3.5 Computer Science

Group Leaders: Michael Heath, David Padua, and Daniel Reed

Faculty: Eric de Sturler, Herbert Edelsbrunner, Michael Heath, Laxmikant (Sanjay) Kale, David Padua, Daniel Reed, Shang-Hua Teng, Paul Saylor, and Marianne Winslett

Research Scientists/Programmers: Milind Bhandarkar, Jay Hoeflinger, and Kent Seamons

Post-Doctoral Associates: Alla Sheffer and Mario Pantano

Graduate Research Assistants: Gheorge Almasi, Yong Cho, Cinda Heeren, Xiangmin (Jim) Jiao, Szu-Wen Kuo, Jonghyun Lee, Xiangyang Li, Yuan Lin, Xiaosong Ma, Girish Maiya, Boyana Norris, James Oly, Ali Pinar, P. Ramachandran, Alper Ungor, Akhil Vidwans, Terry Wilmarth, Peng Wu, Shengke Yu, and Afra Zomorodian

Visitor: Tamal Dey (Indian Institute of Technology, Kharagpur)

Overview

Two teams, Computational Mathematics and Geometry, and Computational Environment, address research in Computer Science. Work in computational mathematics and geometry focuses on the areas of linear solvers, mesh generation and refinement, and interpolation between diverse discretizations. The target of our computational environment program is to provide the broad computational infrastructure necessary to support complex, large-scale simulations in general, and for rocket simulation in particular. Areas of research include parallel programming environments, compiler optimization and parallelization, parallel performance analysis and tools, and parallel input/output.

Computational Mathematics and Geometry

Mesh Generation

Teng, Li, and Ungor have developed a new mesh generation algorithm for both two and three dimensions. This new algorithm, called *biting*, combines the strengths of existing advancing front and Delaunay-based sphere packing methods. We have proved that the biting method generates a well-shaped mesh that is within a constant factor of optimal. We have also developed a new method for generating anisotropic meshes for problems where the underlying function is strongly directional. Our method uses an advancing front technique to generate a close to optimal packing of ellipses that yields a close to optimal sized anisotropic mesh. Software implementing both the biting method and anisotropic meshing has been completed for two dimensions and is currently under development for three dimensions. Other related work in progress includes implementation of geometric mesh partitioning to support parallel applications, analysis of a new sliver removal method based on smoothing, and development of space-time meshing capability. The meshing group also addresses local needs for special purpose mesh generation software for both simplex and hexahedral meshes, and plans to combine Sandia's *Cubit* with our local hex meshing software.

Dynamic Mesh Movement

Sheffer is addressing the problem of ensuring good quality of a dynamically moving mesh, such as in an ALE (Arbitrary Lagrangian Eulerian) treatment of combustion interface regression in rocket simulation. At each stage of the finite element analysis of the burning solid propellant, a mesh must be produced or adapted for the next stage of the solution. Given an initial mesh and a modified boundary, the boundary change must be incorporated into the mesh. We are developing an algorithm that will accomplish this in three stages, based on the severity of the deformation. First, propagate the boundary movement using constrained propagation (do not move nodes when the element Jacobian is small). We plan to examine the use of Winslow-, Laplacian-, and elasticity-based smoothing for the change propagation. Next, check mesh quality (we are examining various quality measures). Finally, if quality needs improvement, develop a combination of mesh smoothing techniques for improving both overall mesh quality (we plan to consider variations on Winslow, Laplacian, and equipotential smoothing) and local improvement of severely distorted elements using optimization-based techniques. If mesh quality or sizing require change following the smoothing, we expect to introduce procedures for changing mesh topology by mesh coarsening or refinement. We plan to develop coarsening/refinement techniques using the topological mesh dual (whiskers and whisker sheets).

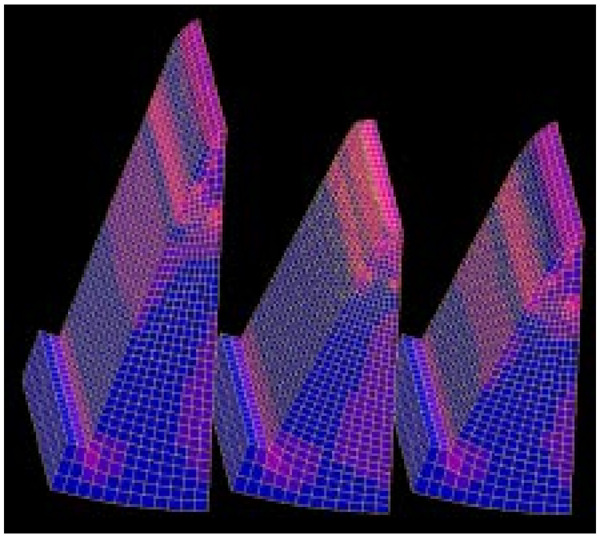
Fig. 3.5.1. Dynamic mesh movement and refinement shown for fins in head-end of Space Shuttle solid propellant grain.

