3.3 Computer Science

Group Leaders: Michael Heath and Laxmikant Kale

Faculty: Eric de Sturler, Michael Heath, Laxmikant Kale, and Marianne Winslett

Research Scientists/Programmers: Phillip Alexander, Michael Campbell, Robert Fiedler, Damrong Guoy, Xiangmin (Jim) Jiao, Orion Lawlor, and John Norris

Post-Doctoral Associates: Jonghyun Lee, Terry Wilmarth, and Eric Shaffer

Graduate Research Assistants: William Cochran, Rebecca Hartman-Baker, Rajeev Jaiman, Soumyadeb Mitra, Michael Parks, Manoj Parmer, Christopher Seifert, Rishi Sinha, Pornput Suriyamongkol, Evan Vanderzee, Hanna Vanderzee, Terry Wilmarth, and Gengbin Zheng

Overview

The computer science group continues to provide technologies and tools used by the rest of CSAR. Production runs of the integrated *Rocstar* code use our automatic tool *Makeflo* to partition the structured mesh for *Rocflo*, the tools provided by the Charm++ FEM Framework to partition the unstructured mesh for *Rocflu*, and our adaptive version of MPI, AMPI, to provide parallel communication along with enhanced latency tolerance, automated load balancing, and numerous other benefits. Various aspects of our research in computer science are summarized in the following sections.

Two teams — Computational Environment, and Computational Mathematics and Geometry — address research in the Computer Science Group. The target of our Computational Environment team 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, parallel input/output, and visualization. Work in Computational Mathematics and Geometry focuses on the areas of linear solvers, mesh generation and adaptation, and interpolation between diverse discretizations.

Computational Environments

The Computer Science Group has provided support for several key technologies used by the rest of the Center. Our unstructured mesh support library, the FEM Framework based on *Charm++,* is in full production use by the Fluids Group and in the integrated code. During the next year, we plan to improve the meshing, mesh partitioning, and remeshing support in the *Charm++* FEM framework.

Lagrangian Interface Propagation (Jiao)

In *Rocstar*, the interface must be tracked as it regresses due to burning. In recent years, Eulerian methods, especially level set methods, have made significant advancements and have become the dominant methods for moving interfaces. In our context, Lagrangian representation of the interface is crucial to describe the boundary of volume meshes of physical regions. However, previous numerical methods, either Eulerian or Lagrangian, have difficulties in capturing evolving singularities (such as ridges and corners) in solid rocket motors.

To meet this challenge, we have developed a novel approach, called *face-offsetting methods*, based on a new *entropy-satisfying Lagrangian formulation.* Our face-offsetting methods deliver an accurate and stable entropy-satisfying solution without requiring Eulerian volume meshes. A fundamental difference between face-offsetting and traditional Lagrangian methods is that our methods solve the Lagrangian formulation face by face, and then reconstruct vertices by constrained minimization and curvature-aware averaging, instead of directly moving vertices along some approximate normal directions. This method allows part of the surface to be fixed or to be constrained to move

along certain directions (such as constraining the propellant to burn along the case). It supports both structured and unstructured meshes, with an integrated node redistribution scheme that suffices to control mesh quality for moderately moving interfaces. Figure 3.3.1 shows the propagation of a block-structured surface mesh for the fluids domain of the Attitude Control Motor (ACM) rocket, where the front and aft ends burn along the cylindrical case.

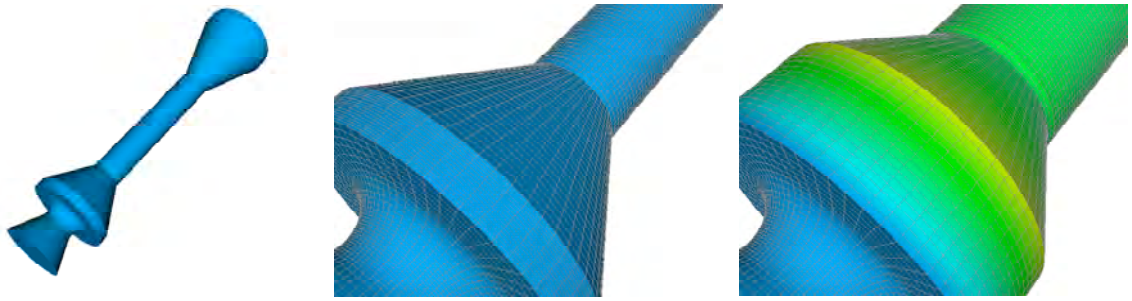

Fig. 3.3.1: Simulation of burning of Attitude Control Motor along case with block-structured meshes using face offsetting. Left subfigure shows initial geometry; middle and right subfigures show meshes of initial geometry and after 30% burn, respectively. Colors indicate magnitude of total displacements of vertices.

