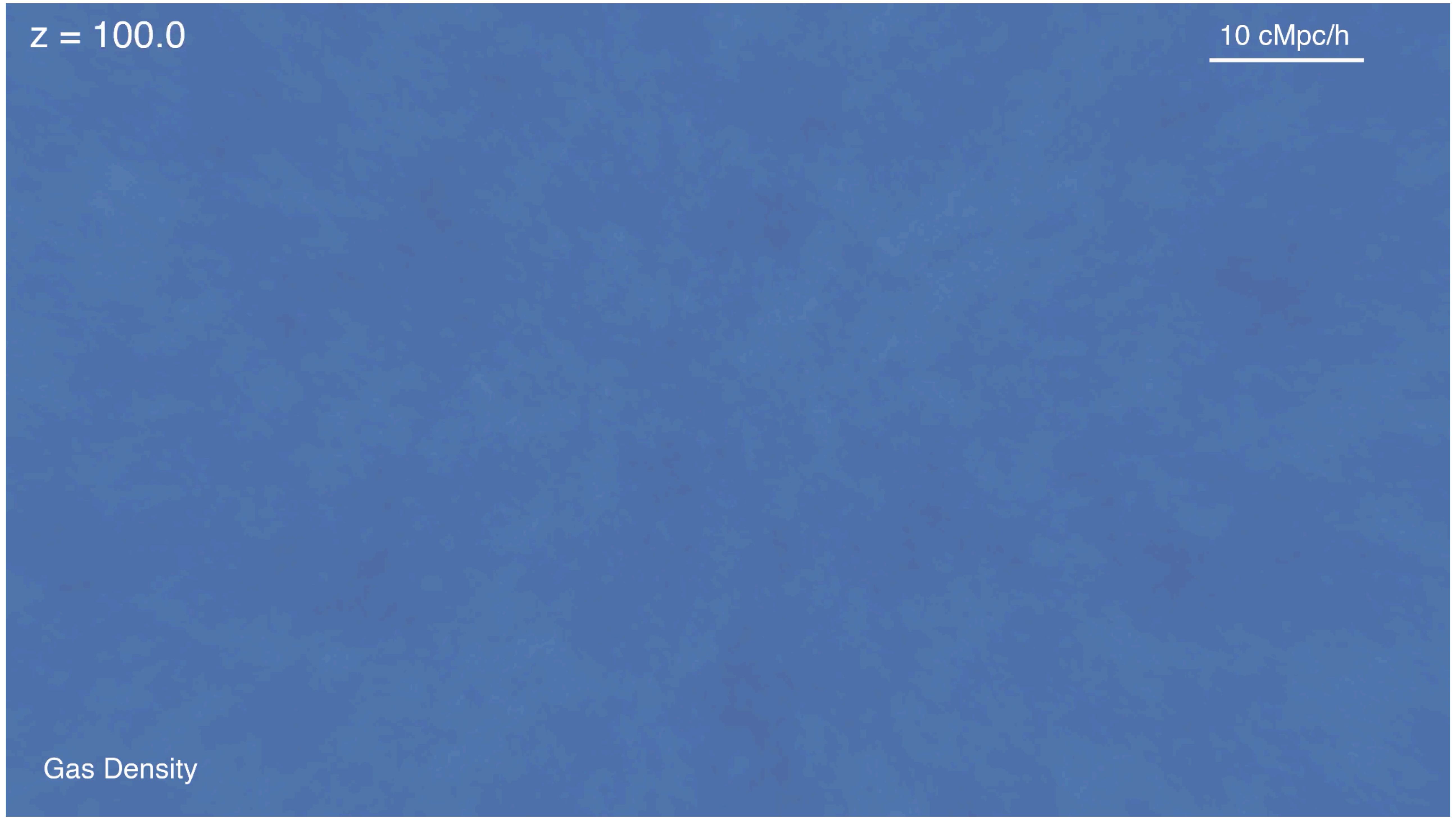
A tool for astrophysical simulations

- Cholla is a GPU-native, massively-parallel, finite-volume **hydrodynamics** code developed for astrophysics simulations

Most of the baryonic* matter in the Universe is gas.

(Some gas has been converted into stars inside galaxies.)

On astrophysical scales, gas can be simulated using fluid dynamics techniques.



