Extreme-Scale AMR

Carsten Burstedde*, Omar Ghattas*†‡, Michael Gurnis§, Tobin Isaac*,
Georg Stadler*, Tim Warburton‖, Lucas C. Wilcox*
*Institute for Computational Engineering & Sciences, The University of Texas at Austin
†Jackson School of Geosciences, The University of Texas at Austin
‡Department of Mechanical Engineering, The University of Texas at Austin
§Seismological Laboratory, California Institute of Technology
‖Department of Computational and Applied Mathematics, Rice University

***Abstract*—Many problems are characterized by dynamics occurring on a wide range of length and time scales. One approach to overcoming the tyranny of scales is adaptive mesh refinement/coarsening (AMR), which dynamically adapts the mesh to resolve features of interest. However, the benefits of AMR are difficult to achieve in practice, particularly on the petascale computers that are essential for difficult problems. Due to the complex dynamic data structures and frequent load balancing, scaling dynamic AMR to hundreds of thousands of cores has long been considered a challenge. Another difficulty is extending parallel AMR techniques to high-order-accurate, complex-geometry-respecting methods that are favored for many classes of problems. Here we present new parallel algorithms for parallel dynamic AMR on forest-of-octrees geometries with arbitrary-order continuous and discontinuous finite/spectral element discretizations. The implementations of these algorithms exhibit excellent weak and strong scaling to over 224,000 Cray XT5 cores for multiscale geophysics problems.**

I. Introduction

Many grand challenge problems in computational science and engineering are modeled by partial differential equations (PDEs) that exhibit dynamics occurring on a wide range of length and time scales. In many cases, high resolution is required only in localized (possibly dynamically evolving) regions, such as near fronts, discontinuities, material interfaces, reentrant corners, boundary and interior layers, and so on. In such cases, the tyranny of scales can in principle be overcome through *adaptive mesh refinement and coarsening* (AMR) methods, which locally and dynamically coarsen and refine the mesh to resolve spatio-temporal features of interest, driven by error estimates [1]–[4]. AMR methods are capable of orders-of-magnitude reductions in the number of unknowns arising from the discretized PDEs.

While AMR promises to help overcome the challenges inherent in modeling multiscale problems, the benefits are difficult to achieve in practice on petascale computers that are essential for the most difficult of problems. Due to complex dynamic data structures and frequent load balancing, scaling dynamic AMR to hundreds of thousands of processors has long been considered a challenge, e.g. [5]. Another difficulty is extending parallel AMR techniques to high-order-accurate, complex-geometry-conforming finite/spectral element methods that are favored for many classes of elliptic and parabolic problems (in continuous Galerkin form) and hyperbolic problems (in discontinuous Galerkin form).

To overcome these challenges, the ALPS project was initiated in 2007 to create a new framework for parallel, high-order, complex-geometry dynamic AMR based on forests of octrees. At SC08 [6], we presented our initial implementation for single octrees on Cartesian geometries, and gave some preliminary results for spherical shells. Here, we present new algorithms for parallel dynamic high-order-accurate AMR on forest-of-octrees geometries. These algorithms are encapsulated in the `p4est` forest-of-octrees AMR library, and the `mangll` high-order discretization library that provides arbitrary-order continuous and discontinuous finite/spectral element discretizations on `p4est` meshes. Here we present evidence that our AMR algorithms exhibit excellent strong and weak scaling, to the full core counts of the most powerful petascale supercomputers available today, i.e. to over 224,000 cores on *Jaguar*, Oak Ridge National Laboratory's 2.33 petaflops Cray XT5 system. To the best of our knowledge, our AMR algorithms and libraries are the first to support high-order discretizations and non-Cartesian geometries on full petascale systems.

In the following sections, we describe our parallel, high-order, forest-of-octrees AMR algorithms (§II), assess the parallel efficiency and scalability of our algorithms on the full 224K-core *Jaguar* system (§III), and present applications to solid earth geophysics problems (§IV), global mantle convection with nonlinear rheology and global seismic wave propagation. Here we give only brief overviews of `p4est` and `mangll` and their applications to solid earth geophysics problems. A detailed presentation of the algorithms underlying `p4est` is provided in [7], while more information on `mangll` and its applications to seismic wave propagation can be found in [8]. In [9] we report new geodynamic insights obtained from the first global mantle convection simulations that

SC10 November 2010, New Orleans, Louisiana, USA 978-1- 4244-7558-2/10/$26.00

resolve kilometer-scale dynamics at plate boundaries; through the use of `p4est`, these simulations require three orders of magnitude fewer unknowns, reducing what would be an exascale problem if it were solved on a uniform mesh, to one that can be solved routinely on a petascale system with AMR.

II. Scalable Algorithms for Parallel AMR on Forests of Octrees

While AMR is critical for making tractable simulations of strongly-localized multiscale problems, designing effective AMR algorithms on highly parallel systems presents formidable challenges. Parallel scalability, dynamic load balancing, complex domain geometries, and high-order discretizations often lead to conflicting goals. These goals have been approached in several different ways in the past [5]. Here provide a brief review and then present our new algorithms for forest-of-octrees AMR.

A. Approaches to Parallel AMR

Block-structured AMR, e.g. [10]–[17], utilizes unions of regular grids, which permit reuse of regular grid sequential codes. Load-balancing can be achieved by distributing the regular grid patches to cores according to their geometric location. However, block-structured AMR increases numerical complexity due to possibly overlapping patches of differing resolutions and can be difficult to extend to high-order discretizations on general geometries.

Unstructured AMR, on the other hand, provides greater geometric flexibility and a straightforward path to high-order accuracy at the cost of explicitly storing all neighborhood relations between mesh elements; see e.g. [18]–[20]. The challenge in these mainly tetrahedral-based methods is to maintain element quality as the mesh is coarsened and refined, which can generate significant communication. Hierarchical hybrid grids [21] split the domain into unstructured macro-elements which are then uniformly refined; while being able to handle large numbers of unknowns, adaptivity is confined to the coarse scale.

Tree-based methods make use of recursive encoding schemes while enabling hierarchical refinement (see e.g. [22]–[27]). In this sense they strike a balance between structure and flexibility, enabling arbitrary-order accuracy (through high-order hexahedral finite/spectral elements along with enforcement of hanging node constraints) and implicitly providing good element quality and a recursive space-filling curve ordering that enables lightweight load balancing. Parallel octree-based algorithms have recently demonstrated high scalability and low computational overhead [6], [28]–[33]. However, a single octree can encode only a topologically cube-shaped domain, and thus is unsuitable for representing general geometries.

The essential question with octree-based AMR is thus how to represent geometries that cannot be mapped to the unit cube. The approach we take to support general geometries is to join multiple adaptive octrees to create a "forest". The forest-of-octrees approach has been implemented previously using fully replicated mesh storage, limiting scalability to several hundred processor cores [34], [35]. We have developed new algorithms that strictly adhere to fully distributed storage and thus extend the scalability of single-octree algorithms to forests of octrees [7]. These algorithms have been implemented in the `p4est` software library; their functionality is described below.

B. Scalable Forest-of-Octrees Parallel AMR

The term *octree* denotes a recursive tree structure in which each node is either a leaf or has eight children. Octrees can be associated with (logically) cubic domains where nodes are called *octants*, and the root node corresponds to the whole domain, which is recursively subdivided according to the tree structure. We use the term *forest* to describe a collection of such logical cubes that are connected in a conforming manner through faces, edges, or corners; see Figure 1 for 2D and 3D examples.