When coupled with mesh adaptation, the face-offsetting method can capture significant burns. Figure 3.3.2 shows a sample result of the burning of a star grain section of a rocket motor using the face offsetting method coupled with surface remeshing using MeshSim from Simmetrix (http://www.simmetrix.com). The interior (the fins) of the propellant burns at uniform speed and exhibits rapid expansion at slots and contraction at fins. The fin tips transform into sharp ridges during propagation, as captured by the face offsetting method.

Fig. 3.3.2: Simulation of uniform burning of section of star grain of solid rocket using face offsetting and mesh repair. Green curves indicate ridges in evolving geometry.

Our methods are extensible in many ways. One important direction is to integrate mesh repair and collision detection to allow propagation under more complex flows. In the coming years, we will work on surface mesh adaptation for handing further geometrical complexities associated with large deformation and substantial burns (such as burn-out of fins), and for handling topological changes. In addition, volume and mass conservation are important for which face offsetting is promising in delivering a simple and reliable solution because of its flexibility in reconstructing vertices. Our current numerical schemes use piecewise linear faces, and hence allow second order accuracy in space. The concept of the geometric construction allows extension to higher order schemes.

Interface propagation has many applications in science, engineering, and computer graphics to which face-offsetting methods are applicable. In addition, our methods can also be applied to surface mesh smoothing, which shares two difficulties with surface propagation: point projection, and enforcing nodal constraints. By posing smoothing as a propagation problem with zero speed, the con-

strained minimization and eigenvalue analysis of face offsetting can help address both of these problems.

Inter-Mesh Solution Transfer (Jiao, Jaiman, Geubelle, and Loth)

Rocface is the interface code that transfers data between non-matching surface meshes in an accurate and conservative manner. To achieve these goals, *Rocface* constructs a common refinement, that is, a mesh whose polygons simultaneously subdivide the polygons of the given input meshes. In collaboration with Jaiman, Geubelle, and Loth, we have performed a systematic comparison of the common-refinement based method used in *Rocface* with some other methods in the literature for fluid-solid interactions, identified conclusively the source of large errors in those traditional methods, and demonstrated the superiority of *Rocface*'s methods.

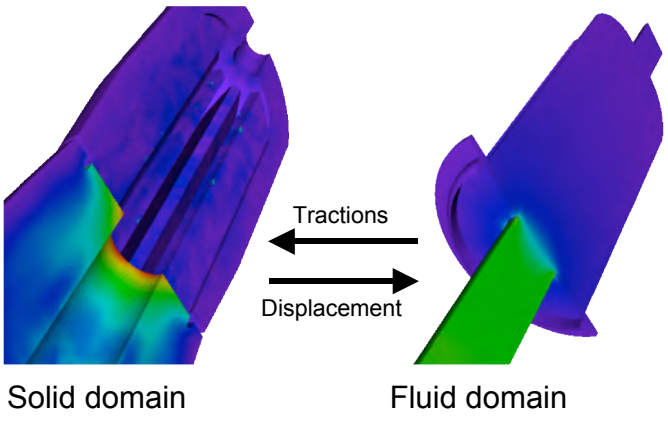

Fig. 3.3.3: *Rocface* used to transfer load and motion in conservative and accurate manner across Nonmatching interfaces in Multiphysics simulations.

Mesh-correlation and inter-mesh solution transfer are important in many scientific and engineering applications. Besides transfer of interface data in fluid-solid interactions, other examples include data transfer after remeshing, crack propagation, and contact problems. This correlation problem is also critical to a number of emerging numerical methods, including the generalized finite element methods (GFEM), immersed and embedded boundary methods, which are likely to be of strategic importance to CSAR's longer-term goal of simulating rockets under abnormal conditions.

The future work will focus on developing efficient and robust parallel algorithms for correlating different meshes that overlap in space but may move across, or slide along, each other, and implementing these algorithms in an easy-to-integrate software toolkit to facilitate various advanced simulations, such as fluid-solid-combustion interaction in rocket simulations, and crack propagations using generalized finite element methods (GFEM).

