Scalable Algorithms for Adaptive Mesh Refinement: Extension to General Element Types and Application to Fluid Dynamics

C. Burstedde, D. Calhoun, J. A. Fonseca, J. Holke, S. Kollet, M. Lahnert, M. Mehl

published in

NIC Symposium 2018
K. Binder, M. Müller, A. Trautmann (Editors)

Forschungszentrum Jülich GmbH,
John von Neumann Institute for Computing (NIC),
Schriften des Forschungszentrums Jülich, NIC Series, Vol. 49,
ISBN 978-3-95806-285-6, pp. 361.
http://hdl.handle.net/2128/17544

© 2018 by Forschungszentrum Jülich
Permission to make digital or hard copies of portions of this work for personal or classroom use is granted provided that the copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise requires prior specific permission by the publisher mentioned above.

Scalable Algorithms for Adaptive Mesh Refinement: Extension to General Element Types and Application to Fluid Dynamics

Carsten Burstedde[1], Donna Calhoun[2], Jose A. Fonseca[1], Johannes Holke[1], Stefan Kollet[3], Michael Lahnert[4], and Miriam Mehl[4]

[1] Institut für Numerische Simulation, Rheinische Friedrich-Wilhelms Universität Bonn, Germany
E-mail: {burstedde,fonseca,holke}@ins.uni-bonn.de

[2] Boise State University, Idaho, USA
E-mail: donnacalhoun@boisestate.edu

[3] Agrosphere (IGB-3), Forschungszentrum Jülich GmbH, Germany
E-mail: s.kollet@fz-juelich.de

[4] Institut für Parallele und Verteilte Systeme (IPVS), Universität Stuttgart, Germany
E-mail: {michael.lahnert,miriam.mehl}@ipvs.uni-stuttgart.de

We discuss the development of parallel algorithms for adaptive mesh refinement (AMR) and their application to large-scale problems in simulating fluid dynamics. Our approach to AMR can be described as using a forest of octrees (or quadtrees in 2D) that are adaptively refined. The storage of elements is distributed in parallel, and fast and scalable algorithms exist for dynamic refinement/coarsening and other important tasks, such as partitioning and the extraction of one layer of off-process (ghost) neighbours.

Our contributions are twofold: (a) We use the `p4est` software as a basis to create numerical applications to simulate the flow of gas in the atmosphere (advection equations), variably saturated subsurface flow (Richards), and the free flow of liquid (Navier-Stokes). (b) In addition to using quadrilateral/cubic elements, we are developing space filling curves and high-level AMR algorithms for triangles and tetrahedra.

We include scalability results and simulation snapshots obtained on the JUQUEEN supercomputer.

1 Adaptive Simulation of Atmospheric Flow

The ForestClaw project provides a software layer between the highly scalable `p4est`[1] library and application solver libraries such as Clawpack[2–4]. This application layer provides the infrastructure for time stepping, dynamic mesh adaption, I/O, coordinate mappings, and communication between logically Cartesian grids stored in the leaves of a forest of octrees. Because ForestClaw is based on `p4est`, we inherit the scalability of its routines for mesh adaptation. In ForestClaw, the smallest parallel unit is a fixed-size grid (typically 32×32), a number of which are distributed in parallel using `p4est`'s space filling curve (SFC) paradigm.

We have now extended ForestClaw with several solver libraries, including two versions of Clawpack (4.6 and 5.0), GeoClaw (for depth averaged geophysical flows) and, recently, Ash3d, developed at the United States Geological Survey (USGS) for modelling the transport and dispersion of volcanic ash in the atmosphere. With the Clawpack solvers, we can solve a wide range of hyperbolic problems in both conservative and non-conservative form,

