Enzo-P/Cello

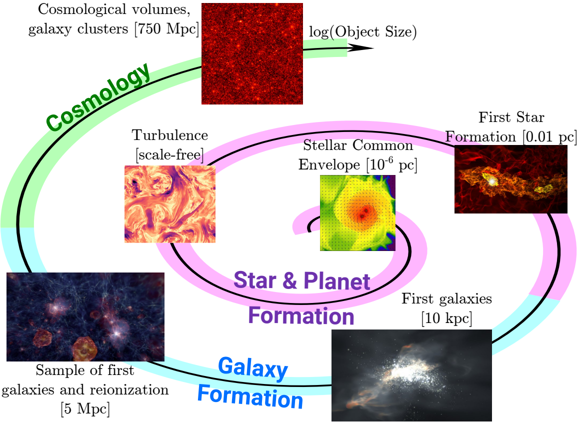
Computational Cosmology and Astrophysics on Adaptive Meshes using Charm++

James Bordner, Michael L. Norman

University of California, San Diego
San Diego Supercomputer Center

Supercomputing 2018
2018-11-11/16

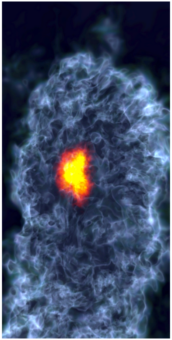
Scientific questions in astrophysics and cosmology



[John Wise]

The ENZO MPI-parallel AMR astrophysics application

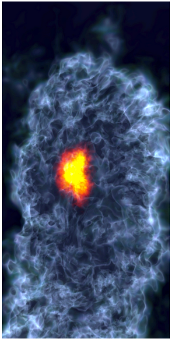
- **ENZO is a very powerful application**
  - science: applicable to wide range of problems
  - equations: rich multiphysics capabilities
  - methods: wide range of numerical solvers
  - data structures: SAMR, fields + particles
- **Its main limitation is parallel scalability**
  - ENZO was developed starting in 1994
  - "massive parallelism" meant $P \approx 10^3$
  - $P \approx 10^7$ today
- **Motivated AMR data structure redesign**
  - **Enzo-P**: "Petascale" branch of ENZO
  - keep ENZO's physics and many methods
  - **Cello** highly scalable AMR framework



[Sam Skillman, Matt Turk]

The ENZO MPI-parallel AMR astrophysics application

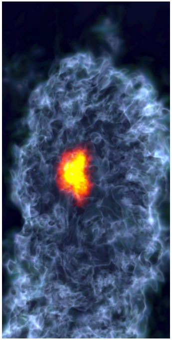
- **ENZO is a very powerful application**
  - science: applicable to wide range of problems
  - equations: rich multiphysics capabilities
  - methods: wide range of numerical solvers
  - data structures: SAMR, fields + particles
- **Its main limitation is parallel scalability**
  - ENZO was developed starting in 1994
  - "massive parallelism" meant $P \approx 10^3$
  - $P \approx 10^7$ today
- **Motivated AMR data structure redesign**
  - **Enzo-P**: "Petascale" branch of ENZO
  - keep ENZO's physics and many methods
  - **Cello** highly scalable AMR framework



[Sam Skillman, Matt Turk]

The ENZO MPI-parallel AMR astrophysics application

- **ENZO is a very powerful application**
  - science: applicable to wide range of problems
  - equations: rich multiphysics capabilities
  - methods: wide range of numerical solvers
  - data structures: SAMR, fields + particles
- **Its main limitation is parallel scalability**
  - ENZO was developed starting in 1994
  - "massive parallelism" meant $P \approx 10^3$
  - $P \approx 10^7$ today
- **Motivated AMR data structure redesign**
  - **Enzo-P**: "Petascale" branch of ENZO
  - keep ENZO's physics and many methods
  - **Cello** highly scalable AMR framework



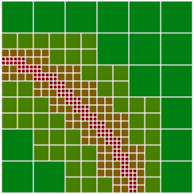
[Sam Skillman, Matt Turk]

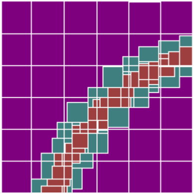
Cello scalable adaptive mesh refinement

Key differences between ENZO and Enzo-P

Enzo-P/Cello

- array of octrees AMR
- Charm++ parallelization
- reusable AMR framework





ENZO

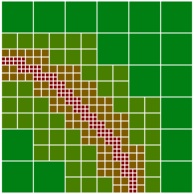
- structured AMR
- MPI+OpenMP parallelization
- non-encapsulated AMR data structure

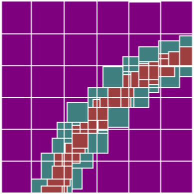
Cello scalable adaptive mesh refinement

Key differences between ENZO and Enzo-P

Enzo-P/Cello

- array of octrees AMR
- Charm++ parallelization
- reusable AMR framework





ENZO

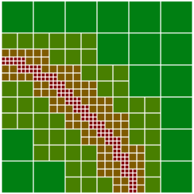
- structured AMR
- MPI+OpenMP parallelization
- non-encapsulated AMR data structure

