LA-UR 97-2877

Approved for public release; distribution is unlimited

Title: AMR++: A Design for Parallel Object-Oriented Adaptive Mesh Refinement

CONF-9703118--

Author(s): Dan Quinlan

Submitted to: Proc. of the IMA Workshop on Structured Adaptive Mesh Refinement
Minneapolis, Minnesota
March 1997

Los Alamos
National Laboratory

DISCLAIMER

AMR++: A DESIGN FOR PARALLEL OBJECT-ORIENTED ADAPTIVE MESH REFINEMENT

DAN QUINLAN

SCIENTIFIC COMPUTING GROUP
COMPUTING INFORMATION AND COMMUNICATIONS DIVISION
LOS ALAMOS, NM, USA, 87545
HTTP://WWW.C3.LANL.GOV/~DQUINLAN
DQUINLAN@LANL.GOV

Abstract. Adaptive mesh refinement computations are complicated by their dynamic nature. In the serial environment they require substantial infrastructures to support the regridding processes, intergrid operations, and local bookkeeping of positions of grids relative to one another. In the parallel environment the dynamic behavior is more problematic because it requires dynamic distribution support and load balancing. Parallel AMR is further complicated by the substantial task parallelism, in addition to the obvious data parallelism, this task parallelism requires additional infrastructure to support efficiently [6]. The degree of parallelism is typically dependent upon the algorithms in use and the equations being solved. Different algorithms have significant compromises between computation and communication. Substantial research work is often required to define efficient methods and suitable infrastructure. The purpose of this paper is to introduce AMR++ as an object-oriented library which forms a part of the OVERTURE framework, a much larger object-oriented numerical framework developed and supported at Los Alamos National Laboratory and distributed on the Web for the last several years.

1. Introduction. Adaptive mesh refinement is a numerical technique for locally tailoring the resolution of computational grids. AMR permits the addition of finer grids to the global computational grid in an adaptive way so as to permit locally more accurate computations or the removal of global error introduced by local singularities. AMR as a numerical technique, is largely independent of the equations being solved, though numerous numerical and algorithmic issues are involved and are the subject of significant research. Unfortunately, AMR is not common place due largely to its inherent complexity. While numerical issues concerning the best ways to handle refinement interfaces are extremely important, the development of parallel AMR has not become common place because its complexity has far out striped improvements in the software productivity required for its implementation and use. A special role for libraries specific to AMR seems obvious, so that the complexity of the AMR process can be properly encapsulated [3]. The development of parallel AMR is of course even more problematic and tedious because of the dynamic distribution of data, and nonregular communication patterns that adaptive solvers necessitate. The development of elliptic solvers for adaptively refined grids is still more complex because of the global nature of the elliptic solution process in the more efficient elliptic methods (e.g. multigrid solvers). AMR represents an impressive opportunity to demonstrate a potentially significant simplifying object-oriented mechanism to build adaptive mesh refinement applications. The purpose of this paper is to introduce AMR++ as a library within OVERTURE[2].

2. Object-Oriented Frameworks. AMR++ is a library contained within OVERTURE an object-oriented framework for the numerical solution of partial differential equations developed at Los Alamos National Laboratory. AMR++ supports the development of structured grid based AMR applications. AMR++ uses objects from the OVERTURE framework to provide adaptive mesh refinement with complex geometry. OVERTURE uses A++/P++[5], a C++ array class library for serial and parallel environments, to provide array class support and abstract issues of parallelism within distributed memory environments. A++/P++ provides an efficient array language like

support for the development of numerical software. A++/P++ has been distributed for 3 years. OVERTURE has been distributed for the last two years. Both are aggressively supported and improved, recent improvements include expression templates as an optional feature. Both A++/P++ and OVERTURE are supported on a wide number of architectures and available via the Web.