Data Management and I/O (Winslett, Mitra, and Sinha)

During the past year, we designed and implemented a new indexing facility for use with *Rocketeer* (and possibly with *Rocstar* in the longer term). This facility allows a user to specify a region of interest in the artifact being visualized, by supplying limit values on major attributes (e.g., temperature, physical location, pressure). The indexing facility will locate the qualifying regions of the data quickly. When used in concert with the GODIVA buffering facility, the indexing facility allows loading of only the data of interest into memory, rather than the entire data set. The indexing facility is based on bitmaps, and breaks new ground in the data management world in a variety of respects. The new facility is currently undergoing performance testing; a preliminary description of it has been accepted for publication in the Storage Network Architecture and Parallel I/O workshop collocated with PACT05.

To support *Rocpanda's* role as the main I/O module for *Rocstar*, we also updated *Rocpanda* to work with *Rocstar* 3. This required major modifications to support *Rocstar's* new data formats and to change the way that data are sent to *Rocpanda* servers. Also, our GODIVA library for data-format-independent prefetching and caching of scientific data continued to be used in the production versions

of both the batch and interactive versions of the *Rocketeer* visualization tools. The figure included below shows the performance benefits that accrue from the use of GODIVA in a typical batch run of *Rocketeer*.

Finally, we designed, implemented, and evaluated a log-based facility that uses idle local disks to minimize the apparent run-time cost of data writes performed by simulation applications, and evaluated it with rocket simulation data. This facility can be used by *Rocstar* in the future if the cost of immediate writes to remote storage becomes prohibitively high, which may happen as the level of computational parallelism continues to increase on *Rocstar's* platforms. The results of this work will appear in the Cluster Computing 2005 conference.

During the next two years, we plan to complete our new indexing facility and integrate it into *Rocketeer* (and, if appropriate, into *Rocstar*). This should provide big performance gains for situations where one only wants to look at part of a large data set. We will also continue to support the evolution of Rocstar and *Rocketeer* through updates to *Rocpanda* and GODIVA as needed. Further, we will evaluate the performance improvement potential of passing I/O related hints to the storage servers used in large parallel computations, to allow the storage servers to tune themselves better for the upcoming I/O requests. Finally, we will continue our effort to transfer the technology of active buffering and data migration to the ROMIO parallel I/O library. We intend the fruit of these technology transfer activities to be a lasting legacy of CSAR that will benefit the thousands of scientists who use ROMIO and netCDF.

Remeshing (Wilmarth, Lawlor)

Over the past six months, we have worked on the *Rocrem* remeshing module. *Rocrem* is used when a physics solver reaches a point of failure due to deteriorated mesh quality. Separate mesh smoothing and local mesh repair efforts are ongoing. Remeshing handles the worst-case scenario when smoothing and local repair fail to improve mesh quality sufficiently for the solver to continue.

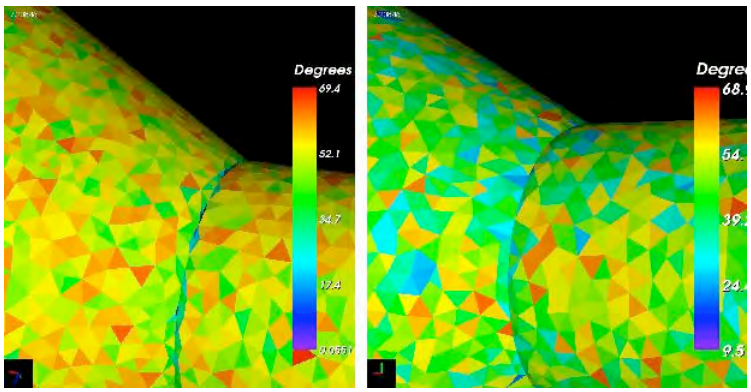

Fig. 3.3.4: Minimum dihedral angle before and after remeshing

Remeshing efforts were begun by Orion Lawlor via a proof-of-concept code using YAMS and TetMesh to perform serial remeshing and Charm++ for parallel solution data transfer. Simmetrix meshing software was designated to replace YAMS and TetMesh, since it has a usable API and runs in parallel. Terry Wilmarth has extended this to convert *Rocrem* to use Simmetrix (Figure 3.3.4).

Rocrem was tested with various meshes, resulting in improvements in the initial surface mesh stitcher and parallel solution data transfer phases of the code. It now produces higher quality meshes with correct solution data. We have added functionality to make the regional granularity of the new mesh similar to that of the original mesh.

Recent efforts are focused on restarting. Four types of *Rocflu* restart files are needed for each partition. Rocrem now extracts data from the new mesh and prepares these files. We are ready to test the solver restart capabilities from a remeshed rocket. Additional efforts involved improving the mesh quality produced by Simmetrix. Initial experience has yielded lower quality meshes than those obtained with YAMS and TetMesh.

