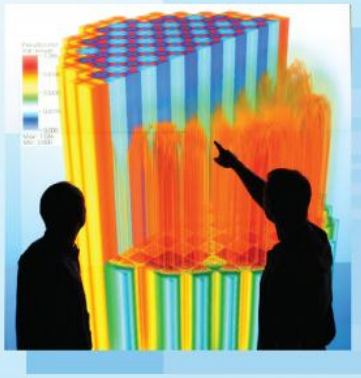
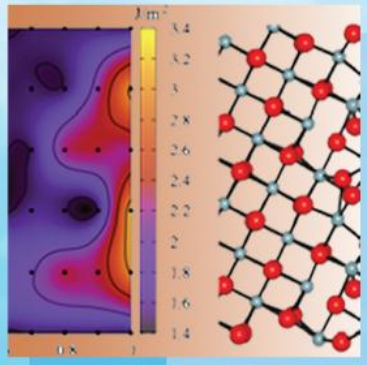
CASL-U-2011-0019-000

Power uprates
and plant life extension



Engineering design
and analysis

Science-enabling
high performance
computing



Fundamental science

Plant operational data

Consortium for Advanced Simulation of LWRs

L3:VUQ.VVDA.P2-2.04

Kevin Copps
SNL

Completed: 6/30/11

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

Sandia National Laboratories

Operated for the U.S. Department of Energy's
National Nuclear Security Administration
by **Sandia Corporation**

Albuquerque, New Mexico 87185

date: July 1, 2011

to: James R. Stewart, MS-1318 (01441)
Randall M. Summers, MS-1321 (01444)
CASL SharePoint site (distribution)

from: Kevin D. Copps, MS-0897 (01544)

subject: Open source the *Percept* software and demonstrate verification of a Grid-to-Rod Fretting problem
(Completion of CASL L3 Milestone VUQ.VVDA.Y1-2.04)

Summary

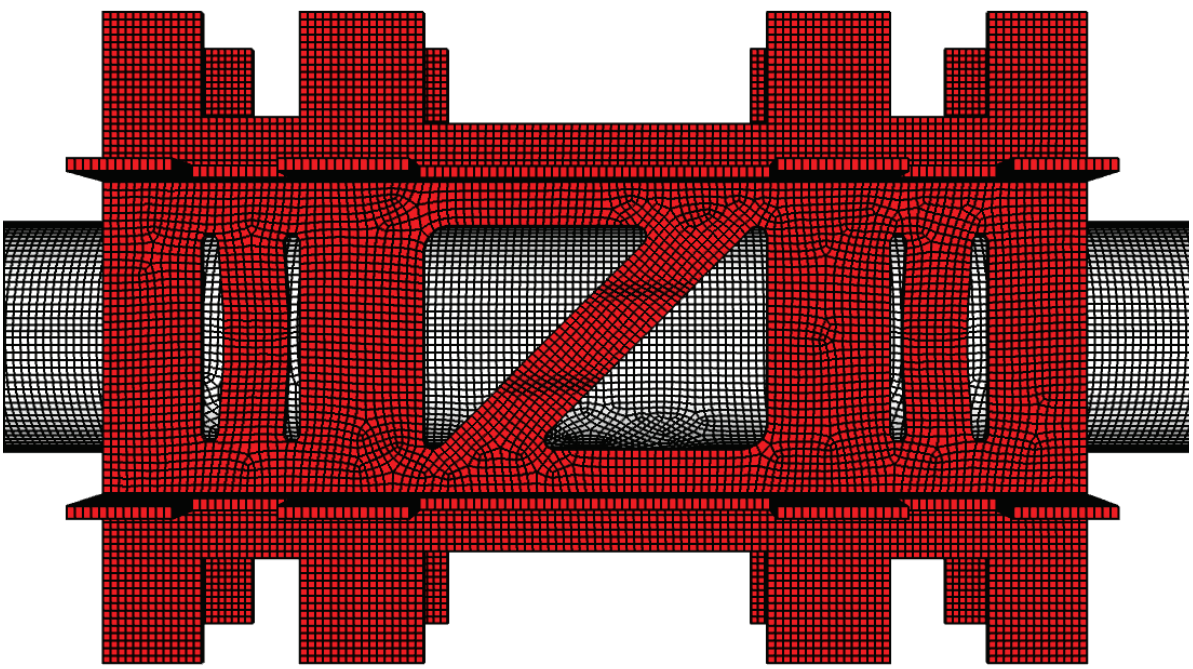
The CASL L3 Milestone (VUQ.VVDA.Y1-2.04)—the last phase of which was to use the new *Percept* verification tools software on a demonstration problem—was completed in June 2011. Because of limited availability of CASL project funds at Sandia National Laboratories, this milestone was completed at a reduced scope. The original intended scope included executing multiple solid mechanics computer simulations of a GTRF (Grid-to-Rod Fretting) problem with successively refined meshes, and investigating the grid convergence of contact energy. Although at reduced scope, we demonstrated that the new *Percept* software package indeed efficiently produces a sequence of refined meshes in parallel—including maintaining fidelity with the original CAD geometry—but the project team did not execute the relatively more computationally expensive solid mechanics simulation code.

