Computational Cosmology and Astrophysics on Adaptive Meshes using Charm++

James Bordner
San Diego Supercomputer Center
University of California, San Diego
La Jolla, CA, U.S.
jobordner@ucsd.edu

Michael L. Norman
San Diego Supercomputer Center
University of California, San Diego
La Jolla, CA, U.S.
mlnorman@ucsd.edu

***Abstract*— Astrophysical and cosmological phenomena involve a large variety of physical processes, and can encompass an enormous range of scales. To effectively investigate these phenomena computationally, applications must similarly support modeling these phenomena on enormous ranges of scales; furthermore, they must do so efficiently on high-performance computing platforms of ever-increasing parallelism and complexity. We describe Enzo-P, a Petascale redesign of the ENZO adaptive mesh refinement astrophysics and cosmology application, along with Cello, a reusable and scalable adaptive mesh refinement software framework, on which Enzo-P is based. Cello's scalability is enabled by the Charm++ Parallel Programming System, whose data-driven asynchronous execution model is ideal for taking advantage of the available but irregular parallelism in adaptive mesh refinement-based applications. We present scaling results on the NSF Blue Waters supercomputer, and outline our future plans to bring Enzo-P to the Exascale Era by targeting highly-heterogeneous accelerator-based platforms.**

***Index Terms*—Adaptive mesh refinement, Astrophysics, Octrees, Petascale computing**

I. Introduction

There are numerous astrophysics and cosmology topics of scientific interest, such as early star formation, galaxy formation, galaxy clusters, and cosmic reionization. These astrophysical phenomena typically involve a variety of physical processes, including hydrodynamics; gravity; gas chemistry, heating, and cooling; radiative transfer; and cosmological expansion. Investigating astrophysical phenomena computationally thus requires a powerful multiphysics software application with a wide variety of numerical methods. Astrophysical phenomena also encompass an enormous range of spatial and temporal scales (see Fig. 1). Astrophysical simulations must be able to span these scales, sometimes in the same simulation, such as when simulating early star formation using cosmological initial conditions [1]. Astrophysics applications thus require some means to efficiently represent a wide dynamic range of scales.

ENZO [2] is an MPI parallel application designed for multiphysics astrophysical and cosmological science simulations, and was the first such application to use adaptive mesh refinement (AMR) [3], [4] to increase the spatial and temporal dynamic range [5]. However, while powerful in terms of multiphysics and multiresolution capabilities, ENZO does have some limitations with respect to its parallel scalability. In particular, although ENZO scales well in uniform grid mode, it does not scale well beyond a few thousand cores when AMR is used, limiting its applicability. This is not surprising, since development on ENZO began in the early 1990's when "extreme scale" meant hundreds of CPU's, not today's millions of cores. Since then, attempting to improve ENZO's scalability by over $\times 1000$ to keep pace with the relentless increase in HPC platform parallelism has become increasingly difficult, and many of ENZO's remaining scaling bottlenecks cannot be resolved without a fundamental overhaul of its parallel AMR design and implementation.

This motivated us to develop the "Petascale ENZO"

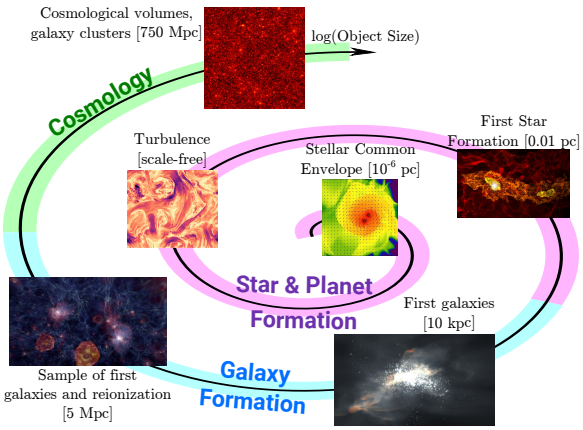

Fig. 1. Astrophysical phenomena can range over extreme scales.

Funding for this work has been provided by the National Science Foundation through grants SI2-SSE-1440709, PHY-1104819, AST-0808184, and OAC-1835402

fork of the ENZO code called *Enzo-P*, using a highly scalable AMR framework called *Cello* [6], which we are developing concurrently. Key features of Cello are that it implements a fully distributed "array-of-octrees" AMR approach, and Cello is parallelized using Charm++ [7] rather than MPI.

Below we discuss the Enzo-P / Cello / Charm++ software stack in §II, we present weak-scaling results of Enzo-P / Cello / Charm++ in §III, and we conclude with our future plans in §IV.

