MENUS-PGG : A MAPPING ENVIRONMENT FOR UNSTRUCTURED AND STRUCTURED NUMERICAL PARALLEL GRID GENERATION

NIKOS CHRISOCHOIDES, GEOFFREY FOX, AND JOE THOMPSON

ABSTRACT. MENUS-PGG is a problem solving environment (PSE) for developing parallel algorithms that generate structured and unstructured static and adaptive grids (or meshes) required for the implementation of scalable parallel partial differential equation (PDE) solvers based on domain decomposition methods. Whereas the first generation PSEs for the numerical solution of PDEs on distributed memory multiprocessor systems are based on the data mapping of sequentially generated grids and support only the data parallel programming model, MENUS-PGG generates and maintains grids on the processors of parallel/distributed systems and combines the most valuable aspects of the data parallel programming model with the flexibility of the task parallel programming model. MENUS-PGG assumes a machine model that consists of homogeneous and heterogeneous clusters of processors operating in a distributed address space implemented on remote memory modules via message passing through a high-speed interconnection network. The major contribution of MENUS-PGG should be the reduction of the pre-processing overhead required by the data parallel PDE solvers and the efficient maintenance of the distributed data structures that support h, p, and hp-refinements. We present preliminary results indicating that the parallel grid generation results in a substantial reduction of the pre-processing overhead needed for the solution of the data mapping problem.

1. INTRODUCTION

Parallel computing and specifically high performance software for scientific computing will be made more attractive to scientists and engineers if these systems will provide support for all aspects of a simulation : grid generation, PDE solvers, adaptive refinement, and I/O. While a fair amount of work has been carried out for field solvers limited or none made towards parallel algorithms and software modules for numerical grid generation. Our aim in this paper is to describe a software system for the parallel numerical generation of structured and unstructured static and adaptive grids.

Besides the geometric modeling and grid visualization we identify and focus on four additional stages for parallel numerical grid generation. The first stage requires

The research of the first author was supported by the Alex G. Nason Prize Award.
AMS(MOS) key words : parallel, numerical, grid, software.

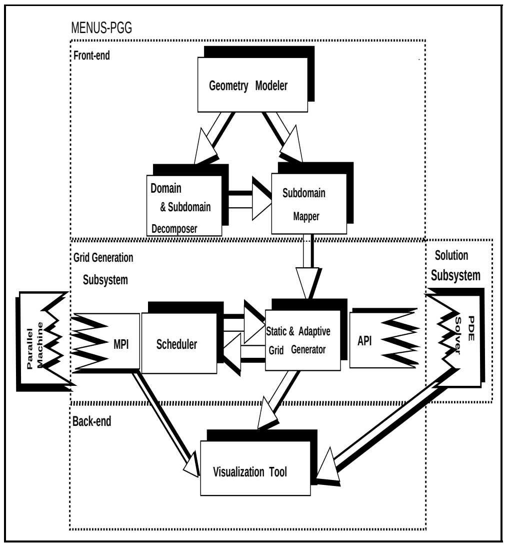
the decomposition of the continuous domain into a set of non-overlapping subdomains with simpler shape (e.g. four- or six-sided polygons for 2- or 3-dimensional spaces). The second stage requires the partitioning and placement of the subdomains on the processors of the target machine so that certain criteria are satisfied. The third stage requires independent grid generation on the subdomains and sometimes maintenance of grid conformity and continuity on the interfaces of the subdomains. Finally, the fourth stage requires the run-time migration of grids among the processors so that processors workload (computation and communication) is balanced and local communication and synchronization among the processors are minimized during the parallel execution of PDE computation. The MENUS-PGG design is based on these stages. In the rest of the paper we describe the MENUS-PGG software infrastructure and the individual software components that correspond to the above stages.

2. Software Infrastructure

The design objective of MENUS-PGG is to provide a uniform environment and the basic software components to implement, analyze and test scalable algorithms for parallel grid generation. The software infrastructure of MENUS-PGG consists of three major subsystems, namely, the *front-end subsystem*, corresponding to the domain definition and the first two stages described above, the *grid generation subsystem*, corresponding to the last two stages, and the *back-end subsystem*, corresponding to the visualization of grid and performance data. Next we describe the software modules of the subsystems and their functionalities. Fig. 1 depicts the software architecture of MENUS-PGG and the interaction among these modules.