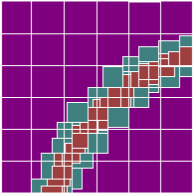
Cello scalable adaptive mesh refinement

Key differences between ENZO and Enzo-P

Enzo-P/Cello

- array of octrees AMR
- Charm++ parallelization
- reusable AMR framework





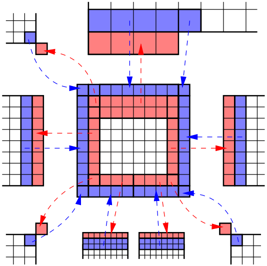
ENZO

- structured AMR
- MPI+OpenMP parallelization
- non-encapsulated AMR data structure

Cello distributed adaptive mesh refinement

How field data are communicated between blocks

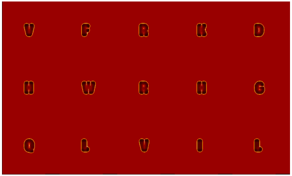
- **Data-driven execution**
  - send `Field` face data when available
  - indexed using bit-coding in hierarchy
  - count face data received
  - last receive triggers computation
- **Dynamic task scheduling**
  - multiple `Blocks` per process
  - overlapped communication/computation

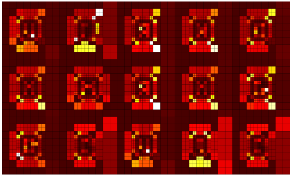


Enzo-P/Cello NSF Blue Waters scaling

"Alphabet Soup" test: hydro and particles

We tested basic Enzo-P hydrodynamics and particles scalability

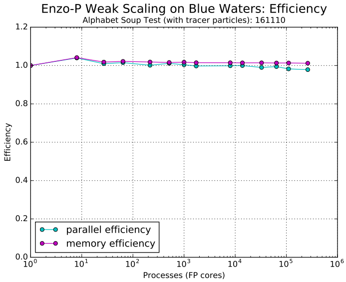


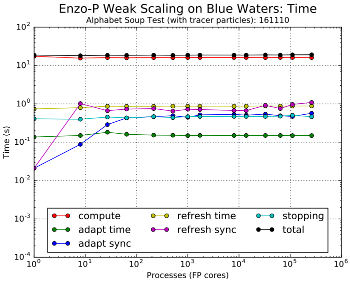


- variation of "array of Sedov Blast" test
- letters instead of spheres
  - inhibits lockstep coarsen/refine
- one blast per BW fp-core
- tested with/without tracer particles
- 32^3 or 24^3 cells per block
- decent sized AMR problem for 2016
  - $262K$ fp-cores
  - $50M$ Blocks
  - $1.7T$ cells; $0.7T$ (cells + particles)
- ENZO would need $72GB$ per process!

Enzo-P/Cello NSF Blue Waters scaling

"Alphabet Soup" test: hydro and particles

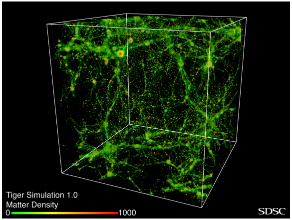




Enzo-P/Cello NSF Blue Waters scaling

"Unigrid Cosmology" test: hydro, particles, gravity

We tested scaling of more recent support for cosmology

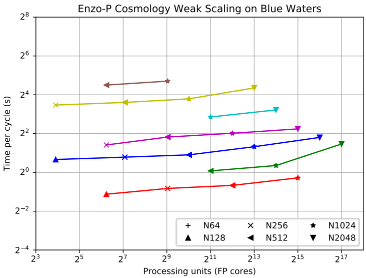


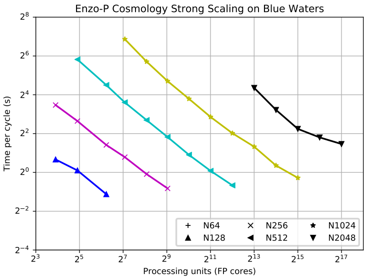
[Renyue Cen]

- PPM hydrodynamics
- "dark matter" particles
- CIC particle-mesh gravity
- multigrid solver—"unigrid" only
- tested up to 131K fp-cores
- have since implemented AMR solver
  - Dan Reynolds "HG" algorithm
  - MG preconditioned BiCG-STAB
  - "semi-scalable"
  - DD and AFACx MG solvers soon

Enzo-P/Cello NSF Blue Waters scaling

“Unigrid Cosmology” test: hydro, particles, gravity





Conclusions and next steps

Enzo-P/Cello is a highly scalable branch of ENZO

- Charm++ enables fully-distributed AMR mesh hierarchy
- excellent hydro weak scaling through $P = 262K$
- very good "unigrid" cosmology scaling through $P = 131K$
- scalable AMR cosmology soon (< 1 month)

Next steps include

- improve I/O scalability
- implement block-adaptive time-stepping
- run large-scale AMR cosmology simulations
- next phase: *Enzo-E*
  - improve strong scaling
  - support heterogeneous hardware
  - increase ENZO developer engagement

Acknowledgements

Funding for development of Enzo-P/Cello has been provided by the National Science Foundation grants OAC-1835402, SI2-SSE-1440709, PHY-1104819, and AST-0808184.