II. Enzo-P / Cello / Charm++

In this section we describe the software stack of Enzo-P / Cello / Charm++ from the bottom up: in §II-A we review the Charm++ parallel programming system, in §II-B we describe the Cello adaptive mesh refinement framework, and in §II-C we discuss the Enzo-P science application layer.

A. The Charm++ parallel language

Charm++ [7], [8], developed at the Parallel Programming Laboratory (PPL) at the University of Illinois, is the visionary parallel programming system on which Cello is built. Since its first public release over 20 years ago, Charm++ has been continuously enhanced and improved by researchers in the PPL in collaboration with application developers in diverse areas of science and engineering.

In Charm++ programs, the fundamental parallel object is a *chare*. Chares are C++ objects that contain special methods called *entry methods*. Entry methods may be invoked remotely by other chares via *proxies*, and communicate with each other using *messages*. Related chares may be grouped together into a *chare array*, in which individual chares are accessed using an array proxy plus element index. Additionally, the Charm++ runtime system supports automatic dynamic load balancing of chares within chare arrays. The *runtime system* manages chares, assigning their location in distributed memory, dynamically migrating chares to balance load, communicating message data between chares, and dynamically scheduling and executing entry methods.

Numerous science and engineering applications have been developed using Charm++. Perhaps the best known is NAMD, a Gordon Bell and Sidney Fernbach Award-winning parallel molecular dynamics code, which has scaled to beyond $500K$ cores [9]. Other Charm++ codes include OpenAtom [10], a highly scalable quantum chemistry application; ChaNGa [11], a scalable collisionless N-body code for cosmological simulations; and EpiSimdemics[12], a simulation system for studying the spread of contagion over large interaction networks.

Applications built on Charm++ directly benefit from its natural latency tolerance, overlap of communication and computation, dynamic load balancing, and checkpoint / restart capabilities. Emerging scalability issues, including fault-tolerance and improving energy efficiency, are also provided by Charm++, with active research in energy-aware rollback-recovery [13], fault-tolerance protocols [14], automated checkpoint / restart mechanisms [15], and automated thermal-aware load balancing [16]. Since Charm++ is based on an introspective and adaptive runtime system, it is well-suited to meet the challenges of increasingly complex hardware and software, and is poised to be a programming model of choice for the Exascale Era.

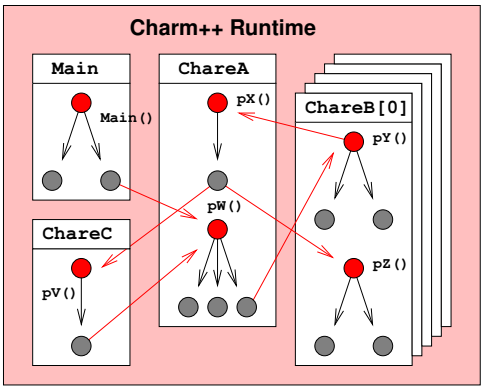

Fig. 2. Charm++ supports object-based data-driven asynchronous parallel programming.

B. The Cello AMR framework

To enable highly scalable multi-resolution simulations in Enzo-P, we are developing Cello, an extremely scalable adaptive mesh refinement framework. Of the two commonly used AMR approaches, structured AMR (SAMR) and octree-based AMR, we decided to implement an octree-based approach in Cello, even though ENZO itself uses SAMR (see Fig. 3). This design decision was made due to its demonstrated high-scalability [17], its relatively uniform parallel task sizes, and the relative simplicity of its mesh hierarchy data structure. Cello implements a slightly more general "array-of-octrees" approach, to allow for non-cubical computational domains.

Cello uses Charm++ to implement two parallel data structures: a "Simulation" process group for storing data associated with each process, and a Block chare array for representing the distributed AMR hierarchy, where each Block is associated with a single node in the array-of-octrees. Each Block contains a Data object, which stores the field and particle data associated with that block. Field and Particle objects in Cello provide applications with easy-to-use API's for accessing field and particle data on the Block.