This CASL project involved the creation of a new open source software package, *Percept*. The Percept package is a collection of tools for verifying simulation codes and solutions. The development of Percept is also partially funded by the DOE ASC program for application to the safety and surety of nuclear weapons. Percept software is now under an open source license and exists as part of the Sandia National Laboratories *Trilinos* software suite (http://trilinos.sandia.gov). Percept executes on distributed parallel machines and is scalable on super computing hardware. Some of the current tools in Percept include providing analytic functions for the method of manufactured solutions, postprocessing of quantities of interest, reporting element quality, and uniform and locally adaptive mesh refinement. Mesh operations in Percept, such as refinement can be integrated with CAD geometry models such that nodes, edges, faces can be matched to the high fidelity 3D curved surfaces of the CAD solid geometry model.

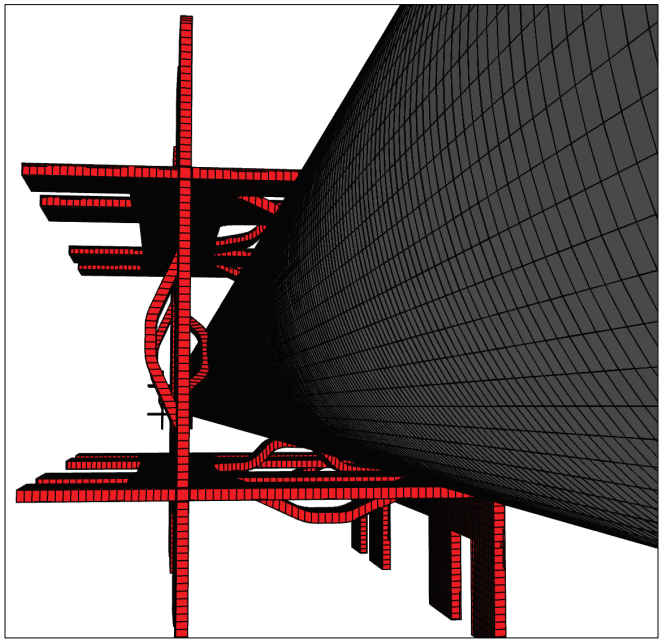
Background

Analysis and improved understanding of the vibrations of fuel rods caused by fluid turbulence and radial velocities is important for reducing grid-to-rod-fretting (GTRF) wear of nuclear fuel assemblies. This wear impacts the cost, efficiency and life of pressurized and light water reactors (PWR and LWR). Numerical modeling and computer simulation of the GTRF phenomena are central to the mission of the Consortium for Advanced Simulation of LWRs (CASL) DOE Energy Innovation Hub. The GTRF is a multi-physics phenomenon, wherein transient fluid/hydraulic forces are exerted on the nuclear fuel rod resulting in a solid mechanical vibrational response. A fluid/solid coupled physics model provides better predictions of fuel rod vibration and a more accurate representation of the excitation forces and important quantities of interest (Elmahdi, et al. 2011).

The verification of software codes and the verification of approximations produced by numerical modeling software is necessary if large-scale computer modeling and simulation is to be predictive. The Percept software is intended to satisfy the need for various kinds of tools to verify codes and calculations. One critical need is a capability to efficiently produce sequences of refined grids, or meshes—this is a necessary ingredient in the analysis of numerical convergence. The component parts of the typical nuclear fuel assembly in LWR design involves complex geometry—modeled using CAD software—usually as part of the spacer grid between rods, which also serve the purpose of directing and controlling fluid flow.