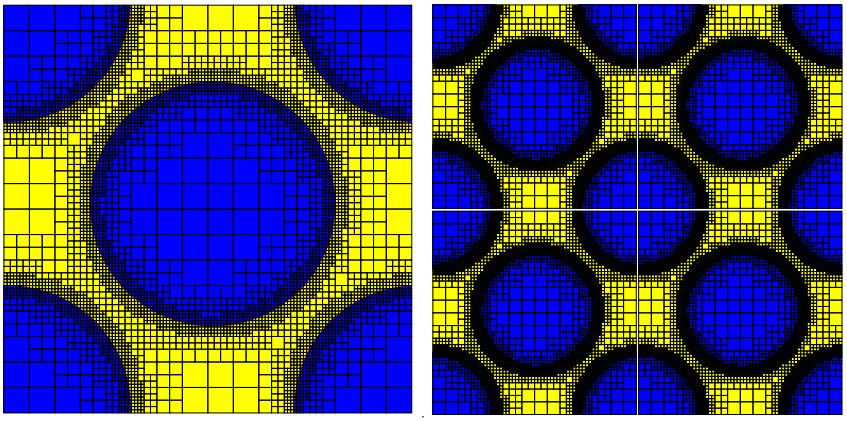

Figure 1. Initial tracer field for the replicated scalar advection problem (left). The mesh is adapted to the tracer field, which is set to 1 inside of one of five disks of radius 0.3 and to 0 outside. We use refinement levels 4 through 7 (grid lines not shown). The domain on the right shows the unit quadtree replicated four times on a $[0, 2] \times [0, 2]$ domain.

including problems from acoustics, gas dynamics, transport, shallow water wave equations and Burgers equation. We have extended these solvers to work on Riemannian surfaces, including the cubed sphere and the torus. These developments are currently established for two-dimensional manifolds, which we are planning to extend to extruded 2.5D and general 3D geometries.

We have developed simple test cases that allow us to examine both weak and strong scaling, beginning on 1 process and scaling up to 65,536 processes. In a basic test, we solve the scalar advection problem on a unit square and advect a disk some distance within this square. We solve this on both uniformly refined and adaptive meshes. To test the weak scaling, we replicate the problem across processes to ensure that each process has the same work load. The numerical code, however, is oblivious to this replication and simply solves a problem with an increased number of disks. Using a relatively small number of grids per process (compared to methods with a single degree of freedom per `p4est` leaf), we observe good scalability up to 16,384 processes; see Figs. 1 and 2.

2 Efficient Simulation of Subsurface Flow

The accurate simulation of variably saturated flow through porous media is a relevant component in understanding the physical processes that govern water resources problems. Such simulations require large scale computations that are enabled by the efficient usage of the latest high performance parallel computing systems. The ParFlow code[5] targets such simulations by computing a numerical solution of the three dimensional Richard's equation[6]. ParFlow is built using MPI distributed parallelism and suitable for large scale problems. Its solver scalability is excellent up to 16,384 processes. Nevertheless, the code has presented issues when trying to scale to higher process counts.

We have audited ParFlow's source code and identified its way of parallelisation of the mesh as the main bottleneck impeding scalability. Essentially, the work load per process

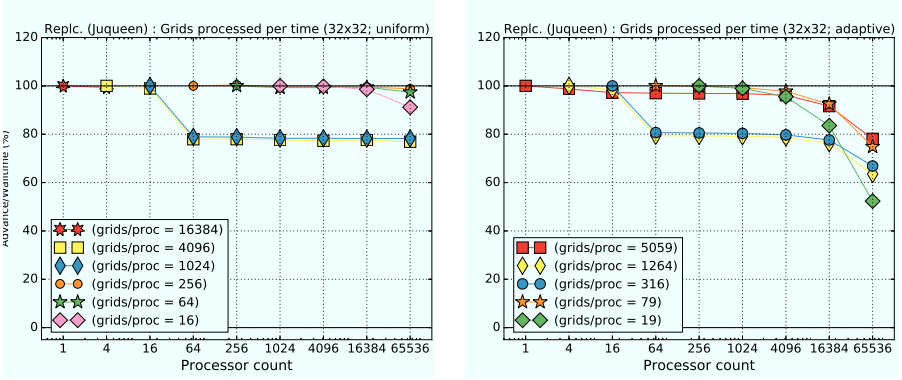

Figure 2. Weak scaling results for the replicated scalar transport problem. Uniformly refined run shown on the left, adaptive run on the right. Each plot shows the efficiency (%) of the particular run, which is computed as the ratio of the number of grids updated per wall time for a simulation run on P processes to the number of grids updated per time on 1 process. The reduction in efficiency going from 16 to 64 processes results from running 1 MPI process per core on 1, 4, and 16 cores to 2 processes per core for 64 and higher process counts, e.g. 32 ranks per node on each JUQUEEN node.

was increasing when more processes where added. Our proposed solution is a reorganisation of ParFlow's mesh subsystem using state-of-the-art partitioning algorithms provided by `p4est`.