Wilmarth has continued work on *ParFUM*, the Charm++ Parallel Framework for Unstructured Meshes, in collaboration with Prof. Geubelle and his students. She has implemented parallel refinement and coarsening algorithms that can be used by a solver to refine mesh regions where solution accuracy is of the highest priority and to coarsen where there is less activity (Figure 3.3.5).

Plans for the future of *Rocrem* include:

- Test the restart capabilities: proceed with this in parallel with working out the mesh quality issues, since in some cases Simmetrix produces adequately improved meshes.
- Resolve remaining Simmetrix mesh quality issues; these efforts are ongoing.
- Integrate *Rocrem* for automatic on-line remeshing: the goal of *Rocrem* is to enable full burns of rockets. We are already at the testing stage for this, so the next logical step is to enable full burn with little or no user interaction required. This will involve implementing an on-line interface using *Roccom* to extract the mesh directly from memory and perform *Rocrem's* task. It will also involve changes to *Rocflu* to enable restart with a new in-memory mesh, and remove the dependency on the several restart files currently required.

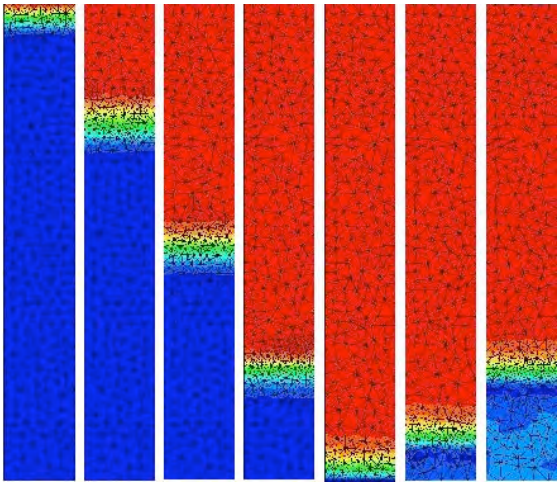

Fig. 3.3.5: Refining along wavefront, coarsening behind

Performance Tools and Tuning (Campbell, Alexander, and Haselbacher)

Rocprof, the profiling module in *Rocstar*, has been redesigned and adapted for use with *Roccom* 3 and *Rocstar* 3. The new *Rocprof* has been designed as a light, stand-alone CS service module rather than a tightly integrated submodule of *Roccom*. As a new CS service module, *Rocprof* compliments *Roccom's* automatic high level profiling by providing enhanced profiling services to any code in the Rocstar suite. *Rocprof's* services are provided through native and standard MPI_PCONTROL interfaces, allowing use by any serial or parallel application. The *Rocprof* suite now also includes post processing tools with which to archive, report, and analyze multiple sets of parallel and serial performance data. *Rocprof's* new design conslidates the stand-alone and integrated code bases, streamlining further development. Support for MacOS X 10.3 and 10.4 has been added.

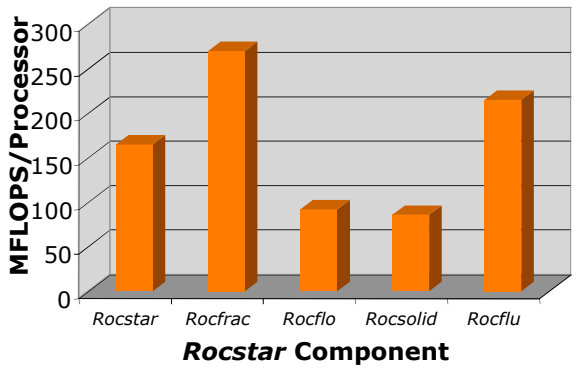

Fig. 3.3.6: *Rocprof* used to profile individual *Rocstar* modules

We have been putting *Rocprof* to good use in profiling and tuning *Rocstar* 3 components. Submodule profiling has enabled us to identify and eliminate many performance bottlenecks in *Rocstar* physics and service modules. Ongoing performance analysis and tuning efforts to *Rocflu*, *Rocstar's* unstructured fluid solver, have yielded substantial performance boosts of up to factors of two for

critical code sections. We have been collaborating with Livermore Computing in the Rocflu tuning effort. *Rocprof* has also identified several bottlenecks in *Rocstar's* mesh smoothing service module, Rocmop. Subsequent elimination of these bottlenecks have reduced overall smoothing overhead by a factor of two.