Software Module	**Computational Phase / Functionality**
Front-end :	
Geometry Modeler	Specify and discretize the boundary curves or surfaces of the domain.
Domain and Subdomain Decomposer	Decompose the domain into four or six-sided subdomains and discretize internal subdomain boundaries.
Subdomain Mapper	Map subdomain to the processors of the target machine.
Grid Generation Subsystem :	
Static & Adaptive Grid Generator	Concurrently generate grids on the subdomains and maintain grid conformity - if it is necessary.
Scheduler	Maintain computation and communication load balance of the processors during the PDE computation.
Back-end :	
Visualization Tool	Visualize grids, solution and performance data.

The **geometry modeler** and the **3D visualization tool** are provided by the National Grid Project which is under development at NSF Engineering Research Center for Computational Field Simulation, ERC-CFS, at Mississippi State University [14]. The geometry modeler constructs the boundary surface (or curve) of the region we want to discretize. The boundary representation of the region can be received as patches from an external CAD system or it can be computed by an



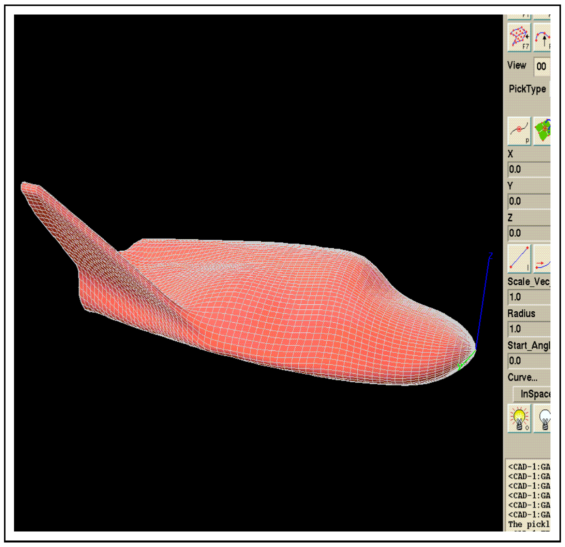


FIGURE 1. MENUS-PGG software architecture and NGP's Geometry Modeler.

internal available CAD system. An instance of the geometry modeler and the 3D visualization tool are depicted in Fig. 1. The **3D visualization tool** is a user friendly interface with functionalities allowing graphical movement of block structures (for the graphical construction of grid topology) as well as scaling, rotation, and transformation of surface segments.

The **domain and subdomain decomposer** is the module that decomposes the given domain into four or six-sided subdomains. Such decompositions can be achieved either interactively using graphical user interface tools [14] or automatically using computational geometry tools, like the Medial Axis Transformations [12]. The domain decomposer is essential for the conventional transfinite mapping techniques used for the generation of structured grids. The composite block structures generated by this module can also be used for parallel unstructured grids.

The **subdomain mapper** is based on a library of Mapping Templates for Load Balancing (MTLB) [4]. The MTLB library of templates provides algorithmic and software infrastructure for the mapping of the subdomains to the processors of the target parallel machine. Such mappings often involve the solution of an intractable combinatorial optimization problem that optimizes a number of criteria, like the minimization of the number of grid points that separate the subdomains residing on different processors, the proper distribution of grids so that the computation and communication work load of processors is balanced, or the appropriate placement of subgrids so that network contention is minimized. Although many heuristics for finding good suboptimal mapping solutions have been proposed in the literature (see in [6], [1], [7], [11], [8] and [2]) there is no general-purpose software that can be used independently of the specific characteristics and data structures of the application. The MTLB library aims to provide a common software framework for the implementation and evaluation of existing and new heuristics for the data mapping problem.

