Overview of the PYRAMID Parallel AMR Library

Charles D. Norton, John Z. Lou, and Thomas A. Cwik
Jet Propulsion Laboratory
California Institute of Technology MS 169-315
4800 Oak Grove Drive, Pasadena, CA 91109-8099

Abstract— **PYRAMID is a Fortran 90 based software library designed to simplify unstructured adaptive refinement of large finite element meshes on parallel computers and clusters. Developed under the Computational Technologies (formerly known as the Earth and Space Sciences) Project, we will present an overview of the numerous features and advanced software engineering technologies employed in this library. This includes the object-based software architecture, automatic mesh quality control, element refinement techniques, mesh migration, load balancing, partitioning, and visualization. The impact of advances in software modernization, cluster computing, and high performance networking for efficient support for automatic adaptive refinement will also be described through examples drawn from a variety of application meshes.**

Keywords— **AMR, parallel, cluster computing, Fortran 90.**

I. Introduction

PARALLEL adaptive methods support the solution of complex problems using grid-based techniques. Software development for unstructured adaptive mesh refinement focuses on simplifying how the user interacts with the grid for problems that exhibit complex geometry. Research on how this is best achieved is on going since meshes can be adaptively refined, partitioned, and load balanced using a variety of approaches.

Our library supports automatic mesh quality control ensuring that elements with poor aspect ratios are not created. This is achieved by actively redefining how an element is refined to prevent the creation of narrowly shaped triangles and tetrahedrons. The ParMetis graph partitioning software, from the University of Minnesota, is used with our mesh migration algorithms to load balance the adaptively refined mesh [1]. Our development in Fortran 90/95 allows for object-based design where a mesh can be created and manipulated using high-level library commands. A complete, minimal, PYRAMID program is shown in Fig. 1.

Many of these routine accept Fortran 90/95 optional arguments that allow for additional control over the actions taken. When these options are not specified a reasonable default action is taken. Furthermore, for the most sophisticated routines, keyword arguments may optionally be specified to visually remind the user how various arguments are used.

The new features of Fortran 90/95 allow for complex data structures to be created, and used, while supporting encapsulation of related concepts in modules. Use of modules make the data and routines they contain available to program units and they can interact to provide increased functionality. Features such as smart pointers, generic interfaces, whole and array subset operations, and dynamic storage provide the modern software principles required for this work. This Fortran-based approach will simplify the union of adaptive meshing with existing Fortran-based solvers. This has resulted in a following within NASA, and elsewhere, regarding our progress with the library.

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

```
program pyramid_example
use pyramid_module
implicit none
type (mesh), dimension(2) :: meshes
  call PAMR_INIT()
  call PAMR_CREATE_INCORE(meshes(1),"mesh_data")
  call PAMR_REPARTITION(meshes(1))
  do i = 1, refinement_depth
    call PAMR_ERROR_EST(meshes(1), meshes(2))
    call PAMR_LOGICAL_AMR(meshes(1))
    call PAMR_REPARTITION(meshes(1))
    call PAMR_PHYSICAL_AMR(meshes(1), meshes(2))
  end do
  call PAMR_VISUALIZE(meshes(2),"mesh.plt")
  call PAMR_FINALIZE( mpi_active = .true.)
end program pyramid_example
```

Fig. 1. A minimal PYRAMID program for adaptive mesh refinement given the mesh and a per-element error indicator.

Fig. 2 shows part of an artery segment that has been adaptively refined, load balanced and repartitioned with mesh migration using PYRAMID. (The original geometry is due to Taylor et. al [2] and the initial mesh was generated by Flaherty and Shepard's group at Rensselaer [3]. The initial mesh contains over 1 million elements, but this number grow significantly under adaptive refinement.

In this paper we briefly describe the technology and capabilities of the PYRAMID library. Much of its current functionality and features have been described at length in other publications, most recently in [4], [5]. In addition to this overview, however, we will also introduce related work where PYRAMID is playing a role in understanding the effectiveness of the Common Component Architecture (CCA) Forum's tools for component-based development of software. This is another ESTO/CT project where PYRAMID is a testbed application to examine the effectiveness of the proposed tools, the user experience using the tools,

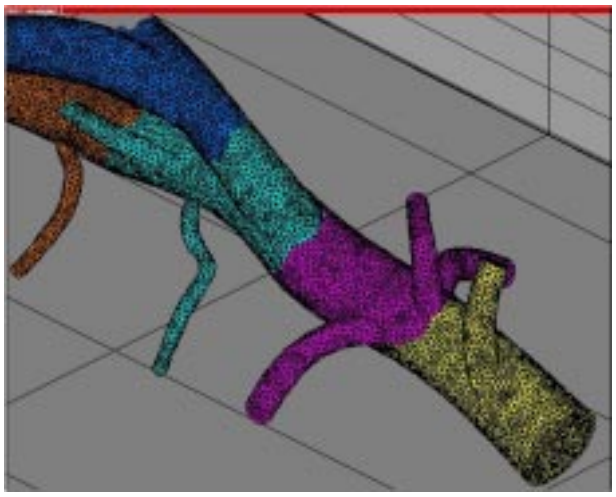

Fig. 2. Repartitioning and migration of artery mesh segment.