Surface Tracking

Maintaining the mesh of a constantly moving and deforming spatial domain is challenging for algorithmic, geometric, and numerical reasons. The approach to this problem pursued in Edelsbrunner's group is based on novel discrete mathematics that describes geometric shapes as envelopes of spheres. From a finite set of defining spheres, the theory produces an infinite family whose envelope is finitely describable. That surface is smooth in the sense that the tangent plane and the maximum curvature vary continuously on the surface and in space. Motion is induced by growing, shrinking, and moving spheres, which includes the possibility of creating new spheres from nothing or shrinking them back to nothing. Edelsbrunner and his students have implemented an algorithm that maintains a triangular mesh representing that surface. The maximum curvature is used to control the local length scale, which is possible because of the special continuity result mentioned above. Besides curvature, the algorithm adapts the surface to changing shape and topology. The algorithm detects when the surface changes genus, and it automatically adjusts the local connectivity between triangles accordingly.

Mesh Association

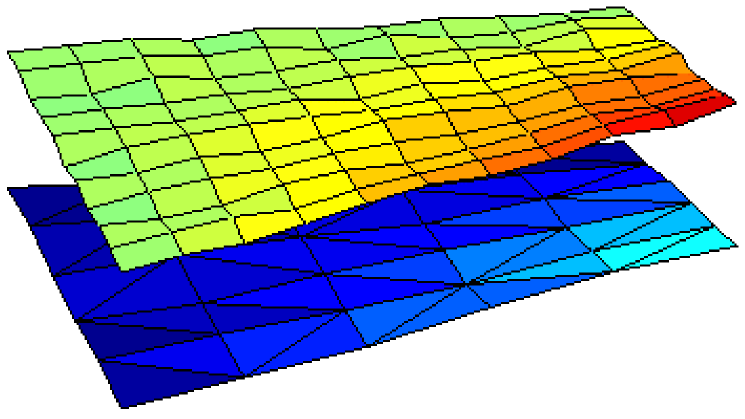
Fig. 3.5.2. Generalized mesh association algorithms enable data transfer between disparate meshes

In multicomponent simulations, the need often arises to transfer data between adjacent component meshes that may differ in structure, resolution, and discretization methodology. Though in principle the two interface meshes should abut, since they discretize the same surface, in practice this cannot be assumed because of discretization or rounding errors. Jiao, Edelsbrunner, and Heath have developed general mesh association algorithms that efficiently determine which element of one interface mesh is associated with each point in the other interface mesh (Fig. 3.5.2). The local coordinates of the associated element can then be used to interpolate relevant field values in a physically conservative manner.

Grid-Based Linear Solvers

Vidwans and Heath have developed an object-oriented framework for grid solvers. Unlike conventional sparse solvers, a grid solver solves the sparse linear system embedded in a computational grid without creating the sparse matrix explicitly at any stage of the solution process. This approach takes advantage of information available within the grid structure to produce a good partitioning and ordering for parallel execution, efficient implementation of both direct and iterative methods, and an effective structural preconditioner, all of which have been implemented within a unified, object-oriented framework.

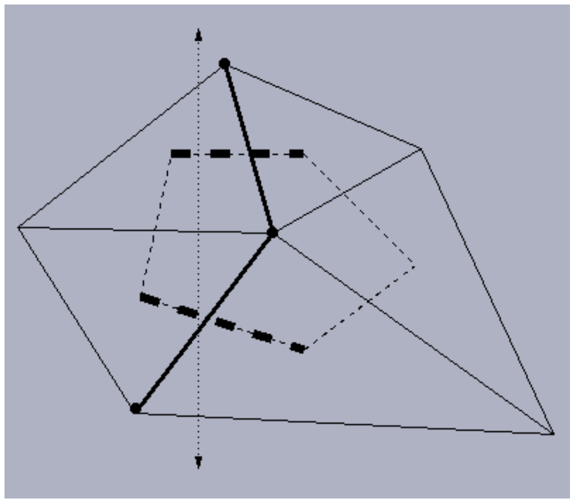
Fig. 3.5.3. Partitioning of irregular grid by Cartesian nested dissection using edge separator for dual graph to determine vertex separator for original mesh.

Improving Performance of Sparse Matrix-Vector Multiplication

Sparse matrix-vector multiplication is the principal kernel in iterative methods for solving sparse linear systems. Pinar and Heath have developed new algorithms and data structures for sparse matrix-vector multiplication that enhance performance by reducing memory indirection. Specifically, we pack contiguous nonzeros into blocks so that only one indirection is required per block. We have also implemented algorithms, based on heuristics for the Traveling Salesperson Problem, to permute the nonzeros of a sparse matrix into contiguous locations.

