# *ALPS*: A framework for parallel adaptive PDE solution

**Carsten Burstedde**[⋆] **Martin Burtscher**[⋆] **Omar Ghattas**[⋆,†]
**Georg Stadler**[⋆] **Tiankai Tu**[⋆] **Lucas C. Wilcox**[⋆]

[⋆]Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, 1 University Station C0200, Austin TX 78712, USA
[†]Jackson School of Geosciences and Department of Mechanical Engineering, The University of Texas at Austin, 1 University Station C0200, Austin TX 78712, USA

E-mail: {carsten,burtscher,omar,georgst,ttu,lucasw}@ices.utexas.edu

**Abstract.** Adaptive mesh refinement and coarsening (AMR) is essential for the numerical solution of partial differential equations (PDEs) that exhibit behavior over a wide range of length and time scales. Because of the complex dynamic data structures and communication patterns and frequent data exchange and redistribution, scaling dynamic AMR to tens of thousands of processors has long been considered a challenge. We are developing *ALPS*, a library for dynamic mesh adaptation of PDEs that is designed to scale to hundreds of thousands of compute cores. Our approach uses parallel forest-of-octree-based hexahedral finite element meshes and dynamic load balancing based on space-filling curves. *ALPS* supports arbitrary-order accurate continuous and discontinuous finite element/spectral element discretizations on general geometries. We present scalability and performance results for two applications from geophysics: seismic wave propagation and mantle convection.

## 1. Introduction

The advent of the age of petascale computing brings unprecedented opportunities for breakthroughs in scientific understanding and engineering innovation. However, the raw performance made available by petascale systems is by itself not sufficient to solve many challenging modeling and simulation problems. For example, the complexity of solving evolutionary partial differential equations often scales as $n^{\frac{4}{3}}$, where $n$ is the number of unknowns.[1] Thus, the three-orders-of-magnitude improvement in peak speed of supercomputers over the past dozen years has meant just a factor of 5.6 improvement in spatio-temporal resolution—not even three successive refinements of mesh size. For many problems of scientific and engineering interest, there is a desire to increase the resolution of current simulations by several orders of magnitude. Thus, although supercomputing performance has outpaced Moore's Law over the past several decades due to increased concurrency [1], the curse of dimensionality imposes much slower scientific returns.

The work requirements of scientific simulations typically scale as $n^{\alpha}$. The power $\alpha$ can be reduced through the use of optimal solvers such as multigrid for PDEs and fast multipole for integral equations and N-body problems. Once $\alpha$ has been reduced as much as possible, further

[1] Optimal solvers require $\mathcal{O}(n)$ work per time step, and time accurate integration often implies $\mathcal{O}(n^{\frac{1}{3}})$ time steps.

reductions in work can be achieved only by reducing $n$ itself. This can be accomplished in two ways: through the use of adaptive mesh refinement/coarsening (AMR) techniques, and the use of higher order accurate discretizations (in space and time). AMR places mesh points only where needed to resolve features of the solution, whereas high-order approximations reduce the number of mesh points necessary to achieve a given accuracy.

Fortunately, many problems have local multiscale character, i.e., resolution is needed only in localized (possibly dynamically evolving) regions, such as near fronts, discontinuities, material interfaces, reentrant corners, boundary and interior layers, and so on. In this case, AMR methods can deliver orders-of-magnitude reductions in the number of mesh points. However, AMR methods can also impose significant overhead, in particular on highly parallel computing systems, due to their need for frequent re-adaptation and load-balancing of the mesh over the course of the simulation. Because of the complex data structures and communication patterns and frequent data exchange and redistribution, scaling dynamic AMR to tens of thousands of processors has long been considered a challenge. Space constraints preclude a proper review of existing methods and software for parallel AMR, but see the discussion and references in [2, 3].