A forest of octrees can be understood as a two-level decomposition of a geometric domain. At the *macro*-level, we decompose the domain into K conforming subdomains, each mapped from a reference cube by a smooth function. Specifically, we allow any manifold that can be covered by smooth images of a finite collection of K reference octrees subject to the restriction that any macro-face or macro-edge shared between two octrees is shared in full or not at all. This approach is more general than domains mapped by a single octree, since any macro-edge can be shared by any number of octrees (not just four), and any macro-corner can be shared by any number of octrees (not just eight). We can thus represent such volumes as tori, spheres, and spherical shells, and in general arbitrary volumes that can be decomposed into hexahedral subdomains. The macro-structure of the forest is static and shared among all cores (which is not problematic, since the number of octrees is generally small and independent of the problem size).

The *micro*-level of the forest is characterized by the division of each octree into octants and the partition of these octants among cores, strictly distributing octant storage in parallel. The micro-structure contains non-conforming (hanging) faces and edges and changes dynamically due to refinement, coarsening, and repartitioning. We permit unconstrained size relations between neighboring octants, and additionally provide the 2:1 balance algorithm to guarantee at most 2:1 size relations when required or preferred by the discretization

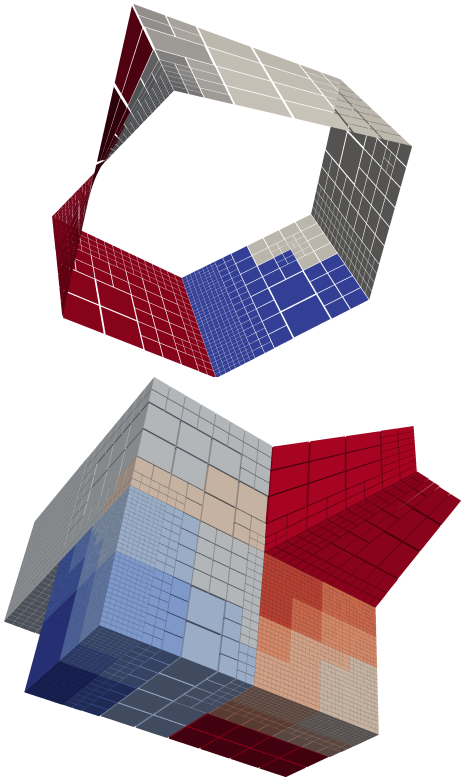

Fig. 1. Examples of forest-of-octrees domains, in which color represents the core number. Top: 2D forest composed of five adaptive quadtrees that represent the periodic Möbius strip. Bottom: 3D forest composed of six adaptive octrees whose orientations are rotated with respect to one another, with five octrees connecting through the horizontal center axis.

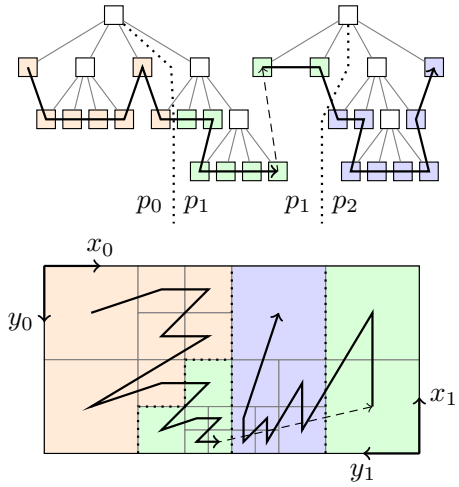

Fig. 2. One-to-one correspondence between a forest of octrees (top) and a geometric domain partitioned into elements (bottom). The leaves of the octrees correspond bijectively to elements that cover the domain with neither holes nor overlaps. A left-to-right traversal of the leaves through all octrees creates a space-filling z-curve that imposes a total ordering of all octants in the domain. For each octree the space-filling curve follows the orientation of its coordinate axes. In this example the forest is partitioned among three cores, p_0, p_1, and p_2. This partition divides the space-filling curve and thus the geometric domain into three segments of equal (± 1) element count.

scheme. Neighborhood size relations are respected both for octants within the same octree and for octants that belong to different octrees and connect through an octree macro-face, -edge, or -corner.

A key concept enabling large-scale parallelism for octree-based AMR is the *space-filling curve*, identified by the traversal of an octree across its leaves. By the equivalence of tree nodes and octants this natural ordering of leaves corresponds to a z-shaped curve in the geometric domain [36]. We extend this concept to a forest of octrees by connecting the space-filling curve between octrees, thus generating a total ordering of all octants in the domain. This total ordering can then be used for fast binary search, finding any of N_p local octants in $\mathcal{O}(\log N_p)$ steps. A parallel partition is created by dividing the curve into P segments, where P is the number of cores. To encode the parallel partition, we share the number of octants on each core and the octree number and coordinates of the first octant on each core; this amounts to globally shared meta-data of just 32 bytes per core. A 2:1 balanced forest and its space-filling curve and load-balanced partition are shown in Figure 2.

C. Core Forest-of-Octrees Algorithms

Beyond the 2:1 balance and space-filling curve-based partitioning algorithms described above, we have designed and implemented a suite of general purpose algorithms required to enable AMR for PDEs on forest-of-octrees meshes. These algorithms are made available in `p4est` as the following functions:

`New`. Create an equi-partitioned, uniformly refined forest. The initial refinement level can be as low as zero, in this case creating only root octants with potentially many empty cores.

`Refine`. Adaptively subdivide octants based on a refinement marker or callback function, optionally once or recursively.

`Coarsen`. Replace families of eight child octants by their common parent octant where specified, once or recursively.

`Partition`. Redistribute the octants in parallel, according to a given target number of octants (optionally weighted) for each core.

`Balance`. Ensure at most 2:1 size relations between neighboring octants by local refinement where necessary.

`Ghost`. Collect one layer of non-local octants touching the parallel partition boundary from the outside.

`Nodes`. Create a globally unique numbering of all unknowns used in high-order non-conforming nodal polynomial discretizations.

The functions `New`, `Refine`, and `Coarsen` require no communication. `Partition` uses the space-filling curve to determine the prospective owners of all local octants, which requires one call to `MPI_Allgather` with one long integer per core. The octants are then transferred point-to-point. `Balance` and `Ghost` require communication between cores whose octants are geometrically close to each other; their communication volumes scale roughly with the number of octants on the partition boundaries. `Ghost` requires a `Balance`'d mesh, and `Nodes` requires the `Ghost` layer.

D. Forest-of-Octrees Approach to General Geometries

Mapping general geometries with a collection of transformed cubes makes it impossible to identify a consistent orientation that encompasses all cubes. To achieve maximum flexibility we endow each cube with a right-handed coordinate system that can be positioned arbitrarily in space. This approach permits rotations of any two cubes with respect to one another. Any two faces can meet in four different ways, and any two edges in two. Any corner of a cube can connect to any other corner. The number of simultaneously connecting corners and edges is arbitrary. Periodicity is naturally included in this scheme, either between different cubes or connecting different faces of the same cube.

To pass information between neighboring octrees we introduce (a) exterior octants that exist in the coordinate system of an octree but outside of its root domain, and (b) transformations of interior and exterior octants between octrees that connect through a macro-face, -edge, or -corner. These transformations account for all possible relative rotations between two neighboring octrees. An example situation is displayed in Figure 3.

An important feature of `p4est` is that connectivity and neighborhood relations are computed discretely (i.e. integer-based). No floating-point arithmetic is used, avoiding topological errors due to roundoff. While smooth geometries are represented by subjecting the octrees to diffeomorphic transformations, `p4est` uses these transformations only for visualization, and to pass the geometry to an external application (such as the PDE solver).