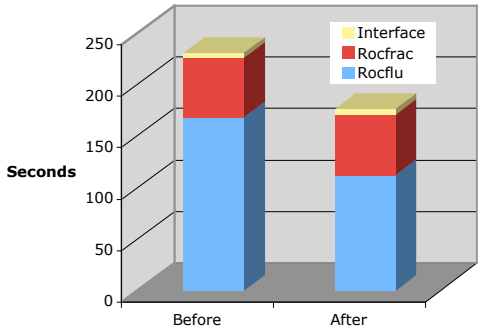

Fig. 3.3.7: *Rocprof* used to identify bottlenecks in *Rocflu*. Reduced overhead by factor of two.

Project Plans for 2006

- Recovery - We will recover functionalities that we have left behind in the redesign of *Rocprof* - Intermediate data dumps allow very long simulations to continue without introducing profiling overhead. This important feature is key to allowing detailed profiling during production runs where performance is important. This is a high priority *Rocprof* item. - V.I. PAPI support. Version independent PAPI support allows us to obtain hardware counter access on Linux systems. As we still have counter access on IBM SP systems, this is a medium priority item. - Thread safety shoring to ensure proper operation under CHARM++. This is a high priority item.
- Enhancement - We plan the following enhancements to *Rocprof* capabilities. – Multiple profile levels to allow varying degrees of detail. Low priority. – Module-based control allowing indepedent profiling control for each Rocstar module. Med priority. - Control file input to allow fine-grain control of profiling. This is high priority. - CHUD tool interface allowing statistical kernel based profiling in the Mac OS X environment. Medium priority
- Utilization - We plan to conduct the following performance studies in addition to nominal Rocstar performance monitoring - *Rocflu* profiling and tuning on multiple platforms. This activity is already underway. There is limited literature available on performance enhancement of unstructured finite element codes and we hope to produce a conference paper and a journal paper out of this activity. - *Rocpart* profiling and tuning as integrated with both structured and unstructured fluid codes.

Mesh Smoothing (Alexander, Campbell, Jiao, Shaffer, E. Vanderzee)

In 2004, we introduced a new module, *Rocmop*, to alleviate the degradation of mesh quality as a simulation progresses. This deterioration necessitates ever-shorter global time steps, and may even result in simulation failure. The intent of *Rocmop* is to provide black-box parallel mesh-smoothing routines supporting both volume mesh smoothing and feature-preserving surface mesh smoothing.

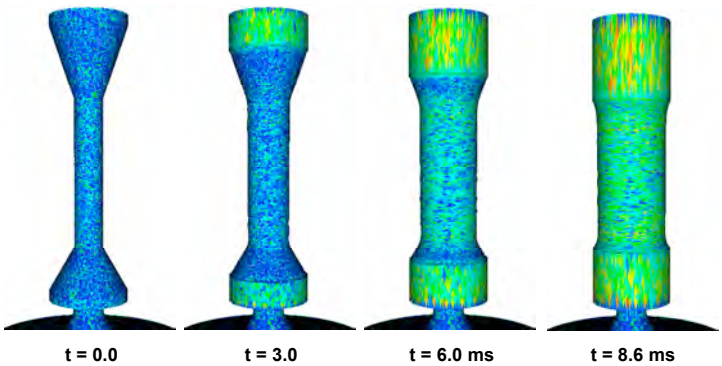

Fig. 3.3.8: Mesh smoothing used to minimize requirement for full motor remeshing. ACM motor shown.

In our simulations, interior mesh quality issues tend to appear earlier than problems on the surface. Therefore, our initial efforts have concentrated on volume mesh smoothing. Our implemen-

tation uses *MESQUITE*, a national lab developed mesh quality toolkit developed at SNL, as a serial core that is used in conjunction with simple partition boundary smoothing techniques to provide volume mesh optimization in parallel. Recent results have proved that *Rocmop* is a viable tool for prolonging simulation life, and *Rocmop* is now utilized in some of our largest runs, including those involving the NASA space shuttle's RSRM rocket booster and the Titan IV PQM-1 rocket motor.

Mesh Smoothing Related DOE Interactions CSAR's continuing development of a parallel mesh-optimization module utilizes the MESQUITE mesh quality toolkit developed primarily at Sandia National Laboratories. This effort is aided by ongoing interaction with national laboratory personnel. To date, we have conferred with Lori Freitag from LLNL as well as Patrick Knupp and Mike Brewer, both employed at SNL. Two items in particular underline the increasing depth of CSAR's relationship with this project's researchers. The first is that at our request, a version of *MESQUITE* was delivered which provides support for additional mesh element types critical to our simulations. This level of commitment to improving *MESQUITE's* viability as a general purpose tool has exceeded our expectations. The other is the recent internship of a CSAR student, Evan VanderZee, with the *MESQUITE* group at Sandia. CSAR will soon adopt reference-mesh-based quality metrics, metrics with which Evan now has first-hand knowledge.

