Forestclaw: Hybrid forest-of-octrees AMR for hyperbolic conservation laws

Carsten BURSTEDDE [a,1], Donna CALHOUN [b], Kyle MANDLI [c] and Andy R. TERREL [c]

[a] *Institut für Numerische Simulation, Universität Bonn, Germany*
[b] *Boise State University, Idaho, USA*
[c] *Institute for Computational Engineering and Sciences, The University of Texas at Austin, USA*

Abstract. We present a new hybrid paradigm for parallel adaptive mesh refinement (AMR) that combines the scalability and lightweight architecture of tree-based AMR with the computational efficiency of patch-based solvers for hyperbolic conservation laws. The key idea is to interpret each leaf of the AMR hierarchy as one uniform compute patch in $\mathbb{R}^d$ of dimensions m^d, where m is customarily between 8 and 32. Thus, computation on each patch can be optimized for speed, while we inherit the flexibility of adaptive meshes. In our work we choose to work with the `p4est` AMR library since it allows us to compose the mesh from multiple mapped octrees and enable the cubed sphere and other nontrivial multiblock geometries.

Keywords. adaptive mesh refinement, hyperbolic conservation laws, clawpack, HPC, manycore

1. Introduction

With the advent of many-core chips, such as GPUs and the MIC architecture, comes the opportunity to sustain unprecedented rates of floating point operations at comparably high integration density and low cost. These architectures, however, require careful structuring of the data layout and memory access patterns to exhaust their multithreading and vectorization capabilities.

Consequently, it is not clear a priori how to accelerate PDE solvers that use adaptive mesh refinement. Of course, it was realized early that it helps to aggregate degrees of freedom (DOF) at the element level, as has been done with high-order spectral element [1], low order continuous Galerkin via methods that accumulate many elements simultaneously [2], or discontinuous Galerkin [3] methods. GPU implementations of the latter have been proposed recently [4, 5]. The finite volume method has typically been implemented using a single degree of freedom per cell on structured [6,7] or unstructured meshes [8], where higher order methods have been constructed by widening the stencil, for instance in WENO methods [9].

To facilitate hardware acceleration for parallel dynamic AMR, we build upon the forest-of-octrees paradigm because of its low overhead and proven scalability [10]. This

[1] Corresponding author. E-mail: `burstedde@ins.uni-bonn.de`

approach identifies each octree leaf with a mesh element. In our work, we go beyond the traditional high-order element and define each element to be a dense computational patch with m^d DOFs. In fact, this approach resembles block-structured AMR [11–14] except that the patches are not overlapping, which enables us to capitalize on our previous experience with scalable FE solvers for PDEs [15]. The CLAWPACK software [16] provides a popular implementation of such a patch. It has been designed to solve hyperbolic conservation laws and successfully used in the context of block-structured AMR [17–19].

In this paper we describe our design for the coupling of forest-of-octree AMR with CLAWPACK at the leaf level. We comment on challenges that arise in enabling multiblock geometries and efficient parallelism and conclude with a range of numerical examples that demonstrate the conceptual improvements in relation to other approaches.

2. Design principles

The starting point of our work is defined by the `p4est` algorithms for forest-of-octrees AMR on the one hand, and the CLAWPACK algorithms for the numerical solution of hyperbolic conservation laws on the other. Both are specialized codes with the following characteristics:

	`p4est`	CLAWPACK
subject	hexahedral nonconforming mesh	hyperbolic PDE on $[0,1]^d$
toplevel unit	forest of octrees	patch of m^d FV cells
atomic unit	octree leaf	one DOF in each cell
parallelization	MPI	threads (Manyclaw variant)
memory access	distributed	shared on each MPI rank
data type	integers	floating point values
language	C	Fortran 77

Each leaf as the atomic unit of `p4est` houses a toplevel unit of CLAWPACK. The term cell is used to identify a single DOF within a CLAWPACK patch. The proposed 1:1 correspondence between a leaf and a patch thus combines two previously disjoint models in a modular way:

1. We permit the reuse of existing, verified, and performant codes.
2. We preserve the separation between the mesh on one hand and the discretization and solvers on the other.
3. The AMR metadata (`p4est`: under 1k bytes per octree, 8 bytes per MPI rank, 24 bytes per leaf. Forestclaw: $84+28d$ bytes per patch) is insignificant compared to the numerical data (m^d floating point values per patch).
4. The resulting parallel programming model is a hybrid (often referred to as MPI+X). Only rank-local leaves/patches are stored and computed on.