Combining the concepts introduced above, namely the space-filling curve inspired by the multi-octree structure and the specification of inter-octree geometric relations and transformations, we obtain a scalable general-geometry AMR framework that requires only minimal globally shared meta-data, ensures good data locality and cache efficiency, provides lightweight search facilities for octants and owner processes, and allows for fast dynamic adaptation and load-balancing [7]. These features lead to excellent scalability and parallel efficiency, as will be demonstrated in §III.

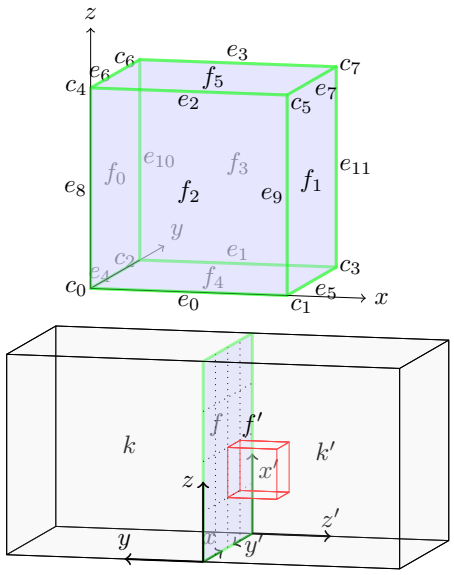

Fig. 3. Top: z-order numbering convention for an octree. The symbols indicate corners c_i, edges e_i, and faces f_i. Octrees have right-handed coordinate systems that can be arbitrarily rotated in space. Octants inherit the coordinate system from the octree they belong to. Bottom: face connection between two octrees with non-aligned coordinate systems. The face numbers seen from the left and right octrees k and k' are 2 and 4, respectively. The red octant of size $1/4$ is exterior to k and interior to k'. Its coordinates are $(2, -1, 1)$ with respect to k and $(1, 1, 0)$ with respect to k'.

E. High-order PDE Discretization on Adaptive Forest-of-Octree Meshes

At each adaptivity step, `p4est` creates a forest-of-octrees-based mesh of hexahedral elements. The software library `mangll` provides the functions needed to discretize PDEs using this mesh structure created by `p4est`. Many of the core functions of `mangll` (e.g., creation of high-order element shape functions and quadrature rules) do not require communication and are straightforward to implement. However, the association of local unknowns is more complicated, especially for high-order elements and at 2:1 refinement boundaries. In this section, we discuss parallel algorithms to achieve this association of corresponding degrees of freedom.

For most applications, one layer of non-local elements, sorted in the total order defined by the space-filling curve, provides sufficient neighborhood information to associate and number the unknowns. We produce such a "ghost layer" in `Ghost`. (Multiple layers, for example as needed by a semi-Lagrangian method, can be enabled by a minor extension of `Ghost`.) In discontinuous Galerkin (dG) methods, such as the one used for seismic wave propagation in §IV-B, all unknowns are associated with an element, resulting in discontinuous approximations. Computing fluxes across faces requires access to unknowns on neighboring elements. We accomplish this by

fast binary searches in the local octant storage, or in the ghost layer when a parallel boundary is encountered. The rotation of coordinate systems between octrees needs to be taken into account when aligning unknowns across inter-octree faces. For 2:1 non-conforming faces, the unknowns on the larger face are interpolated to align with the unknowns on the four connecting smaller faces.

High-order continuous Galerkin (cG) discretizations employ global basis functions that are continuous across element boundaries. The unknowns in this formulation can be associated with element volumes, faces, edges or corners. A 2:1 balanced mesh contains non-conforming, or *hanging*, faces and edges. A face is hanging if it is one of four half-size faces that neighbor a full-size face; an edge is hanging if it is one half of a full-size neighboring edge. For a cG discretization, nodal values on half-size faces or edges are generally not associated with independent unknowns; instead we constrain them to interpolate neighboring unknowns associated with full-size faces or edges. In our algorithm `Nodes` we identify independent unknowns by binary searching for neighbor elements in local octants and the ghost layer where necessary, and then construct a globally unique numbering. The independent nodes on octree boundaries need to be canonicalized first, i.e., they are assigned to the lowest numbered participating octree and transformed into its coordinate system (cf. Figure 3). We use this node numbering for our trilinear finite element mantle convection described in §IV-A.

We have encapsulated all algorithms for cG and dG discretizations in the `mangll` software library, which interfaces to `p4est` through `Ghost` and `Nodes`. This includes routines for construction of polynomial spaces, numerical integration, and high-order interpolation on hanging faces and edges, and parallel scatter-gather algorithms for unknowns that are shared between cores. The strict separation between adaptation and partitioning of the forest of octrees itself, and the numerical operations for PDE discretization, has proven to be one of the keys to the versatility and scalability of our parallel AMR framework.

III. Assessment of Scalability and Parallel Efficiency of AMR framework

First, we report two sets of results illustrating the performance and scalability of the parallel AMR algorithms presented in §II on *Jaguar*, the ORNL Cray XT5 system. The first set stresses the forest-of-octrees operations themselves using an idealized mesh. The second set examines the performance of the algorithms for high-order dG solution of the advection equation on a dynamically refined and coarsened forest-of-octrees mesh. Because the PDE is linear, scalar, and explicitly-integrated, this provides an extreme test of parallel AMR performance, since there are few PDE solver flops over which to amortize the AMR operations.

The timings and performance evaluations for AMR solution of PDEs presented in this section and the next reflect *complete end-to-end simulations*, including mesh initialization, coarsening, refinement, 2:1 balancing, partitioning, solution transfer between meshes, and PDE time integration.

A. Weak Scalability of Forest-of-Octrees AMR

First, we study the weak scalability of the `p4est` algorithms on a six-octree forest configuration, as seen in Figure 1, designed to activate many types of inter-octree connectivity pairings. Using a fractal-type refinement derived from a uniform initial configuration (as described in the caption of Figure 4), an increase in the refinement level by one yields eight times as many octants. Thus, we multiply the core count by eight for each increment in refinement level. This results in a problem size of approximately 2.3 million octants per core; the largest problem contains over 500 billion octants on 220,320 cores. We display the measured runtimes of the core `p4est` algorithms in Figure 4. As can be seen in the top bar chart, the runtimes of `New`, `Refine`, and `Partition` are negligible (`Coarsen` is not used here but is as fast as `Refine`), and `Balance` and `Nodes` consume over 90% of the total runtime. The bottom bar chart of Figure 4 displays the absolute runtimes of the two most expensive algorithms, `Balance` and `Nodes`, normalized by one million octants per core. The runtimes rise mildly from 6 seconds for 12 cores to between 8 and 9 seconds for 220,320 cores. This yields a parallel efficiency of 65% for `Balance` and 72% for `Nodes` for an 18,360-fold increase in core count and mesh size. These are excellent parallel efficiencies for dynamic mesh adaptivity, particularly considering that there are no PDE solver operations to offset the AMR costs.

B. Scalability of High-Order AMR for Advection Equation

Next, we study the performance of `p4est` and `mangll` applied to parallel dynamic mesh adaptation for the time-dependent advection equation,

$$\frac{\partial C}{\partial t} + \boldsymbol{u} \cdot \nabla C = 0, \quad \boldsymbol{x} \in \Omega,\ t \in (0, T), \qquad (1)$$

with given initial condition $C(\boldsymbol{x}, 0) = C_0(\boldsymbol{x})$ and appropriate boundary conditions. The PDE (1) models a time-dependent unknown $C(\boldsymbol{x}, t)$ (e.g., a temperature field or chemical concentration) that is transported by a given flow velocity field $\boldsymbol{u}$.