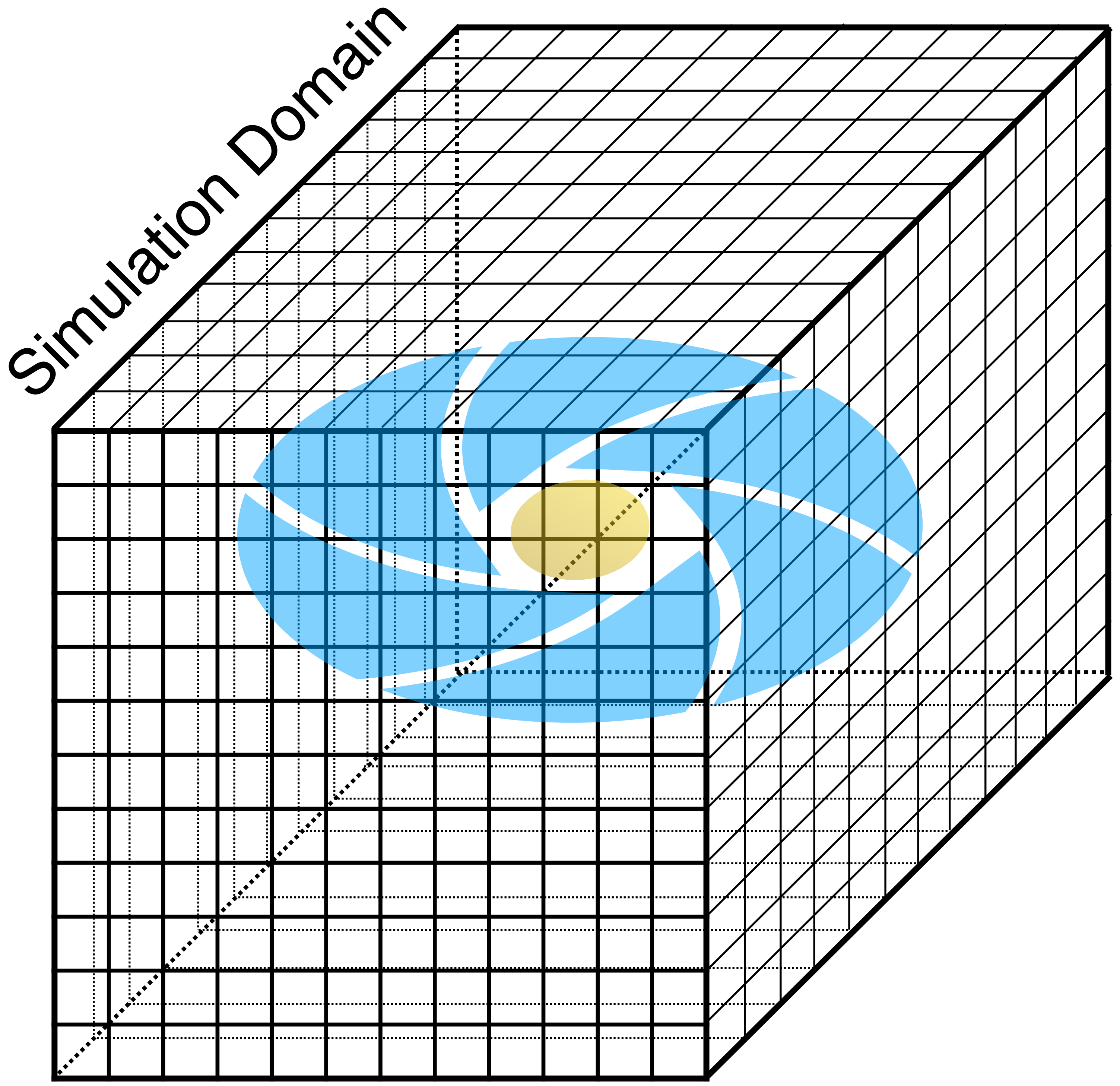
Cholla cosmological simulation by Bruno Villasenor

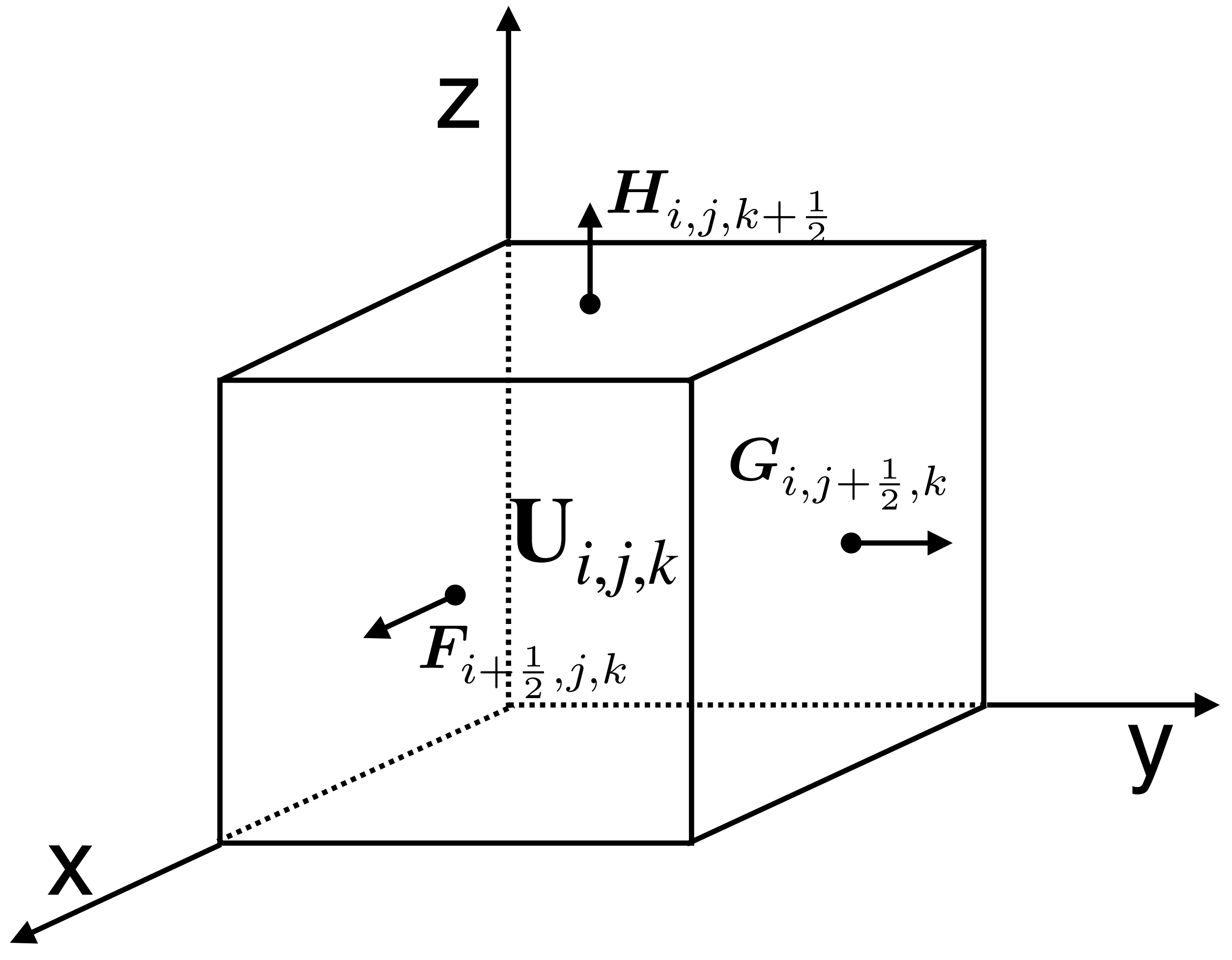
*baryonic = not dark

Cholla: **C**omputational **H**ydrodynamics **on ll A**rchitectures

A tool for astrophysical simulations

- Cholla is a GPU-native, massively-parallel, **finite-volume** hydrodynamics code developed for astrophysics simulations