We have developed the *ALPS* (**A**daptive **L**arge-scale **P**arallel **S**imulations) framework for parallel adaptive solution of PDEs [3]. *ALPS* includes the *octor* [4] and *p4est* libraries for parallel dynamic mesh adaptivity on single-octree-based and forest-of-octree-based geometries, respectively, and the *mangll* library for arbitrary-order hexahedral continuous and discontinuous finite/spectral element discretizations on general multi-octree geometries. *ALPS* has been shown to scale well weakly and strongly to over 60,000 processor cores [3]. In this paper, we describe several applications of the *ALPS* framework to two continuum mechanics problems—the propagation of elastic waves (with application to global seismology) and the buoyancy-driven creeping (Stokes) flow of a non-Newtonian incompressible fluid (with application to global mantle convection)—and provide sample performance results on Ranger, the 579 teraflops, 62,976-core Sun system at the Texas Advanced Computing Center (TACC).

## 2. Elastic wave propagation: Global seismology

The elastic wave equation models the propagation of longitudinal and shear waves in a linearly elastic medium. For isotropic media, the material stiffness tensor involves only the Lamé parameters $\lambda$ and $\mu$. In mixed velocity–stress form, the elastic wave equation reduces to

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} = \nabla \cdot \boldsymbol{S} + \boldsymbol{f}, \tag{1a}$$

$$\frac{\partial \boldsymbol{S}}{\partial t} = \mu \left( \nabla \boldsymbol{v} + \nabla \boldsymbol{v}^{\top} \right) + \lambda \left( \nabla \cdot \boldsymbol{v} \right) \boldsymbol{I}, \tag{1b}$$

where $\boldsymbol{v}$ and $\boldsymbol{S}$ are the unknown velocity vector and Cauchy stress tensor of the medium, and $\rho$, $t$, $f$, and $\boldsymbol{I}$ are the mass density, time, body force, and identity tensor, respectively.

We have built an elastic wave propagation code based on the *mangll* and *p4est* libraries from *ALPS*. Velocity and stress fields are discretized using the discontinuous Galerkin (dG) method with a Godunov flux [5] in space and a five-stage fourth-order explicit Runge-Kutta (RK) method in time. The elements are the same as those used in the spectral element method [6], having nodes at the tensor product Legendre-Gauss-Lobatto (LGL) points. The dG method duplicates degrees of freedom between adjacent element faces. On nonconforming interfaces between elements (arising due to 2:1 subdivision between adjacent hexahedra), the numerical flux is computed by introducing a face integration mesh that incorporates the contributions from each smaller face individually using the two-dimensional tensor LGL quadrature based on the nodes of the smaller face. The LGL quadrature reduces the block diagonal dG mass matrix to a diagonal matrix, permitting faster application of its inverse.