We have successfully coupled the ParFlow code with `p4est`, establishing the latter as the mesh management backend of the former and reducing the setup time by a large margin. This modified version of ParFlow is still using uniform meshes, yet it offers a wider range of parallel mesh configurations in comparison with the upstream version of the code. Previously, ParFlow enforced a rectangular partition into subgrids where each process held exactly one such grid, whereas the modified version permits that one process may hold multiple grids. Due to savings in memory usage, the modified version of ParFlow can be executed using higher process counts (MPI ranks) and with improved parallel scalability[7]; see Fig. 3.

3 Adaptive Lattice-Boltzmann Simulations

ESPResSo[8,9] is a versatile software package for performing coarse-grained molecular dynamics (MD) simulations. A molecular ensemble can be subjected to a background flow. To this end, ESPResSo provides an implementation of the lattice-Boltzmann method (LBM)[10,11] based on a uniform Cartesian mesh. The interaction between particles and fluid is realised using a bi-directional force coupling[12]. ESPResSo is parallelised using MPI.

If we need a specific resolution to capture certain aspects in a part of the domain, the number of LBM cells grows cubically when considering larger scenarios. This is an issue for simulating real-world applications with realistic system sizes and time scales. A relevant exemplary simulation scenario is the translocation of biomolecules embedded in an aqueous solution through nanopores. The biomolecules are charged, which leads to the formation of a so-called double-layer. The double-layer is a thin layer on the scale

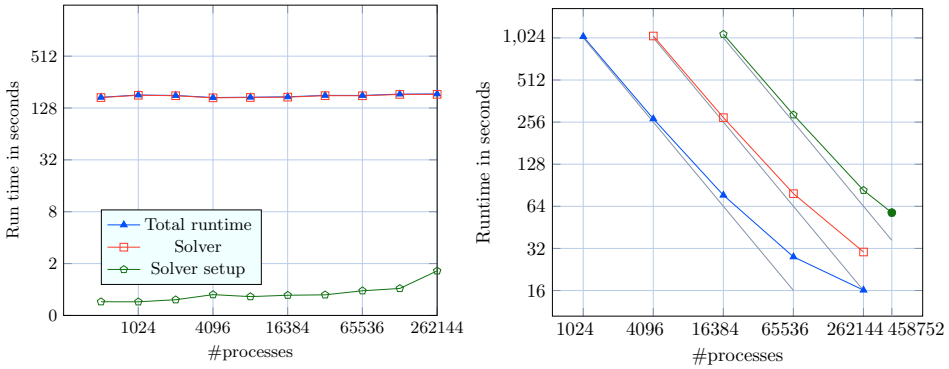

Figure 3. Left: weak scaling timing results of the modified version of ParFlow. Compared to the upstream version, the solver setup executes in negligible time, eliminating a major scalability impediment. Right: strong scaling timing results of our modified version of ParFlow for three different problem sizes (colour coded: blue with 671 million grid points, red with 2.68 billion grid points and green with roughly 10.6 billion grid points) showing good scalability to the full JUQUEEN system. These results granted the modified version of ParFlow membership in the High-Q Club maintained by the JSC.

of one to a few hundred nanometres and of particular interest to physicists, as it is both stable and the only part of the volume that is not net-neutral. The size of the system, however, is on the order of hundreds of micrometers. To mitigate this problem, dynamic spatial adaptivity allows for simulating areas of interest with a high local resolution while reducing the number of unknowns in the remaining domain.