Mesh Smoothing Plans

Vast improvements are planned for *Rocmop's* volume smoothing capabilities. The current version of the module has inefficiencies that will be addressed during fall 2005. Later in the year, and extending into the spring of 2006, development will focus on incorporating the newly available version of *MESQUITE,* with its support for pyramids and prisms. This work will enable all of the physics codes to utilize Rocmop, whereas the current version supports only *Rocflu*. At the same time, a new reference-mesh-based quality metric will be adopted. Useful characteristics of this class of metrics include robust detection of poor quality elements (dihedral angle is not robust), increased respect for mesh sizing, and the absence of start-up issues currently caused by initial smoothing displacements.

A parallel effort will restart development of surface smoothing in *Rocmop*. With simulations progressing further towards burnout, this ability becomes increasingly important. In 2005, CSAR continued to develop surface smoothing capabilities indirectly under the auspices of *Rocprop*, a surface propagation module. Core surface routines for feature detection and classification that are needed by both *Rocmop* and *Rocprop* will be separated from *Rocprop*. We will then concentrate on building higher-order geometric representations of the surface mesh and corresponding schemes to project points onto these surfaces. During the first half of 2006, CSAR will use these surface routines to introduce parallel feature-preserving surface smoothing either through custom code, or via further integration with *MESQUITE*.

I/O Capability Enhancements (Norris)

During the last year, we have worked to improve the I/O capabilities of *Rocstar*. Norris has expanded the two new *Roccom* I/O modules, *Rocin* and *Rocout*, to support CGNS 2.4 and also worked with the CGNS development team to include support for ghost data in unstructured grids, a necessary feature for our data. CGNS will be a welcome change to HDF4, which presents many problems, including slow file access and nondeterministic I/O errors. The CGNS reader plug-in for *Rocketeer* is also being updated to support the new version.

Rocman (Zheng, Jiao)

We redesigned *Rocman*, the control and orchestration module, to coordinate multiple physics modules in coupled simulations and provide facilities to extend and implement new coupling schemes. With a completely new design, using the idea of action-centric specification and automatic scheduling of reusable actions to describe the intermodule interactions, the new *Rocman* facilitates

the diverse needs of different applications and coupling schemes in an easy-to-use fashion. The new *Rocman* also provides better readability and maintainability by promoting code reuse and modularity.

Rocman currently supports FluidAlone without burn, FluidAlone with burn, SolidAlone without burn, Solid/Fluid coupling schemes without burn, Fluid/Solid coupling schemes without burn, and the fully coupled scheme with Fluid/Solid/Burn. The new *Rocman* also provides a visualization tool that allows a user to examine the coupling scheme and verify its correctness of the scheme visually. The visualization of a coupling scheme shows the internal control flow of actions and the data flow as shown in the following figure.