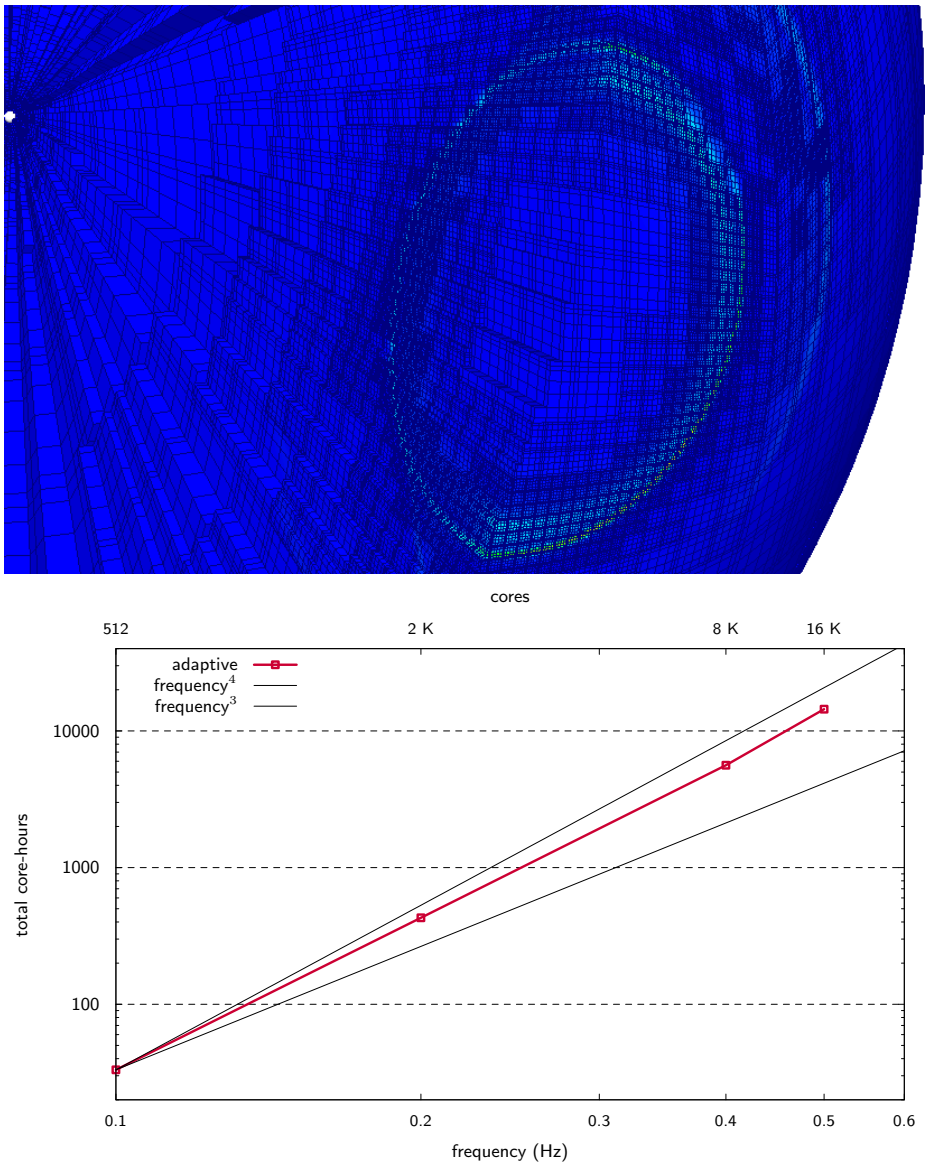


**Figure 1.** *Top:* cut of wavefront-adapted spherical mesh for a double-couple source. *Bottom:* scaling of run time with the number of cores for dynamic-adaptive wave propagation solver.

Figure 1 presents results from the solution of an elastic wave propagation problem in a spherical geometry. The material parameters approximate those of the Earth and the source is a simple double-couple. The top image depicts a snapshot of a mesh that has been adapted to propagating wavefronts. The significant reduction of mesh points can be seen in the image. The bottom plot presents scaling results to 16K cores, showing the growth in total processor-hours as a function of maximum source frequency. A perfectly scalable static mesh explicit wave propagation solver would have a total solution time that scales as the fourth power of frequency. The scaling is shown as the top black line (with 512 cores as the base case). Dynamic adaptivity has the potential to reduce the complexity to be closer to the third power of frequency (shown as the bottom black line), since adapting to moving wavefronts effectively eliminates one dimension from the mesh. The cubic scaling cannot be achieved for problems on finite domains or with interfaces, due to reflections and generation of new waves (the number of wavefronts

