Optimizing CAD and Mesh Generation Workflow for SeisSol

Sebastian Rettenberger*, Cameron Smith†, Christian Pelties‡
*Department of Informatics, Technische Universität München, Munich, Germany
†Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY, U.S.A.
‡Department of Earth and Environmental Sciences, Ludwig-Maximilians-Universität München, Munich, Germany

I. Introduction

SeisSol is a simulation software for seismic wave propagation and earthquake scenarios. It solves the fully elastic wave equations in heterogeneous media. Incorporating dynamic rupture simulation it performs complex multiphysics earthquake simulations. To account for complicated geometries SeisSol uses a fully unstructured tetrahedral mesh. Recent publications [1], [2] have shown that SeisSol is able run at petascale performance on modern supercomputers.

In this poster we present SeisSol workflow optimizations for three scalability-critical stages: CAD model generation, parallel mesh generation, and initialization of the mesh database in SeisSol. CAD model generation is automated through a combination of point cloud surface meshing techniques and discrete surface boolean opertions. On these CAD models the PUMGen tool couples Simmetrix[1] unstructured parallel mesh generation to a novel parallel mesh file format. Using this format we were able to initialize a 3.6×10^9 element tetrahedral mesh within seconds on 9216 processes in SeisSol.

II. Automated CAD generation

Parallel mesh generation of complex domains requires creation of a valid CAD model. SeisSol CAD data comes from various sources. Typically, these sources consist of point clouds for topographical and sub-surface features, analytic surface definitions, and CAD geometry. Manual construction of a valid CAD model from these sources is tedious and error prone. It often consumes multiple weeks of researcher time.

Initial efforts to automate these processes in a robust manner have focused on point cloud and analytic surface data sources. Towards this a command-line-based workflow combines a Python PyProj-based[2] tool for projecting point cloud sets to a common reference frame, MeshLab[3] for surface reconstruction and trimming procedures, and Simmetrix GeomSim for discrete surface booleans. Using these tools a physically consistent model for the 1992 Landers fault system depicted in Figure 1 is generated from USGS DEM topological data and an interpretation of the sub-surface fault structure-based on the surface trace.

[1]http://simmetrix.com/
[2]https://code.google.com/p/pyproj/
[3]http://meshlab.sourceforge.net/

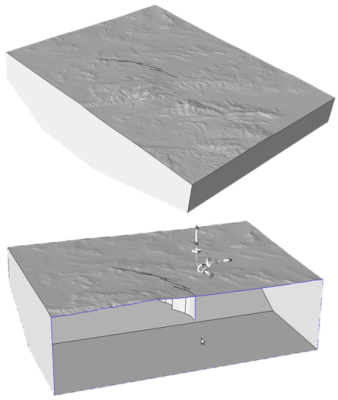

Fig. 1. Non-manifold 1992 Landers fault system CAD generated with automated workflow.

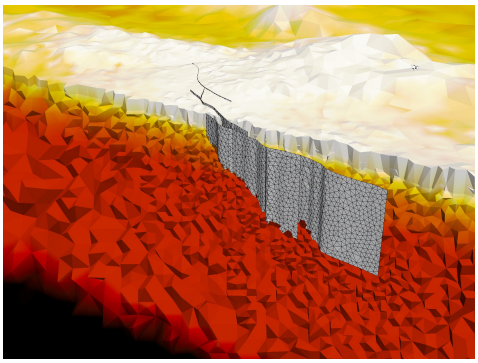

Fig. 2. Tetrahedral mesh of the 1992 Landers fault system generated with Simmetrix tools. For visualization purpose a mesh with 7.2 million elements was used. (Heinecke et al. [2])

III. Parallel mesh generation

The GAMBIT serial mesh generation tools have historically been used to generate unstructured meshes for SeisSol simulations. Given a complex geometric model though, such as Landers, additional geometric model faces are required to subdivide the domain and aid GAMBIT mesh generation element size gradation. The need for this additional geometry required iterative model and mesh control modifications, which often required expert CAD knowledge, and further increased the already high costs of simulation setup.

Efforts to automate mesh generation have focused on use of the Simmetrix CAD-based parallel mesh generation tools on as-simulated CAD models. Using these tools the 184 million element mesh depicted in Figure 2 has been successfully generated on the Landers model of Figure 1.

IV. Highly scaling mesh format

Previously, SeisSol initialization procedures read the entire mesh and partition file, generated with a graph-based partitioner such as ParMETIS [3], on each process. From this data the local elements were extracted and communication patterns determined. As the serial GAMBIT format used an ASCII format this approach required each process to iterate over the

entire mesh to locate local entity information. Directly parsing the ASCII-based files in SeisSol puts a high pressure on the underlying store system and does not scale; this usually limited meshes to only several million tetrahedrons. An alternative broadcast-based approach is also rendered impossible for the large meshes of interest by the limited on-node memory.

To avoid parsing variable length strings – as required for GAMBIT files – our new mesh format is based on netCDF-4[4]. Since netCDF-4 is based on HDF5 our reader can use parallel I/O and thus exploit parallel file systems of supercomputers such as GPFS or Lustre. However, to efficiently read the mesh in parallel each process must know a priori where its own part is stored in the file. Therefore, we include the partition information directly in the our mesh format. The first dimension of each array in the netCDF file is used to identify the partition of the data. Information on MPI boundaries – e.g. vertex coordinates – are stored multiple times in the file to simplify the access.

