

## **OpenMP-based Frameworks for Interoperable Structured Adaptive Methods**

Dinshaw S. Balsara  
National Center for Supercomputing Applications  
and  
Center for Simulation of Advanced Rockets  
University of Illinois at Urbana-Champaign  
[dbalsara@ncsa.uiuc.edu](mailto:dbalsara@ncsa.uiuc.edu)

Charles D. Norton  
National Aeronautics and Space Administration  
Jet Propulsion Laboratory, California Institute of Technology  
High Performance Computing Systems and Applications Group  
and  
Center for Space Mission Information and Software Systems  
[nortonc@bryce.jpl.nasa.gov](mailto:nortonc@bryce.jpl.nasa.gov)

### **Extended Abstract**

Parallel adaptive mesh refinement (AMR) is an important numerical technique that leads to the efficient solution of many physical and engineering problems. While some AMR libraries have been designed, there are many advantages to considering alternative approaches based on language paradigms and standards. Furthermore, it is even more desirable to develop a framework that allows one to easily compose solutions to new problems where solvers and multi-grid methods can interoperate freely. This kind of framework-oriented design is even amenable to the features of computational power grid processing.

This abstract describes recent work where a very general approach to AMR has been devised by combining the best aspects of object-oriented programming using modern aspects of Fortran 90/95, the parallelizing features of OpenMP, and solvers designed for interoperability. The approach combines efficiency, portability, and maintainability for the application scientist, building on results described in Balsara and Norton [2].

Our approach, based entirely on well-defined standards, reduces programming complexity, preserves the investment in existing Fortran-based solvers, and benefits from years of compiler optimization techniques. Our very general approach is scalable, efficient, and complete, integrating emerging trends in high performance computing with the desire to create interoperable frameworks that simplify the modification of simulations to new problems. Our work has been applied to Balsara's RIEMANN framework, see Balsara [1] and references therein.

## Technical Significance

Our approach combines the parallelizing directives of OpenMP with the Fortran 90/95 standard for structured AMR. We have developed efficient, parallel, and scalable methods for performing all of the tasks required. This includes creation and deletion of AMR hierarchies, processing of inter-grid transfers across/within levels, and the solution of these grids anywhere in the hierarchy in a load balanced, and parallel, way.

Introducing object-oriented programming techniques with the new features of Fortran 90/95 (see Decyk, Norton, and Szymanski [4,5]) makes it possible build intricate AMR structures that are efficient. While the array-syntax and dynamic memory management features are most familiar, new features including modules, derived-types (user defined types), use-association, generic interfaces, (safe) pointers, and recursion simplify AMR data structure design.

Fortran 90/95 allows us to create, manage, and delete grid types that are used in the solution process. These grids can overlap and support parent, child, and sibling relationships across AMR levels along with the interpolation of boundary conditions. Collections of grids at a given level in the AMR hierarchy totally cover the regions that need refinement. Fortran 90/95 modules allow one to define specific features that can be applied to the grids, either as a collection or individually. Additional features useful for the solution process can be included in the module as well, and when used in main programs that allows objects to be created. State changes in the objects are limited to the routines that the module makes public. This object-oriented design allows all grid operations to be completely parallelized including the regridding strategies (see Berger and Rigoutsos [3]).

The features of OpenMP that complete our approach are the directives that support data distribution, generation of threads for independent loops, and the `do schedule` clause that allows one to support the “owner-computes” rule for efficient processing. We have also implemented a very efficient load-balancer to ensure that grid objects are created and processed in the hierarchy in a balanced and parallel way. Figure1 briefly shows the use of Fortran 90/95 object abstractions and the directives.

```
type (single_grid), pointer :: this
!$OMP PARALLEL DO SCHEDULE (static, 1)
!$OMP PRIVATE (igrid, this)
!$OMP& SHARED(level, grid_is_active, pointers_to_grids)
  do igrid = 1, max_single_grids
    if (grid_is_active(level, igrid) == 1) then
      this => pointers_to_grids(level, igrid)%sgp
      call wrapper_solver_single_grid (this,...)
    end if
  end do
!$OMP END PARALLEL DO
```

The code segment illustrates how a series of dynamically defined grids is processed at an AMR level, and how a Fortran 90/95 wrapper is used to call an existing Fortran 77 solver. The `do schedule` ensures that processors work on the grids that they themselves own. (Most of the arrays could be replaced with lists, but arrays are demonstrated for simplicity. Module and object definitions have also been omitted in this abstract.)

In AMR, where changes in computational work can only be estimated at run-time, applications require dynamic load balancing over each level in the AMR hierarchy. We have designed a specialized load balancer that is uniquely well suited for AMR applications. The load balancing is iterative, and improves in quality with successive iterations. It utilizes a pairwise exchange of load assigned to available processors such that an exchange causes a maximal reduction in load imbalance between pairs of processors. The computational cost of the algorithm is low, and it can be parallelized easily.

### Framework-Based Design

OpenMP provides the mechanism for bringing together the many components of AMR processing into an interoperable framework suitable for a wide variety of applications. Entire levels in the AMR hierarchy can be built/rebuilt on different subsets of processors in a load balanced fashion. The adaptive process can be tailored for radiative, self-gravitating flow, reactive flow, and particle problems without affecting the parallelization of how grid regions are created and processed based on an OpenMP strategy. Our general load balancer ensures efficient processing and can adapt to the features of computational power grids where varying speeds of the processors and high latencies must be tolerated. Finally, the Berger and Rigoutsos [3] regridding scheme allows one to automatically identify regions for refinement and dynamic mesh generation.

We have experimented with multi-grid methods that interoperate with Newton-Krylov methods leading to a “plug-and-play” algorithm for radiative transfer. That is, once the user has devised the core algorithms for radiative transfer multi-grid and Newton-Krylov are easily adapted for the radiative transfer algorithm. This allows one who knows the core algorithms well to benefit without knowing much about multi-grid or Newton-Krylov. Combining all of these issues, where OpenMP supports parallelism, has been powerful. Indeed, building such a system from scratch would be a substantial effort. Performance results for scalability are shown in Table 1.

Scalability Performance Results for Processing and AMR Hierarchy			
# of Processors	MFlops	Relative Speedup	Cumulative Speedup
1	94.51	---	---
4	362.00	3.83	3.83
8	687.20	1.90	7.27
16	1,393.25	2.02	14.74

32	2,578.70	1.85	27.28
48	3,766.67	1.46	39.85
64	4,885.72	1.30	51.70
96	7,612.74	1.56	80.55
128	9,978.20	1.31	105.57
192	15,002.40	1.50	158.35
256	17,317.65	1.15	182.10

## References

1. D. Balsara. Linearized Formulation of the Riemann Problem for Radiation Magnetohydrodynamics. *J. Quant. Spectroscopy and Radiation Transfer*, 62:167-189, 1999.
2. D. Balsara and C. D. Norton. Highly Parallel Structured Adaptive Mesh Refinement Using Parallel Language-Based Approaches. *Parallel Computing*, to appear 2000.
3. M. Berger and I. Rigoutsos. An Algorithm for Point Clustering and Grid Generation. *IEEE Trans. on System, Man, and Cybernetics*, 21:61-75, 1991.
4. V. K. Decyk, C. D. Norton, and B. K. Szymanski. How to Express C++ Concepts in Fortran 90. *Scientific Programming*, 6(4):363-390, Winter 1997. IOS Press.
5. C. D. Norton. Object-Oriented Scientific Programming with Fortran 90. In J. Schaeffer, editor, *High Performance Computing Systems and Applications*, pages 47-58, Kluwer Academic Publishers, 1998.