A particular feature of Forestclaw is that the generic handling of multiblock geometries is inherited from `p4est`, identifying each octree as a block. Each block is understood as a reference unit cube with its own geometric mapping. The connectivity of the blocks can be created by external hexahedral mesh generators, eliminating the need to encode it by hand.

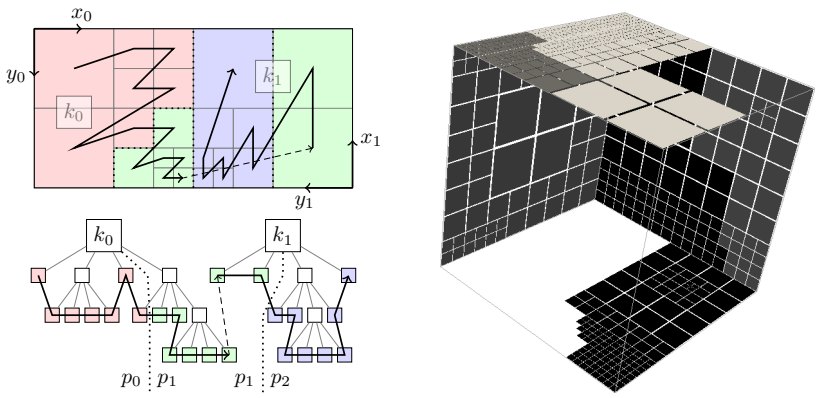

Figure 1. Left: Forest of two quadtrees, partitioned among three MPI processes. Each quadtree has its own coordinate orientation. The situation in 3D (octrees) is analogous. Right: The leaves in a forest of six quadtrees that serves as the computational domain for the cubed sphere. An adhoc refinement pattern has been 2:1 balanced in terms of neighbor sizes and partitioned between five MPI processes (the three middle ones are shown from black to white).

A main challenge is presented by the fact that the patch neighborhood is only known to `p4est`. This patch connectivity information needs to be propagated to the numerical code in Forestclaw that implements the interaction with neighbor patches via the use of a layer of ghost cells surrounding each patch. To this end, we define an interface that allows read-only access to the sequence of blocks, the list of patches for each, and a lookup of neighbor patches (and their relative orientation which is nontrivial between blocks). Suitably informed by `p4est`, Forestclaw stores only the patches local to each MPI rank. The exchange of ghost data and parallel repartitioning is provided transparently by `p4est`. Mesh modification directives, such as adaptive refinement and coarsening, are called from Forestclaw and relayed to `p4est`.

3. Parallelization

The MPI layer is addressed from within `p4est` and not exposed to the Forestclaw code. The order of leaves is maintained in `p4est` according to a space filling curve. Each MPI rank has a local view on its own partition, augmented where necessary with information about one layer of ghost leaves (see Figure 1).

Forestclaw uses iterators over all rank-local leafs, optionally restricted to a given level. Random access is possible and used when executing $O(1)$ time neighbor lookups. Looping over the patches in the order prescribed by the forest and accessing neighbors only relative to the current patch leads to a high percentage of cache reuse [20].

When Forestclaw accesses neighbor patches, they can be on the same or a different block. In the latter case, coordinate transformations are carried out. The structure of Forestclaw is oblivious to the fact that it only has a local view of the distributed mesh and data which relieves the developer from programming to the MPI interface. Parallel neighbor exchanges are hidden inside `p4est` and called by Forestclaw at well-defined synchronization points. There is one such parallel data exchange per time step for a global value of dt, or one exchange per discretization level per time step if dt is chosen

per-patch depending on its size (this is sometimes called sub-cycling). When using sub-cycling, the load balance is attained per level as it is done for example in Chombo [13] or recent geometric adaptive multigrid schemes [21].