The **static and adaptive grid generator** will provide the algorithmic and software infrastructure for the concurrent grid generation of the subdomains and for the maintenance of grid continuity and conformity on the interfaces. Grids (or meshes) are classified into *structured grids*, formed by intersecting grid lines, and *unstructured grids or meshes*, formed by first creating all the node points and then connecting the nodes to form "best" possible triangles. Another classification of grids is based on their static or dynamic evolution during the PDE computation. Grids that remain the same throughout the PDE computations are called static and grids that evolve according to the behavior of predefined error estimators are called dynamic or adaptive. Fig. 2 depicts a taxonomy of the five different methods and types of grids that are under development within MENUS-PGG environment and they are : (1) parallel structured grids for 2 and 3-dimensional complex domains based on composite block structures [15] and (2) parallel unstructured grids using Delaunay triangulation, (3) parallel adaptive semi-structured grids based on a combination of nested structured grids and local equidistribution approaches (4) moving structured grids and (5) adaptive unstructured grids. A detailed discussion of the individual tools and issues related to parallel numerical grid generation is out of the scope of this paper and appears elsewhere [3], [4].

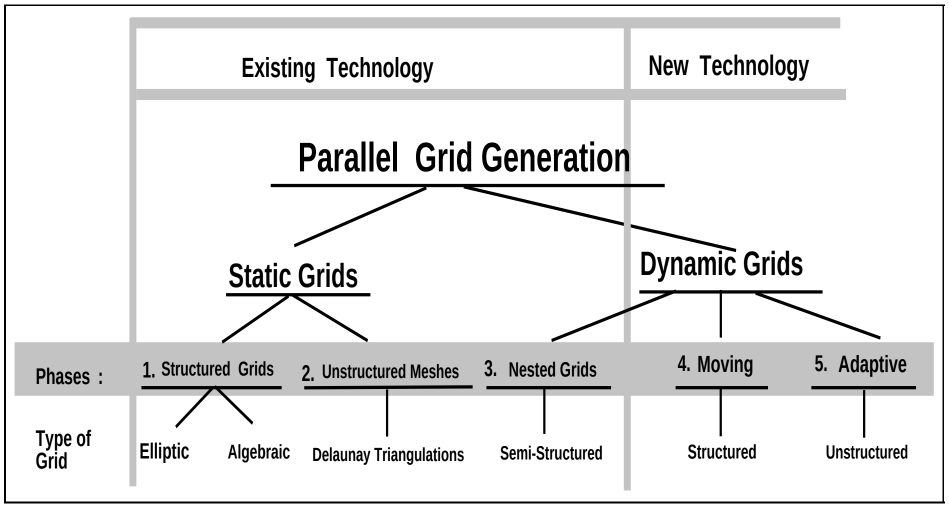


FIGURE 2. Taxonomy of grid types to be generated by MENUS-PGG.

The **scheduler** is a machine independent library of communication and synchronization routines required for the implementation of the computational engine on distributed memory MIMD/SIMD machines using the data and task-parallel message passing programming model. The portability of the computational engine will be guaranteed by developing the scheduler on the Message Passing Interface standard (MPI) defined recently [10]. A run-time support system suitable for parallel adaptive grid generation computations is the major software component of the scheduler. The run-time support system is limited to a very small set of parallel adaptive PDE computations. The system is organized in four layers (from the lower to higher layer) : (0) Network Interface Layer (interrupt and polling driven), (1) Multithread priority based scheduling layer (2) Thread Scheduling Layer (scheduling algorithms for different pairs of grid/architecture), and (3) Interface Layer to static and adaptive grid generator. The run-time support system will be available in the second version of the software. In the mean time the system will be based on existing incremental data mapping methods.

3. Summary and Conclusions

In this paper we discussed the MENUS-PGG software architecture and the modules that correspond to the six steps needed for the parallel numerical grid generation. Preliminary results indicate substantial reduction in the overhead introduced for the solution of the data mapping problem. Another advantage of the MENUS-PGG is that it generates and maintains grids on the processors of parallel/distributed systems and combines the most valuable aspects of the data parallel programming model with the flexibility of the task parallel programming model. With the use of composite block structures (CBS approach) as a tool for contracting the size of the problem not only we reduce the pre-processing overhead but we achieve optimality of the mapping.

Table 1 depicts performance data associated to the time (in sec) required to sequentially generate (Grid-Gen.), partition and store (Mapping) the subgrids onto the 64 nodes of the nCUBE 2, and the time (in sec) to sequentially generate and partition $C_f(\Omega)$ ($C_f(\Omega)$-Proc.) plus the time (in sec) to generate on a 64 node nCUBE 2 an algebraic grid using $C_f(\Omega)$. Columns fourth and seventh indicate the total sum of times (in sec) for the sequential grid generation and CBS approaches respectively. A SPARC workstation was used for the sequential generation and partitioning of grids and $C_f(\Omega)$.

