Enzo-P / Cello
Pursuing Petascale Astrophysics

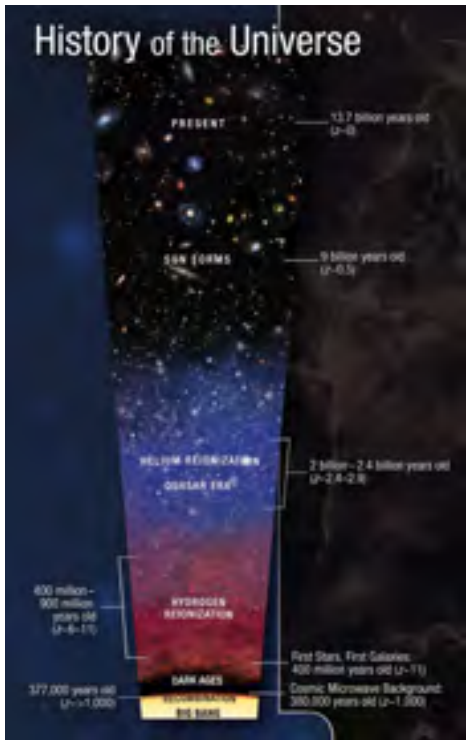
PI Michael L. Norman, Co-PI James Bordner

University of California, San Diego
San Diego Supercomputer Center

NCSA Blue Waters Symposium
16–19 May 2017

The search for missing physics

Realistic simulations of the intergalactic medium



Hubble 2010
Science Year in Review

Our project has two components:

- **cosmological simulations** (Norman)
  - investigating Helium reionization epoch
  - simulations don't match new observations
  - ran extreme simulations using *Enzo* on BW
  - more: Norman BW 2016, Pengfei Chen thesis
- **software development** (Bordner)
  - Enzo's AMR is Terascale not Petascale
  - developing next-generation *Enzo-P*
  - parallelized using Charm++
  - scalable array-of-octree AMR (*Cello*)
  - using BW for performance testing

The Enzo cosmology and astrophysics application

Enzo's strengths

Applicable to a wide range of astrophysical/cosmological problems

Physics Equations: *mathematical models*

• Hydrodynamics (Euler equations) • Gravity ($\nabla^2\Phi = 4\pi G\rho$) • Chemistry/cooling • Star formation • Magnetism • Radiation . . .

Numerical Methods: *approximate and solve*

• PPM, • ZEUS, • MUSCL • FFT • multigrid • Gadget cooling • Cloudy cooling • Grackle • Dedner MHD • MHD-CT • Implicit FLD • Moray . . .

Data Structures: *computer representation*

• Structured Adaptive Mesh Refinement (SAMR) • Eulerian fields • Lagrangian particles . . .

The Enzo cosmology and astrophysics application

Enzo's strengths

Implements a variety of sophisticated numerical methods

Physics Equations: *mathematical models*

• Hydrodynamics (Euler equations) • Gravity ($\nabla^2\Phi = 4\pi G\rho$) • Chemistry/cooling • Star formation • Magnetism • Radiation . . .

Numerical Methods: *approximate and solve*

• PPM, • ZEUS, • MUSCL • FFT • multigrid • Gadget cooling • Cloudy cooling • Grackle • Dedner MHD • MHD-CT • Implicit FLD • Moray . . .

Data Structures: *computer representation*

• Structured Adaptive Mesh Refinement (SAMR) • Eulerian fields • Lagrangian particles . . .

The Enzo cosmology and astrophysics application

Enzo's strengths

Enabled by adaptive mesh refinement with particles and fields

Physics Equations: *mathematical models*

• Hydrodynamics (Euler equations) • Gravity ($\nabla^2\Phi = 4\pi G\rho$) • Chemistry/cooling • Star formation • Magnetism • Radiation . . .

Numerical Methods: *approximate and solve*

• PPM, • ZEUS, • MUSCL • FFT • multigrid • Gadget cooling • Cloudy cooling • Grackle • Dedner MHD • MHD-CT • Implicit FLD • Moray . . .

Data Structures: *computer representation*

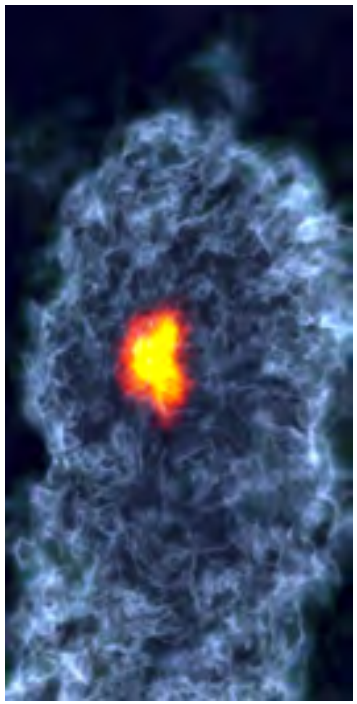
• Structured Adaptive Mesh Refinement (SAMR) • Eulerian fields • Lagrangian particles . . .