Cello also provides simple C++ base classes for Cello applications to inherit from to implement computational

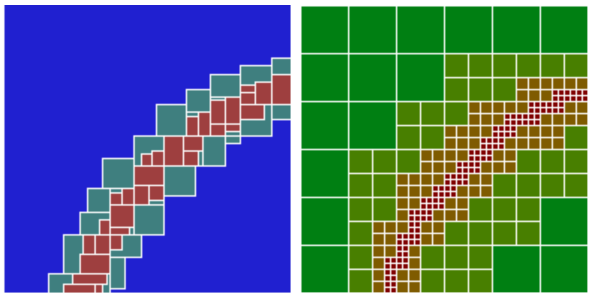

Fig. 3. Whereas ENZO uses structured AMR (left), Enzo-P uses "array-of-octree" AMR (right) provided provided by Cello.

methods (Method), initial and boundary conditions (Initial and Boundary), refinement criteria (Refine), inter-level data interpolation or coarsening (Restrict and Prolong), linear solvers (Solver), I/O (Output), etc. Adding new functionality to a Cello application, say a new physics kernel or a new refinement criterion, is straightforward; typically, it involves adding a new inherited C++ class and implementing one or two virtual methods, which operate on a single Block. This object-oriented approach helps provide extensibility, manageability, and composability to Cello applications.

User-implemented Method's acting on a Block typically require a layer of "ghost" data surrounding the block to be up-to-date, despite the data being assigned to a neighboring Block (see Fig. 4). This is handled entirely by Cello, with the application only needing to specify which Field's ghost zones need to be updated, and how many cells wide the ghost data layer is.

Data locality and optimizing data movement are increasingly crucial for high performance scalable parallel software. While inter-node data locality and movement is controlled by Charm++ through its wide variety of leading-edge dynamic load balancing algorithms, intra-node data locality is handled by Cello. Field and Particle objects organize data in memory to improve cache memory hierarchy performance. This can be done by specifying the order of field and particle attributes, aligning field memory addresses in memory, adding extra padding between field arrays, allocating particle data in fixed-sized batches to reduce memory management overhead, and interleaving field values or particle attributes.

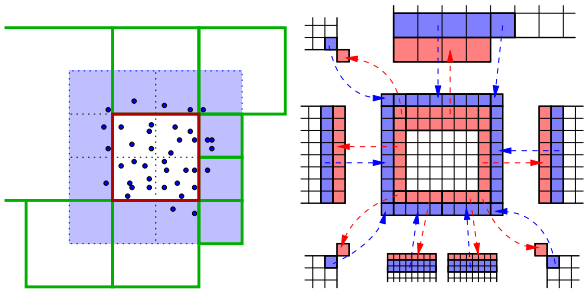

Fig. 4. Cello supports both Lagrangian particles and Eulerian grid computations. Particles may move between Blocks (left), and grid boundary "ghost" data are communicated between Blocks asynchronously, with the last update received triggering the next Method.

C. The Enzo-P astrophysics application

Enzo-P is the astrophysics and cosmology application being developed using Cello. While it does not yet support the full range of physics capabilities of its parent application ENZO, it has reached the point where it is capable of running scientifically viable cosmological simulations of sizes limited only by the available HPC hardware.

Enzo-P supports a growing range of numerical methods, including the core hydrodynamics and gravity solvers, as well as chemistry and cooling via the GRACKLE software library [18]. The main hydrodynamical solver is a modified piece-wise parabolic method (PPM) [19], [20], implemented as a Cello Method (ppm). Another PPM method, PPML [21], is available for magnetized supersonic turbulence simulations (ppml). Enzo-P currently solves for the gravitational potential using a cloud-in-cell particle-mesh (PM) method (gravity). The Method can one of several linear solvers, including preconditioned CG and BiCG-STAB Krylov solvers (cg and bicgstab), or a geometric multigrid V-cycle solver (mg0). Our current linear solver of choice is HG, developed by Dan Reynolds, and implemented as a composite of other available Solvers. Enzo-P also supports cosmological units, reading cosmological initial conditions from HDF5 files, and cosmological comoving expansion terms.

III. Parallel Scaling Results

Demonstrative weak scaling tests on the NSF Blue Waters (BW) Petascale platform indicate that Enzo-P AMR hydrodynamics and (non-AMR) cosmology simulations can scale extremely well in both time and memory. (AMR cosmology scaling tests are underway, and results are expected by the time this paper is available.) All tests were run using Charm++ configured for SMP-mode, and to use the native Cray GNI network layer.

Fig. 5 shows the weak scaling results of a hydrodynamic test problem with AMR and tracer particles. The problem involves a 3D array of blast waves, with one blast wave per processor core. The weak scaling test involves varying the array size from 1^3 to 64^3. To inhibit lockstep mesh refinement, instead of a sphere, the blast is sourced by a high pressure region in the shape of a letter of the alphabet chosen at random for each core. The initial state for each blast problem is refined by a 5-level octree, which resulted in on average about 200 Blocks (chares) per octree (core), (such over-decomposition is