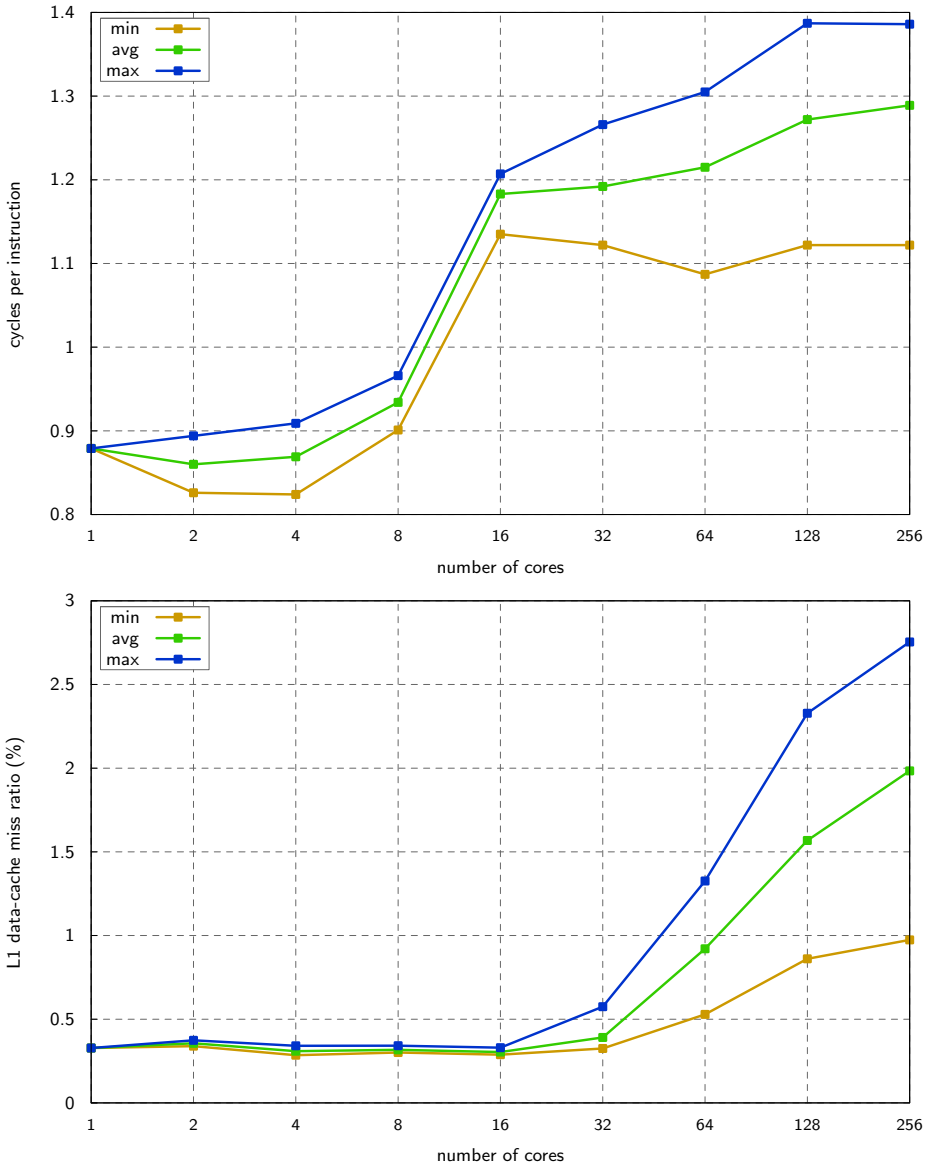


**Figure 2.** CPU performance for wave propagation code with 10K total elements of polynomial degree 6. *Top:* cycles per instruction. *Bottom:* L1 data-cache miss ratio in percent.

that must be tracked does not remain constant). The actual observed scaling is plotted as the red line, showing that, even in a layered Earth model, the "optimal" fourth power scaling can be improved using adaptivity.

We also studied the efficiency of the implementation using performance counter measurements from the *PAPI* library [7]; the results are summarized in Figure 2. Up to eight cores, more than one instruction is executed per cycle on average, indicating that there is significant instruction-level parallelism. As the compute nodes become saturated when using all 16 cores per node, we observe a jump in the cycles per instruction (CPI). Yet, even above 16 cores, the CPI is low because we carefully tuned important sections of the source code so that the compiler can vectorize them [8]. Beyond 16 cores, the CPI grows due to an increase in cache misses, which is caused by an increase in data that need to be exchanged between compute nodes, which results in less regular memory access. Nevertheless, the L1 data-cache miss ratio is surprisingly low for