The Enzo cosmology and astrophysics application

Enzo's struggles

Limitations primarily due to relentless growth of HPC platforms

- Ever-changing software requirements
  - Enzo was born in early 1990's
  - "massive parallelism" meant $P \approx 100$
  - $\times 1000$ parallelism today
- Affects different parts of Enzo differently
  - ☺ physics requirements unchanged
  - ☺ numerical methods mostly viable
  - ☹ *data structures* limit Enzo's scalability
- Motivates AMR data structure redesign
  - **Enzo-P**: "petascale" version of Enzo
  - keep Enzo's physics and many methods
  - built on **Cello** scalable AMR framework



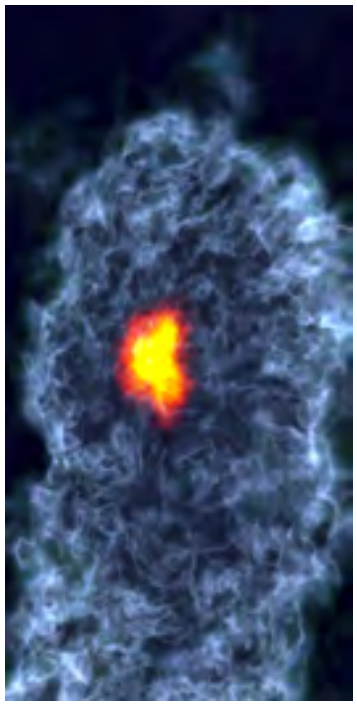
[Sam Skillman, Matt Turk]

The Enzo cosmology and astrophysics application

Enzo's struggles

Limitations primarily due to relentless growth of HPC platforms

- Ever-changing software requirements
  - Enzo was born in early 1990's
  - "massive parallelism" meant $P \approx 100$
  - $\times 1000$ parallelism today
- Affects different parts of Enzo differently
  - ☺ physics requirements unchanged
  - ☺ numerical methods mostly viable
  - ☹ *data structures* limit Enzo's scalability
- Motivates AMR data structure redesign
  - **Enzo-P**: "petascale" version of Enzo
  - keep Enzo's physics and many methods
  - built on **Cello** scalable AMR framework



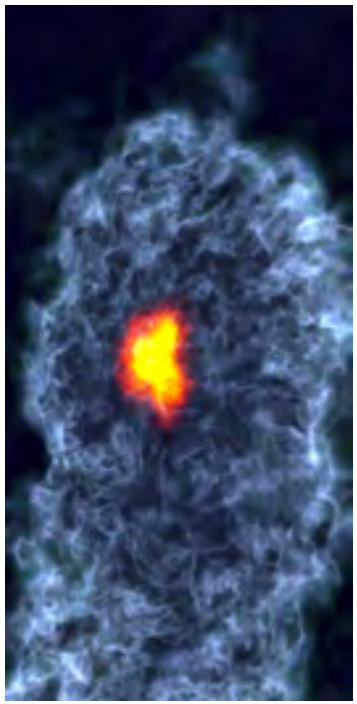
[Sam Skillman, Matt Turk]

The Enzo cosmology and astrophysics application

Enzo's struggles

Limitations primarily due to relentless growth of HPC platforms

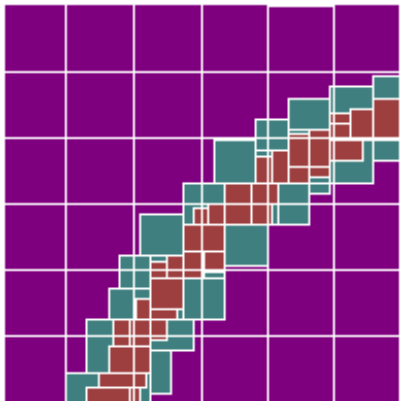
- Ever-changing software requirements
  - Enzo was born in early 1990's
  - "massive parallelism" meant $P \approx 100$
  - $\times 1000$ parallelism today
- Affects different parts of Enzo differently
  - ☺ physics requirements unchanged
  - ☺ numerical methods mostly viable
  - ☹ *data structures* limit Enzo's scalability
- Motivates AMR data structure redesign
  - **Enzo-P**: "petascale" version of Enzo
  - keep Enzo's physics and many methods
  - built on **Cello** scalable AMR framework



[Sam Skillman, Matt Turk]

Cello scalable adaptive mesh refinement

Differences between Enzo and Enzo-P



Enzo

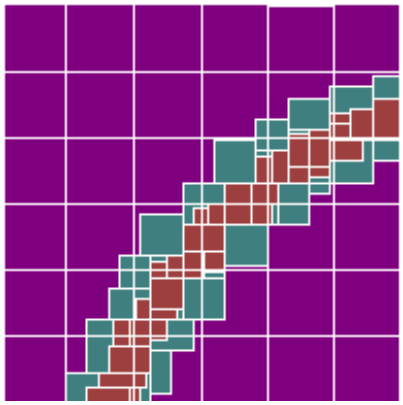
structured AMR
variable shaped patches
neighbor- & parent-communication
MPI parallelization

Enzo-P/Cello

array of octrees
fixed shaped blocks
neighbor-only communication
Charm++ parallelization

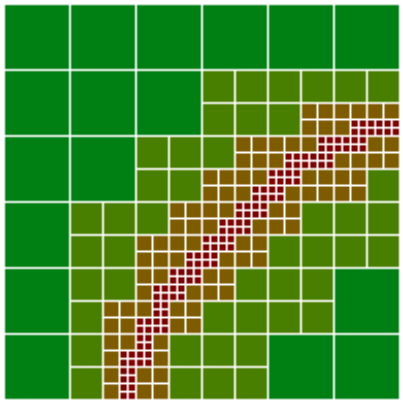
Cello scalable adaptive mesh refinement

Differences between Enzo and Enzo-P



Enzo

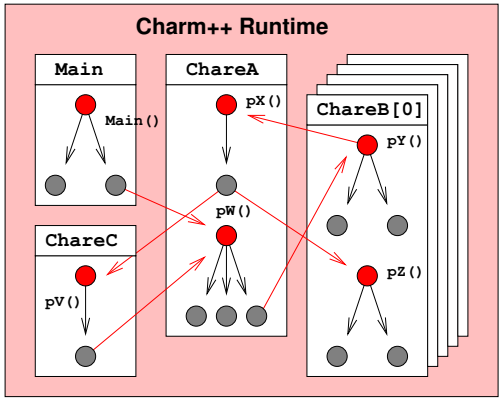
structured AMR
variable shaped patches
neighbor- & parent-communication
MPI parallelization



Enzo-P/Cello

array of octrees
fixed shaped blocks
neighbor-only communication
Charm++ parallelization