The threaded parallelism over the degrees of freedom of a patch can be handled by Forestclaw alone without the need to involve `p4est`. Additionally many-core implementations, such as Manyclaw [22], can be used for the hyperbolic integration on a leaf-patch. The design of leaf-patches allows for the smart management of data local to accelerator and the host.

4. Patch-based numerics at the leaf level

For hyperbolic problems, we integrate the solution on a single uniform patch, containing m^d cells, using the wave propagation algorithm described by R. J. LeVeque [23] and implemented in CLAWPACK [7, 24]. We assume a single degree of freedom per cell and reconstruct a piecewise constant solution to obtain left and right states at cell edges. At each edge, we solve Riemann problems to obtain left and right going waves propagating at speeds determined from the solution to the Riemann problem. For scalar advection, the speed of each wave is simply the local advection speed at the cell interface. For nonlinear problems and systems, an approximate Riemann solver, such as a Roe solver [25], is typically used. Since much of the physics of an application can be contained in the Riemann solver, Forestclaw adopts CLAWPACK's interface to Riemann solvers effectively allowing problems solved with CLAWPACK to be solvable in Forestclaw. In order to achieve second order accuracy wave limiters are used to suppress spurious oscillations at sharp gradients in the flow.

Data exchanges between neighboring patches are done via layers of ghost cells extending the dimensions along the edges of each patch. The interior edge values of a given patch overlap the ghost cell region of a neighboring patch. For the second order wave propagation algorithm, two layers of ghost cells are sufficient. This implies that one layer of ghost patches is sufficient for $m \geq 4$. Neighboring patches at the same level of refinement simply copy their interior edge values into a neighbors' ghost cells. Neighboring fine grid patches average their interior edge data to a coarser neighbor's ghost cell values. And neighboring coarse grid patches interpolate data from their interior edge cells to their fine grid neighbor's ghost cell values. To avoid loss of conservation and the creation of spurious extrema, we use a standard conservative, limited interpolation scheme to interpolate values from the coarse grid to fine grid coarse cells [18, 26]. When sub-cycling, time accurate data between coarse grids is used to fill in ghost cells for fine grids. As mentioned in the previous section, this procedure can be extended transparently to distributed parallelism by defining an abstract exchange routine for ghost patch data.

Grid refinement and coarsening requires interpolation from coarse grids to newly created fine grids, and the averaging of fine grid data to a newly created underlying coarse grid. This operation is rank-local, analogous to the general dynamic AMR procedures used in `p4est`-based FE codes [27, Fig. 4].

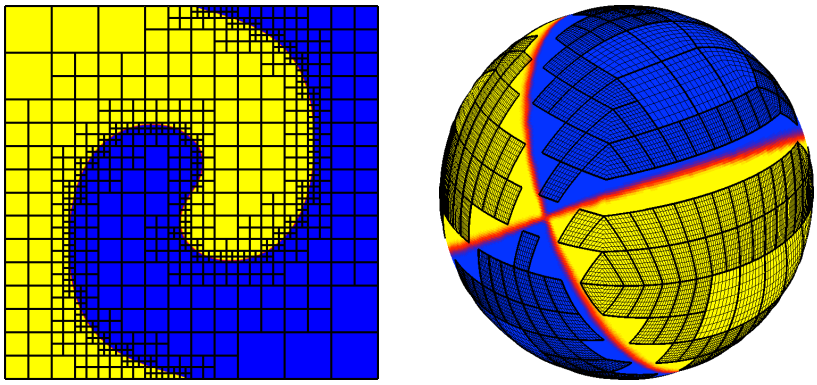

Figure 2. Numerical results for solving the advection equation. Left: Unit square (single quadtree) with a twisting velocity field. Right: Spherical ball with a rotational velocity field constructed from two mapped quadtrees. In both cases the concentration is color-coded with a sharp gradient shown in red. The adaptive mesh refinement follows the location of the gradient (the patches are not shown where they become too fine for display). Here we use $m = 8$ cells per CLAWPACK patch.