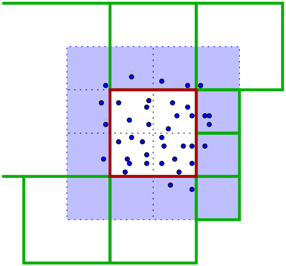
This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and the National Center for Supercomputing Applications.

http://cello-project.org/

Cello distributed adaptive mesh refinement

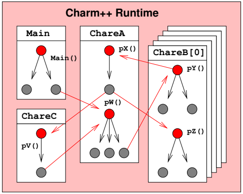
How particle data are communicated between Blocks

- 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

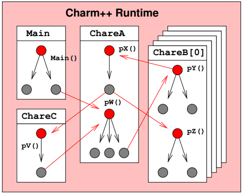
The Charm++ parallel programming system



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
  - can migrate chares
- Additional features
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance

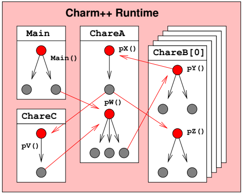
The Charm++ parallel programming system



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
  - can migrate chares
- Additional features
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance

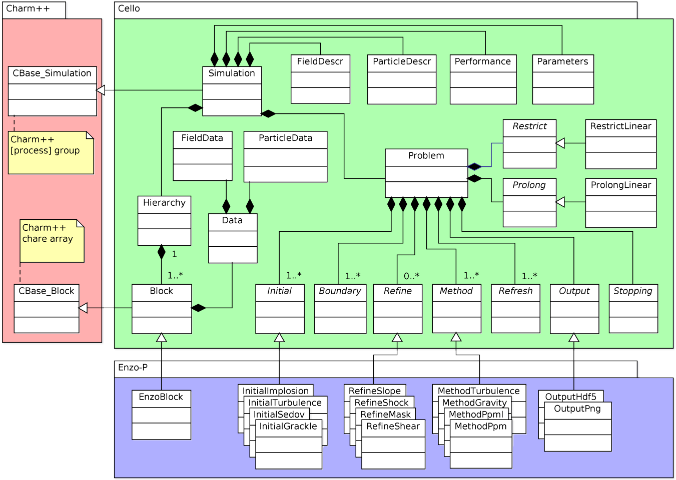
The Charm++ parallel programming system



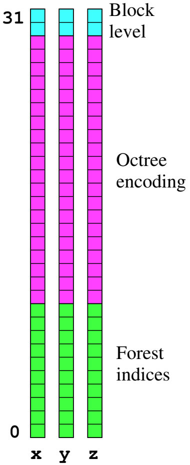
A Charm++ parallel program

- 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
  - can migrate chares
- Additional features
  - checkpoint/restart
  - dynamic load balancing
  - fault-tolerance

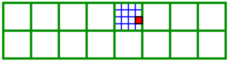
Enzo-P/Cello/Charm++ class organization



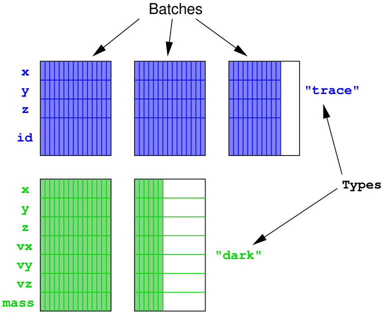
How the `Block` chare array is indexed



- User-defined chare array indices supported
- Cello indices for `Block` arrays:
  - 3×10 bits for *array indices*
  - 3×20 bits for the *octree encoding*
  - 6 bits for the *block level*
- Up to 1024^3 array of octrees
- Up to 20 octree levels
- $-31 \leqslant \text{level} \leqslant 31$
- Block id's use index: e.g. `B100:11_1:01`



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

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	$\times$1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

*not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	$\times$1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

*not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	×1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	×1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

*not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	$\times$1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

* not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	**Enzo-P/Cello**
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	$\times$1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

* not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	$\times$1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

* not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	$\times$1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

* not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	×1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

* not implemented yet

How do Enzo-P and ENZO differ?

	ENZO	Enzo-P/Cello
Parallelization	MPI	Charm++
AMR type	patch-based	octree-based
AMR structure	replicated	fully distributed
Time stepping	level-adaptive	block-adaptive*
Block sizes	$\times$1000 variation	constant
Task scheduling	level-parallel	dependency-driven
Load balancing	patch migration	Charm++-based
Data locality	LB conflict	no LB conflict
Mesh quality	level jumps	2-to-1 constraint

* not implemented yet