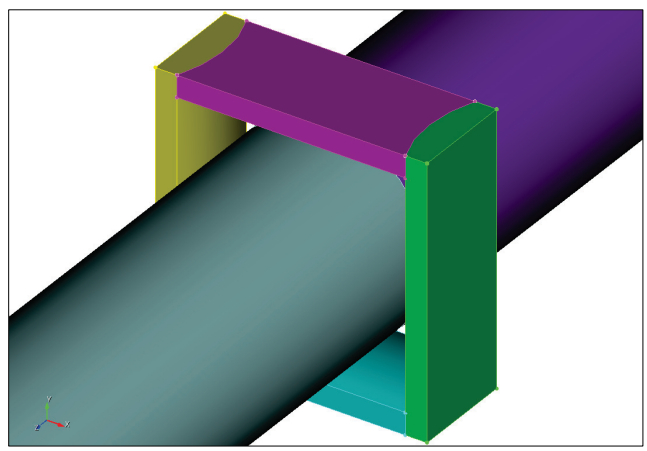


The CUBIT meshing software was used to create the computational grid used for solid mechanics analysis of a CASL GTRF solid mechanics problem (Rick M. Garcia, SNL). A variant of a mesh of this type will become input to the Sierra Mechanics solid mechanics application, Presto (SNL Engineering Sciences Center 2010-3112). The Percept software was used to create uniformly refined meshes starting with this coarse mesh in the ExodusII format (Sjaardema, Yarberry and Schoof 2006) and an associated openNURBS (http://www.opennurbs.org) CAD geometry file output by CUBIT (http://cubit.sandia.gov).



For this milestone, a significant portion of demonstrating Percept is creating the sequence of refined meshes on a decoupled solid mechanical model of the grid-to-rod-fretting problem. The solid mechanics model was created as part of the solid mechanics task of the GTRF CASL effort (Crane 2011), along with a corresponding fluids model (Rodriquez and Turner 2011).

Variants of the solid mechanics model that may be used for some simulations—possibly with no loss in predictive accuracy—include simplifications to the *spacer grid* (shown in red in the complex configurations).



A simplified form of the spacer grid is represented in this CAD solid model, shown at right, versus the complex versions (shown in red, above). In this form, the contact between the cylindrical surfaces of the fuel rod assembly is also simplified.

Percept Software Package

The *Percept* software package consists of multiple binary compiled libraries, which may optionally be linked to other modeling and simulation applications. Percept source code is primarily written in the C++ language. An additional large part of Percept is Python language modules, which in turn depend on dynamically linked (shared object) versions of the libraries. The Percept team intends the Python modules to provide a dynamic and flexible way for analysts and code developers to take advantage of the capabilities provided by the high performance C++ libraries.

The Percept libraries are currently part of the *STK* package within Trilinos. In the near future, the Percept team intends to revise the open source licensing in order than Percept be promoted to be its own package within the Trilinos hierarchy. Percept will continue to heavily depend on other modules of the Trilinos STK package, as well as on other packages in Trilinos and other external packages such as MPI.