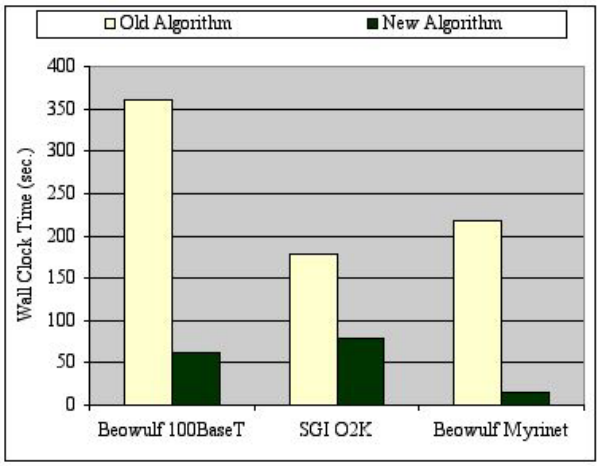

Fig. 3. Migration performance on 8 processors showing improvements due to network inspired message passing algorithm modifications.

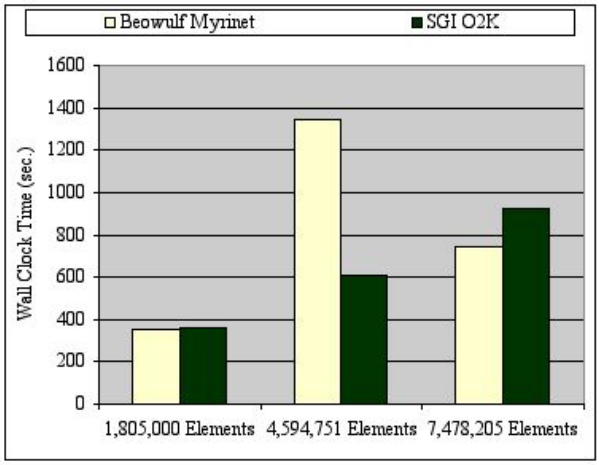

Fig. 4. Performance variation for adaptive refinement of artery mesh on 50 processors.

and the performance implications.

II. Development Activities

The library contains a growing list of commands that support adaptive refinement capabilities in two and three dimensions. These include object-oriented access to the sophisticated mesh data structure and useful mathematical routines as built-in features. Indeed, these are the among the most attractive features to users that collaborate on our software development as the intricacies of the mesh data structure are hidden behind well defined interfaces. The library runs well on traditional computer systems and cluster computers as well. An important consideration for clusters, however, is that AMR is very communication intensive so traditional 100baseT Ethernet networks can severely impact performance.

In this work we have explored the use of new gigabit networking technology, such as Myrinet [6], as a means to improve performance. While such networks are substantially faster than Ethernet for point-to-point communication (Myrinet 2000 is a 2 Gbit/s network) when many processors communicate in an irregular fashion the performance drops significantly.

To address this problem we developed communication algorithms that take better advantage of the properties of such networks in a cluster computing environment as shown in Fig. 3. For unstructured AMR problems where processors may need to communicate in an irregular, but largely predictable, fashion we found that sending a minimum number of messages to precise destinations was not favorable for CLOS-based high degree networks. Instead, we introduced a much more scalable tree-based pair-wise exchange method that will work on any number of processors. Our algorithm is special in that it guarantees that a minimal number of exchanges will be performed meaning that it can determine if a processor has already received the data it needs and skip communication operations as necessary. Although the volume of data sent is larger this is addressed since the method also determines when certain data are no longer needed in the message stream removing it from future communication stages. Fig. 3 shows that this method brings substantial improvements for networks on various systems as demonstrated with an 8 processor earthquake mesh.

Performance results for unstructured AMR can be hard to characterize since there are many contributing factors. The current partitioning of the mesh, the region to be adaptively, refined, the load balancing algorithm, mesh migration scheme, and the adaptive refinement process all play a role. Fig. 4 shows one such example where our artery mesh of Fig. 2 goes though successive adaptive refinements.

When comparing a cluster computer running Myrinet to and SGI Origin 2K system we see that for the 4.5 million element case the cluster performs worse than the SGI O2K. Although migration and adaptive refinement time were quite high in this case the time was dominated by adaptive refinement and the impact was most likely due to a difference in the partitioning. Part of our current work involves examining additional partitioning strategies to explore this matter further.