5. Numerical results

We provide two examples that demonstrate the Forestclaw code. The numerical results to date have been designed to verify that interface layer between `p4est` and Forestclaw is sufficiently flexible and robust. Of particular importance was ensuring that all ghost cell transfers (including averaging and interpolation) have been implemented correctly. The basic CLAWPACK algorithm and corresponding code have been thoroughly tested and need no further verification in our context.

In both sets of numerical results, we solve a scalar advection equation,

$$q_t + (\mathbf{u} \cdot \nabla) q = 0, \tag{1}$$

where the velocity field $\mathbf{u} = (u(\xi, \eta, t), v(\xi, \eta, t)$ is a prescribed function of computational coordinates. The relevant numerical parameters that are set in each case include the size of the patch on each leaf ($m = 8$ in each case), and the minimum and maximum refinement levels, which in turn fix minimum and maximum effective grid resolutions.

In both examples, a patch is tagged for refinement if the difference between its maximum and minimum values exceeds a prescribed threshold. A family of 2^d patches is coarsened to a parent patch if this patch would not meet the criteria for refinement.

Example 1: An initial concentration field q is set to 0 in the left half of a computational square and to 1 in the right half. A time dependent flow field is prescribed that distorts the interface between the 0 and 1 concentration values. The minimum and maximum levels of refinement are set to 3 and 6, respectively. This results in a minimum resolution of 64×64 and a maximum resolution of 512×512. Figure 2 shows the results at an intermediate time step.

Example 2: We demonstrate the multiblock functionality of Forestclaw by considering flow on a sphere. The sphere mapping we use consists of two quadtrees, each defined in the computational space $[-1,1] \times [-1,1]$ [28,29]. Each quadtree is mapped

to cover one hemisphere. The initial conditions is $q = 0$ in one half of the sphere, and $q = 1$ in the other half, where the halves are not necessarily aligned with the equator of the mapping. The flow field u simulates rigid body rotation. We show the results at an intermediate time in Figure 2.

The results in this section are preliminary and will be extended in the final version. We have so far exercised Forestclaw for two 2D manifolds, advecting a concentration field with a sharp gradient that we track dynamically with adaptive mesh refinement. We observe that the refinement follows the front and allows for an accurate representation of its location over time.

`p4est` is parallelized with MPI and well tested on large supercomputers, both in 2D and 3D. Parallelization of Forestclaw is not completed yet but envisioned for the final version of the paper.

Acknowledgements

The authors acknowledge valuable discussion with Randy LeVeque, Marsha Berger, and Hans-Petter Langtangen. We also acknowledge David Ketcheson and the KAUST sponsored HPC^3 numerics workshop at which the initial phases of this project were first discussed. The second author would like to also acknowledge the Isaac Newton Institute (Cambridge, UK), where much of the preliminary development work for Forestclaw was done. The leaf/patch paradigm was independently presented by B. as part of a talk at the SCI Institute, Salt Lake City, Utah, USA in July 2011.

References

[1] H. M. Tufo and P. F. Fischer. Terascale spectral element algorithms and implementations. In *Proceedings of the ACM/IEEE SC99 Conference on High Performance Networking and Computing*, 1999.

[2] Matthew G. Knepley and Andy R. Terrel. Finite Element Integration on GPUs. *Transactions of Mathematical Software*, 39(2), 2013.

[3] Jan S. Hesthaven and Timothy Warburton. Nodal high-order methods on unstructured grids. I. Time-domain solution of Maxwell's equations. *Journal of Computational Physics*, 181(1):186–221, 2002.

[4] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven. Nodal discontinuous Galerkin methods on graphics processors. *Journal of Computational Physics*, 228(21):7863–7882, 2009.

[5] Carsten Burstedde, Omar Ghattas, Michael Gurnis, Tobin Isaac, Georg Stadler, Tim Warburton, and Lucas C. Wilcox. Extreme-scale AMR. In *SC10: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM/IEEE, 2010.