We discretize (1) using an upwind nodal dG method [37] in space and an explicit five-stage fourth-order Runge-Kutta method [38] in time. The unknowns are

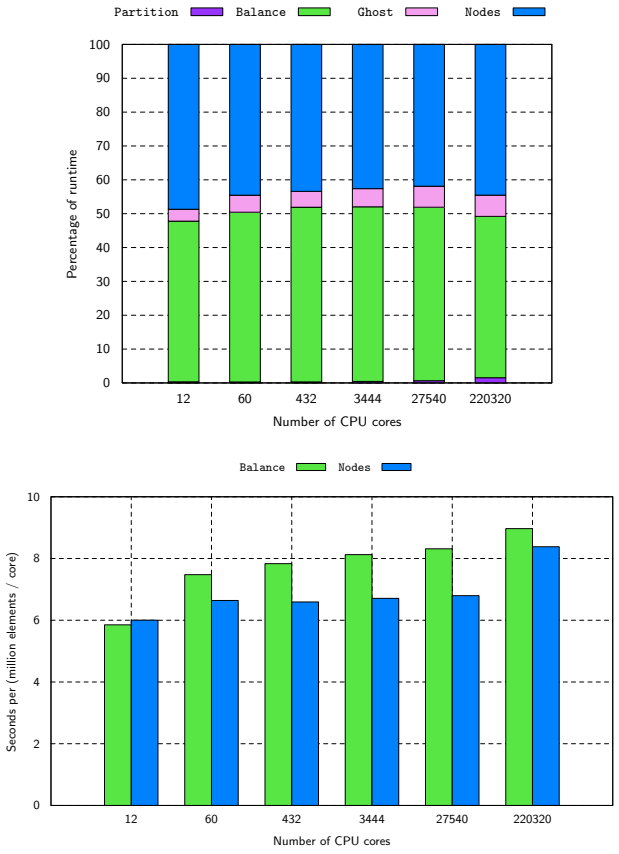

Fig. 4. Weak scaling results for a six-octree forest on up to 220,320 cores. We define a fractal-type mesh by recursively subdividing octants with child identifiers 0, 3, 5 and 6 while not exceeding four levels of size difference in the forest. To scale from 12 to 220,320 cores the maximum refinement level is incremented by one while the number of cores is multiplied by eight. Top: Percentages of runtime for each of the core `p4est` algorithms. Runtime is dominated by `Balance` and `Nodes` while `Partition` and `Ghost` together take up less than 10% (`New` and `Refine` are negligible and not shown). Bottom: Performance assessed by normalizing the time spent in the `Balance` and `Nodes` algorithms by the number of octants per core, which is held constant at approximately 2.3 million (ideal scaling would result in bars of constant height.) The largest mesh created contains over 5.13×10^{11} octants.

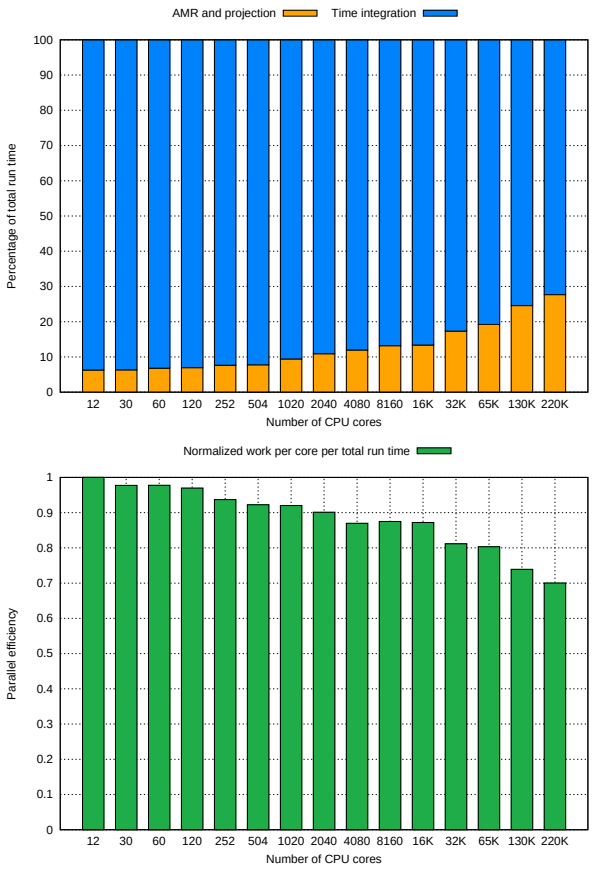

Fig. 5. Weak scaling for a dynamically adapted dG solution of the advection equation (1) from 12 up to 220,320 cores. The mesh is adapted and repartitioned, maintaining 3200 tricubic elements per core. The maximum number of elements is 7.0×10^8 on 220,320 cores, yielding a problem with 4.5×10^{10} unknowns. The top bar chart shows the overhead imposed by all AMR operations, which begins at 7% for 12 cores and grows to 27% for 220,320 cores. The bottom bar chart demonstrates an end-to-end parallel efficiency of 70% for an increase in problem size and number of cores by a factor of 18,360.

associated with tensor product Legendre-Gauss-Lobatto (LGL) points, as in the spectral element method [39]. All integrations are performed using LGL quadrature, which reduces the dG mass matrix to diagonal form.

To examine the scalability of `p4est` and `mangll`, we solve (1) on a spherical shell domain using mesh adaptivity to dynamically resolve four advecting spherical fronts. The spherical shell domain is split into six caps as used in a cubed-sphere decomposition. Each cap is further divided into four octrees, resulting in 24 adaptive octrees overall. The element order in this example is 3, and the mesh is coarsened/refined and repartitioned every 32 times steps. The weak scaling results presented in Figure 5 reveal 70% end-to-end parallel efficiency for weak scaling from 12 cores (with 2.5 million unknowns) to 220,320 cores (with 45 billion unknowns). This problem is a severe test of the AMR framework; not only are there few flops to hide the parallel AMR operations behind (as mentioned above), but the aggressive adaptivity (about 40% of the elements are coarsened and about 5% are refined in each adaption step of the largest run, keeping the overall number of elements constant) results in exchange of over 99% of the elements among cores during repartitioning at each adaptivity step.

IV. AMR Simulations in Solid Earth Geophysics

In this section we present applications of `p4est` and `mangll` to two problems in solid earth geophysics, one in global mantle convection and global seismic wave

propagation.

A. Global Mantle Convection

Our first target problem, for which the AMR capabilities of `p4est` have proven to be essential, is the simulation of convective flow in earth's mantle. Mantle convection is a fundamental physical process within earth's interior and is the principal control on the thermal and geological evolution of the planet. As the most efficient heat transfer mechanism, it controls the removal of heat from earth's mantle and core, and hence its cooling history. Plate tectonics, volcanism, and mountain building are surface manifestations of this process.

Our parallel AMR algorithms are enabling the first global mantle flow simulations that are able to resolve the nonlinear rheology at plate boundaries, which requires local mesh resolution of about 1 km. Using a mesh with 1 km uniform resolution to discretize the mantle would lead to $\mathcal{O}(10^{12})$ unknowns, well beyond the reach of contemporary petascale supercomputers. Using `p4est`, we have been able to reduce the number of unknowns by over three orders of magnitude. This has allowed us to perform ultra-high resolution global mantle convection simulations with realistic rheological parameters on $\mathcal{O}(10^4)$ cores. These have been instrumental for providing us with new insights into the dynamics of plate boundaries [9] (see Figure 6).