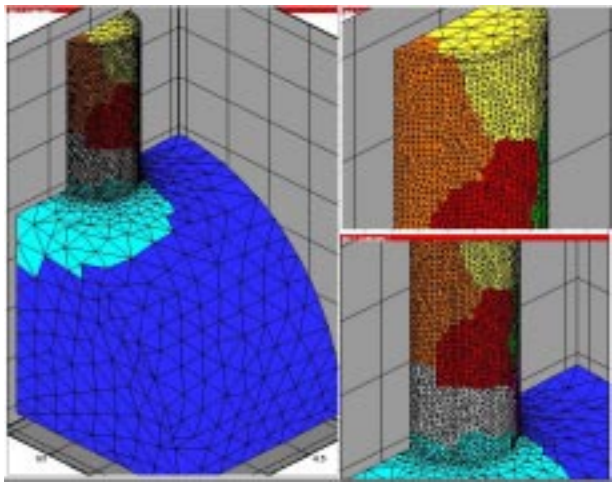

Fig. 5. Repartitioning and migration of muzzle brake segment.

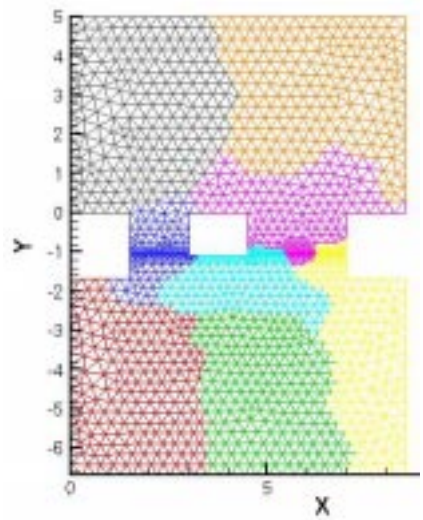

Fig. 6. Active device modeling mesh.

Fig. 5 shows a muzzle-brake mesh, used in shock simulations, that we have adaptively refined in PYRAMID. Our library has support software to convert mesh formats provided by others, as in this case, into the format that PYRAMID requires. Although we have used our software to generate new meshes from existing ones we do not generate meshes from initial solid models as is typically done in the mesh generation community. We have been fortunate to have a number of collaborators in the mesh generation community that provide meshes to aid in the testing and development of our software.

Recently the 2D version of this software has been used as a testbed to evaluate features of the Common Component Architecture Forum's tool to support development of frameworks for scientific applications [7]. The library and main program were converted into components that interact through the framework system. The example problem was drawn from adaptive refinement for an active device model [8] shown in Fig. 6. We are in the process of analyzing the effort involved in converting the library to a component as well as the performance impact/overhead the component introduces.

Components have the promise of reducing the effort involved in community code development while simplifying the ability of scientists to interface their software to other applications. As AMR is an important feature that can benefit scientific codes introducing standards that simplify how new and existing software can interact with codes that support adaptive methods is very beneficial.

III. Conclusions

Progress in developing the basic infrastructure for parallel unstructured AMR has been strong so our attention is now focused on portability issues and support for large 3D differential equation solvers. This includes adding user-controllable boundary zones, interpolation methods among mesh levels, and defining straightforward ways to include error-estimation routines to drive the adaptive process.

We have tried to stay ahead of solver technology allowing work in that community to help influence development issues in our library. We have established a number of important collaborations with leaders in tetrahedral-based finite element solvers and are currently enhancing the library to include features to support their needs. To ensure our design philosophy is preserved we add new functionality carefully. Anyone interested in contributing ideas is welcome to contact the authors.

Acknowledgments

We acknowledge the on-going support of the ESTO/CT Program and numerous others that have contributed their expertise to the development of this software. This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] G. Karypis, K. Schloegel, and V. Kumar, "ParMetis: Parallel Graph Partitioning and Sparse Matrix Ordering Library," Tech. Rep., Dept. of Computer Science, U. Minnesota, 1997.

[2] Charles A. Taylor, Thomas J. R. Hugues, and Christopher K. Zairns, "Finite Element Modeling of Blood Flow in Arteries," To appear, *Comp. Meth. in Appl. Mech. and Engng.*, 1999.

[3] Joseph E. Flaherty and James D. Teresco, "Software for Parallel Adaptive Computation," in *Proc. 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation*, Michel Deville and Robert Owens, Eds., Lausanne, 2000, IMACS, Paper 174–6.

[4] C. D. Norton and T. A. Cwik, "Early Experiences with the Myricom 2000 Switch on an SMP Beowulf Class Cluster for Unstructured Adaptive Meshing," in *2001 IEEE International Conference Conference on Cluster Computing*, D. Katz and et. al. T. Sterling, Eds., Newport Beach, CA, October 8-11 2001, IEEE Task Force on Cluster Computing, IEEE Computer Society.

[5] C. D. Norton and T. A. Cwik, "Parallel Unstructured AMR and Gigabit Networking for Beowulf-Class Clusters," *Lecture Notes in Computer Science*, vol. 2328, 2002.

[6] Myricom, Inc, *Myricom Creators of Myrinet*, 2001, http://www.myri.com.

[7] *Common Component Architecture Forum*, 2002, http://www.cca-forum.org.

[8] T. A. Cwik, R. Coccioli, G. Wilkins, J. Z. Lou, and C. D. Norton, "Multi-Scale Meshes for Finite Element and Finite Volume Methods: Active Device and Guided-Wave Modeling," in *Proc. AP2000 Millennium Meeting*, Davos, Switzerland, April 2000.