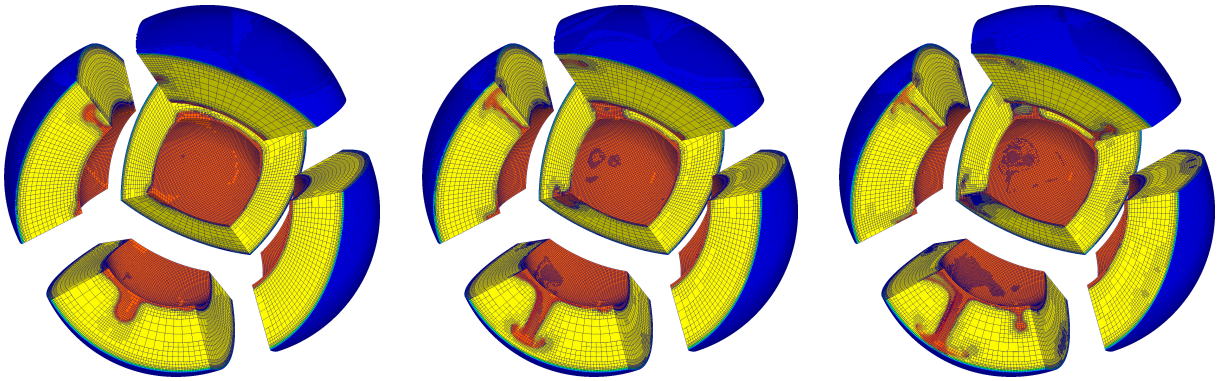

**Figure 3.** Snapshots of three time steps from a mantle convection simulation using *Rhea*, showing dynamically-adapted refinement near rising thermal plumes and thermal boundary layers.

code that processes about 1.75, 1.12 and 0.96 million degrees of freedom per second per core on 1, 32 and 256 cores, respectively. This low ratio is due to data being accessed in mostly regular patterns, which allows data to be prefetched effectively.

## 3. Non-Newtonian Stokes flow: Mantle convection

Our second example is flow of a viscous incompressible creeping non-Newtonian fluid, which is governed by equations for mass, momentum, and energy balance,

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

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

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

where $\boldsymbol{v}$, $p$, and $T$ are the unknown velocity, pressure, and temperature; and $\eta = \eta(T, \boldsymbol{v})$, $\rho$, $\boldsymbol{g}$, $c_p$, $k$ and $H$ are the viscosity, density, gravitational acceleration, specific heat, thermal conductivity, and internal heat generation rate. We are interested in convection in Earth's mantle, where the flow is driven by buoyancy. Thus we invoke the Boussinesq approximation, which replaces the density in the gravitational force term in (2b) by $\rho = \rho_0 [1 - \alpha(T - T_0)]$, where $\rho_0$ and $T_0$ denote reference temperature and density and $\alpha$ is the coefficient of thermal expansion. Various constitutive laws are used for the mantle, but in general the viscosity depends nonlinearly on both temperature and second invariant of the deviatoric strain rate tensor.

We have built a parallel AMR mantle convection code, *Rhea*, that solves (2a)–(2b) with appropriate boundary conditions and temperature- and strain-rate-dependent viscosity [3] using *ALPS* components. The images in Figure 3 illustrate dynamic mesh adaptivity in resolving thermal upwellings and boundary layers. The first version of *Rhea* discretizes the velocity, pressure, temperature fields with mapped trilinear hexahedral finite elements. Mantle flows are strongly advection-dominated, and therefore we employ the streamline-upwind Petrov-Galerkin (SUPG) scheme to stabilize the energy equation (2c). The equal-order discretization of the Stokes equations (2a), (2b) is stabilized with pressure projection [9]. 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 a variable-viscosity Stokes solve, which is discussed below.