The dynamics of mantle convection are governed by the conservation of mass, momentum, and energy,

$$\nabla \cdot \boldsymbol{v} = 0, \tag{2a}$$

$$-\nabla \cdot \left[\eta \left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top\right) - \boldsymbol{I} p\right] = \rho \boldsymbol{g}, \tag{2b}$$

$$\rho c_p \left(\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T\right) - \nabla \cdot (k \nabla T) = \rho H, \tag{2c}$$

where $\boldsymbol{v}$, p, and T are the unknown velocity, pressure, and temperature; $\eta = \eta(T, \boldsymbol{v})$, ρ, $\boldsymbol{g}$, c_p, k, and H are the temperature- and strain-rate-dependent viscosity, density, gravitational acceleration, specific heat, thermal conductivity, and internal heat generation rate. Under the Boussinesq approximation for a mantle with uniform composition and the assumption that the mantle deforms as a viscous medium, we can replace the density ρ on the right side of (2b) by $\rho_0 \left[1 - \alpha(T - T_0)\right]$, with reference density and temperature ρ_0 and T_0. Various constitutive laws are used for the mantle, but in general the viscosity depends nonlinearly on both temperature and the second invariant of the deviatoric strain rate tensor, i.e.,

$$\eta(\boldsymbol{v}, T) = c_1 \exp\left(\frac{c_2}{T}\right) \left[\left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top\right) : \left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top\right)\right]^{c_3},$$

with parameters c_i. Additionally, the rheology often incorporates yielding under high strain rates.

Our parallel AMR mantle convection code *Rhea*, which interfaces to `p4est`, solves (2a)–(2c) with appropriate boundary conditions [6]. *Rhea* discretizes the velocity, pressure, and temperature fields with trilinear hexahedral finite elements on a forest of 24 octrees, which are mapped to earth's mantle using a modified cubed sphere transformation (see Figure 6 and the description of the shell domain in §III-B). Mantle thermal transport is strongly advection-dominated; we thus employ the streamline-upwind Petrov-Galerkin (SUPG) scheme to stabilize the discretization of the energy equation (2c). The equal-order discretization of the Stokes equations (2a)–(2b) is stabilized with pressure projection [40]. Explicit integration of the energy equation decouples the temperature update from the nonlinear Stokes solve; the latter is carried out at each time step using the updated temperature via a lagged-viscosity (Picard) iteration. Each Picard iteration requires an (implicit) variable-viscosity Stokes solve, which is carried out with a MINRES Krylov solver, preconditioned in the (1,1) block by one V-cycle of the algebraic multigrid solver ML [41], and in the (2,2) block by a mass matrix (with inverse viscosity) approximation of the pressure Schur complement. Details are given in [42], where optimal algorithmic scalability of the preconditioner is demonstrated.

As mentioned above, we are using *Rhea* to carry out the first global mantle flow simulations that resolve plate boundaries [9]. These global models are driven entirely by temperature variations, which are based on the thermal age of the surface, seismicity-defined slabs, and seismic tomography in the lower mantle. The use of this present-day temperature model replaces solution of (2c). Thus, we solve (2a)–(2b) with realistic Rayleigh number and a nonlinear rheology that incorporates diffusion and dislocation creep, as well as yielding at high strain rates [9], [43]. Plate boundaries are modeled by narrow (about 10km wide) zones, for which the viscosity is lowered by 5 orders of magnitude (see the red lines on earth's surface in Figure 6).

Adaptivity is carried out as follows. The initial temperature field is interpolated on a finite element mesh with about 22 million elements. First, this initial mesh is coarsened and refined based on temperature variations. Then, the mesh is refined down to 1–2 km element size in the narrow low viscosity zones that model the plate boundaries, which results in a mesh with about 90 million elements. Iterative solution of the nonlinear Stokes equation begins after these static AMR steps. To resolve localized flow features as they evolve from the nonlinear yielding rheology, we interleave the nonlinear iterations with dynamic local mesh refinement steps. In particular, we further refine the mesh every 2–8 nonlinear iterations based on error indicators that involve strain rates and (dynamically evolving) viscosity gradients. Typically, 5–7 of these dynamic solution-adaptive (*a posteriori*) refinements are performed, resulting in meshes with about

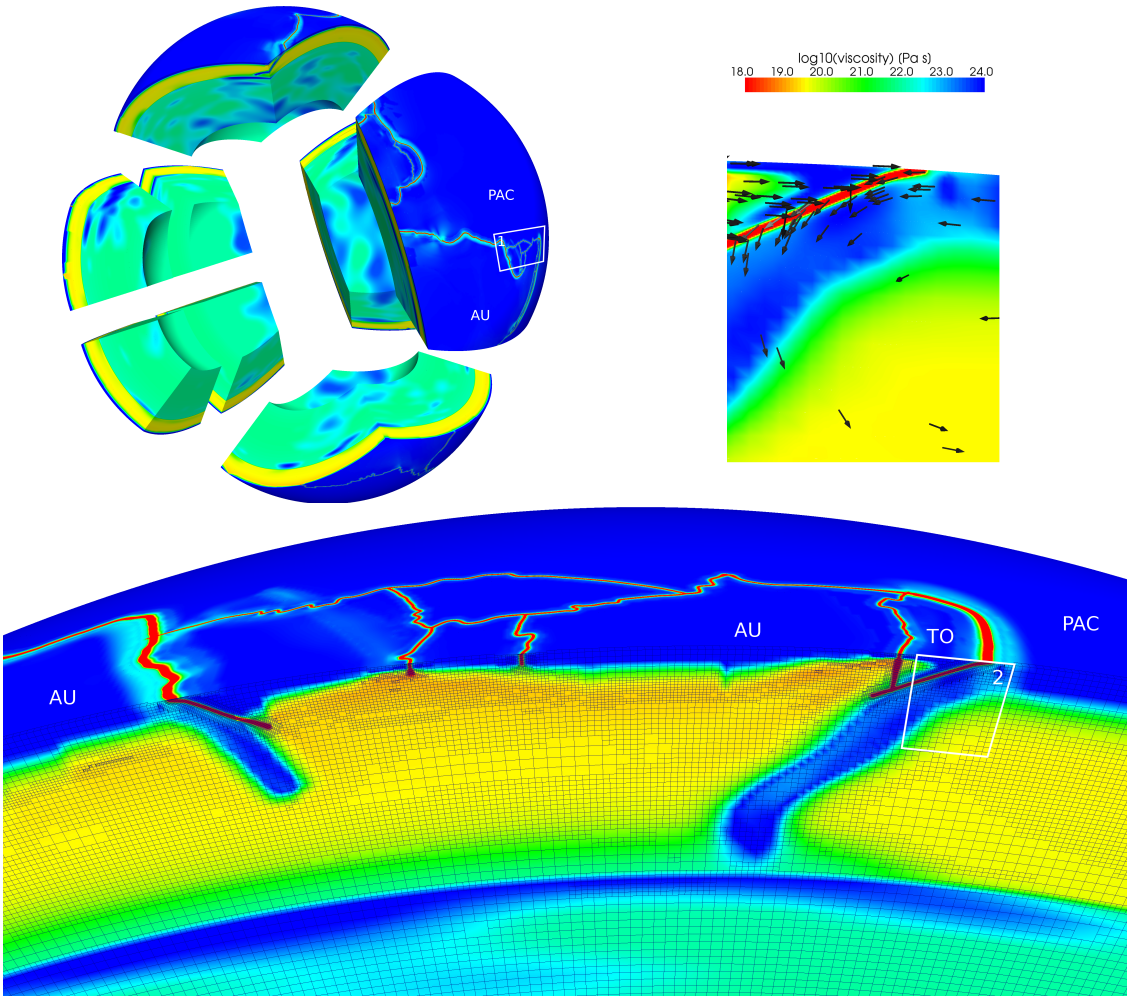

Fig. 6. Adaptive, ultra-high resolution mantle flow simulation. *Top, left*: Decomposition of earth's mantle into 24 octrees using the `p4est` library. The viscosity field is shown; very narrow low-viscosity zones (red lines on the surface) indicate plate boundaries. Box 1 between the Pacific (PAC) and the Australian (AU) plates indicates the region of the cross-section shown below. *Bottom*: Enlarged cross-section showing the refinement that occurs both around plate boundaries and dynamically in response to the nonlinear viscosity and plastic failure in the region around Fiji and Tonga (TO) in the SW Pacific. AMR, which is essential to resolve the plate boundaries, dynamically creates a mesh that contains elements at 8 refinement levels, with finest resolution of about 1 km. A zoom into Box 2, where the Pacific plate subducts underneath the Australian plate, is shown in the top right figure. *Top, right:* Viscosity and flow vectors for a zoom into the hinge zone of the Pacific slab (indicated by Box 2). The narrow low viscosity zone (red) allows shearing of the flow and, thus, plate subduction to occur. The opposite directions of the plates (the blue regions separated by the weak zone; arrows point in opposite directions) shows that our simulation predicts trench rollback, as is known to occur in this region. Resolving these local phenomena is critical for global mantle flow models to fit observations; this is the first time that a dynamic mantle flow model predicts these phenomena [9].