In our collaboration, we have replaced the uniform Cartesian mesh in ESPResSo's LBM algorithm by a tree-structured Cartesian mesh based on `p4est`. Our integration is

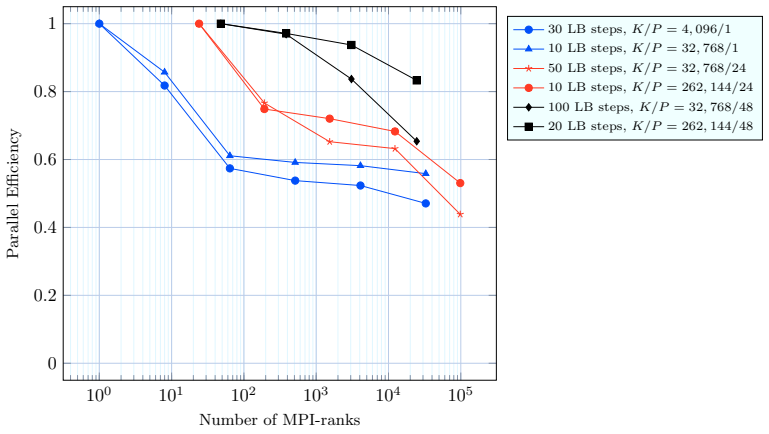

Figure 4. Weak scaling experiments performed on Hazel Hen for a uniform discretisation of a single octree using different refinement levels using different quadrant (K) to process (P) ratios. Blue lines compare to serial execution, red lines to execution on a single fully-occupied node, that is, 24 MPI ranks per node, and black lines to execution on two fully occupied nodes.

minimally invasive and modular to minimise the amount of newly written code.

As a first test, we use a uniform discretisation to reproduce the results of the upstream version of ESPResSo[13]. As can be seen in Fig. 4, the implementation shows promising scaling behaviour on up to 100 k cores in a weak-scaling setting on "Hazel Hen" at the High Performance Computing Center Stuttgart (HLRS). We see a significant drop in parallel efficiency after going from one node to the network, which we hope to mitigate by implementing optimisations like communication hiding, i.e., overlapping communication with computation. We are currently executing this new version on JUQUEEN.

4 Tree-Based Parallel AMR with Simplices

We are developing `t8code`, a collection of algorithms that provide all necessary operations to perform parallel adaptive refinement for hybrid meshes. The `t8code` algorithms are implemented as a software library. We currently support 0D points, 1D lines, 2D quadrilaterals and triangles, and 3D hexahedra and tetrahedra. `t8code` is designed to implement all parallel meshing routines and thus allows the application developer to concentrate on the formulation and solution of the numerical problem.

In order to efficiently manage triangular and tetrahedral elements, we have developed a new SFC for these shapes. The tetrahedral Morton index (TM-index) is motivated by the Morton SFC for cubical elements[14] that computes the SFC index of a quadrilateral/hexahedron via bitwise interleaving of the coordinates of a designated vertex, the anchor node. In addition to the coordinates, we introduce the type of a simplex. A triangle can have type 0 or 1 and a tetrahedron a type from 0 to 5, depending on their orientation in a reference coordinate system. To compute the TM-index, we interleave the coordinates of the anchor

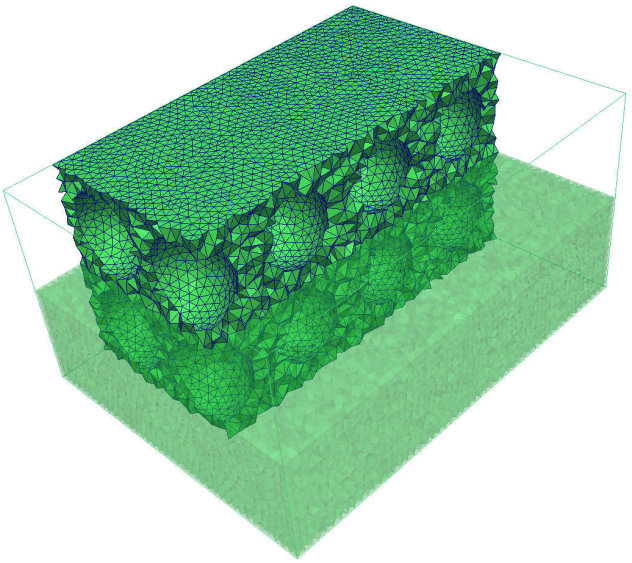

Figure 5. A coarse mesh built from $n_x \times n_y \times n_z$ cubes with one spherical hole each. For this picture we use $n_x = 4$, $n_y = 3$, $n_z = 2$. Each cube is triangulated with approximately 7.575 tetrahedra.