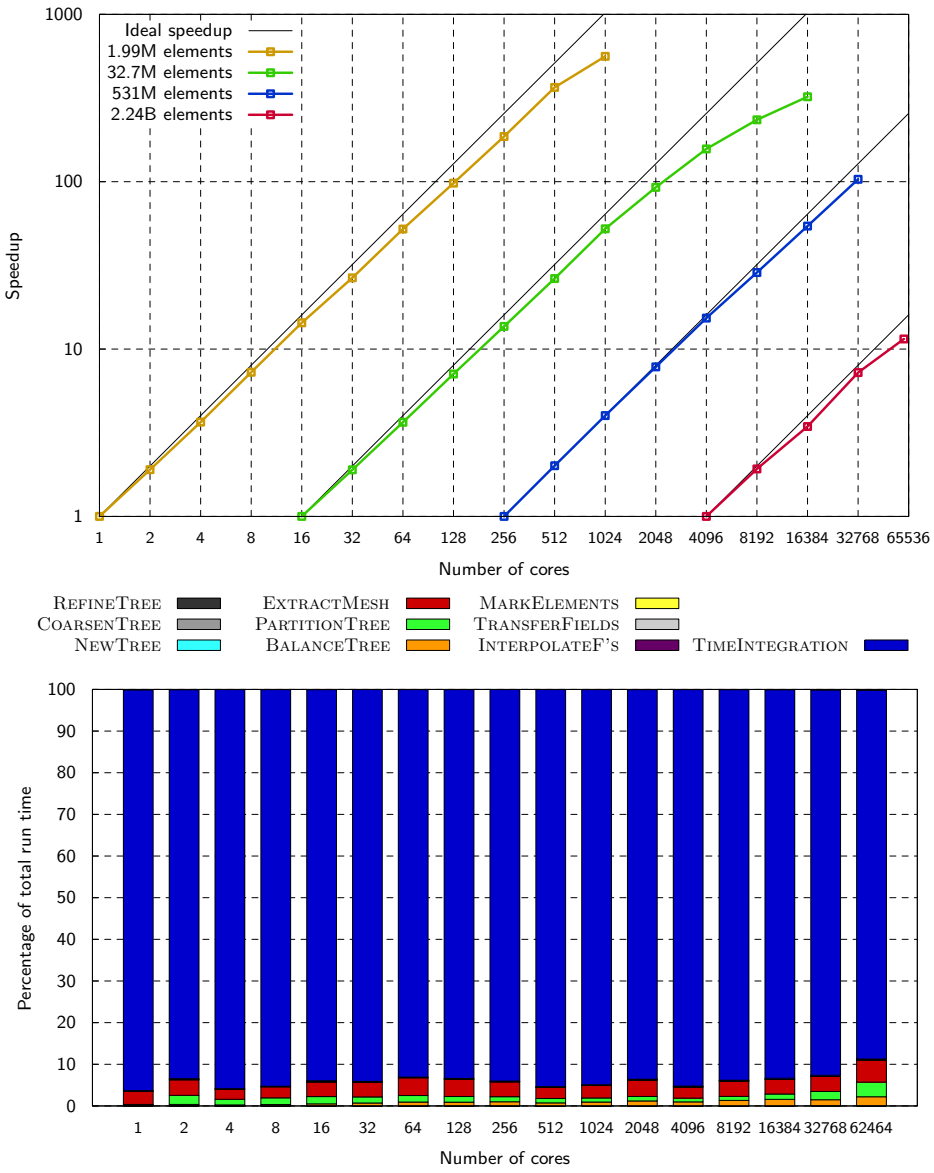


**Figure 4.** *Top:* fixed-size (strong) scalability for adaptive solution of energy equation for four problem sizes on cubic domain. Mesh is adapted every 32 time steps. *Bottom:* weak scalability for adaptive solution of energy equation. Total run time is broken down into numerical PDE solution (blue) and AMR functions (all other colors). Problem size increases isogranularly at 131,000 elements per core; largest problem has approximately 7.9 billion elements.

First, we study the scalability of AMR for solution of the energy equation (2c). This low-order-discretized, explicitly-solved, scalar, linear equation is a severe test of AMR, since there is very little numerical work over which to amortize AMR. As can be seen from the top plot in Figure 4, the fixed-size scaling speedups are nearly optimal over a wide range of core counts. For instance, solving the problem with 531 million elements (blue line) exhibits a speedup of 101 on 32,768 cores over the same problem running on 256 cores (128 would be optimal). Moreover, scaling weakly from 1 to 62,464 cores (bottom of Figure 4), we see that all mesh adaptation functions—including coarsening, refinement, interpolation, rebalancing and repartitioning of the mesh—together impose little overhead on the PDE solver. Only for 62K cores does the total