150–300 million finite elements. These meshes typically contain 8 different refinement levels and about a billion (velocity and pressure) unknowns. As the mesh adapts and is repartitioned, all solution fields are interpolated between meshes and redistributed according to the mesh partition.

Figure 7 presents runtime percentages for the solution of a typical global mantle convection problem using *Rhea* as outlined above. The percentages are broken down into AMR operations, solver time (which includes nonlinear residual formation, Picard operator construction, and Krylov iteration matrix-vector products and inner products), and AMG V-cycle time. As can be seen from the table, the time spent in AMR components is completely overwhelmed by the solver (solve + V-cycle) time; in this case, the AMR components together require no more than 0.12% of runtime. Thus, parallel AMR has transformed a problem that would have required an exascale computer to obtain 1 km uniform resolution, to one that can fit on a petascale cluster by employing

core count	13.8K	27.6K	55.1K
solve	33.6%	21.7%	16.3%
V-cycle	66.2%	78.0%	83.4%
AMR	0.07%	0.10%	0.12%

Fig. 7. Runtime percentages for adaptive solution of global mantle flow problem described in §IV-A on *Jaguar*. The first row is the percentage of time for all solver operations (including residual formation, Picard operator construction, and Krylov iterations), excluding the AMG V-cycle time, which is given in the second row. (The AMG setup cost is negligible since it is performed just once per several hundred MINRES iterations.) The third row gives the percentage of time for 5 data-adaptive and 5 solution-adaptive refinements (including the `p4est` operations `Nodes`, `Partition`, `Balance`, and `Refine`/`Coarsen`) error indicator computation and marking of elements, and interpolation of solution fields between meshes. The overall time spent in AMR averages a tenth of a percent while scaling over a range of 13.8K to 55.1K cores and reaching a final adapted mesh size of 150 million elements.

# proc cores	meshing time (s)	wave prop per step (s)	par eff wave	Tflops
32,640	6.32	12.76	1.00	25.6
65,280	6.78	6.30	1.01	52.2
130,560	17.76	3.12	1.02	105.5
223,752	47.64	1.89	0.99	175.6

Fig. 9. Strong scaling of global seismic wave propagation for a problem with a source frequency of 0.28 Hz) on up to 224K cores of *Jaguar*. Degree $N = 6$ elements with at least 10 points per wavelength. Mesh consists of 170 million elements, corresponding to 53 billion unknowns. *Meshing time* refers to time taken for parallel generation of the mesh (adapted to local wave speed) prior to wave propagation solution. *Wave prop per step* is the runtime per time step of the wave propagation solver. *Par eff wave* denotes parallel efficiency of the wave propagation solver (time for mesh generation is in the noise), defined by ratio of measured to ideal speedup on an increasing number of cores for fixed problem size. *Tflops* is double precision teraflops/s based on performance counters from the *PAPI* library [45].

a highly-heterogeneous, dynamically-adapted mesh. Yet, the overhead imposed by parallel AMR is completely negligible.

B. Global Seismic Wave Propagation

Our second target problem is the simulation of seismic waves propagating through coupled acoustic-elastic heterogeneous media, both in seismically-active regions such as Southern California, as well as at global scales. Adaptivity is important in two ways. First, parallel adaptive meshing is used to tailor the mesh size to the minimum local seismic wavelength, given a description of the elastic properties (and hence seismic wave speeds) within the region of interest. This avoids over-refinement and over-restrictive time step sizes, and can result in orders of magnitude reduction in number of unknowns for highly-heterogeneous regions, such as sedimentary basins [28]. This adaptivity must be done online to avoid the transfer of massive meshes. Second, we can optionally coarsen and refine the mesh during the simulation to track propagating waves (Figure 8) when the sparsity of wavefronts can be profitably exploited or to resolve the dynamics of earthquake fault rupture.

In mixed velocity–strain form, the governing equations for elastic wave propagation through isotropic media are

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} = \nabla \cdot (2\mu \boldsymbol{E} + \lambda \operatorname{tr}(\boldsymbol{E}) \boldsymbol{I}) + \boldsymbol{f}, \tag{3a}$$

$$\frac{\partial \boldsymbol{E}}{\partial t} = \frac{1}{2} \left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^{\top} \right), \tag{3b}$$

where $\boldsymbol{v}$ and $\boldsymbol{E}$ are the unknown velocity vector and strain tensor of the medium, and ρ, $\boldsymbol{f}$, and $\boldsymbol{I}$ are the mass density, body force per unit volume, and identity tensor, respectively.

We have designed an adaptive elastic wave propagation code, *dGea*, based on the `mangll` and `p4est` libraries. Velocity and strain fields are discretized using the discontinuous Galerkin method with a Godunov flux in space [8], [37] and a five-stage fourth-order explicit Runge-Kutta method in time [38]. The first-order velocity-strain formulation allows us to simulate waves propagating in acoustic, elastic and coupled acoustic-elastic media within the same framework. We use the same high-order nodal dG discretization as for the advection equation in §III-B.

We now study the scalability and the performance of *dGea* on adaptively refined meshes for seismic wave propagation in an earth model. Figure 9 presents strong scaling results of *dGea* on up to 224K cores of *Jaguar* for a problem with 53 billion unknowns. Timings are broken down into the runtime needed to statically generate and adapt a mesh to the earth structure and seismic velocity (defined by the PREM model), and the average runtime needed for a single time step for solving the wave propagation equations. Resolving high frequencies characteristic of broadband observations results in a need for $\mathcal{O}(10^4$–$10^5)$ time steps. Thus, the 20–40 seconds of runtime required to generate an adaptive mesh is completely overwhelmed by the many hours of runtime required to solve the wave propagation equations. As shown in Figure 9, strong scaling from 32K to 224K cores exhibits 99% parallel efficiency. Due to the efficiency of the algorithms underlying `p4est`, the overall parallel efficiency is unchanged from 99% when considering end-to-end simulations (i.e., adaptive mesh generation plus wave propagation solution).