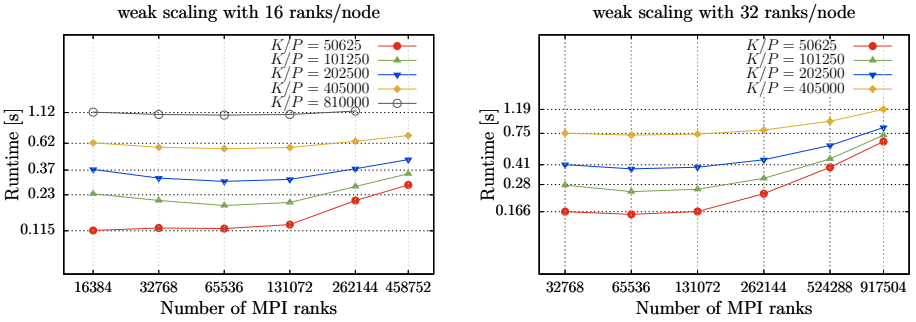

Figure 6. Weak scaling of coarse mesh partitioning for a mesh of disjoint hexahedral trees. Left: 16 ranks per node. Right: 32 ranks per node. We show the run times for the baseline on the y-axis and provide graphs for different ratios between the total number of coarse cells K and MPI processes P. On the left hand side the time for the biggest run, on 458,752 processes, is 0.72 seconds; on the right hand side the time for the biggest run on 917,504 is 1.19 seconds. We obtain efficiencies of $0.62/0.72 = 86\%$ and $0.75/1.19 = 63\%$ compared to the baselines of 16,384/32,768 MPI ranks, respectively (yellow lines). The time for the $P = 262{,}144$ run with 810,000 trees per process (black line) increases from 1.12 to 1.15 seconds, which translates to a weak scaling efficiency of 97.4%.

node with the types of all ancestors of a triangle/tetrahedron. Based on this concept, we have demonstrated near optimal scalability of mesh initialisation and refinement on up to 131 k processes on JUQUEEN[15].

To manage meshes for complex geometries, we connect the roots of the adaptive trees to form a coarse mesh. These trees are then further refined to form the (fine) mesh of elements. We take, for example, a mesh generated with Triangle[16], Tetgen[17], or gmsh[18], store it as an unstructured mesh and use each element as an individual refinement tree (see Fig. 5). To allow for geometries with more than about one million coarse mesh elements, partitioning the coarse mesh among the processes becomes necessary. We have recently developed a new partitioning scheme that minimises the necessary parallel communication while maintaining an optimal load of fine mesh elements[19]; see also Fig. 6.

5 Conclusions

In collaboration with geophysicists, we are planning to extend the simulation of atmospheric and subsurface flow to more realistic example problems. The lattice-Boltzmann simulations are still in an early phase of development, and we are working to complete a functional Navier-Stokes solver on dynamic-adaptive meshes. The development of parallel AMR on hybrid meshes is ongoing, and several follow-up publications are in preparation.

Acknowledgements

B. and F. gratefully acknowledge the financial support by the collaborative research initiative SFB/TR32 “Patterns in Soil-Vegetation-Atmosphere Systems: Monitoring, Modelling, and Data Assimilation,” project D8, funded by the Deutsche Forschungsgemein-

schaft (DFG). B. , F. and H. gratefully acknowledge travel support by the Bonn Hausdorff Centre for Mathematics (HCM) also funded by the DFG.

C. gratefully acknowledges the National Science Foundation's grant No. 1419108.

L. and M. gratefully acknowledge funding provided by the DFG as part of the Collaborative Research Center (SFB) 716 as well as the computing resources provided by High Performance Computing Center Stuttgart (HLRS) and Global Scientific Information and Computing Center (GSIC).