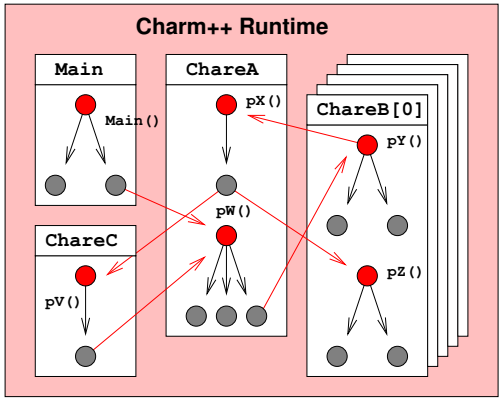
The Charm++ parallel programming system



A Charm++ parallel program

- Charm++ program
  - Decomposed by *objects*
  - Charm++ objects called *chares*
  - invoke *entry methods*
  - *asynchronous*
  - communicate via *messages*
- Charm++ runtime system
  - maps chares to processors
  - schedules entry methods
  - migrates chares to load balance
- Additional features
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance

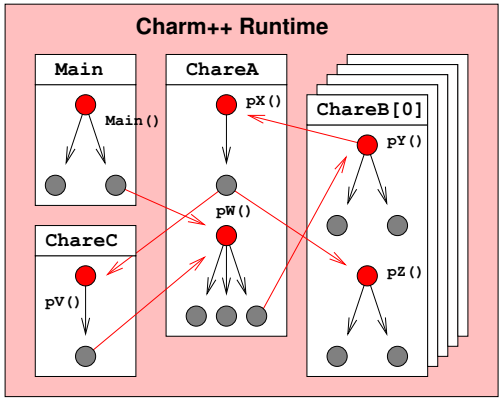
The Charm++ parallel programming system



A Charm++ parallel program

- Charm++ program
  - Decomposed by *objects*
  - Charm++ objects called *chares*
  - invoke *entry methods*
  - *asynchronous*
  - communicate via *messages*
- Charm++ runtime system
  - maps chares to processors
  - schedules entry methods
  - migrates chares to load balance
- Additional features
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance

The Charm++ parallel programming system



A Charm++ parallel program

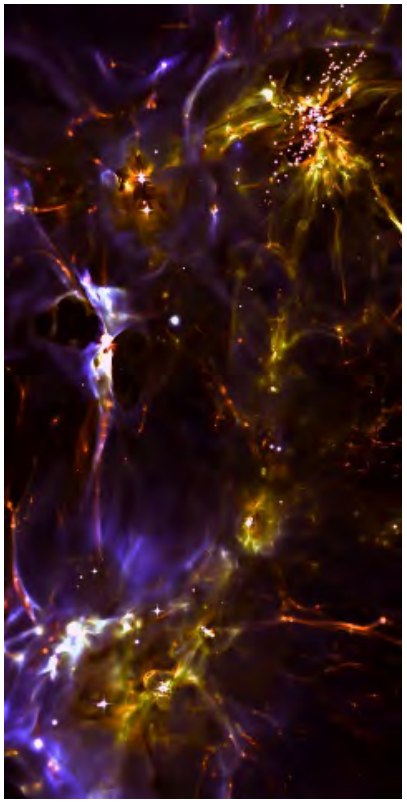
- Charm++ program
  - Decomposed by *objects*
  - Charm++ objects called *chares*
  - invoke *entry methods*
  - *asynchronous*
  - communicate via *messages*
- Charm++ runtime system
  - maps chares to processors
  - schedules entry methods
  - migrates chares to load balance
- Additional features
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance

Cello scalable adaptive mesh refinement

How Enzo-P addresses Enzo's limitations

- Memory usage
  - AMR structure is fully distributed
  - uniform blocks reduce fragmentation
- Mesh quality
  - 2-to-1 refinement constraint maintained
- Parallel task definition
  - uniform field array sizes in blocks
  - sizes determined by user
- Parallel task scheduling
  - asynchronous data-driven
- Data locality
  - primarily nearest-neighbor comm.

Enzo



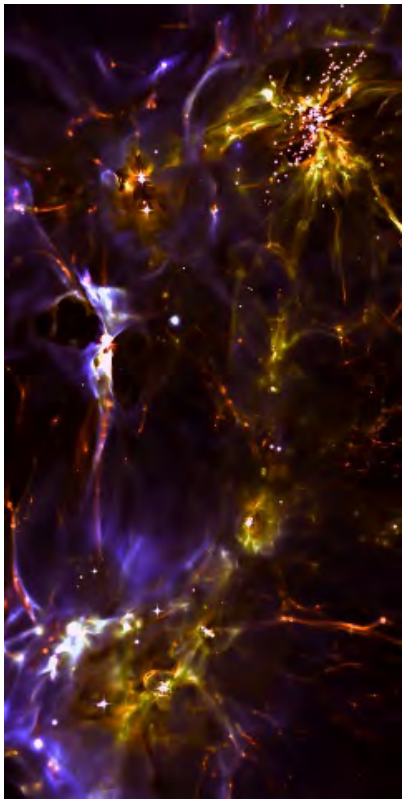
[John Wise]

Cello scalable adaptive mesh refinement

How Enzo-P addresses Enzo's limitations

- **Memory usage**
  - AMR structure is fully distributed
  - uniform blocks reduce fragmentation
- Mesh quality
  - 2-to-1 refinement constraint maintained
- Parallel task definition
  - uniform field array sizes in blocks
  - sizes determined by user
- Parallel task scheduling
  - asynchronous data-driven
- Data locality
  - primarily nearest-neighbor comm.