Command line tools written in Python provide an easy to use interface to Percept binary executable. An excerpt from the command line help documentation is shown here:

```
usage: percept [-h] [--version] [-C DIR] [-j NP] [--proc-rank-field]
               [-v VERBOSE] [--echo-command-line] [-o OUTPUT_LOG]
               [--show-defaults] [--pause-for-debugging] [--pout POUT]
               [--dout DOUT] [--writer WRITER] [--timer TIMER]
               {enrich,convert,refine} ... infile [outfile]

Percept is a tool for verifying modeling and simulation codes and solutions.

positional arguments:
  {enrich,convert,refine}
  infile                input mesh file name
  outfile               output mesh file name

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -C DIR, --directory DIR
                        change working directory

parallel control:
  -j NP, --num-proc NP  number of processors to use (default: 1)
  --proc-rank-field     store the processor-rank in an element field on mesh
                        output

informative output:
  -v VERBOSE, --verbose VERBOSE
                        print more verbose information int >= 0 (default: 0)
  --echo-command-line   output original command line and continue as normal
  -o OUTPUT_LOG, --output-log OUTPUT_LOG
                        path where output log file will be written (default:
                        percept.log)
  --show-defaults       show percept defaults

diagnostic and debugging:
  --pause-for-debugging
                        wait for your input to allow attaching debugger
  --pout POUT           path where per processor log files will be written
                        (default: -)
  --dout DOUT           path, or stream (one of cout, cerr), where diagnostic
                        log file will be written (default: cout)
  --writer WRITER       name of the diagnostic writer, for example "all"
  --timer TIMER         name of the diagnostic timer, for example "mesh"

Some commonly used percept commands are:
   refine    Divide blocks of elements uniformly
   convert   Convert blocks of elements to another specified type of element
   enrich    Add more nodes to blocks of elements to increase polynomial order

See "percept COMMAND --help" for more information on a specific command.
```

Additional help for specific Percept subcommands is available. An example is the documentation for the refine command:

```
usage: percept refine [-h] [-n NUMBER_REFINES] [-t REFINE_TYPE]
                      [-b BLOCK_LIST] [-B BLOCK_FILE]

optional arguments:
  -h, --help            show this help message and exit
  -n NUMBER_REFINES, --number-refines NUMBER_REFINES
                        number of refinements (default: 1)
  -t REFINE_TYPE, --refine-type REFINE_TYPE
                        type of elements to refine into another type (one of:
                        Quad4_Quad4_4, Tri3_Tri3_4, Tet4_Tet4_8, Hex8_Hex8_8,
                        Wedge6_Wedge6_8, Tri6_Tri6_4, Quad9_Quad9_4,
                        Hex27_Hex27_8, Tet10_Tet10_8, Wedge18_Wedge18_8,
                        ShellTri3_ShellTri3_4, ShellQuad4_ShellQuad4_4)
  -b BLOCK_LIST, --blocks BLOCK_LIST
                        include or exclude block numbers in a quoted, comma
                        delimited string.
  -B BLOCK_FILE, --blocks-from BLOCK_FILE
                        include or exclude block numbers listed in a file.
```

Percept uses MPI communication to execute in parallel on distributed super-computing architectures. Related STK modules perform most of the work of data management and synchronization, primarily the `stk_mesh` and `stk_io` modules.

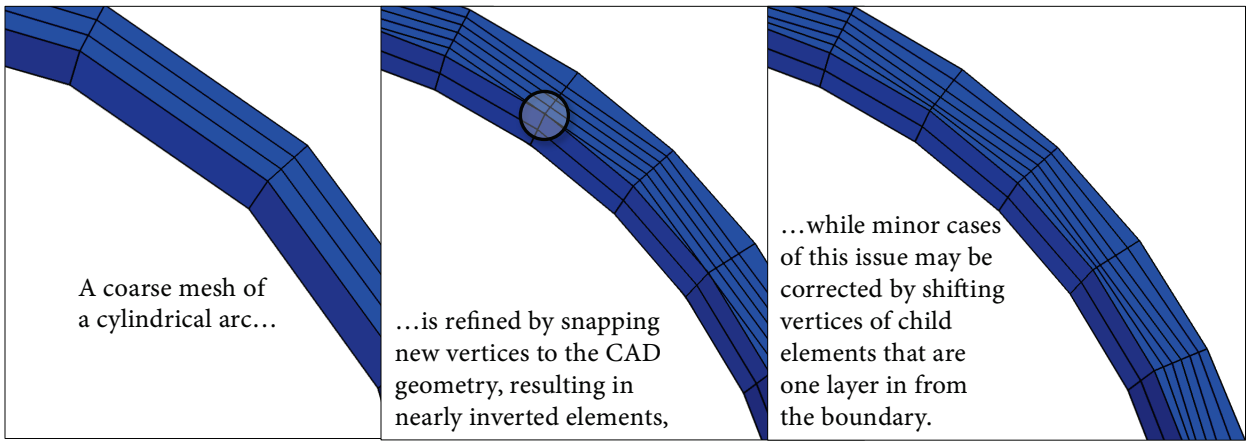
Parallel Uniform Refinement

The critical use case for solution verification is creation of a sequence of uniformly refined meshes. If a numerical modeling simulation cannot converge quantities of interest under uniform mesh refinement in spatial discretization (and, similarly refinements in the time discretization), serious issues are almost certainly the cause. For simulations at the petascale, hundreds of millions of grid cells, or elements, are required. A proper verification study should contain a sequence of three or four grids at successively finer resolution.