Table 1. Performance data for sequentially generating, partitioning and storing grids on the 64 processors of nCUBE 2 and data for sequentially generating, partitioning and storing composite block structures together with the data for parallel grid generation on a nCUBE 2 with with 64 processors.

Grid Points	Grid-Gen.	Mapping	Total	$C_f(\Omega)$-Proc.	(//) Grid-Gen.	Total
2.5×10^3	0.38	17.81	18.19	3.44	0.08	3.52
10×10^3	1.61	46.45	48.06	4.38	0.35	4.73
22.5×10^3	3.61	100.15	103.76	8.48	0.71	9.19
40×10^3	6.45	195.13	201.58	16.06	1.25	17.31

ACKNOWLEDGMENTS
The authors are grateful to Wayne Mastin and Adam Gaither for their help with the NGP code. We also thank David Keyes for his detailed and constructive comments that improved the presentation of the paper.

References

1. Charbel, F., A simple and efficient automatic FEM domain decomposer, *Computers and Structures*, Vol. 28, pp 579–602, 1988.
2. Chrisochoides, N., E. Houstis and J. Rice, Mapping Algorithms and Software Environment for Data Parallel PDE Iterative Solvers, *Special Issue of the Journal of Parallel and Distributed Computing on Data-Parallel Algorithms and Programming,* Vol. 21, No 1, pp 75–95, 1994.
3. Chrisochoides, N., An Alternative to Data Mapping for Parallel PDE Solvers : Parallel Grid Generation, *Proceedings of Scalable Parallel Libraries Conference*, pp 36–44, October, 1993.
4. Chrisochoides, N., Mapping Templates for Load Balancing. To be submitted to ACM Trans. of Mathematical Software. (In preparation)
5. Fox, G., M. Johnson, G. Lyzenga, S. Otto, J. Salmon and D. Walker *Solving Problems on Concurrent Processors.* Prentice Hall, New Jersey, 1988.
6. Fox, G., P. Messina R.Williams. Parallel Computing Works! Morgan Kaufmann, San Mateo, CA 1994
7. Hammond, W. S., Mapping Unstructured Grid Computations to Massively Parallel Computers, Ph.D Thesis, Rensselaer Polytechnic Institute, Troy, NY.
8. Mansour N., *Physical Optimization Algorithms for Mapping Data to Distributed-Memory Multiprocessors.* PhD thesis, Computer Science Department, Syracuse University, 1992.
9. Mansour N., R. Ponnusamy, A. Choudhary, and G. Fox. Graph Contraction for Physical Optimization Methods: A Quality-Cost Tradeoff for Mapping Data on Parallel Computers. *International Supercomputing Conference*, Japan, July 1993, ACM Press.
10. MPI Forum, Message-Passing Interface Standard, April 15, 1994.
11. Simon, H., Partitioning of Unstructured Problems for Parallel Processing, RNR-91-008, NASA Ames Research Center, Moffet Field, CA, 94035.
12. Tam T.K.H., Price M., Amstrong C., and McKeag. Computing the critical points of the medial axis of planar object using a Delaunay point triangulation algorithm. Submitted to IEEE, PAMI, 1993.
13. Thompson, J., Z. U. A. Warsi and C. Wayne Mastin, *Numerical Grid Generation.* North-Holland, New York, 1985.
14. Thompson, J., The National Grid Project, NSF Engineering Research Center for Computational Field Simulation, 1991.
15. Thompson, J., A survey of composite grid generation for general three-dimensional regions. *Numerical Methods for Engine-Airframe Integration*, S.N.B. Murthy and G. C. Paynter eds., 1984.

Northeast Parallel Architectures Center, Syracuse University, 111 College Place, Syracuse, NY, 13244-4100
E-mail address: nikos@npac.syr.edu

Northeast Parallel Architectures Center, Syracuse University, 111 College Place, Syracuse, NY, 13244-4100
E-mail address: gcf@npac.syr.edu

Engineering Research Center for Computational Field Simulation, Mississippi State University, P.O. Box Drawer 6176, Mississippi State, MS 39762
E-mail address: joe@erc.msstate.edu