Enzo



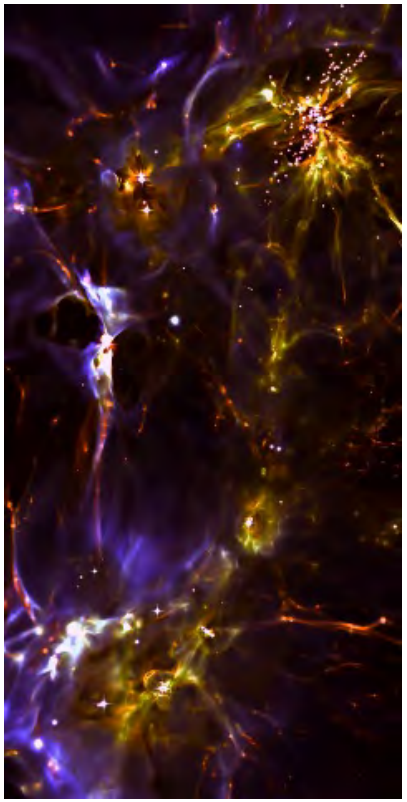
[John Wise]

Cello scalable adaptive mesh refinement

How Enzo-P addresses Enzo's limitations

- Memory usage
  - AMR structure is fully distributed
  - uniform blocks reduce fragmentation
- **Mesh quality**
  - 2-to-1 refinement constraint maintained
- Parallel task definition
  - uniform field array sizes in blocks
  - sizes determined by user
- Parallel task scheduling
  - asynchronous data-driven
- Data locality
  - primarily nearest-neighbor comm.

Enzo



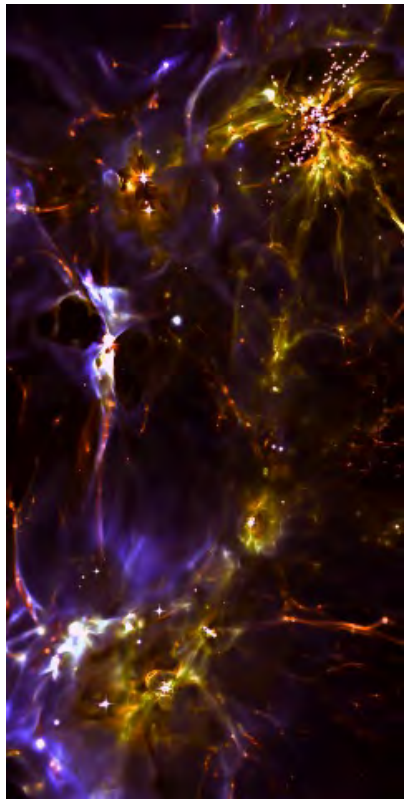
[John Wise]

Cello scalable adaptive mesh refinement

How Enzo-P addresses Enzo's limitations

- Memory usage
  - AMR structure is fully distributed
  - uniform blocks reduce fragmentation
- Mesh quality
  - 2-to-1 refinement constraint maintained
- **Parallel task definition**
  - uniform field array sizes in blocks
  - sizes determined by user
- Parallel task scheduling
  - asynchronous data-driven
- Data locality
  - primarily nearest-neighbor comm.

Enzo



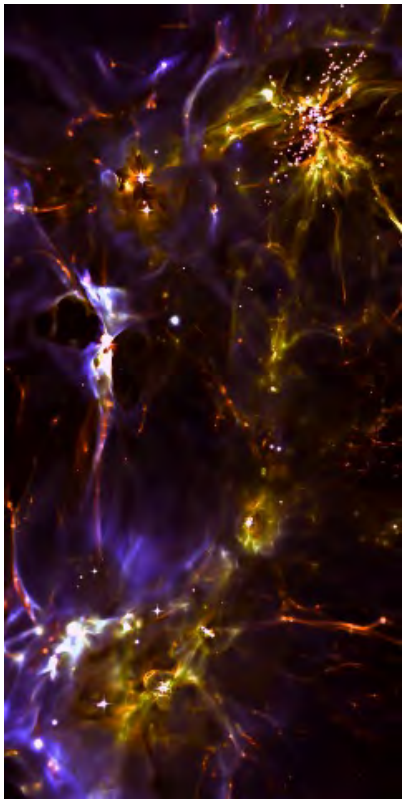
[John Wise]

Cello scalable adaptive mesh refinement

How Enzo-P addresses Enzo's limitations

- Memory usage
  - AMR structure is fully distributed
  - uniform blocks reduce fragmentation
- Mesh quality
  - 2-to-1 refinement constraint maintained
- Parallel task definition
  - uniform field array sizes in blocks
  - sizes determined by user
- **Parallel task scheduling**
  - asynchronous data-driven
- Data locality
  - primarily nearest-neighbor comm.

Enzo



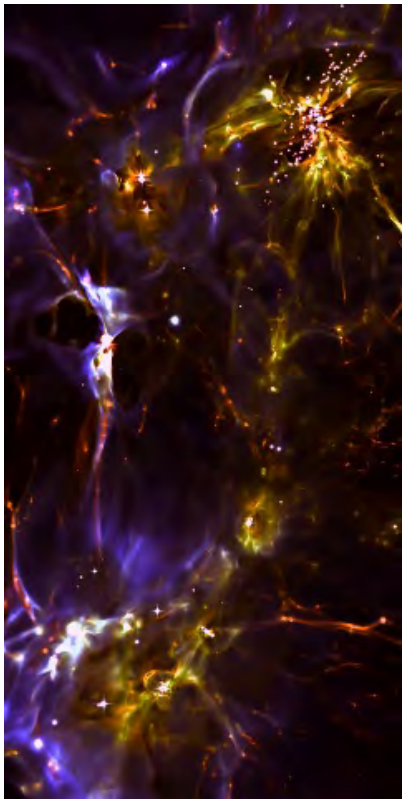
[John Wise]

Cello scalable adaptive mesh refinement

How Enzo-P addresses Enzo's limitations

- Memory usage
  - AMR structure is fully distributed
  - uniform blocks reduce fragmentation
- Mesh quality
  - 2-to-1 refinement constraint maintained
- Parallel task definition
  - uniform field array sizes in blocks
  - sizes determined by user
- Parallel task scheduling
  - asynchronous data-driven
- **Data locality**
  - primarily nearest-neighbor comm.