Convergence with respect to spatial discretization often occurs far more uniformly and monotonically when mesh sequences are *embedded,* in the sense that the function space, or basis, of the coarser grid is contained within the finer grid. Therefore, one of the design goals of the initial version of Percept refinement is that the refined sequences exhibit this embedding of child elements within a coarser parent element, at least wherever possible. Maintaining the embedding is not possible near curved boundaries, at least if finer meshes are constructed according to curved geometry such that geometric error is also reduced. Percept will attempt to maintain the embedding property at all child elements that are within one parent removed from part/component external boundaries.

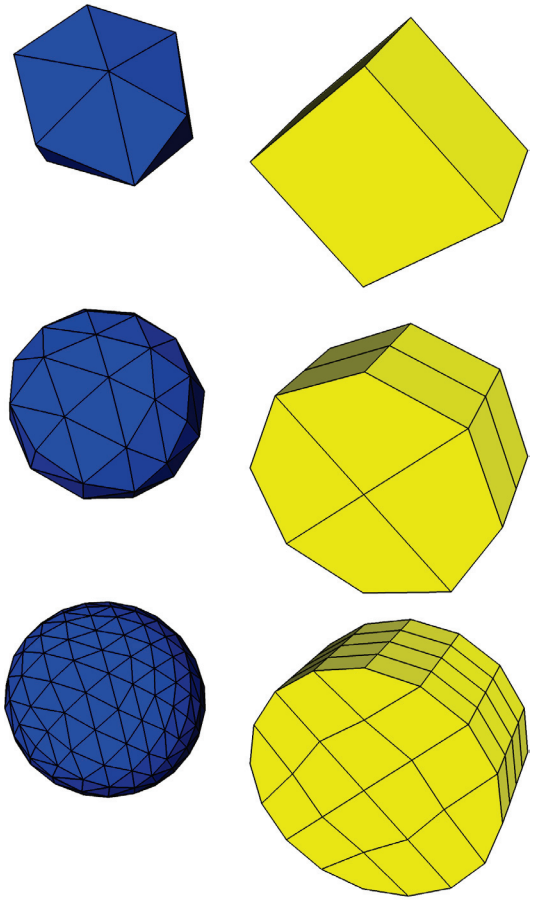
Reducing error in the discretization of geometry requires that the child vertices are placed with respect to CAD geometry definitions. Dynamically changing part boundaries and material interfaces is another difficult issue, and this is not addressed in the current version of *Percept* software. The CUBIT mesh generation software can export an associated geometry definition in the form of an openNURBS geometry file format (Robert McNeel and Associates 2011). At the same time, an associated ExodusII mesh database is also output containing references to the openNURBS geometry. The *Percept* package has been developed such that we can read a coarse mesh and geometry, and use the geometry definitions during parallel refinement.

One issue with modifying meshes that are initially well formed, while respecting curved geometry, is that it is easy to form highly deformed elements—even inverted elements. The cylindrical ring is representative of the cladding layer of a nuclear fuel rod, and the refinement of one level shows that child elements near the domain boundary are nearly inverted.