However, we do not only store the mesh in partitioned way but also explicitly include precomputed ghost layer information. With this optimization we avoid finding communication partners in parallel which usually requires complicated and expensive global communication patterns.

Generation of the SeisSol mesh format is through the MPI-based PUMGen tool. PUMGen currently supports reading and partitioning serial GAMBIT meshes or directly generating meshes in parallel using the Simmetrix Simulation Modeling Suite C++ API. Through the use of a modular design that seperates CAD, mesh, and model concerns [4] new mesh sources can easily be integrated.

Burstedde et al. [5] propose an alternative workflow. They have only a small unstructured mesh and use adaptive mesh refinement to generate quasi structured cells. Even so, the additional refinement does not allow to better resolve geometric features like topography or faults which is crucial for earthquake simulations. To avoid reading a mesh from file while maintaining the advantages of fully unstructured meshes integrating the mesh generator directly into the solver is also an option (e.g. [6]). In SeisSol, we decided against the integration to provide more flexibility for the user.

V. Results

In this section we analyze the performance of PUMGen and the initialization of the mesh in SeisSol. For converting a mesh with 191,098,540 tetrahedrons PUMGen takes 1 hours 32 minutes on eight dual socket 8-core Intel Xeon E5-2680 nodes. However, the performance is strictly dominated by reading the serial GAMBIT file which takes more than 80% of the time. This emphasizes that the GAMBIT mesh format is not designed to store meshes at this size. We will include results of the parallel meshing with PUMGen in the final version of the poster which avoids serial I/O completely.

The scalability of mesh initialization in SeisSol is tested using a cubic domain which is recursively subdivided into tetrahedrons. For this scenario the load is kept constant to exactly 400,000 tetrahedrons per process and node (SeisSol is implemented with hybrid MPI+OpenMP parallelization and requires only one process per node). We compare the results from SuperMUC[5] and Stampede[6] in Figure 3. On both systems we use the dual socket 8-core Intel Xeon E5-2680 nodes for reading the mesh but on SuperMUC it is stored on the GPFS file system while Stampede uses Lustre.

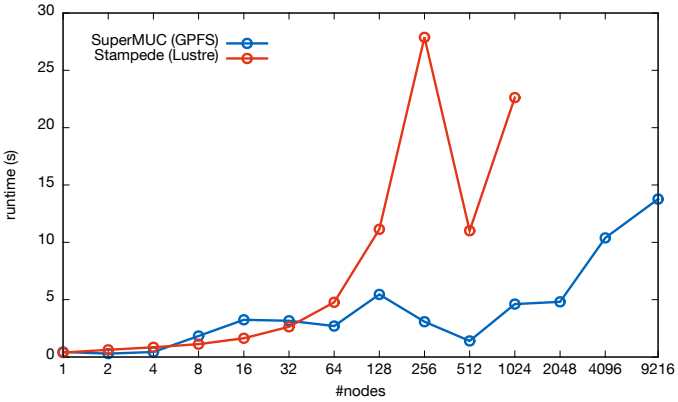

Fig. 3. Total time for reading and initializing a mesh from the new format. In all runs, the load was kept constant with 400,000 tetrahedrons per node. All measurements with less than 4,096 nodes were performed during normal user operation and include variations due to different occupancy rates of the file system.

References

[1] A. Breuer, A. Heinecke, S. Rettenberger, M. Bader, A.-A. Gabriel, and C. Pelties, "Sustained Petascale Performance of Seismic Simulations with SeisSol on SuperMUC," in *Supercomputing - 29th International Conference, ISC 2014*, ser. Lecture Notes in Computer Science, J. Kunkel, T. T. Ludwig, and H. Meuer, Eds., vol. 8488. Heidelberg: Springer, Jun. 2014, pp. 1–18.

[2] A. Heinecke, A. Breuer, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties, A. Bode, W. Barth, X.-K. Liao, K. Vaidyanathan, M. Smelyanskiy, and P. Dubey, "Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '14, Nov. 2014, accepted for publication.

[3] G. Karypis and V. Kumar, "A Coarse-Grain Parallel Formulation of Multilevel k-way Graph Partitioning Algorithm." in *SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 1997.

[4] M. S. Shephard, M. W. Beall, R. M. O'Bara, and B. E. Webster, "Toward simulation-based design," *Finite Elements in Analysis and Design*, vol. 40, no. 12, pp. 1575 – 1598, 2004, the Fifteenth Annual Robert J. Melosh Competition.

[5] C. Burstedde, G. Stadler, L. Alisic, L. C. Wilcox, E. Tan, M. Gurnis, and O. Ghattas, "Large-scale adaptive mantle convection simulation," *Geophysical Journal International*, vol. 192, no. 3, pp. 889–906, 2013.

[6] L. Carrington, D. Komatitsch, M. Laurenzano, M. M. Tikir, D. Michéa, N. L. Goff, A. Snavely, and J. Tromp, "High-frequency simulations of global seismic wave propagation using SPECFEM3D_GLOBE on 62K processors." in *International Conference for High Performance Computing, Networking, Storage and Analysis, 2008*. IEEE Press, 2008, pp. 60:1–60:11.

[4] http://www.unidata.ucar.edu/software/netcdf/

[5] http://www.lrz.de/services/compute/supermuc/

[6] https://www.tacc.utexas.edu/stampede/