Finite Volume

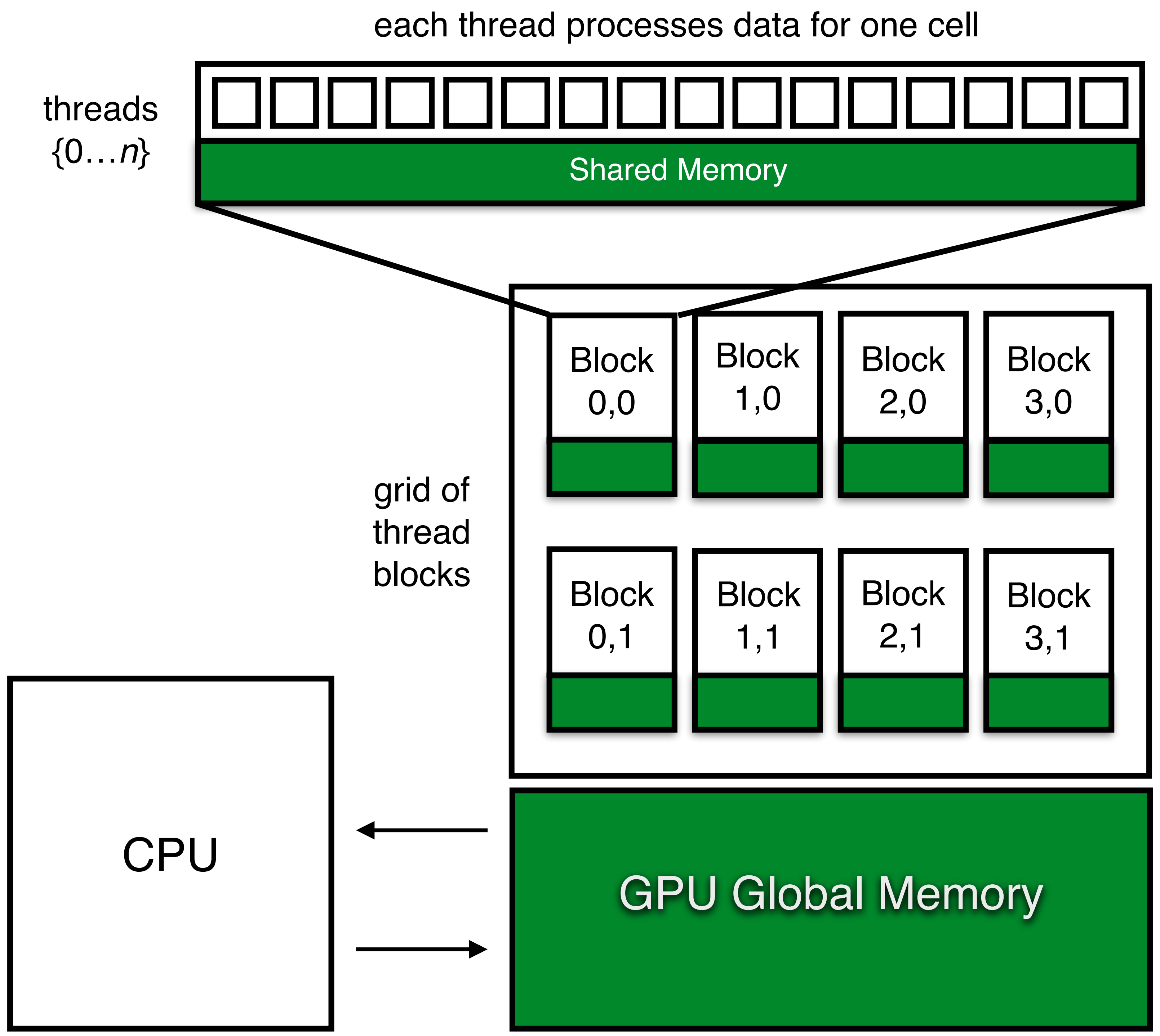
$$U = [\rho, \rho u, \rho v, \rho w, E]^{\mathrm{T}}$$

$$\begin{aligned}\boldsymbol{U}_{i,j,k}^{n+1} = \boldsymbol{U}_{i,j,k}^{n} &- \frac{\delta t}{\delta x}\left(\boldsymbol{F}_{i+1/2,j,k}^{n+1/2} - \boldsymbol{F}_{i-1/2,j,k}^{n+1/2}\right) \\ &- \frac{\delta t}{\delta y}\left(\boldsymbol{G}_{i,j+1/2,k}^{n+1/2} - \boldsymbol{G}_{i,j-1/2,k}^{n+1/2}\right) \\ &- \frac{\delta t}{\delta z}\left(\boldsymbol{H}_{i,j,k+1/2}^{n+1/2} - \boldsymbol{H}_{i,j,k-1/2}^{n+1/2}\right)\end{aligned}$$

Cholla: **C**omputational **H**ydrodynamics **on ll A**rchitectures

A tool for astrophysical simulations

- Cholla is a **GPU-native**, **massively-parallel**, finite-volume hydrodynamics code developed for astrophysics simulations



