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
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 . . .

James Bordner  Enzo-P / Cello  Pursuing Petascale Astrophysics  BW Symposium 2017
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

- **Charm++ program**
  - Decomposed by *objects*
  - Charm++ objects called *charles*
  - invoke *entry methods*
  - *asynchronous*
  - communicate via *messages*

- **Charm++ runtime system**
  - maps charles to processors
  - schedules entry methods
  - migrates charles to load balance

- **Additional features**
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance
The Charm++ parallel programming system

- **Charm++ program**
  - Decomposed by *objects*
  - Charm++ objects called *charms*
  - *asynchronous*
  - communicate via *messages*

- **Charm++ runtime system**
  - maps *charms* to processors
  - schedules entry methods
  - migrates *charms* to load balance

- **Additional features**
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance
The Charm++ parallel programming system

- Charm++ program
  - Decomposed by objects
  - Charm++ objects called *char*es
  - invoke *entry methods*
  - *asynchronous*
  - communicate via *messages*

- Charm++ runtime system
  - maps *char*es to processors
  - schedules *entry methods*
  - migrates *char*es 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.
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.

[John Wise]
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.
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.
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.

[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.
Cello implements a fully-distributed array-of-octree mesh hierarchy using a Charm++ char array of Blocks. Block char array 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
  - overlapped 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
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 Weak Scaling on Blue Waters: Efficiency

Enzo-P Weak Scaling on Blue Waters: Time
Enzo-P/Cello Blue Waters hydro/particle scaling
Projections analysis: hidden $O(N)$ or $O(P)$

Bytes received

Messages received

James Bordner
Enzo-P / Cello  Pursuing Petascale Astrophysics  BW Symposium 2017
Enzo-P/Cello Blue Waters hydro/particle scaling
Charm++ SMP mode: excellent efficiency through 256K fp-cores

Enzo-P Weak Scaling on Blue Waters: Efficiency
Alphabet Soup Test (with tracer particles): 161110

Enzo-P Weak Scaling on Blue Waters: Time
Alphabet Soup Test (with tracer particles): 161110

James Bordner
Enzo-P / Cello Pursuing Petascale Astrophysics
BW Symposium 2017
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
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/