- The A++/P++ class library
  A++P++ is a portable C++ array class library for serial/parallel computers. A++ represents the serial array class and P++ represents the parallel array class, both have an identical interface. It provides a simple syntax for the creation and manipulation of arrays, similar to FORTRAN 90. In the serial environment the arrays represent contiguous storage and can be shared with other FORTRAN or C applications and provide for indexing, indirect-addressing, complex views (subarrays), dynamic manipulation, etc. In the parallel environment the arrays form distributed objects whose distribution can readily be specified and manipulated dynamically. Within the array class overloaded operators define a conventional array syntax with the usual array semantics of FORTRAN 90 like operations. In the parallel environment the operators handle communication, as required, for correct operation. A++/P++ optionally uses expression templates as a mechanism for better performance, it is implemented as an optional feature so that portability and compile time performance can be readily maintained.
- The OVERTURE framework
  OVERTURE is an object-oriented numerical framework which derives from the A++/P++ class library to define mechanisms for much more complex computations than the array objects themselves readily permit by themselves. In addition to support for rectangular grid computations such computations include complex geometry, adaptive mesh refinement (AMR), and overlapping grid computations which permit modeling of more complex geometries than single grid techniques would allow. A++P++ forms a part of the OVERTURE framework, additional libraries within OVERTURE include: MLB[1] Load Balancing for structured grids, Finite Difference Operator libraries, Finite Volume Operator libraries, Mapping libraries for defining analytic and discrete mappings, Grid libraries for defining computational grids and collections of grid (using mapping objects), and a Grid Function Library for defining data on grid objects (derived from A++/P++ array objects). OVERTURE runs in parallel because its principle objects use A++ arrays and in the parallel environment use the distributed P++ array objects, having the identical interface.

3. An Object-Oriented Approach to Parallel AMR. The development of parallel AMR is distinguished by its lack of general availability and use within sophisticated applications. The development of such applications, especially where complex geometries are involved requires significant expertise that is not generally available except within large research groups. The development of complete AMR applications most often requires significant specialized expertise, however the use of libraries permits the encapsulation of the many details related to AMR, this is especially the case for parallel AMR. Even such parallel AMR libraries must rely upon multiple lower level libraries to avoid the mixing of the many levels of abstraction that are represented. The result is a hierarchy of libraries such as those within the object-oriented OVERTURE framework. The use of object-oriented techniques provides a simplifying means to make AMR accessible to users unwilling to develop the required support infrastructure for AMR.

The principle purpose of AMR++ is to provide mechanisms for:

1. management of the adaptive grid within time-dependent problems
2. the flexible control over addition and deletion of local refinement
3. the distribution of the adaptive grid (using MLB load balancer within OVERTURE)
4. the definition of transfer operators

Using these mechanisms, an additional purpose of AMR++ is to provide a simple interface that encapsulates many conventional practices within AMR, yet provides full exposure to the underlying mechanisms. AMR++ takes advantage of a templated design to accomplish some of these goals. C++ provides a templating mechanism to define default object methods, while permitting the use to use template specialization as a means for the user to specialize the behavior of AMR++ for a particular application, often using the underlying mechanisms. Thus AMR++ is an extensible library and readily adapted to multiple application domains.

An express goal of AMR++ is the same level of portability as A++/P++ and OVERTURE, so the use of templates has been restricted in initial work to the limits of the most common C++ compiler. Innovative solutions to many issues are possible if this condition were relaxed.

4. Design of AMR++. AMR++ is designed to make simple AMR applications trivial to build, but also permit the specialization of AMR++ for more complex applications, which we can not readily anticipate. To do this we define many specific instances to support the more common objectives within an AMR application. We build these from general mechanisms which we make available to the user as well. These general mechanisms permit greater specialization of the AMR++ library to particular applications areas. For example, we provide a library of interpolation and projection operators but make available the base classes from which each is derived so that the user can readily have a mechanism to define intergrid transfer operations that are specific to a particular application or application area. Thus AMR++ can be readily built on top of to define more specialized AMR mechanism to a particular application area, such as, ocean modeling, multi-material modeling, hyperbolic flows, elliptic solvers, etc.