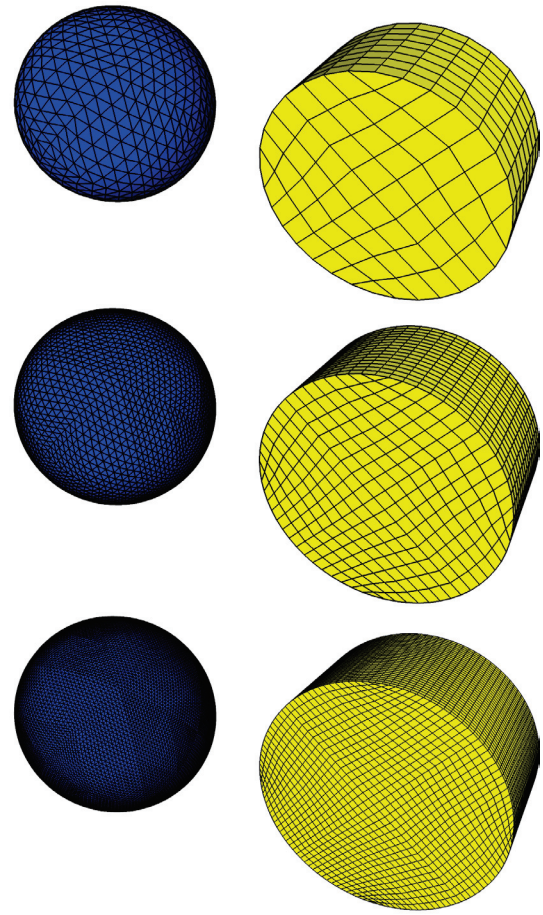
Violating the embedding property further, Percept may be able to fix badly formed elements due to maintaining boundary geometry by shifting positions of child elements near the boundary. Better smoothing capabilities are required for other cases. Additional smoothing strategies may be added to Percept in later versions.

The reduction in geometry error and the extreme levels of refinement that are possible using Percept are not easily visible in pictures of the CASL solid mechanics meshes. In addition, complex and arbitrarily trimmed NURBS curves and surfaces are not yet fully supported by the CUBIT exporting capability for subsequent reading by Percept. These should be supported before the end of the year 2011. For production use, all quadrics, such as spheres, cylinders, parabolas, etc., are currently supported. Because of these reasons it is worthwhile to examine in detail a sequence of uniform refinements on an extremely coarse initial mesh representation of a sphere and cylinder.



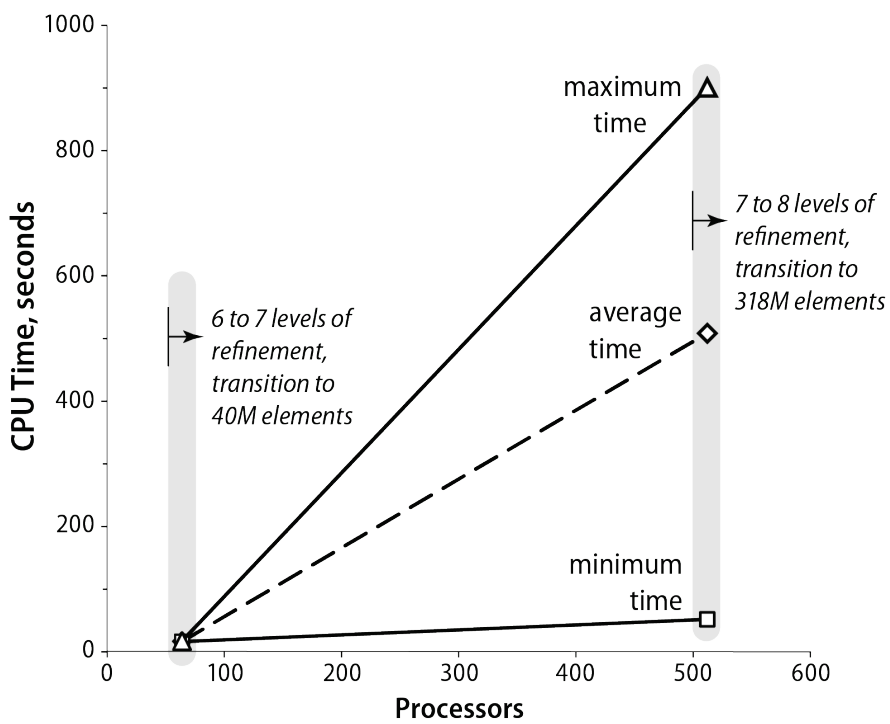
We begin with a coarse mesh, modeling a disconnected sphere and cylinder geometry, wherein the sphere is represented by tetrahedral and the cylinder by hexahedra, respectively. These meshes are refined a total of eight levels, where we denote the initial coarse mesh as the level zero mesh. The number of elements present in each of the meshes is listed in the following table. The sequence of meshes is shown at the right; only levels 0 through 5 are included here.