Enzo

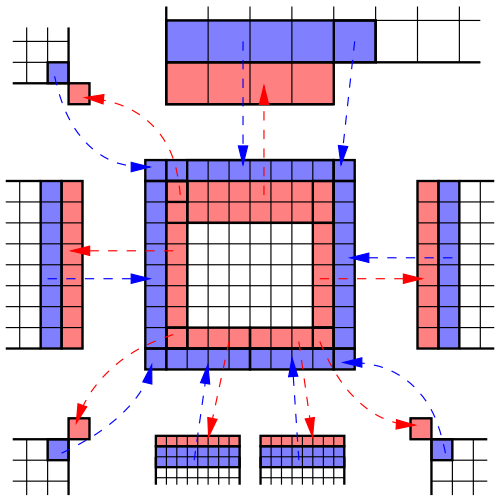


[John Wise]

Cello distributed adaptive mesh refinement

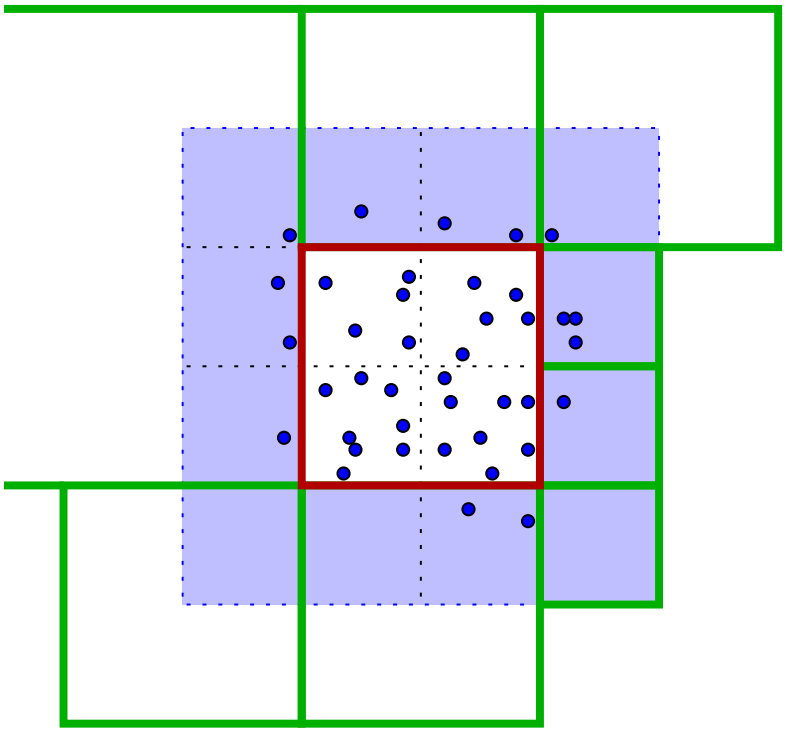
Cello implements a *fully-distributed* array-of-octree mesh hierarchy using a Charm++ *chare array* of `Blocks`. `Block` chare elements are indexed using a bit-coding of the array and octree indices.

- **Data-driven execution**
  - `Field` face data sent when available
  - last update triggers computation
- **Dynamic task scheduling**
  - multiple `Blocks` per process
  - overlaped communication/computation
- Charm++ **also provides**
  - cutting-edge load-balancing schemes
  - double in-memory checkpoint/restart



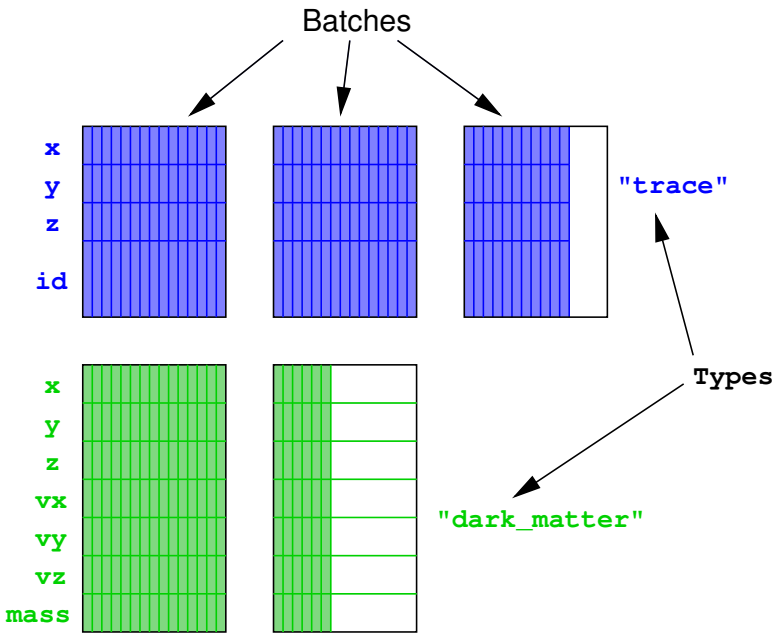
How particle data is communicated between Blocks

- communication is required when particles move outside a Block
- this is done using a 4x4x4 array
  - array contains pointers to ParticleData (PD) objects
  - one PD object per neighbor Block



- migrating particles are
  - `scatter()`-ed to PD array objects
  - sent to associated neighbors
  - `gather()`-ed by neighbors
- one sweep through particles
- one communication step per neighbor