Iterative Methods for Linear Systems and Eigenvalue Problems

Saylor and collaborators have developed a block-Jacobi preconditioner for linear systems arising from the integro-differential equation of radiation transport. This preconditioner is based on LDU factorization of the diagonal blocks of the block-tridiagonal matrix, and is effective even with dense blocks or an indefinite matrix. Saylor has also developed more precise stopping criteria for iterative methods based on Gaussian quadrature.

De Sturler made several advancements in understanding the behavior of iterative methods based on the nonsymmetric Lanczos process. We are implementing these results in a software package to be included in the *ISIS*++ package developed at Sandia Livermore. We plan to continue research on fast solvers for nonsymmetric systems that arise in large-deformation structural analysis of rocket motors. We also continued research on the theoretical background of eigenvalue solvers for large sparse matrices (important for stability analysis) and the use of different projectors in Jacobi-Davidson type methods and plan to apply these methods to rocket motor stability analysis.

Computational Environment

Performance Analysis Tools

The complexity of multidisciplinary, adaptive applications necessitates online analysis and tuning of system and application performance. To analyze and optimize the behavior of the multidisciplinary CSAR codes, Reed's group continued to develop and enhance the *SvPablo*, *Autopilot*, and *Virtue* toolkits for performance tuning and visualization. Using *Globus* as a base, the *Autopilot* toolkit defines performance sensors, control actuators, and decision procedures. These decision procedures implement fuzzy logic control of distributed software using sensor inputs and actuator outputs. To minimize the complexity of sensor specification and configuration, *Autopilot* relies on the *SvPablo* system for automatic code parsing and sensor insertion.

Using this integrated system, we instrumented CSAR's evolving GEN1 simulation code and analyzed its behavior. *Virtue* revealed optimization opportunities in the code's initialization phase, which is dominated by exchange of small MPI messages. We also identified opportunities for dynamically adjusting convergence criteria and execution balance across code components. After inserting *Autopilot* sensors and actuators, *Virtue* will enable the developers to explore the behavioral balance of code components interactively. During the coming year, we plan to continue extension of the integrated *SvPablo*, *Autopilot*, *Virtue* instrumentation, visualization, and control system. Our goal is further to enable adaptive software optimization, both automatically and interactively. This will require both software infrastructure extensions and continued close collaboration with the CSAR code developers to identify those points where software steering would provide maximum benefit.

Parallel I/O

CSAR's rocket simulation codes need high-performance parallel I/O, for example, to take many successive snapshots of data arrays for creating animated visualizations. To provide such facilities, Winslett's group adapted *ROCFLO*, the fluids component of GEN1, to use *Panda*, our prototype library for parallel I/O. Our experimental evaluation on the SGI Origin

2000, the main current platform for the rocket simulations, shows good I/O performance when *ROCFLO* uses *Panda* to perform I/O.

Inexpensive clusters of PCs are a platform of great interest for CSAR. Unfortunately, clusters of symmetric multiprocessors (SMPs) typically incur severe I/O load imbalance when traditional parallel I/O approaches are used because of the shared resources present in SMPs and the high likelihood of heterogeneity in a cluster of significant size. To provide high performance I/O in such environments, we developed algorithms that automatically determine the optimal number and placement of I/O servers for a possibly heterogeneous cluster of SMPs. Our experimental evaluations on a heterogeneous SMP cluster running NT show that our approach correctly places I/O servers to maximize I/O performance.

Parallel visualization tools are an interesting future direction for high-performance simulations. Data placement is an important issue in such an environment, as a poor arrangement of data on disk will slow visualization performance by utilizing less than the full degree of parallelism present in the storage system. We developed algorithms to decluster simulation output data across parallel disks so that the time required to fetch a random subspace for visualization is minimized.

We are currently adapting *ROCSOLID*, the solids component of GEN1, to use *Panda* for I/O, and will continue to provide parallel I/O support for CSAR as the codes evolve. To this end, we are adding facilities to *Panda* that support concurrent I/O of multiple, independent arrays that are not distributed across processors. We are also expanding and enhancing our facilities for migration of data between platforms during computation, with an emphasis on optimization techniques that will minimize the turnaround time for users to see results from a running simulation. We are also continuing our efforts to design a modular, portable framework for optimization of parallel I/O.

Reed’s group also continued to develop adaptive I/O software and libraries for automatically classifying I/O access patterns and dynamically configuring appropriate I/O policies. This PPFS II toolkit was extended with hidden Markov models that can predict irregular I/O patterns. We plan to integrate our PPFS II parallel I/O library with the CSAR code on Linux clusters. This will both provide a shared file system for the cluster and enable us to distribute application checkpoint and visualization data adaptively across multiple storage devices.