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (`www.gauss-centre.eu`) for funding this project by providing computing time on the GCS Supercomputer JUQUEEN at Jülich Supercomputing Centre (JSC).

References

1. C. Burstedde, L. C. Wilcox, and O. Ghattas, *`p4est`: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees*, SIAM Journal on Scientific Computing, **33**, no. 3, 1103–1133, 2011.
2. R. J. LeVeque, *High-Resolution Conservative Algorithms for Advection in Incompressible Flow*, SIAM Journal on Numerical Analysis, **33**, no. 2, 627–665, 1996.
3. R. J. LeVeque, *Wave propagation algorithms for multi-dimensional hyperbolic systems*, Journal of Computational Physics, **131**, 327–353, 1997.
4. K. T. Mandli, A. J. Ahmadia, M. Berger, D. Calhoun, D. L. George, Y. Hadjimichael, D. I. Ketcheson, G. I. Lemoine, and R. J. LeVeque, *Clawpack: Building an open source ecosystem for solving hyperbolic PDEs*, PeerJ Computer Science, **2**, no. 68, August 2016.
5. S. J. Kollet and R. M. Maxwell, *Integrated surface-groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model*, Advances in Water Resources, **29**, 945–958, 2006.
6. L. A. Richards, *Capillary conduction of liquids through porous media*, Physics, **1**, 318–33, 1931.
7. C. Burstedde, J. A. Fonseca, and S. Kollet, *Enhancing speed and scalability of the ParFlow simulation code*, Computational Geosciencies, accepted for publication, 2017.
8. H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm, *ESPResSo – An Extensible Simulation Package for Research on Soft Matter Systems*, Computer Physics Communications, **174**, no. 9, 704–727, May 2006.
9. A. Arnold, O. Lenz, S. Kesselheim, R. Weeber, F. Fahrenberger, D. Roehm, P. Košovan, and C. Holm, *ESPResSo 3.1: Molecular Dynamics Software for Coarse-Grained Models*, in: Meshfree Methods for Partial Differential Equations VI, M. Griebel and M. A. Schweitzer (Eds.), vol. 89 of *Lecture Notes in Computational Science and Engineering*, Springer, 1–23, September 2012.
10. S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Clarendon Press, 2001.
11. U. D. Schiller, *Thermal fluctuations and boundary conditions in the lattice Boltzmann method*, PhD thesis, Johannes Gutenberg-Universität, Mainz, 2008.
12. B. Dünweg and A. J. C. Ladd, *Lattice boltzmann simulations of soft matter systems*, in: Advanced Computer Simulation Approaches for Soft Matter Sciences III, Springer+Business Media, 89–166, 2009.

13. M. Lahnert, C. Burstedde, C. Holm, M. Mehl, G. Rempfer, and F. Weik, *Towards lattice-Boltzmann on dynamically adaptive grids – minimally-invasive grid exchange in ESPResSo*, in: ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering, M. Papadrakakis, V. Papadopoulos, G. Stefanou, and V. Plevris (Eds.), 1–25, 2016.
14. G. M. Morton, *A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing*, Tech. Rep., IBM Ltd., 1966.
15. C. Burstedde and J. Holke, *A Tetrahedral Space-Filling Curve for Nonconforming Adaptive Meshes*, SIAM Journal on Scientific Computing, **38**, no. 5, C471–C503, 2016.
16. J. R. Shewchuk, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, in: Applied Computational Geometry: Towards Geometric Engineering, M. C. Lin and D. Manocha (Eds.), vol. 1148 of *Lecture Notes in Computer Science*, Springer, 203–222, 1996, from the First ACM Workshop on Applied Computational Geometry.
17. H. Si, *TetGen – A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, Weierstraß Institute for Applied Analysis and Stochastics, Berlin, 2006.
18. C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, International Journal for Numerical Methods in Engineering, **79**, no. 11, 1309–1331, 2009.
19. C. Burstedde and J. Holke, *Coarse mesh partitioning for tree-based AMR*, SIAM Journal on Scientific Computing, accepted for publication, 2017.