How `Particle` objects store particle data



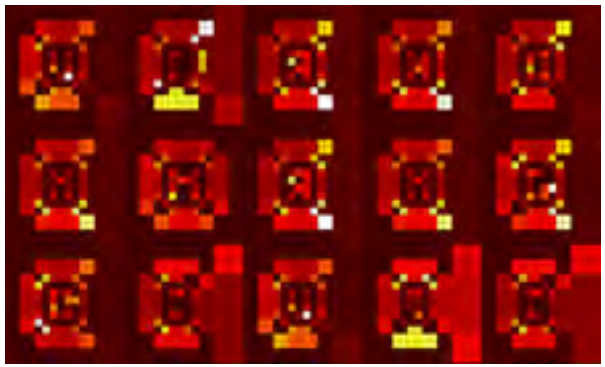
- multiple particle *types*
- particles allocated in *batches*
  - fixed size arrays
  - fewer new/delete operations
  - efficient insert/delete operations
  - potentially useful for GPU's
- batches store particle *attributes*
  - (position, velocity, mass, etc.)
  - 8,16,32,64-bit integers
  - 32,64,128-bit floats

- particle positions may be floating-point or integers
  - floating-point for storing global positions
  - integers for `Block`-local coordinates
    - solves reduced precision issue for deep hierarchies
    - less memory required for given accuracy

Enzo-P/Cello Blue Waters scaling test

Hydro and particles: "Alphabet Soup" problem

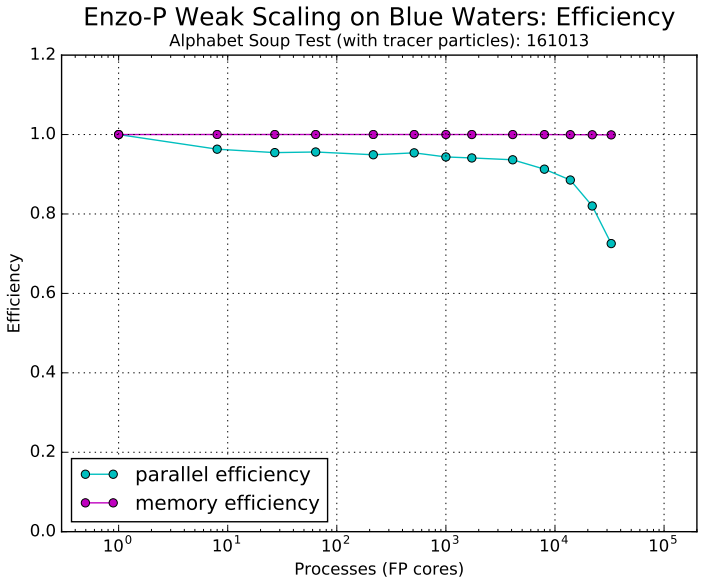
We tested basic Enzo-P hydrodynamics and particles scalability

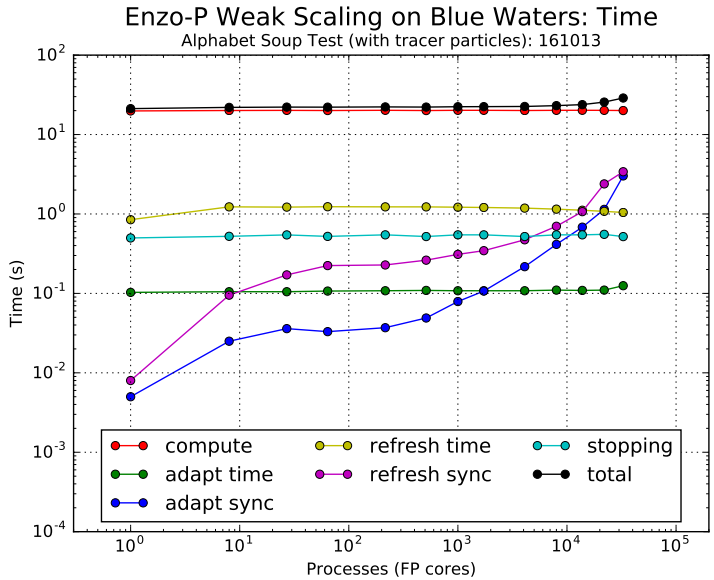


- variation of "array of Sedov Blast" test
- letters instead of spheres
  - inhibits lockstep coarsen/refine
- one letter per BW fp-core
- tested with/without tracer particles
- 32^3 or 24^3 cells per block
- among largest AMR runs ever done
  - $256K$ fp-cores
  - $1.7T$ cells; $0.7T$ (cells + particles)
  - $50M$ Blocks
- Enzo would require $72GB$ / process