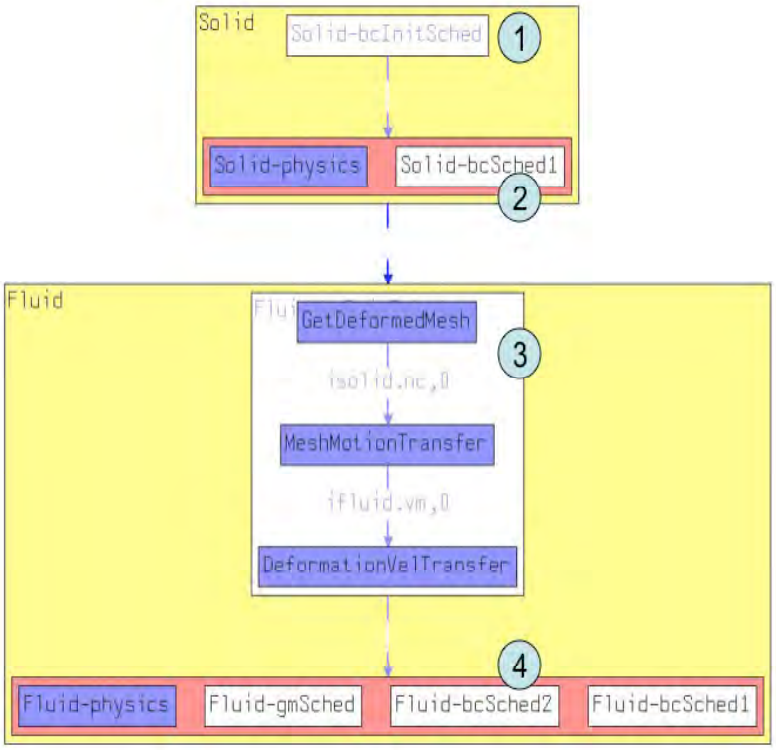

Fig. 3.3.9: Solid/fluid coupling scheme visualization in AiSee

Parallel Computing

AMPI (Kale, Zheng, Lawlor, Wilmarth)

We continue to develop AMPI, our version of MPI that supports virtual processors. We have greatly improved AMPI performance on the new Apple Turing cluster with Myrinet interconnect. We dramatically improved the communication performance with both small and large messages, as well as collective communication such as broadcasting, by optimizing Charm++'s Myrinet machine layer. *Rocstar* running on Turing cluster enjoys this performance improvement since almost every production run of the integrated code running on the Turing cluster uses AMPI.

Load balancing Many scientific applications, including rocket simulation, involve simulations with dynamic geometry, and use adaptive techniques to solve highly irregular problems. In these applications, load balancing is one of the key factors for achieving high performance on large parallel machines. Load balancing for extremely large parallel machines such as Blue Gene/L, is ever challenging. We designed and implemented a new hierarchical load balancing algorithm for very large parallel machines and demonstrated its performance sing the BigSim performance simulator we developed.

Rocman 3 We will continue to work on *Rocman* 3. Future work includes:

- Incorporate all existing features, such as ALE and predictor-corrector, from the older version of *Rocman*.
- Verify the correctness of new *Rocman* by comparing with the physics results of old *Rocman*.
- Continue to work with other physicists in developing new coupling scheme based on their needs.

- Work on a comprehensive *Rocman* manual so that it becomes easier for a physicist to write his/her own coupling scheme.
- We will continue to improve the performance of AMPI, especially the communication performance for small messages and collective operations.

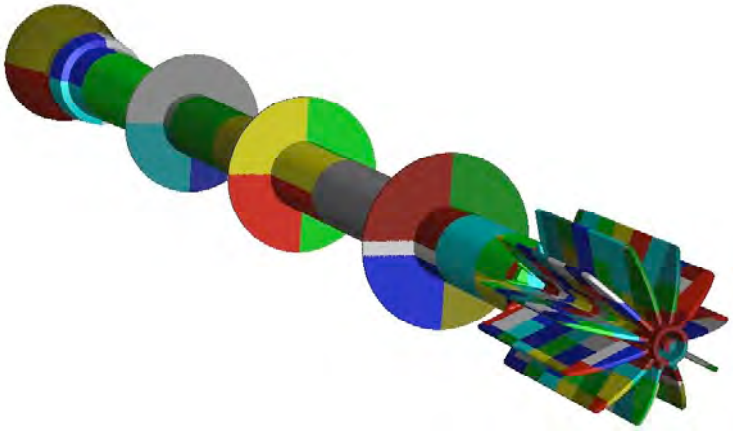

Fig. 3.3.10: New hybrid topological-geometric mesh partitioner used to generate a 200 partition, 36.8 million mixed element, RSRM fluid domain

Rocstar currently does not use the load balancing framework in Charm++ and AMPI. We plan to integrate *Rocstar* with our load balancing framework by extending the Rocstar code with process migration capability.

Parallel Mesh Partitioning (Cochran and Heath)

The hybrid topological-geometric mesh partitioner we have proposed relies on the accurate computation of the medial surface of a mesh. Available algorithms are efficient but require the mesh to be free of sharp features in order to achieve sufficient accuracy for this application. We are developing an algorithm that is free of this requirement, and initial results look promising. A simple, smooth mesh has been subdivided topologically for optimal geometric partitioning. Further, the parallel geometric partitioner has been implemented and successfully partitioned a 37 million element mesh for the RSRM, far exceeding previous partitioning capability in size.

Mesh Smoothing (Guoy and Suriyamongkol)

The meshing research in CSAR is driven by the need to cope with dynamically changing geometry. In integrated simulations, the fluid and solid domains change due to both structural deformation and consumption of solid propellant. Our strategy for mesh improvement involves three levels of aggressiveness to deal with progressively more severe geometric change.