Finally, we have implemented a hybrid CPU-GPU version of *dGea*, in which the wave propagation solver runs on the GPU and `p4est` AMR operations execute on the CPU. The single precision GPU version of *dGea* exhibits a $\sim$50$\times$ speedup on an NVIDIA FX 5800 GPU relative to a single Intel Xeon 5150 core running at 2.666 GHz. These speedups are consistent with previ-

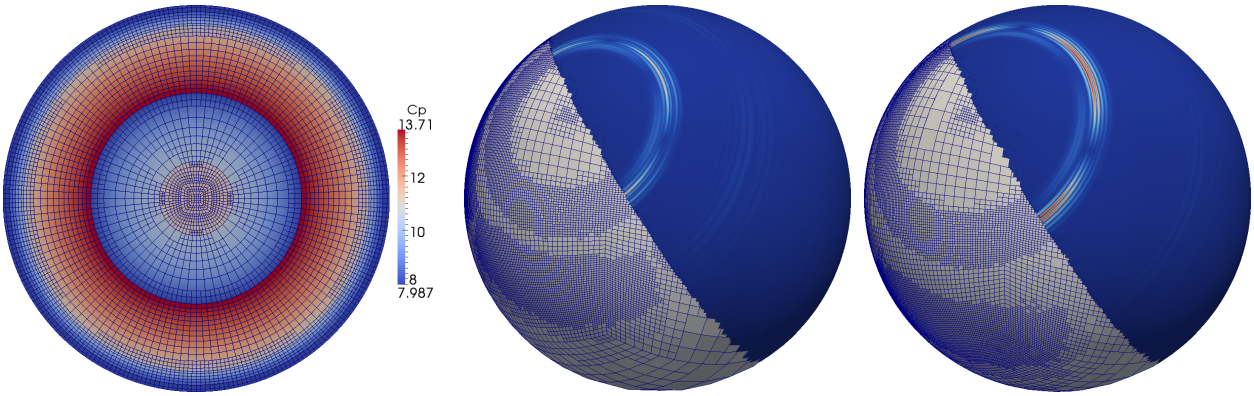

Fig. 8. Left: Section through mesh that has been adapted lcoally according to the size of spatially-variable wavelengths; low frequency used for illustrative purposes. The color scale corresponds to the primary wave speed in km/s. The mesh aligns with discontinuities in wave speed present in the PREM (Preliminary Reference Earth Model) model used [44]. Middle and right: Two snapshots of waves propagating from an earthquake source; the mesh is adapted dynamically to track propagating wavefronts.

ous experience with an efficient GPU implementation of a dG electromagnetics wave propagation code [46]. Figure 10 assesses weak scaling on the TACC Longhorn GPU cluster, composed of FX 5800 GPUs and Intel Nehalem quadcore processors. For these scaling results, a static mesh is adapted to local seismic wavelengths of the PREM model in parallel on the CPUs, the mesh is transferred to the GPUs, and the seismic wave propagation equations are solved in parallel on the GPUs. The wave propagation solver requires communication among GPUs at each time step, which involves transfer of shared data to CPUs and communication via MPI. The table indicates 99.7% parallel efficiency in scaling over a 32-fold increase in number of GPUs and problem size, to 256 GPUs and 3.2 billion unknowns. While the GPU implementation delivers a substantial $\sim$50$\times$ speedup for the wave propagation solver, the CPU-only implementation of AMR does not present any obstacles to excellent scalability, thanks to the efficiency and negligible overhead imposed by our parallel AMR algorithms.

GPUs	elem	mesh (s)	transf (s)	wave prop	par eff wave	Tflops
8	224048	9.40	13.0	29.95	1.000	0.63
64	1778776	9.37	21.3	29.88	1.000	5.07
256	6302960	10.6	19.1	30.03	0.997	20.3

Fig. 10. Weak scaling of GPU version of global seismic wave propagation code on up to 256 GPUs of the TACC Longhorn cluster for up to 6.3 million elements (3.2 billion unknowns). Degree $N = 7$ elements. *Mesh* refers to time to generate adaptive mesh in parallel on CPUs; *transf* indicates the time to transfer the mesh and other initial data from CPU to GPU memory; *wave prop* is the runtime in μsec per time step per average number of elements per GPU (we normalize by number of elements per GPU since AMR results in about 12% variability in granularity with increasing GPU count); *par eff* represents parallel efficiency measured by the degradation in normalized runtime per GPU; and *Tflops* is single precision teraflops/s based on hand-counted operations. Wallclock time averages under a second per time step, which means that meshing time on the CPU and mesh transfer time to the GPU are completely negligible for realistic simulations. Longhorn is composed of 512 NVIDIA FX 5800 GPUs each with 4GB graphics memory and 512 Intel Nehalem quad core processors connected by QDR InfiniBand interconnect.

V. Conclusions

The goal of this paper has been to present and assess the performance of new parallel AMR algorithms for solution of PDEs that (1) respect complex geometries, (2) support high-order accuracy, and (3) scale to the largest parallel systems available. Results on up to 224K cores of the *Jaguar* Cray XT5 demonstrate excellent strong and weak scalability of AMR for challenging test problems involving fractal mesh adaptation and scalar advection. Moreover, applications of our parallel AMR framework to multiscale problems in solid earth geophysics exhibit at least three orders of magnitude reduction in problem size, making tractable global mantle convection simulations that would otherwise require exascale computing in the absence of AMR. In these cases, the overhead imposed by AMR is completely negligible, underscoring the scalability and parallel efficiency of our AMR algorithms and libraries.

Acknowledgments

This work was partially supported by NSF (OCI-0749334, OCI-0748898, CCF-0427985, DMS-0724746, OPP-0941678), AFOSR (FA9550-09-1-0608), and DOE (06ER25782, 08ER25860, 08NA28615, SC0002710). We thank Laura Alisic, George Biros, Martin Burtscher, Rahul Sampath, and Tiankai Tu for extended discussions. We also thank TACC for providing access to *Ranger* under TeraGrid award MCA04N026, and the NCCS/ORNL for early-user access to *Jaguar*.

REFERENCES

[1] M. Ainsworth and J. T. Oden, *A posteriori error estimation in finite element analysis*, ser. Pure and Applied Mathematics (New York). New York: John Wiley & Sons, 2000.

[2] I. Babuška and T. Strouboulis, *The finite element method and its reliability*, ser. Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York, 2001.

[3] R. Becker and R. Rannacher, "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, vol. 10, pp. 1–102, 2001.

[4] L. Demkowicz, J. Kurtz, D. Pardo, M. Paszyński, W. Rachowicz, and A. Zdunek, *Computing with hp Finite Elements II. Frontiers: Three-Dimensional Elliptic and Maxwell Problems with Applications*. CRC Press, Taylor and Francis, 2007.

[5] L. F. Diachin, R. Hornung, P. Plassmann, and A. Wissink, "Parallel adaptive mesh refinement," in *Parallel Processing for Scientific Computing*, ser. Software, Environments, and Tools, M. A. Heroux, P. Raghavan, and H. D. Simon, Eds. SIAM, 2006, no. 20, ch. 8.

[6] C. Burstedde, O. Ghattas, M. Gurnis, E. Tan, T. Tu, G. Stadler, L. C. Wilcox, and S. Zhong, "Scalable adaptive mantle convection simulation on petascale supercomputers," in *SC '08: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2008.

[7] C. Burstedde, L. C. Wilcox, and O. Ghattas, "`p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," 2010, submitted to *SIAM Journal on Scientific Computing*. http://ccgo.ices.utexas.edu/publications/BursteddeWilcoxGhattas10.pdf.

[8] L. C. Wilcox, G. Stadler, C. Burstedde, and O. Ghattas, "A high-order discontinuous Galerkin method for wave propagation through coupled elastic-acoustic media," *Journal of Computational Physics*, 2010, submitted, http://ccgo.ices.utexas.edu/publications/WilcoxStadlerBursteddeEtAl10.pdf.

[9] G. Stadler, M. Gurnis, C. Burstedde, L. C. Wilcox, L. Alisic, and O. Ghattas, "The dynamics of plate tectonics and mantle flow: From local to global scales," 2010, *Science*, In press.