Enzo-P/Cello Blue Waters hydro/particle scaling

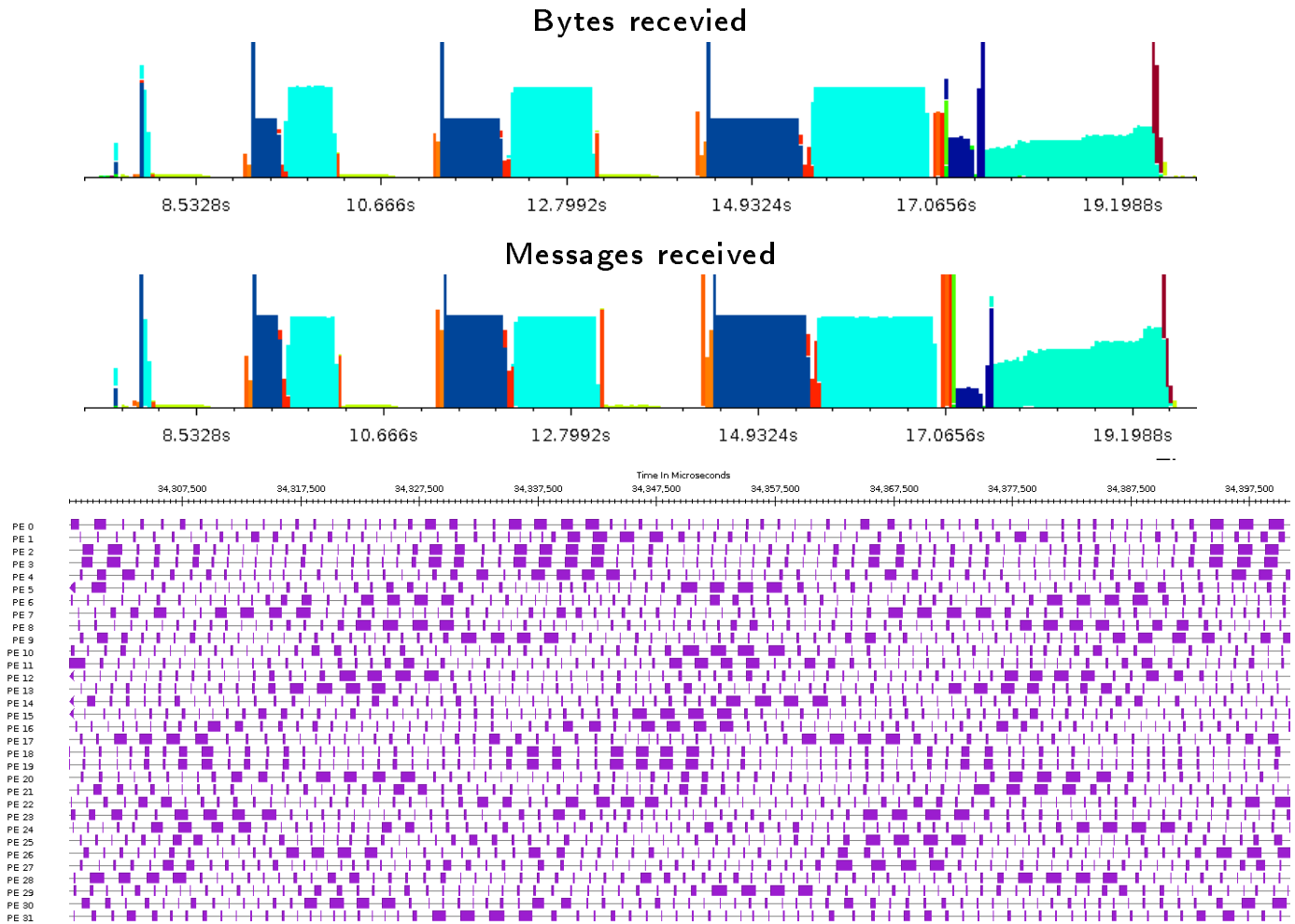
Initial tests: efficiency dropoff before $32K$ fp-cores





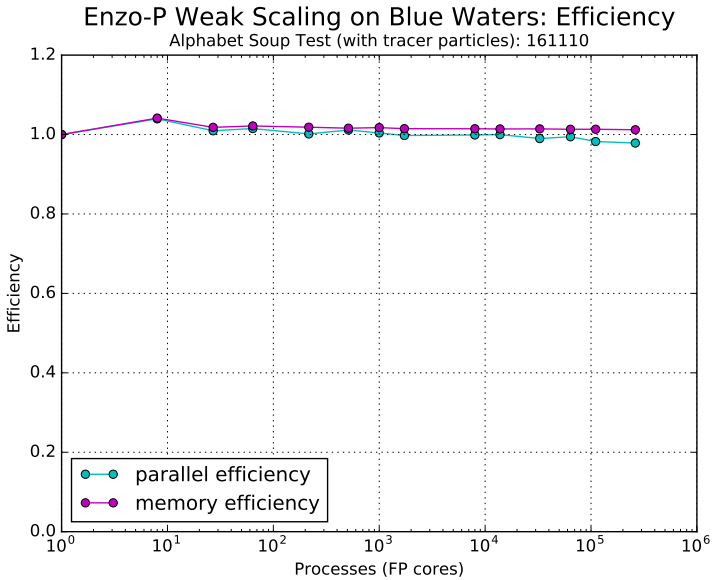
Enzo-P/Cello Blue Waters hydro/particle scaling

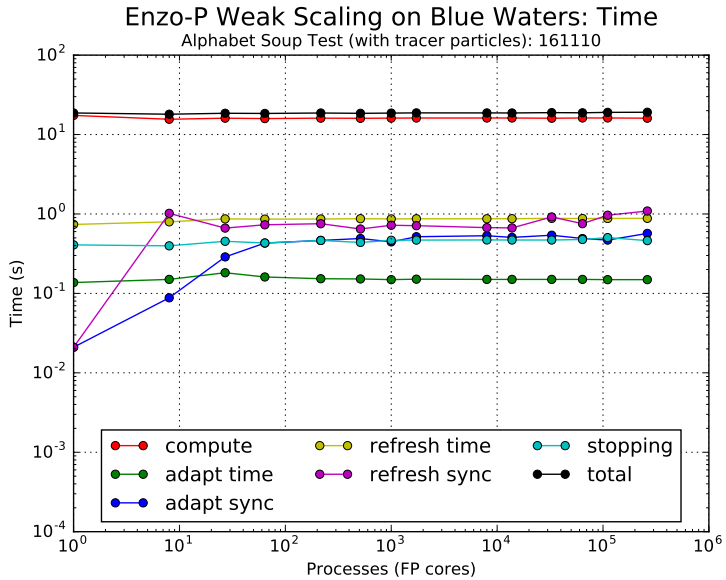
`Projections` analysis: hidden $O(N)$ or $O(P)$