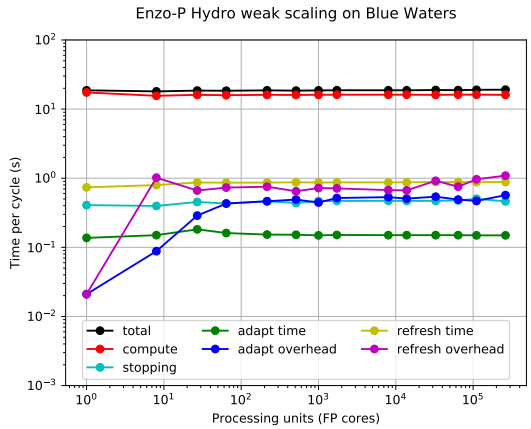

Fig. 5. Weak scaling of Enzo-P / Cello hydrodynamics on up to 262K floating-point cores of Blue Waters

key to Charm++'s high efficiency). Here, the Block's grid size is 32^3 cells, so each blast wave is initialized with about 6.5M cells and particles. The largest problem run had $64^3 = 262,144$ cores (octrees), 52.7M chares, and 1.4 trillion cells plus particles. We note that this demonstration test problem, despite its relative simplicity and balanced workload, is well beyond ENZO's ability to run—it would require about 182GB per process simply to store the AMR metadata!

As shown in Fig. 5, the AMR overhead remains a small fraction of the total cost through to the largest problem run. The parallel efficiency is very good, achieving about 94% at 262K floating-point cores. Memory scaling is virtually ideal, due to the chare array storing the array-of-octrees data being fully distributed (i.e., no replicated data).

Fig. 6 shows scaling results for a simple cosmology simulation on a uniform grid. The problem solves the coupled equations for single species adiabatic gas dynamics, dark matter N-body dynamics, and self-gravity on a uniform mesh seeded with cosmological perturbations. Here a chare is a 32^3 block containing hydrodynamic fields and dark matter particles. Self-gravity is computed using Enzo-P's mg0 multigrid V-cycle solver. Scaling tests consist of grid sizes ranging from 64^3 to 2048^3 cells/particles, fixing the number of chares per core. We see excellent weak scaling results (solid lines), with deviations from ideal reflecting the $O(\log P)$ scaling of the multigrid algorithm itself, and some tailoff due to work starvation at the highest core count (128K) for the relatively small 2048^3 problem. Strong scaling results are excellent as well (dashed lines), and again show some tailoff due to work starvation.

IV. Future Work

Enzo-P has demonstrated excellent scaling on Petascale platforms, as enabled by Charm++'s data-driven asynchronous approach. As we continue augmenting the physics capabilities of Enzo-P to that of its parent code ENZO, we are also preparing for astrophysics and cosmology at the Exascale.

While Charm++ is specifically targeting supporting Exascale applications, achieving the required strong scalability on highly heterogeneous architectures will likely require comprehensive rethinking of software at all levels, including the Cello AMR software framework and even the application layer itself.

Our approach to developing Enzo-E will be to collaborate with the Charm++ group, which has already implemented techniques to simplify programming of heterogeneous systems through additional keywords, generating multiple versions of work unit kernels, and agglomerating work units when required. We will build on Cello's existing capabilities by enhancing its Field, Particle, and other Data objects to support multiple heterogeneous memory spaces, and, working with the Charm++ group, we will develop a dynamic load balancing method that maintains high data locality to improve the efficiency of work unit agglomeration. Together, Cello and Charm++ will help isolate the complexity of heterogeneous hardware from Enzo-E as much as is feasible, while still providing a means for Enzo-E numerical methods to efficiently use multiple accelerators when available.

Acknowledgments

We thank Sanjay Kalé and the Parallel Programming Laboratory at the University of Illinois, Urbana Champaign for their helpful discussions concerning using Charm++, and we thank John Wise for the "extreme scales" image in Fig.1.

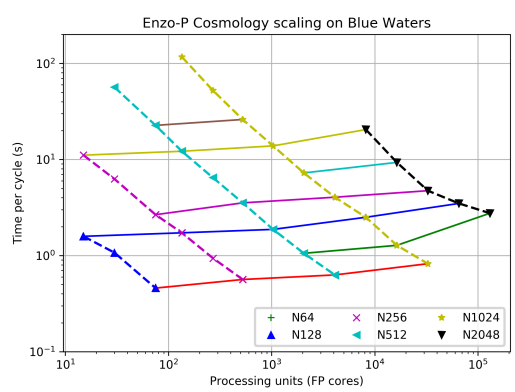

Fig. 6. Weak (dashed line) and strong (solid line) scaling of a basic cosmology simulation (uniform grid with hydrodynamics plus dark matter) on up to 128K cores.