Cholla: **C**omputational **H**ydrodynamics **o**n **ll** **A**rchitectures

Not just hydrodynamics anymore

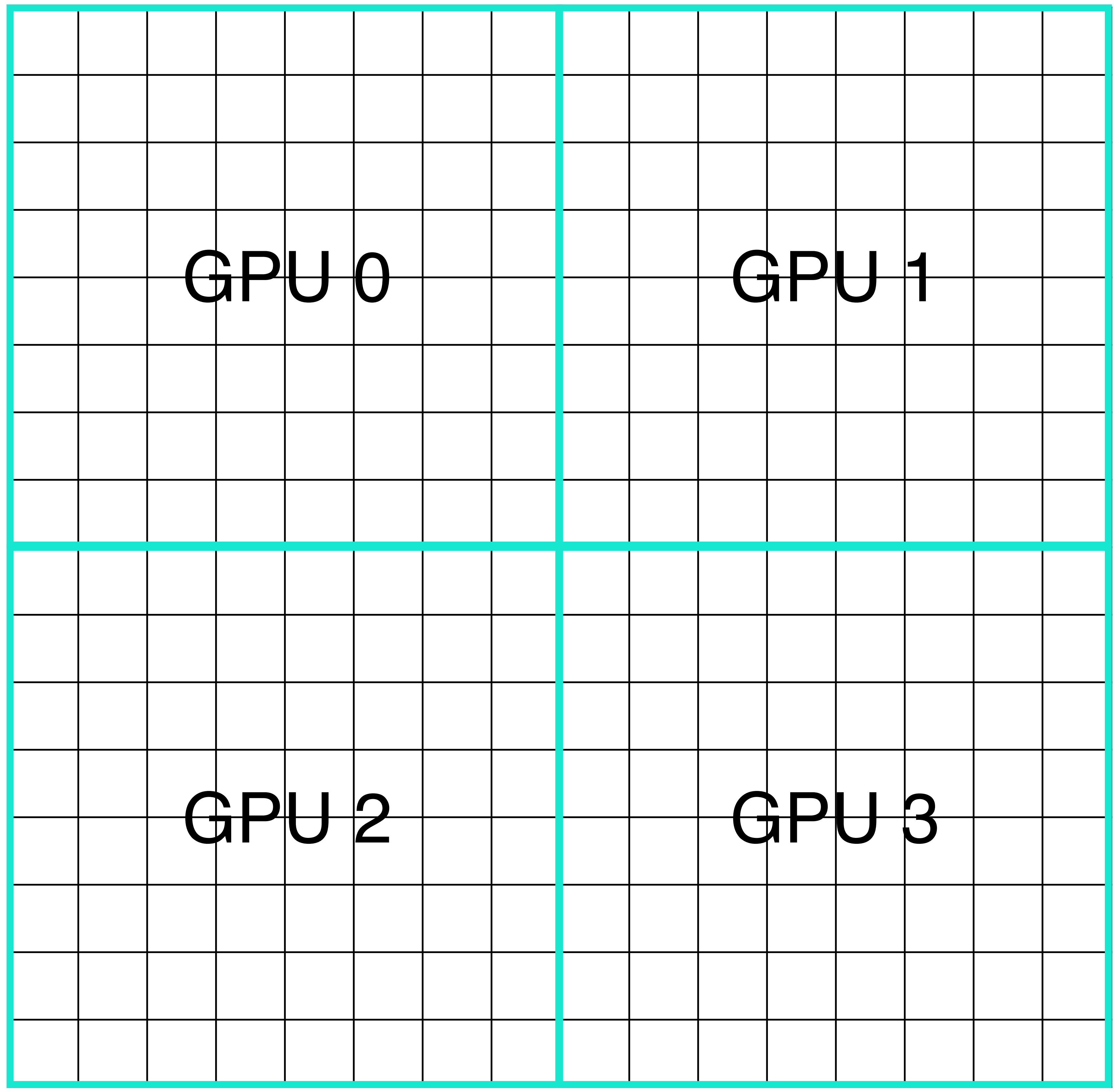
- Cholla has gained many additional physics modules in the last couple of years; thesis work of Bruno Villasenor, UCSC
  - **Self gravity** (FFT-based)
  - Comoving coordinate system (expanding universe)
  - **Dark matter particles**
  - Chemistry and radiative cooling
  - UV background
- Enables massive static-mesh cosmological hydrodynamical simulations

Villasenor+ 2021, arxiv no: 2009.06652

How does it work?