AMR++ is divided into four separate libraries (other libraries are used for infrastructure support and include: OVERTURE, A++/P++, and BoxLib):

1. LevelTransferLib
   Intergrid transfer operator library - This library defines multiple interpolation and projection operators for use within AMR applications and additionally defines mechanisms that a user's application may use to define customized interpolation and projection operators.
2. ParentChildSiblingLib
   Intergrid datastructure support library - This library defines objects which serve as descriptors for the definition of the relationships between refinement levels and connected collections of grids at the same refinement level. This library also defines the mechanisms for the user to build such descriptors specialized to the user's application.
3. RegridLib
   Adaptive regridding support library - AMR++ regridding is designed to greatly simplify the extremely tedious job of taking information about the approximated error (using mechanisms defined within AMR++ or by the user) and generating grids to support the local refinement process. The regrid process leverages isolated parts of work done by Lawrence Berkeley Laboratory to handle aspects of the regrid process.
4. SolverLib
   This library contains the principle interface classes for AMR++. Applications can be developed to use the above libraries directly, but the Solver library represents a simplifying templated interface. These interface classes permit the use of a single grid solver (as templated parameter) within AMR++ to build an adaptive mesh refinment application. The only additional requirement is a user defined adaptive grid solver to define the outer-most iterations upon the collection of refinement levels. In each case the users solver class is derived from AMR++ classes which then define the default behavior and encapsulate key mechanisms.

The use of AMR++ for parallel applications is made possible because AMR++ uses OVERTURE classes internally which in the parallel environment uses P++, a parallel

array class library. Since AMR++ uses OVERTURE classes, it logically sits at the top of the OVERTURE framework, and can use the features within OVERTURE such as mappings, grids, and grid functions; to provide adaptive mesh refinement within the context of complex geometry, and eventually, even overlapping grids[4].

5. Parallel issues in AMR++/OVERTURE. The parallel issues in AMR are the subject of special attention within AMR++. OVERTURE provides simple mechanisms for the specification of the distribution of OVERTURE grid data (via P++ distribution mechanisms). These distribution mechanisms are used by AMR++ and the Multilevel Load Balancer (MLB) to define the distribution of the adaptive grid. Different AMR algorithms place different requirements upon the distribution of the adaptive grid in the parallel environment. AMR++ supports a couple of different distribution mechanisms specialized for the most common sorts of adaptive mesh refinement solution methods. However, the underlying mechanisms for defining new distributions are readily available to the user as well, this allows for the tailoring of AMR++ parallel support for different sorts of applications or for research on distribution mechanisms more generally.

6. Serial and Parallel Results. Currently only some simple AMR++ applications are running in parallel. More complex applications using AMR++ are in development and being used to evaluate robustness and performance in the serial and parallel environments.

REFERENCES

[1] Quinlan, D., Berndt, M. *MLB: Multilevel Load Balancing for Structured Grid Applications*, Published in Preceedings of the SIAM Parallel Conference, Minneapolis, MN. March, 1997

[2] Brown, D., Chesshire, G., Henshaw, W., and Quinlan, D., *OVERTURE: An Object-Oriented Software System for Solving Partial Differential Equations in Serial and Parallel Environments*, Published in Preceedings of the SIAM Parallel Conference, Minneapolis, MN. March, 1997

[3] Balsara, D., Quinlan, D., *Parallel Object-Oriented Adaptive Mesh Refinement*, Published in Preceedings of the SIAM Parallel Conference, Minneapolis, MN. March, 1997

[4] K. D. Brislawn, D. L. Brown, G. Chesshire, and J. S. Saltzman, *Adaptive composite overlapping grids for hyperbolic conservation laws*, LANL unclassified report 95-257, Los Alamos National Laboratory, 1995.

[5] M. Lemke, D. Quinlan, *P++, a C++ Virtual Shared Grids Based Programming Environment for Architecture-Independent Development of Structured Grid Applications*, CONPAR/VAPP V, September 1992, Lyon, France; published in *Lecture Notes in Computer Science*, Springer Verlag, September 1992.

[6] R. Parsons, D. Quinlan, *Run-time Recognition of Task Parallelism within the P++ Parallel Array Class Library*, Proceedings of the Conference on Parallel Scalable Libraries, Mississippi State, 1993.