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

[11] A. Calder, B. Curtis, L. Dursi, B. Fryxell, G. Henry, P. MacNeice, K. Olson, P. Ricker, R. Rosner, F. Timmes, H. Tufo, J. Truran, and M. Zingale, "High-performance reactive fluid flow simulations using adaptive mesh refinement on thousands of processors," in *SC '00: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2000, pp. 56–56.

[12] G. L. Bryan, T. Abel, and M. L. Norman, "Achieving extreme resolution in numerical cosmology using adaptive mesh refinement: Resolving primordial star formation," in *SC '01: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2001.

[13] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf, "The Cactus framework and toolkit: Design and applications," in *Vector and Parallel Processing – VECPAR '2002, 5th International Conference*. Springer, 2003.

[14] J. Luitjens, B. Worthen, M. Berzins, and T. Henderson, "Scalable parallel AMR for the Uintah multiphysics code," in *Petascale Computing Algorithms and Applications*, D. Bader, Ed. Chapman and Hall/CRC, 2007.

[15] P. Colella, D. Graves, N. Keen, T. Ligocki, D. F. Martin, P. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen, *Chombo Software Package for AMR Applications. Design Document.*, Applied Numerical Algoirthms Group, NERSC Division, Lawrence Berkeley National Laboratory, Berkeley, CA, May 2007.

[16] P. Colella, J. Bell, N. Keen, T. Ligocki, M. Lijewski, and B. van Straalen, "Performance and scaling of locally-structured grid methods for partial differential equations," *Journal of Physics: Conference Series*, vol. 78, pp. 1–13, 2007.

[17] J. Luitjens and M. Berzins, "Improving the performance of Uintah: A large-scale adaptive meshing computational framework," in *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2010.

[18] J. E. Flaherty, R. M. Loy, M. S. Shephard, B. K. Szymanski, J. D. Teresco, and L. H. Ziantz, "Adaptive local refinement with octree load balancing for the parallel solution of three-dimensional conservation laws," *Journal of Parallel and Distributed Computing*, vol. 47, no. 2, pp. 139–152, 1997.

[19] C. D. Norton, J. Z. Lou, and T. A. Cwik, "Status and directions for the pyramid parallel unstructured amr library," in *Proceedings of the 15th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2001, p. 120.

[20] O. S. Lawlor, S. Chakravorty, T. L. Wilmarth, N. Choudhury, I. Dooley, G. Zheng, and L. V. Kalé, "ParFUM: a parallel framework for unstructured meshes for scalable dynamic physics applications," *Engineering with Computers*, vol. 22, no. 3, pp. 215–235, 2006.

[21] B. Bergen, F. Hülsemann, and U. Rüde, "Is 1.7×10^{10} unknowns the largest finite element system that can be solved today?" in *SC '05: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2005.

[22] M. Griebel and G. W. Zumbusch, "Parallel multigrid in an adaptive PDE solver based on hashing and space-filling curves," *Parallel Computing*, vol. 25, pp. 827–843, 1999.

[23] S. Popinet, "Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries," *Journal of Computational Physics*, vol. 190, no. 2, pp. 572–600, 2003.

[24] J. R. Stewart and H. C. Edwards, "A framework approach for developing parallel adaptive multiphysics applications," *Finite Elements in Analysis and Design*, vol. 40, no. 12, pp. 1599–1617, 2004.

[25] D. Rosenberg, A. Fournier, P. Fischer, and A. Pouquet, "Geophysical-astrophysical spectral-element adaptive refinement (GASpAR): Object-oriented h-adaptive fluid dynamics simulation," *Journal of Computational Physics*, vol. 215, no. 1, pp. 59–80, 2006.

[26] A. Laszloffy, J. Long, and A. K. Patra, "Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive hp finite element simulations," *Parallel Computing*, vol. 26, pp. 1765–1788, 2000.

[27] M. Paszynski, D. Pardo, C. Torres-Verdín, L. Demkowicz, and V. Calo, "A parallel direct solver for self-adaptive hp-finite element method," *Journal of Parallel and Distributed Computing*, vol. 70, pp. 270–281, 2010.

[28] V. Akçelik, J. Bielak, G. Biros, I. Epanomeritakis, A. Fernandez, O. Ghattas, E. J. Kim, J. Lopez, D. R. O'Hallaron, T. Tu, and J. Urbanic, "High resolution forward and inverse earthquake modeling on terascale computers," in *SC '03: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2003, Gordon Bell Prize for Special Achievement.

[29] T. Tu, D. R. O'Hallaron, and O. Ghattas, "Scalable parallel octree meshing for terascale applications," in *SC '05: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2005.

[30] H. Sundar, R. S. Sampath, S. S. Adavani, C. Davatzikos, and G. Biros, "Low-constant parallel algorithms for finite element simulations using linear octrees," in *SC '07: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM/IEEE, 2007.

[31] C. Burstedde, O. Ghattas, G. Stadler, T. Tu, and L. C. Wilcox, "Towards adaptive mesh PDE simulations on petascale computers," in *Proceedings of Teragrid '08*, 2008.

[32] R. S. Sampath, S. S. Adavani, H. Sundar, I. Lashuk, and G. Biros, "`Dendro`: Parallel algorithms for multigrid and AMR methods on 2:1 balanced octrees," in *SC '08: Proceedings of*

the International Conference for High Performance Computing, Networking, Storage, and Analysis. ACM/IEEE, 2008.

[33] R. S. Sampath and G. Biros, "A parallel geometric multigrid method for finite elements on octree meshes," *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 1361–1392, 2010.

[34] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "`libMesh`: A C++ library for parallel adaptive mesh refinement/coarsening simulations," *Engineering with Computers*, vol. 22, no. 3–4, pp. 237–254, 2006.

[35] W. Bangerth, R. Hartmann, and G. Kanschat, "deal.II – a general-purpose object-oriented finite element library," *ACM Transactions on Mathematical Software*, vol. 33, no. 4, p. 24, 2007.

[36] G. M. Morton, "A computer oriented geodetic data base; and a new technique in file sequencing," IBM Ltd., Tech. Rep., 1966.

[37] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, ser. Texts in Applied Mathematics. Springer, 2008, vol. 54.

[38] M. H. Carpenter and C. A. Kennedy, "Fourth-order 2N-storage Runge-Kutta schemes," NASA Langley Research Center, Hampton, Virginia 23681-0001, NASA Technical Memorandum 109112, June 1994.

[39] M. Deville, P. Fischer, and E. Mund, *High-Order Methods for Incompressible Fluid Flow*, ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2002, vol. 9.

[40] C. Dohrmann and P. Bochev, "A stabilized finite element method for the Stokes problem based on polynomial pressure projections," *International Journal for Numerical Methods in Fluids*, vol. 46, pp. 183–201, 2004.

[41] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala, "ML 5.0 smoothed aggregation user's guide," Sandia National Laboratories, Tech. Rep. SAND2006-2649, 2006.

[42] C. Burstedde, O. Ghattas, G. Stadler, T. Tu, and L. C. Wilcox, "Parallel scalable adjoint-based adaptive solution for variable-viscosity Stokes flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, pp. 1691–1700, 2009.

[43] M. Billen and G. Hirth, "Rheologic controls on slab dynamics," *Geochemistry, Geophysics, Geosystems*, vol. 8, p. Q08012, 2007.

[44] A. M. Dziewonski and D. L. Anderson, "Preliminary reference earth model," *Physics of the Earth and Planetary Interiors*, vol. 25, no. 4, pp. 297–356, 1981.

[45] "Performance applications programming interface (PAPI)." [Online]. Available: http://icl.cs.utk.edu/papi/

[46] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven, "Nodal discontinuous Galerkin methods on graphics processors," *Journal of Computational Physics*, vol. 228, no. 21, pp. 7863–7882, 2009.