Cholla circa 2019

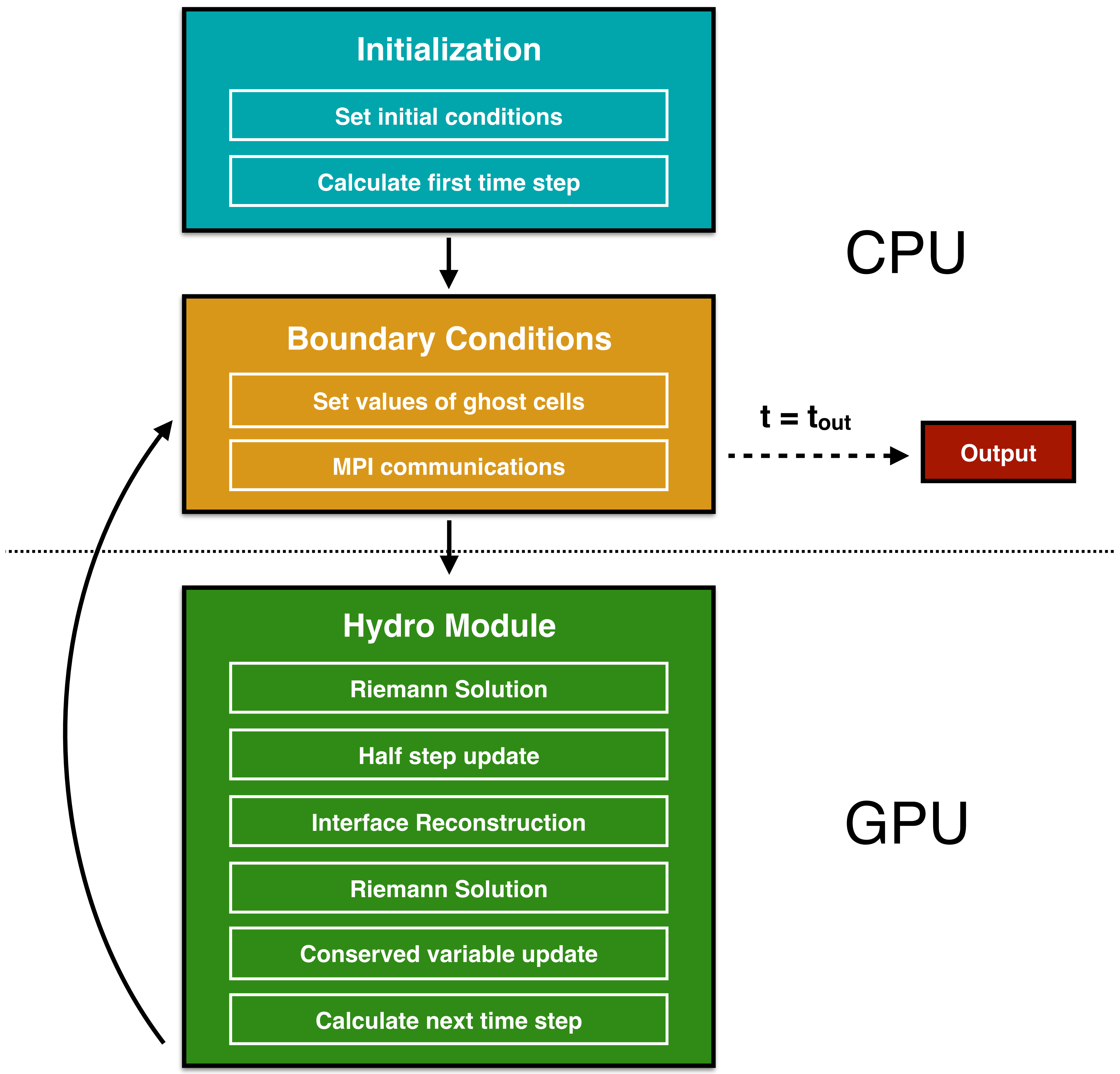
- Simulation domain is divided into sub volumes, each MPI rank is assigned a single sub-volume and a single GPU
- Typical sub-volume is 256^3 cells
- Each cell is mapped to a single thread on the GPU
- Subvolumes can be further divided if data size is too large to fit in memory on a single GPU



How does it work?

Cholla circa 2019

- Serial portions of the code execute on the CPU
- Parallel portions execute on the GPU
- Some of the new physics modules executed partially on the GPU, some executed exclusively on the CPU
- Fundamentally, the grids “lived” on the CPU



CAAR

The Center for Accelerated Application Readiness

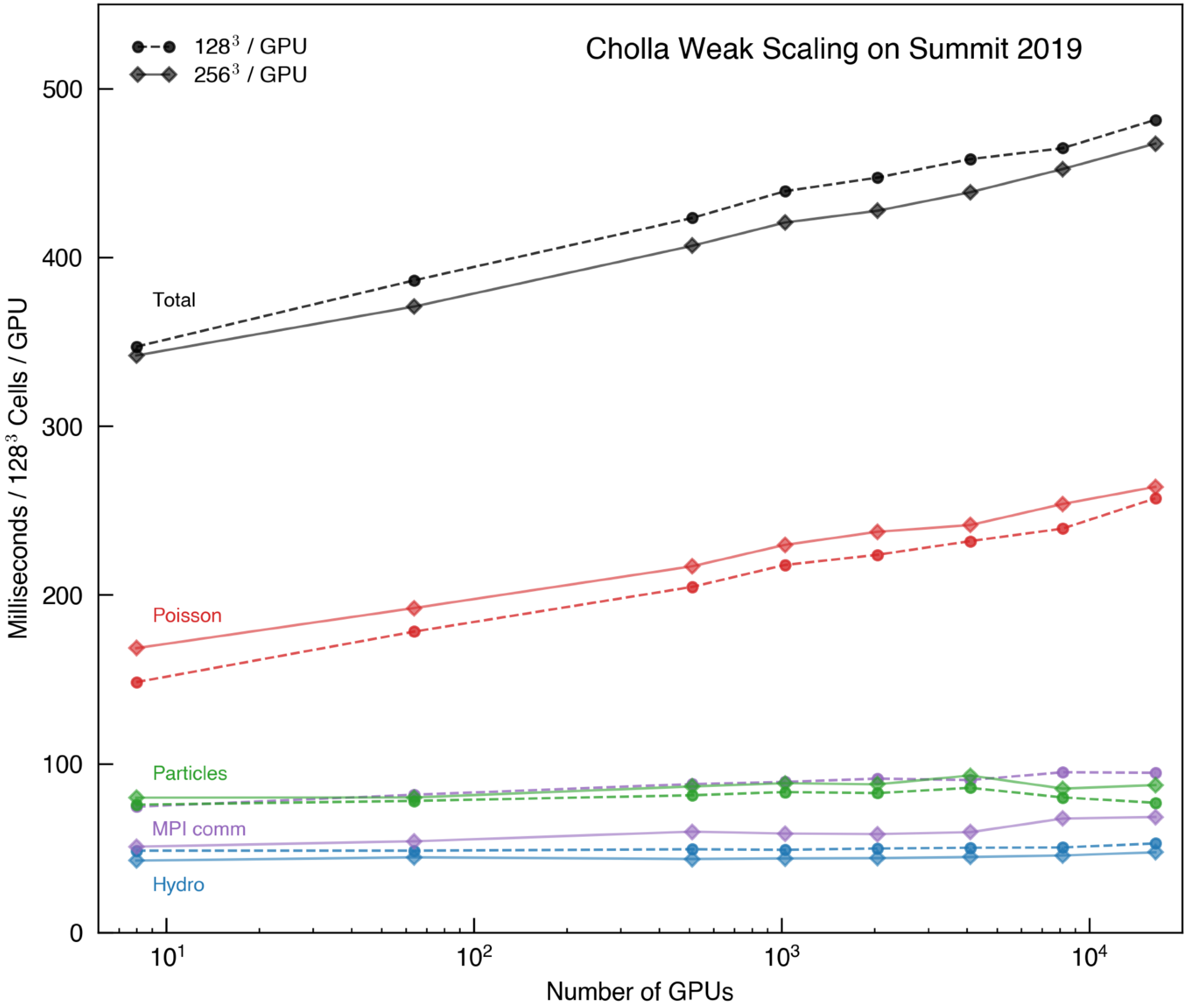
- Cholla was selected as one of 8 CAAR applications preparing for Frontier
- Team includes astrophysicists, OLCF staff, and vendor partners
    - Evan Schneider (Pitt)
    - Orlando Warren (Pitt)
    - Bob Caddy (Pitt)
    - Helena Richie (Pitt)
    - Reuben Budiardja (OLCF)
    - Brant Robertson (UCSC)
    - Bruno Villasenor (UCSC)
    - Nicole Drakos (UCSC)
    - Trey White (HPE)
    - Damon McDougal (AMD)