Compiler Techniques for Irregular Structures

Padua’s group is developing compiler techniques to analyze irregular structures such as linear lists and arrays of linked lists. The result of this analysis is necessary in many cases to enhance locality automatically and to detect parallelism both at the instruction level and the loop level. A method has been developed to analyze accurately heap-allocated objects, especially objects of standard classes (such as a linear arrays, linear lists, and hash tables), which may contain references to arbitrary objects. Because the analysis deals with standard classes, it is possible to include within the compiler the semantics of the classes to be analyzed and, in this way, succeed where earlier techniques have failed. Although not fully general, focusing the analysis on standard classes should have an important effect, given that standard classes are used frequently in real programs and programmers, once informed that the compiler generates better code in the presence of standard classes, will likely try to use these as often as possible. Other compiler research included completion of a new, more efficient version of an interprocedural analysis pass for the Polaris parallelizing compiler.

Investigation and Evaluation of OpenMP

OpenMP is a de facto standard for compiler-driven, shared memory parallel programming. During the past year, Hoeflinger carried out a performance comparison of Fortran 90 with MPI and with OpenMP on the SGI Origin 2000. This study showed that certain Fortran 90 constructs cause bottlenecks when used inside parallel regions with OpenMP, and that the method used for making thread-private data with OpenMP causes less efficient code to be generated for using that data. The Fortran 90 ALLOCATE statement and all I/O statements are serialized when used with OpenMP, and therefore do not show scalable performance when used in parallel regions. During the coming year, we plan to study the use of MPI and OpenMP together, using the results and experience gained from the MPI/OpenMP performance comparison study. First, we will try a hierarchical combination, using MPI at the highest level and OpenMP inside each MPI process. Next, we plan to experiment with non-hierarchical combinations of the two paradigms. In addition, we plan to upgrade *Polaris* to handle Fortran 90, which would allow it to manipulate the CSAR codes.

Object-Based Parallel Programming for Irregular Applications

Many complex, coupled simulations exhibit irregularity that poses several difficulties for their implementation on parallel computers. Patterns of computation and communication within applications change with time, causing severe load imbalance. Kale’s group is developing parallel programming techniques to simplify development of irregular and dynamic applications on dedicated as well as shared parallel computing platforms. Our efforts are based on measurement of actual computational load (rather than load estimation) coupled with object migration-based load balancing in *Charm++,* an object-oriented framework for portable parallel programming.

Our research in the past year focused on exploring and demonstrating various techniques for programming applications using object-based frameworks. For this purpose, we have incorporated an abstraction for arrays of migratable objects into *Charm++*. A framework for load balancing strategies was developed that simplifies writing new plug-in, distributed or centralized load balancing strategies by automated load collection and external as well as internal triggers for these strategies. This allows programming applications in *multi-partition* manner, where the computations are represented by objects (called partitions), which can be mapped many-to-one onto available processors, and which can be migrated periodically under control of the plug-in load balancing strategy to achieve better load balance. A migration path for MPI-based legacy applications was developed and supported by our new MPI library (ArrayMPI) on top of the array abstraction in *Charm++*. This allows us to execute the MPI process in a user-level thread associated with an array element object without any significant change in a legacy code. We demonstrated this by converting *ROCFLO* to the multi-partition paradigm.

We plan to perform detailed evaluation of the multi-partition paradigm with respect to the ease of developing new applications, as well as for converting legacy applications. Existing component codes of the rocket simulation programs are the candidates for such conversion. Specifically, we are planning to explore adaptive refinement in *ROCFLO* and in crack propagation codes from Geubelle's group. We will analyze and reduce overheads associated with the load balancing framework and the multi-partition paradigm. In addition, we are building application frameworks that incorporate commonly used algorithms in applications such as finite difference and finite element methods.

Software Integration Framework

A new Software Integration Framework Team (SWIFT), including Bhandarkar, de Sturler, Hoeflinger, Kale, Padua, and Reed, has begun an extensive evaluation of existing frameworks with respect to the requirements of our simulation codes, mode of collaboration, and assumptions on problems to be solved and programs to be included. We also developed an initial version of our own framework and a test implementation of *ROCFLO* within this framework. This test worked out very well; the (parallel) *ROCFLO* program was ported onto the framework in about one week. This confirms our approach is in the right direction. We reported on this effort in a workshop at the PDPTA conference. We plan to develop a tool, based on *Polaris*, for automatically translating Fortran 90/MPI codes into this framework.

Further development of the framework is continuing in collaboration with the CSAR applications groups. A deeper analysis of software frameworks and of the most important scientific issues that are necessary to advance this area is currently under way, and we expect to make significant advances in the development of our software framework in the next year. An integrated, flexible framework will be developed that will provide services for both the solids and fluids codes and for the mesh association code at the interface between these components. This framework will also be open to other programs, as we plan to follow a modular component approach.