[6] Phillip Colella and Paul R. Woodward. The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations. *Journal of Computational Physics*, 54:174–201, 1984.

[7] AMRClaw website: `http://www.clawpack.org`.

[8] OpenFOAM website: `http://openfoam.org`.

[9] Chi-Wang Shu. High Order Weighted Essentially Nonoscillatory Schemes for Convection Dominated Problems. *SIAM Review*, 51(1):82–126, February 2009.

[10] Carsten Burstedde, Lucas C. Wilcox, and Omar Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[11] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[12] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[13] Phillip Colella, Daniel T. Graves, Noel Keen, Terry J. Ligocki, Daniel F. Martin, Peter W. McCorquodale, David Modiano, Peter O. Schwartz, Theodore D. Sternberg, and Brian Van Straalen. *Chombo Software Package for AMR Applications. Design Document.* Applied Numerical Algoirthms Group, NERSC Division, Lawrence Berkeley National Laboratory, Berkeley, CA, May 2007.

[14] M. Berzins, J. Luitjens, Q. Meng, T. Harman, and C. A. Wight. Uintah: a scalable framework for hazard analysis. In *Proceedings of the 2010 TeraGrid Conference*, 2010.

[15] Carsten Burstedde, Georg Stadler, Laura Alisic, Lucas C. Wilcox, Eh Tan, Michael Gurnis, and Omar Ghattas. Large-scale adaptive mantle convection simulation. *Geophysical Journal International*, 2013. Accepted for publication.

[16] R. J. LeVeque. Wave propagation algorithms for multi-dimensional hyperbolic systems. *Journal of Computational Physics*, 131:327–353, 1997.

[17] M. J. Berger and R. J. LeVeque. A rotated difference scheme for Cartesian grids in complex geometries. AIAA Conference on Computational Fluid Dynamics, Honolulu, HI, CP-91-1602, 1991.

[18] AMRClaw website: `http://www.clawpack.org`.

[19] Randall J. LeVeque, David L George, and Marsha J Berger. Tsunami Propagation and inundation with adaptively refined finite volume methods. *Acta Numerica*, pages 211–289, 2011.

[20] Carsten Burstedde, Martin Burtscher, Omar Ghattas, Georg Stadler, Tiankai Tu, and Lucas C. Wilcox. ALPS: A framework for parallel adaptive PDE solution. *Journal of Physics: Conference Series*, 180:012009, 2009.

[21] Hari Sundar, George Biros, Carsten Burstedde, Johann Rudi, Omar Ghattas, and Georg Stadler. Parallel geometric-algebraic multigrid on unstructured forests of octrees. In *SC12: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM/IEEE, 2012.

[22] Andy R. Terrel and Kyle T. Mandli. ManyClaw: Slicing and dicing Riemann solvers for next generation highly parallel architectures. *TACC-Intel Symposium on Highly Parallel Architectures*, 2012.

[23] Randall J. LeVeque. Wave Propagation Algorithms for Multidimensional Hyperbolic Systems. *Journal of Computational Physics*, 131(2):327–353, March 1997.

[24] R. J. LeVeque. *Finite volume methods for hyperbolic problems*. Cambridge University Press, 2002.

[25] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43(2):357–372, October 1981.

[26] Chombo website: `http://seesar.lbl.gov/anag/chombo`.

[27] Carsten Burstedde, Omar Ghattas, Georg Stadler, Tiankai Tu, and Lucas C. Wilcox. Towards adaptive mesh PDE simulations on petascale computers. In *Proceedings of Teragrid '08*, 2008.

[28] D. A. Calhoun, C. Helzel, and R. J. LeVeque. Logically rectangular grids and finite volume methods for PDEs in circular and spherical domains. *SIAM Review*, 50(4):723–752, 2008.

[29] M. J. Berger and R. J. LeVeque D. Calhoun, C. Helzel. Logically rectangular finite volume methods with adaptive refinement on the sphere. *Phil. Trans. R. Soc. A*, 367(1):4483–4496, 2009.