CAAR

The Center for Accelerated Application Readiness

- Our goal is to run a grand-challenge science problem — a parsec-scale resolution simulation of the Milky Way — on Frontier.

- To do so, we will need
  - Hydrodynamics
  - Self-gravity (Poisson)
  - Particles



First Task: Portability

Or, how we learned to HIPifly

- Cholla was written in C++ / Cuda / MPI / OpenMP
- Frontier will have AMD GPUs, which use HIP
- Solution: use HIP
  - Option one: HIPify
    - Modify all cuda source files, changing all cuda syntax to hip syntax
    - hipcc compiles resulting code for either AMD or NVIDIA hardware
  - Option two: HIPifly!

First Task: Portability

Or, how we learned to HIPifly

Added a single header file, gpu.hpp

```
#ifdef DO_HIP

#define cudaDeviceSynchronize hipDeviceSynchronize

#define cudaError hipError_t

#define cudaError_t hipError_t

#define cudaErrorInsufficientDriver hipErrorInsufficientDriver

#define cudaErrorNoDevice hipErrorNoDevice
```

etc.

This means there is a single CUDA code base for both NVIDIA and AMD GPUs.

Second Task:

Data must live on the GPU

- Originally, Cholla was designed to offload hydro calculations to the GPU every time step (largely to allow bigger grids)
  - As GPUs speeds have increased, CPU-GPU communication speeds have stayed roughly the same

—> Data transfer was taking up a larger portion of the time step than hydro calculation!

- Solution: keep the hydro grid on the GPU, transfer boundary cells using GPU-aware MPI, only transfer the grid back to the CPU for output
  - Currently resulting in a ~4x speedup for hydro

Third Task: Gravity

Do the Poisson solve on the GPU

- The primary gravity solver in Cholla is FFT-based; our domain decomposition is block based - need a block-based FFT library to do Poisson solve
  - Previously, did this with PFFT on the CPU (using FFTW)
  - There was no existing parallel block-based FFT library for GPUs
- Solution: Write one. Trey White (HPE) wrote a block-based Poisson solver, **Paris**, that uses either cuFFT (Nvidia GPUS) or rocFFT (AMD GPUs) to perform FFTs on the GPU, and GPU-direct MPI communication