Enzo-P/Cello Blue Waters hydro/particle scaling

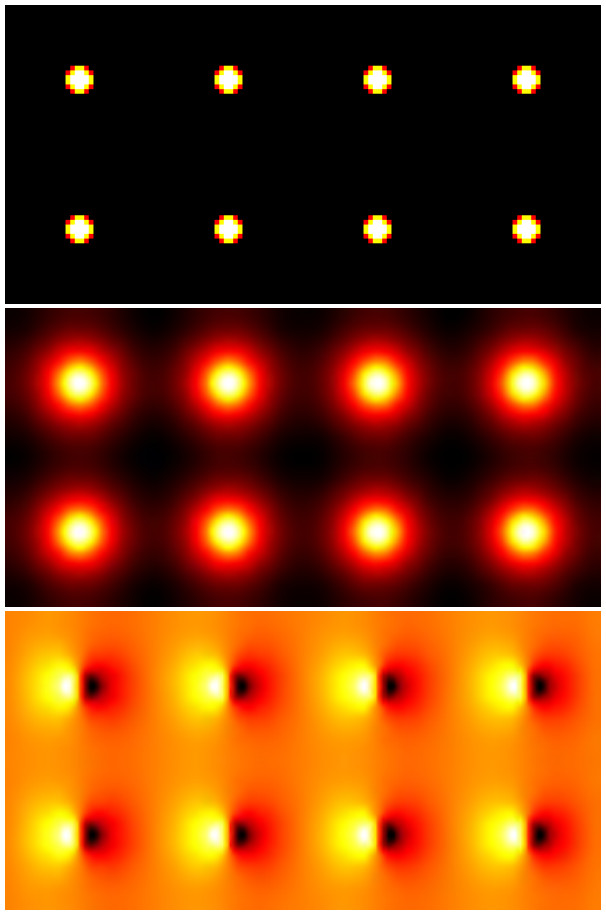
Charm++ SMP mode: excellent efficiency through 256K fp-cores

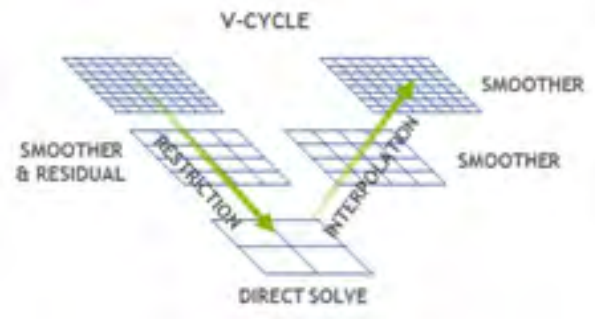




Enzo-P/Cello Blue Waters gravity test

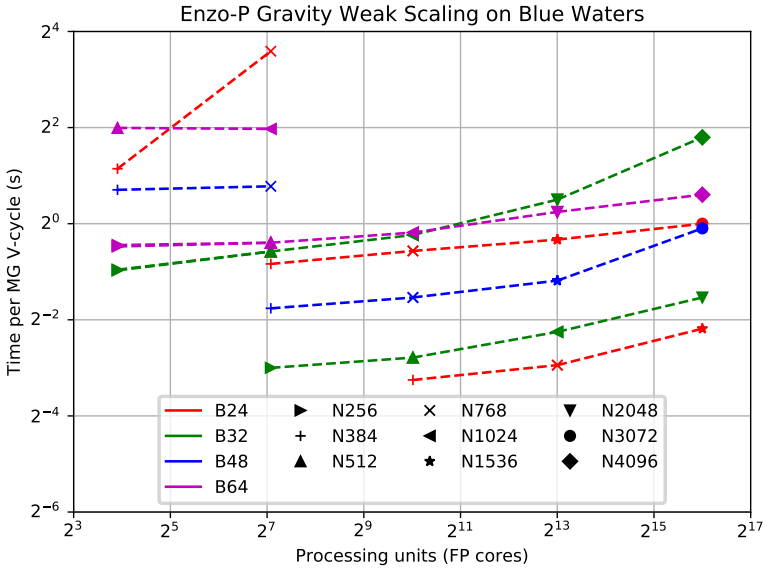
We tested scalability of a recently implemented gravity solver

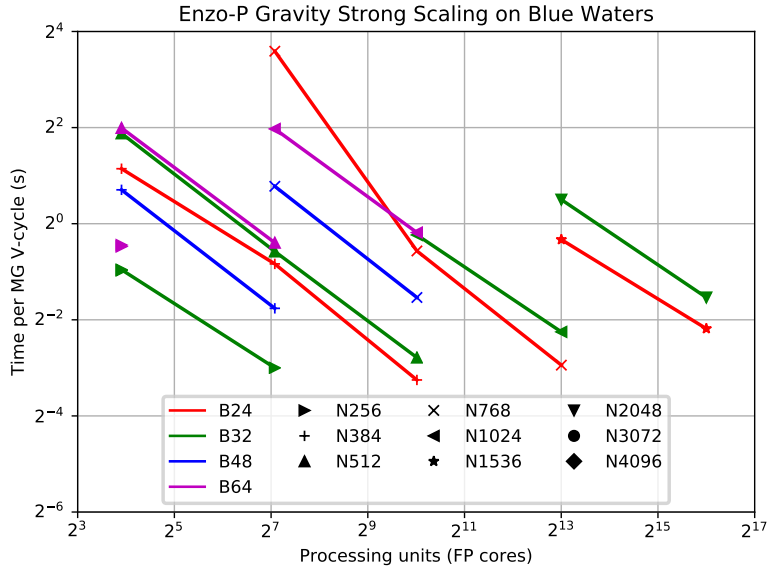




- "array of collapsing spheres" problem
- varied both block size and problem size
- multigrid V-cycle solve (uniform-mesh)
- debugging scalable AMR gravity
  - Reynolds "HG" algorithm
  - BiCG-STAB Krylov solver
  - multigrid-based preconditioner

Enzo-P/Cello Blue Waters gravity scaling





Conclusions and next steps

We are developing Enzo-P/Cello, the next-generation of the Enzo parallel AMR cosmology and astrophysics application

- Scalability of AMR hydro is excellent through $256K$ cores
- Scalability of uniform grid gravity is very good through $64K$ cores
- Scalable AMR gravity capability is around the corner

Next steps include the following

- Analyse and optimize uniform grid gravity
- Test and optimize AMR gravity
- More rigorous and realistic scaling problems
- Exercise Charm++ load balancing algorithms

`http://cello-project.org/`

NSF SI2-SSE-1440709