The first line of defense is a mesh smoothing module called *Rocmop*, in which mesh nodes are repositioned, but topological connectivity is preserved. In our application, mesh smoothing is applied in a dynamic setting. The smoothing procedure must be effi-

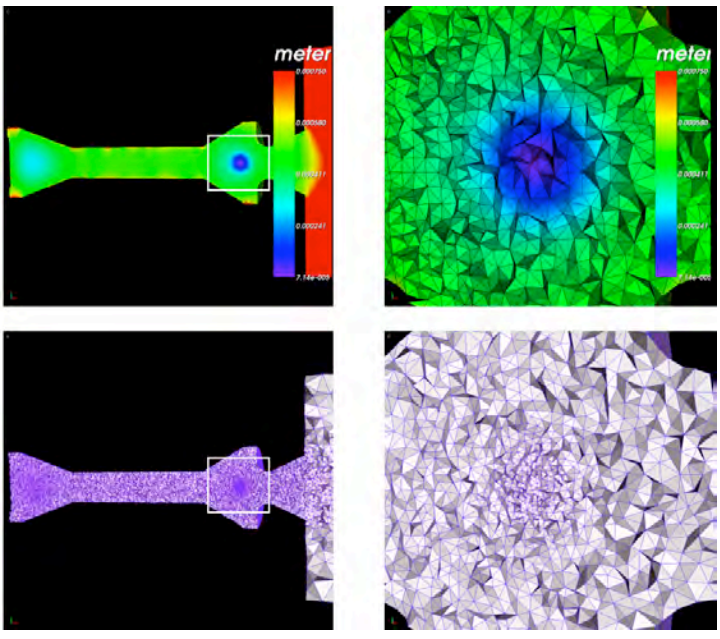

Fig. 3.3.11: Adaptive mesh refinement via MeshSimAdapt. Top pictures show input sizing field with hot spot in input mesh. Bottom pictures show results from local mesh refinement near hot spot.

cient enough to be called at every time step by every processor. *Rocmop* has been deployed successfully and is currently being tuned for efficiency and robustness.

The second line of defense is *local mesh repair*. This procedure is needed for severe deformation in a local area; for example, near the tip of a propagating crack or around the joint slot of the pressurized Titan IV. The goal of local repair is to eliminate poor elements in the problematic area while preserving good elements elsewhere. Only newly created elements need local solution transfer. Parallelization of local mesh repair is challenging because the area in need of repair can be shared by an arbitrary number of processors. In the past year, we collaborated with Prof. Mark Shephard of Rensselaer Polytechnic Institute and Simmetrix Inc. on local mesh repair and adaptation. Our preliminary results showed the effectiveness of Simmetrix's MeshSimAdapt on adaptive mesh refinement (Figure 3.3.11), mesh sizing control, sequential local mesh repair, and on parallel local repair (Figure 3.3.12). The results look promising.

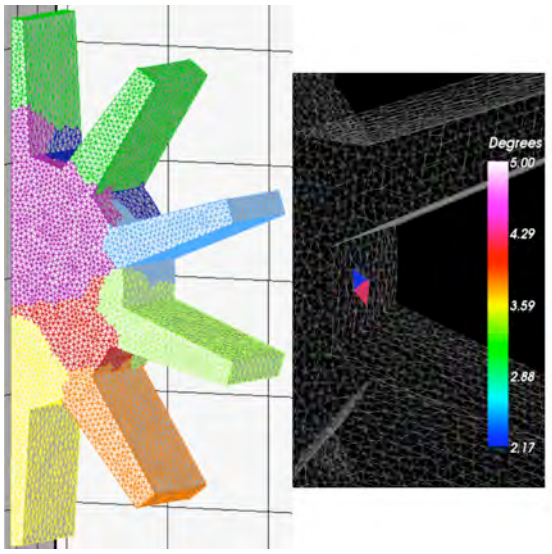

Fig. 3.3.12: Parallel local repair. Top picture shows sliver elements with minimum dihedral angle between 2 and 5 degrees. Bottom picture shows mesh after parallel repair, with minimum dihedral angle was improved to 10 degrees.

The third line of defense is *global remeshing* which is implemented in a module called *Rocrem*. Remeshing the entire domain has the advantage of being independent of the severity of the deformation of the old mesh, but the disadvantage of requiring global solution transfer, which entails an expensive geometric search. In addition, boundary flags must be preserved for the physics solvers that will use the new mesh. For a mesh of moderate size (a few million elements) sequential remeshing may suffice, but for a billion elements or more, parallel mesh generation is necessary.