Paris

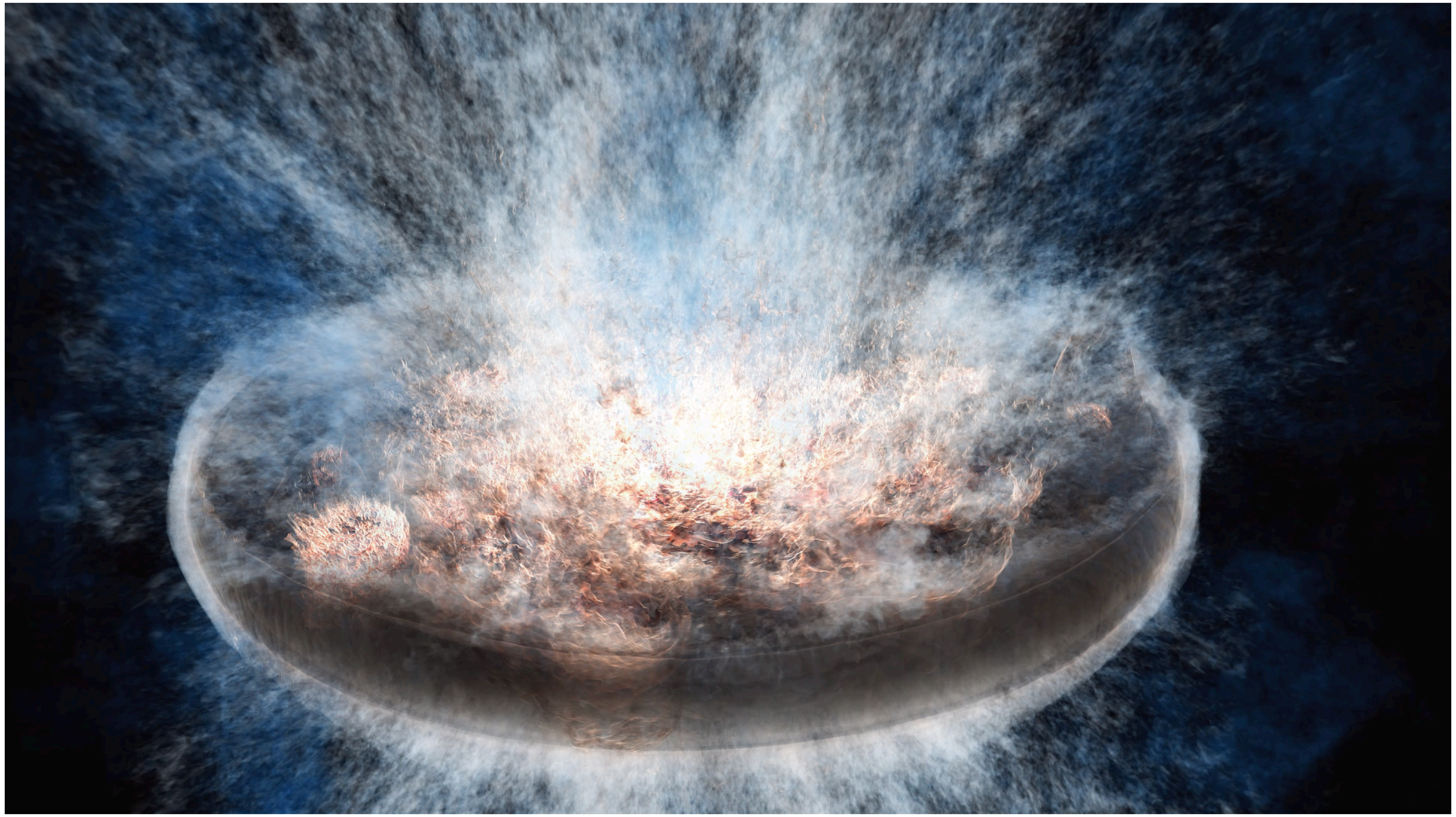
A new 3D FFT-based Poisson solver for distributed memory GPU supercomputers

- Paris moves all the following from the CPU to the GPU:
  - FFTs (now computed using cuFFT or rocFFT)
  - Poisson solve in frequency space
  - Buffers for MPI communication
  - Copies and transposes for changing dimensions in the 3D FFTs
  - Currently seeing at least a 3x speedup when using Paris vs PFFT

Paris

A new 3D FFT-based Poisson solver for distributed memory GPU supercomputers

- Paris can also deal with open boundaries — critical for our challenge problem
- Achieved by converting discrete sine transforms to FFTs (also on the GPU)