REFERENCES

[1] M. L. Norman, "Population III Star Formation and IMF," in *First Stars III*, ser. American Institute of Physics Conference Series, B. W. O'Shea and A. Heger, Eds., vol. 990, Mar. 2008, pp. 3–15.

[2] G. L. Bryan, M. L. Norman, B. W. O'Shea, T. Abel, J. H. Wise, M. J. Turk, D. R. Reynolds, D. C. Collins, P. Wang, S. W. Skillman, B. Smith, R. P. Harkness, J. Bordner, J.-h. Kim, M. Kuhlen, H. Xu, N. Goldbaum, C. Hummels, A. G. Kritsuk, E. Tasker, S. Skory, C. M. Simpson, O. Hahn, J. S. Oishi, G. C. So, F. Zhao, R. Cen, Y. Li, and Enzo Collaboration, "ENZO: An Adaptive Mesh Refinement Code for Astrophysics," *The Astrophysical Journal, Supplement*, vol. 211, p. 19, Apr. 2014.

[3] M. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[4] M. J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics," *Journal of Computational Physics*, vol. 82, pp. 64–84, May 1989.

[5] G. L. Bryan and M. L. Norman, "A Hybrid AMR Application for Cosmology and Astrophysics," *ArXiv Astrophysics e-prints*, Oct. 1997.

[6] J. Bordner and M. L. Norman, "Enzo-P / Cello: Scalable adaptive mesh refinement for astrophysics and cosmology," in *Proceedings of the Extreme Scaling Workshop*, Blue Waters and XSEDE. ACM Digital Library, 2012, ⟨http://cello-project.org/xsede12.pdf⟩.

[7] P. P. Laboratory. (2017) Parallel Programming Laboratory. University of Illinois. [Online]. Available: http://charm.cs.uiuc.edu/

[8] L. V. Kale and A. Bhatele, *Parallel Science and Engineering Applications: The Charm++ Approach*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2013.

[9] Theoretical and C. B. Group. (2018) Namd - scalable molecular dynamics. University of Illinois. [Online]. Available: https://www-s.ks.uiuc.edu/Research/namd/

[10] P. P. Laboratory. (2017) Openatom. University of Illinois. [Online]. Available: http://charm.cs.illinois.edu/OpenAtom/

[11] N.-B. Shop. (2017) Changa. University of Washington. [Online]. Available: https://faculty.washington.edu/trq/hpcc/tools/changa.html

[12] P. P. Laboratory. (2017) Episimdemics. University of Illinois. [Online]. Available: http://charm.cs.uiuc.edu/research/episim

[13] E. Meneses, O. Sarood, and L. V. Kal, "Energy profile of rollback-recovery strategies in high performance computing," *Parallel Computing*, vol. 40, no. 9, pp. 536 – 547, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819114000350

[14] E. Meneses, O. Sarood, and L. Kal, "Assessing energy efficiency of fault tolerance protocols for hpc systems," in *Proceedings - Symposium on Computer Architecture and High Performance Computing*, 10 2012, pp. 35–42.

[15] X. Ni, E. Meneses, N. Jain, and L. V. Kal, "Acr: Automatic checkpoint/restart for soft and hard error protection," in *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Nov 2013, pp. 1–12.

[16] H. Menon, B. Acun, S. Garcia De Gonzalo, O. Sarood, and L. Kale, "Thermal aware automated load balancing for hpc applications," in *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 09 2013, pp. 1–8.

[17] C. Burstedde, L. C. Wilcox, and O. Ghattas, "`p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011.

[18] B. D. Smith, G. L. Bryan, S. C. O. Glover, N. J. Goldbaum, M. J. Turk, J. Regan, J. H. Wise, H.-Y. Schive, T. Abel, A. Emerick, B. W. O'Shea, P. Anninos, C. B. Hummels, and S. Khochfar, "GRACKLE: a chemistry and cooling library for astrophysics," *Monthly Notices of the Royal Astronomical Society*, vol. 466, pp. 2217–2234, Apr. 2017.

[19] P. Colella and P. R. Woodward, "The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations," *Journal of Computational Physics*, vol. 54, pp. 174–201, Sep. 1984.

[20] G. Bryan, M. Norman, J. Stone, R. Cen, and J. Ostriker, "A piecewise parabolic method for cosmological hydrodynamics," *Comput. Phys. Commun.*, vol. 89, pp. 149–168, 1995.

[21] S. D. Ustyugov, M. V. Popov, A. G. Kritsuk, and M. L. Norman, "Piecewise parabolic method on a local stencil for magnetized supersonic turbulence simulation," *J. Comput. Phys.*, vol. 228, pp. 7614–7633, November 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1598103.1598701