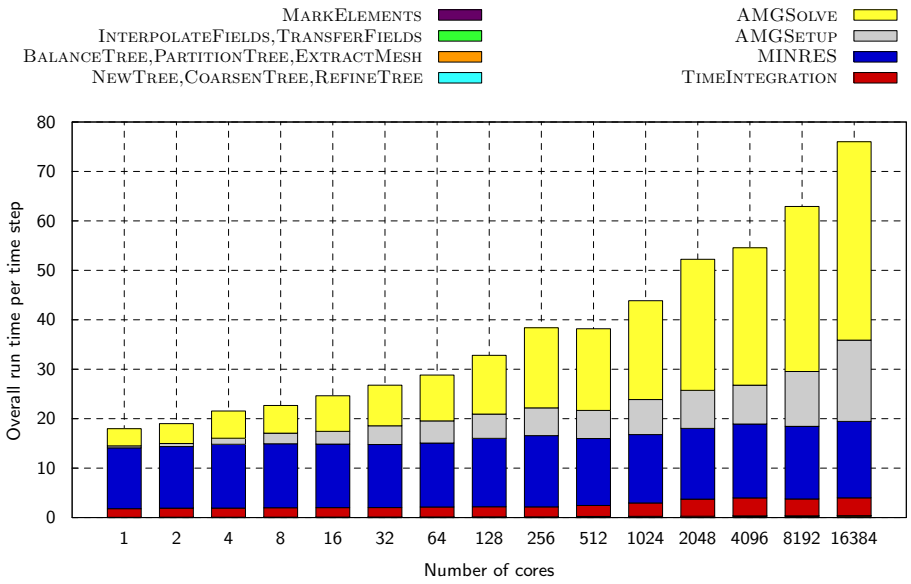


**Figure 5.** Breakdown of mantle convection run time per time step into AMG setup (gray), AMG V-cycle (yellow), MINRES iteration (blue), and time integration of the energy equation (red) using *BoomerAMG* preconditioning. AMR operations (all other colors) have negligible cost. Mesh is adapted every 16 time steps, and problem size increases isogranularly at 50K elements per core (largest problem has 815 million elements).

cost of AMR exceed 10% of the end-to-end run time, and even then just barely.

Next, we study the algorithmic and parallel scalability of the variable-viscosity Stokes solver, which is invoked at each Picard iteration within each time step. We use MINRES to solve the symmetric indefinite Stokes system; the preconditioner is a block-diagonal matrix, which invokes one V-cycle of an algebraic multigrid (AMG) method for the viscous (1,1) block, and an inverse-viscosity-scaled mass matrix for the pressure Schur-complement (2,2) block (see [10] for details). For Cartesian geometry, the viscous vector operator can be approximated by a scalar, variable-coefficient Poisson operator associated with each of the three velocity components (provided viscosity gradients are not too large). The Poisson operators are approximately inverted by one V-cycle of *BoomerAMG* from the *hypre* library [11]. The bar chart in Figure 5 shows the breakdown of different components of an adaptive variable-viscosity Stokes solve using *BoomerAMG* as we scale weakly from 1 to 16K cores. Spherical geometry, on the other hand, induces additional coupling among the velocity components that cannot be effectively approximated by the decomposition into scalar Poisson solves. Thus we employ one V-cycle of the adaptive smoothed aggregation AMG solver *ML* from *Trilinos* [12] on the entire viscous vector block. Table 1 presents weak scalings of a single variable-viscosity Stokes solve on the spherical domain using *ML*. We draw several conclusions from Figure 5 and Table 1. First, for the full AMR mantle convection simulations with *Rhea* (i.e. including energy and nonlinear Stokes), the cost of parallel AMR is in the noise. Second, using a V-cycle of either *BoomerAMG* or *ML* as a preconditioner for the viscous block, in conjunction with a scaled mass matrix approximation of the Schur complement, results in excellent algorithmic scalability, as evidenced by the near-insensitivity of MINRES iterations to a 16,000-fold increase in problem size and number of cores. Third, while both AMG implementations exhibit excellent algorithmic scalability, there is an opportunity for improvement of the parallel scalability of the AMG setup and V-cycle components at $O(10^4)$ cores, based on our tests on a cluster system and viscous operators with several-orders-of-magnitude variation in coefficients. Finally, the examples demonstrate that