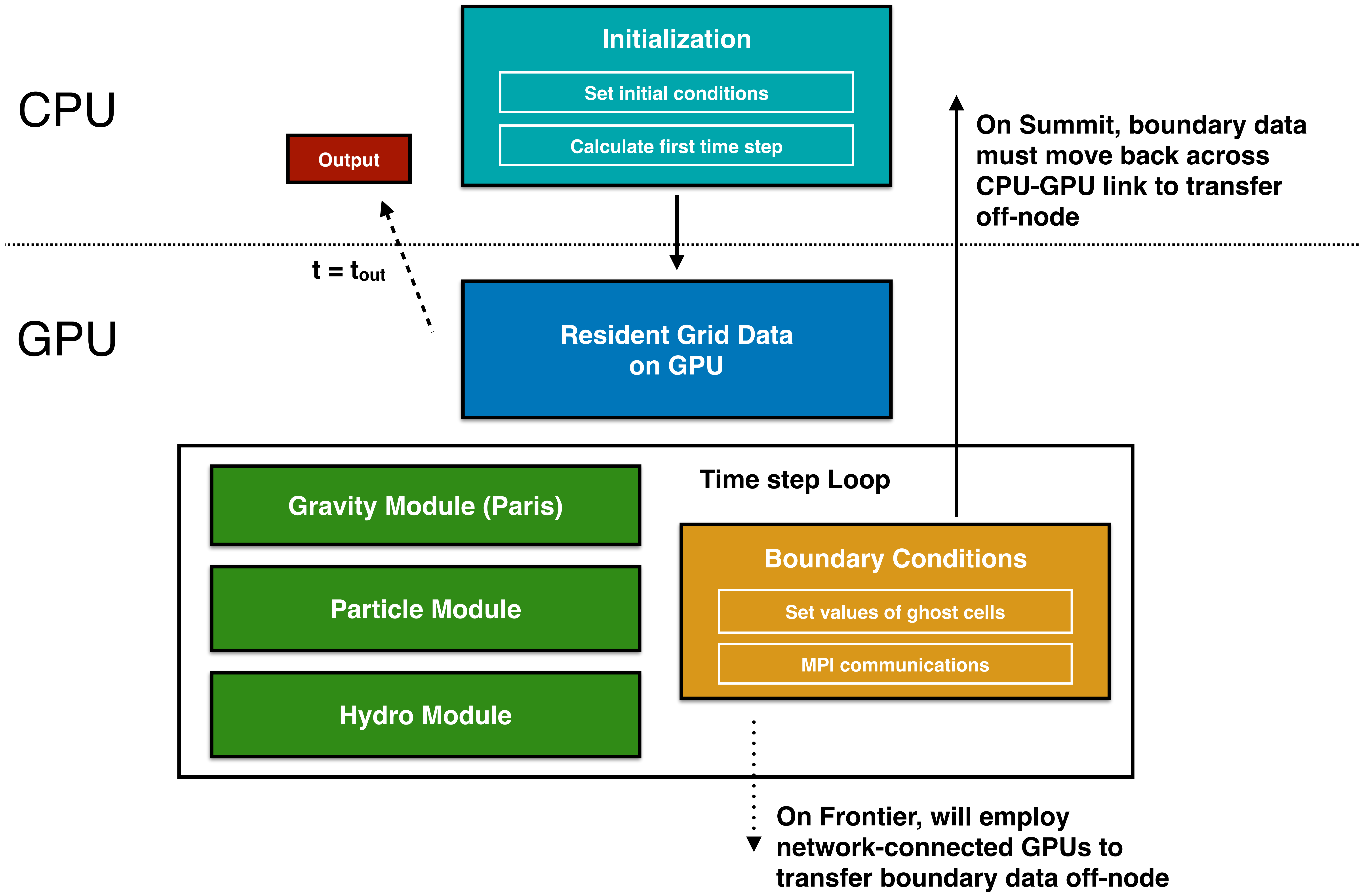


Cholla outflow simulation from INCITE program AST125

How does it work?

Cholla circa 2021

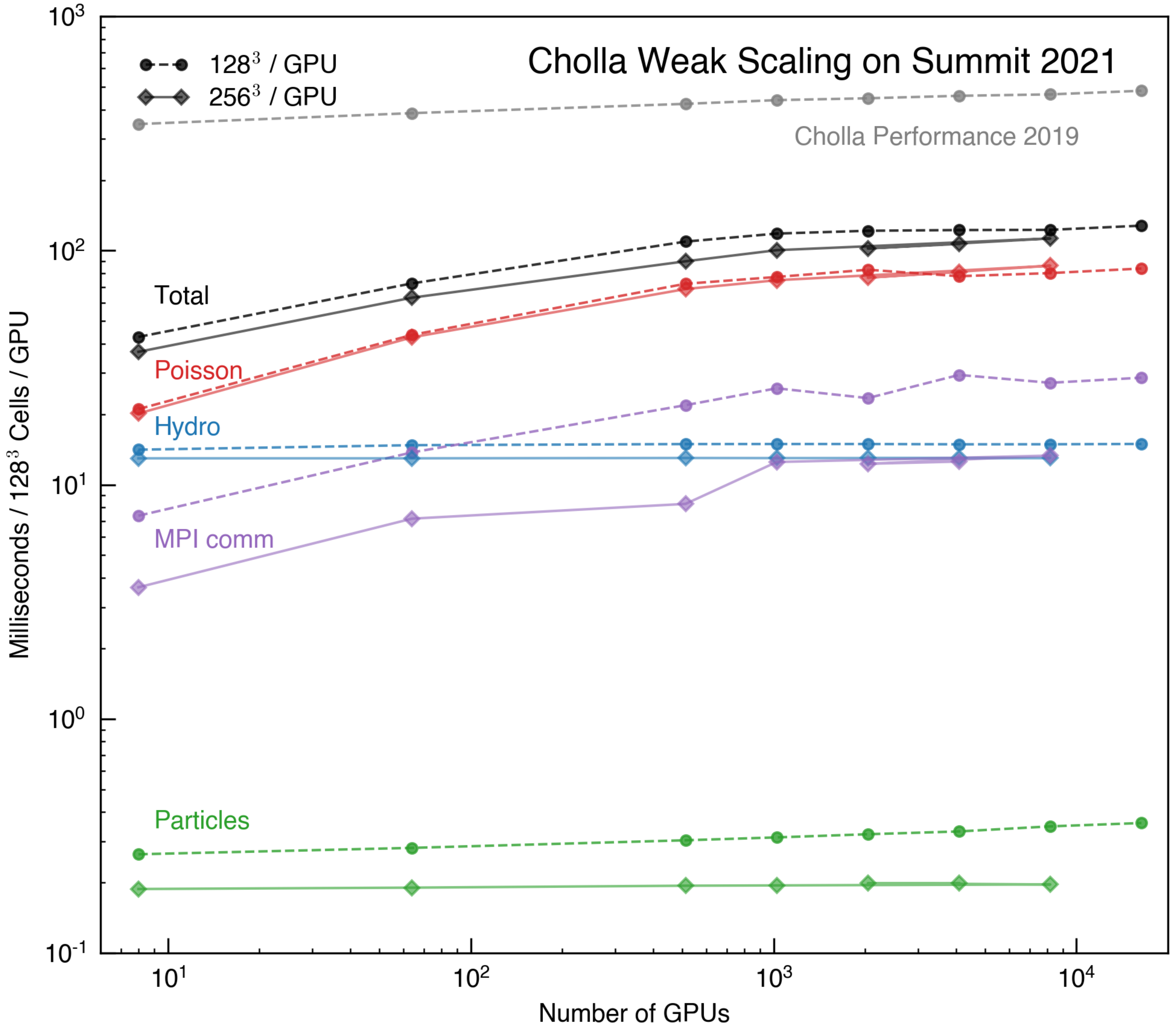
- Final steps in the GPU-resident data transition:
  - Packing boundary buffers on the GPU (achieved using openMP) and sending off-node using GPU-aware MPI
  - Moving particle solver fully to the GPU



Performance Gains

Putting it all together

- Overall performance improvement on Summit since 2019 is 3.8x at scale (8x on 8 GPUs!)
- We anticipate further performance gains from:
  - Optimizing Paris at scale
  - Network-connected GPUs on Frontier



To Summarize:

- Cholla is currently ~4x faster *on the same hardware* than it was 2 years ago, thanks to CAAR improvements
- Frontier is expected to have ~7x the floating point capability of Summit, so we fully expect to hit our CAAR performance target when it comes online
- Challenge problem — simulating the Milky Way at parsec-scale resolution — is just around the corner
- All updates to the code can be found in the "CAAR" branch of Cholla on Github: https://github.com/cholla-hydro/cholla