Refine Level	Number of Tetrahedra	Number of Hexahedra	Number of Elements
0	18	1	57
1	144	8	288
2	1,152	64	1,728
3	9,216	512	11,712
4	73,728	4,096	85,632
5	589,824	32768	653,568
6	4,718,592	262,144	5,104,128
7	37,748,736	2,097,152	39,845,888
8	301,989,888	16,777,216	318,767,104



The meshes that end in levels 1 through 5 can be produced within seconds using Percept on a workstation with 8 processors and 24 GB of system memory. The level 6 mesh can be produced on this class of workstation with some wait time. The level 7 and 8 meshes were produced using the *Red Sky* platform at SNL. These were executed multiple times in order to report average CPU time. The level 7 to 8 transition was executed four separate times, and averaged. Surprisingly, these runs on Red Sky produced wide variations in CPU times. We would expect nearly a flat line to show weak scaling from 6 to 7 and 7 to 8, because the number of processors we used (from 64 to 512) grows at the same rate of 8x in number elements. This variation may be due to resource contention, and we hope to further investigate this anomaly.

A scaling study performed using Percept on the Red Sky computer resulted in a wide variation of CPU times to complete the transition from 7 to 8 levels of uniform refinement. Similar runs on other hardware platforms related to work for the NNSA/ASC program have not shown this behavior, which suggests further investigation on the Red Sky platform.



Next Steps

We will continue our work to demonstrate Percept on the solid mechanics GTRF related problem. We will compute Sierra Mechanics (Presto) simulations on the refined sequence of meshes, and use Percept where applicable to analyze convergence rates of the various energy quantities, including contact energy between the grid spacer and surface of the fuel rod cladding.

Farther out, in FY12 activities for CASL, we will work on a new milestone to perform verification on a coupled fluid/mechanical GTRF simulation.

Acknowledgements

William Rider (SNL) coordinated the collaboration between VUQ and the GTRF analysis teams. Percept would not be possible without the work of Steve Kennon (The Numericus Group), whose contributions to separate contributions to Percept software are funded through the DOE NNSA ASC program. Direct contributors of the geometry and CUBIT integration for Percept were Mark Dewey (ETI) and Matt Staten (SNL).

The simplified 3D model of the solid mechanical assembly for the GTRF problem relied on problem statements, models, and technical guidance provided by Westinghouse Electric Company LLC. Other direct contributors to the solid mechanical GTRF model definition setup and support were primarily Nathan Crane (SNL), with help from Rick Garcia (SNL), and Mary White (SNL). Dan Turner (SNL) and Salvador Rodriguez (SNL) supported the fluids GTRF model definition. SNL's Computer Science Research Institute, and SNL's Computational Computing and Network Services Center provided hardware and software support.

Citations

Crane, Nathan. *Analysis integration procedures for Grid-to-Rod Fretting (GTRF) test problem.* communication, Dept. 01542, Computational Solid Mechanics and Structural Dynamics, Engineering Sciences Center, Albuquerque: Sandia National Laboratories, 2011.

Elmahdi, A. M., R. Lu, M. E. Conner, Z. Karoutas, and E. Baglietto. "Flow Induced Vibration Forces on a Fuel Rod by LES CDF Analysis." *The 14th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-14).* Toronto, Ontario: Westinghouse Electric Company LLC, 2011. NURETH14-365.

Robert McNeel and Associates. "openNURBS 5.0 C++ Toolkit Documentation." *openNURBS Initiative.* February 2, 2011. http://www.opennurbs.org/docs.htm (accessed June 30, 2011).

Rodriquez, S. B., and D. Z. Turner. *Assessment of existing Sierra/Fuego capabilities related to Grid-to-Rod Fretting (GTRF).* Albuquerque, NM: Sandia National Laboratories, 2011.

Sjaardema, G. D., V. R. Yarberry, and L. A. Schoof. *EXODUS II: A finite element data model.* Technical Report Technical Report SAND92-2137, Org. 01544, Simulation Modeling Sciences, Albuquerque: Sandia National Laboratories, 2006.

SNL Engineering Sciences Center. *Presto 4.16 User's Guide.* SAND, Dept. 01542, Computational Solid Mechanics and Structural Dynamics, Albuquerque, NM: Sandia National Laboratories, 2010-3112.