| #cores | #dofs | MINRES #iterations | AMG setup (s) | AMG V-cycle (s) | MINRES matvec (s) | $\zeta_A$ | $\zeta_I$ | $\zeta_V$ | $\zeta$ |
|---|---|---|---|---|---|---|---|---|---|
| 8 | 1.89M | 77 | 7.12 | 41.3 | 130.8 | 1.00 | 1.00 | 1.00 | 1.00 |
| 64 | 12.8M | 76 | 8.47 | 38.9 | 114.9 | 1.01 | 1.12 | 1.05 | 1.10 |
| 512 | 99.7M | 78 | 10.2 | 48.1 | 129.9 | 0.99 | 1.02 | 0.87 | 0.95 |
| 4096 | 672M | 109 | 36.5 | 189.8 | 208.7 | 0.70 | 0.89 | 0.31 | 0.41 |
| 16384 | 2.43B | 109 | 66.0 | 211.3 | 216.5 | 0.70 | 0.86 | 0.28 | 0.36 |

**Table 1.** Weak scaling of Stokes solve on spherical shell domain using *ML*, with $10^3$ viscosity contrast, on a 3 times locally refined mesh. The numbers of cores, degrees of freedom, and MINRES iterations are tabulated, along with the time for AMG setup, AMG V-cycle, and MINRES matvecs. Also shown are algorithmic efficiency $\zeta_A$ (based on MINRES iterations), implementation efficiency $\zeta_I$ (based on MINRES timing, excluding V-cycle), V-cycle efficiency $\zeta_V$ (based on *ML* timing), and overall efficiency $\zeta$ (based on overall wall clock).

our framework for parallel forest-of-octrees AMR with high-order-accurate discretization and general geometries scales very well to at least $O(10^4)$ processor cores.

**Acknowledgement**
This work was partially supported by NSF (OCI-0749334, CCF-0427985, CNS-0540372, CNS-0619838, DMS-0724746) and DOE (06ER25782, 08ER25860). Computing resources at TACC were provided under TeraGrid award MCA04N026. We acknowledge many helpful discussions with Rob Falgout, Ulrike Yang, Rahul Sampath, George Biros, and Tim Warburton. We thank TACC for their outstanding support, in particular Bill Barth, Jay Boisseau, Tommy Minyard, Romy Schneider, and Karl Schulz.

**References**

[1] Colella P, Dunning T H, Gropp W D and Keyes D E 2004 A Science-Based Case for Large-Scale Simulation, Volume 2 Office of Science, U.S. Department of Energy URL `http://www.pnl.gov/scales/`
[2] Diachin L F, Hornung R, Plassmann P and Wissink A 2006 *Parallel Adaptive Mesh Refinement* (SIAM) chap 8 (*Software, Environments, and Tools* no 20)
[3] Burstedde C, Ghattas O, Gurnis M, Tan E, Tu T, Stadler G, Wilcox L C and Zhong S 2008 Scalable adaptive mantle convection simulation on petascale supercomputers *Proceedings of SC08* (IEEE/ACM) Gordon Bell Prize finalist.
[4] Tu T, O'Hallaron D R and Ghattas O 2005 Scalable parallel octree meshing for terascale applications *Proceedings of SC2005*
[5] Hesthaven J S and Warburton T 2008 *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications* (*Texts in Applied Mathematics* vol 54) (Springer)
[6] Deville M, Fischer P and Mund E 2002 *High-Order Methods for Incompressible Fluid Flow* (*Cambridge Monographs on Applied and Computational Mathematics* vol 9) (Cambridge University Press)
[7] Performance applications programming interface (PAPI) URL `http://icl.cs.utk.edu/papi/`
[8] Diamond J, Kim D, Burtscher M, Keckler S, Pingali K and Browne J 2009 Multicore optimization for Ranger *Proceedings of Teragrid '09*
[9] Dohrmann C and Bochev P 2004 *International Journal for Numerical Methods in Fluids* **46** 183–201
[10] Burstedde C, Ghattas O, Stadler G, Tu T and Wilcox L C 2009 *Computer Methods in Applied Mechanics and Engineering* **198** 1691–1700
[11] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory 2007 hypre. *High Performance Preconditioners, User Manual* `https://computation.llnl.gov/casc/linear_solvers/`
[12] Gee M, Siefert C, Hu J, Tuminaro R and Sala M 2006 ML 5.0 smoothed aggregation user's guide Tech. Rep. SAND2006-2649 Sandia National Laboratories