SAND-2018-XXXX

Sandia National Laboratories

SAND2018-11508PE

Computational Multiphysics at Sandia National Laboratories

PRESENTED BY

Dr. Glen Hansen

Computational Multiphysics Department

Sandia National Laboratories

Glen Hansen's Background

- Ph.D. Computer Science 1996. Thesis on parallel numerical methods for stiff nonlinear problems (preconditioning). M.S. Mechanical Engineering, B.S. Petroleum Engineering
- PI/DGL LANL (1996-2006), PI/Group Leader INL (2006-2011)
- PI/Technical Manager, Computational Multiphysics Department @ Sandia (2011-Present)
- Research and development background:
  - Shock hydrodynamics, ALE methods, and mesh adaptivity
  - Newton-Krylov solution methods and code architecture
  - Multiphysics coupling methods
  - Software project leadership, software architecture for advanced computer platforms and HPC, simulation and analysis software development in C++

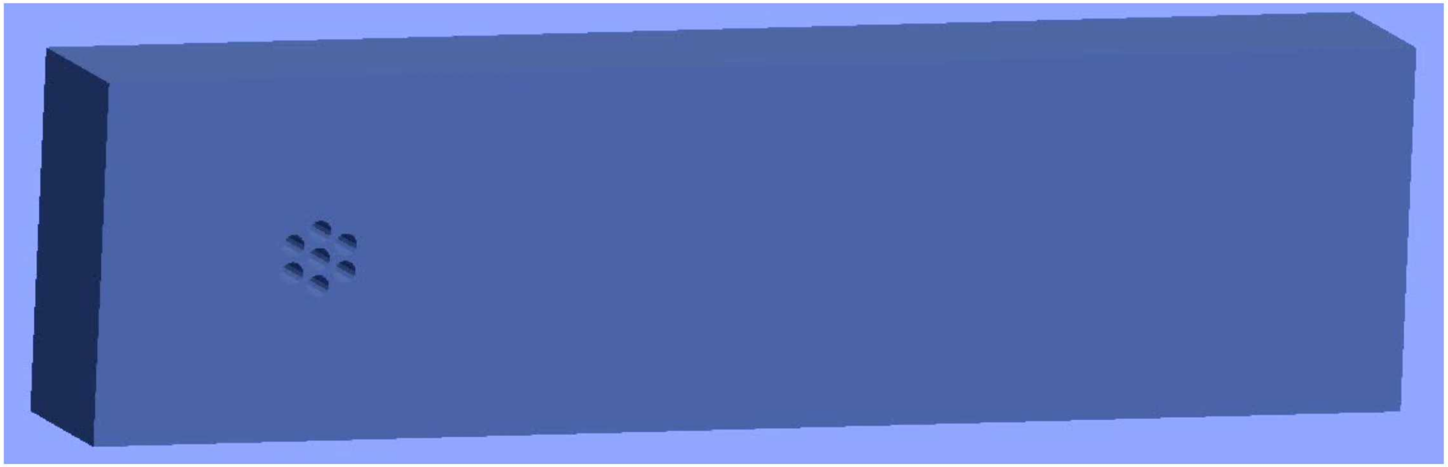
What is “Computational Multiphysics?”

- *The application of computation to coupled processes or systems involving more than one simultaneously occurring physical fields*
- Examples include:
  - Shock hydrodynamics + MHD
  - Rad hydro
  - Conjugate heat transfer + radiation
  - Fluid structure interaction (FSI)
  - Nuclear reactor modeling (CFD + neutronics + FSI + debris)
  - Pulsed power (plasma physics + terminal ballistics + many of the above)

Albany Code

https://gahansen.github.io/Albany/

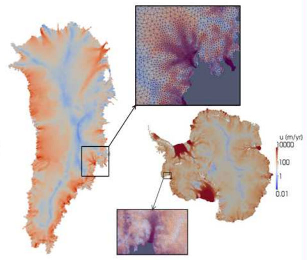
- Albany supports a wide variety of application physics areas including heat transfer, fluid dynamics, structural mechanics, plasticity, quantum device modeling, climate modeling, and many others
- Leverages Trilinos framework for linear and nonlinear solvers, preconditioning, load balancing, hardware portability layers, finite element data structures and multiphysics code support infrastructure
- Advanced analysis capabilities (embedded SA and UQ), supported by parallel scalable RPI adaptive meshing technologies, support for topology optimization



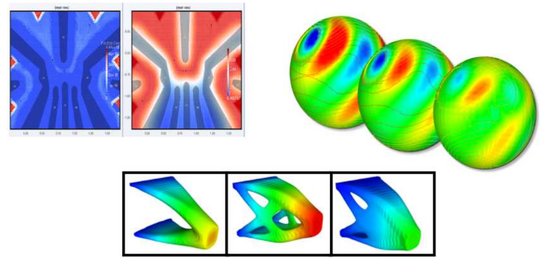
Albany Ecosystem

Application Impact: Ice Sheets

- Surface flow velocities for Greenland and Antarctic Ice Sheets
- Demonstrates nonlinear solves, linear solves, UQ, adaptivity, and performance portability
- Employs automatic differentiation, discretizations, partitioning, mesh database

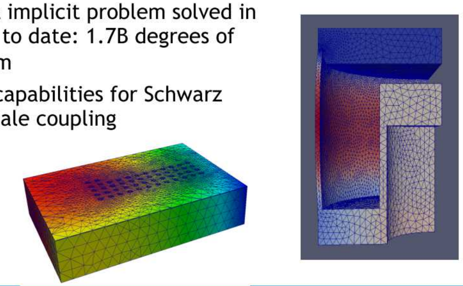


Additional Application Impact



Application Impact: Computational Mechanics

- Largest implicit problem solved in Albany to date: 1.7B degrees of freedom
- Initial capabilities for Schwarz multiscale coupling



Application

Albany

DAKOTA

Trilinos

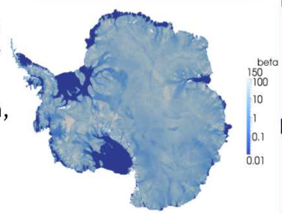
PUMI

Parallel Unstructured Mesh Infrastructure

Nonlinear Solvers and Inversion

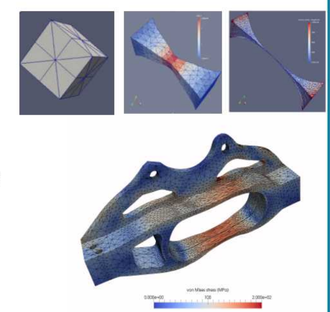
- Homotopy and Anderson Acceleration in Trilinos::NOX
- The robustness of nonlinear solvers are critical when an application is to be called as a sub-component within a larger application code.
- Uses Automatic Differentiation, Preconditioning, Optimization algorithms from Trilinos

$$\frac{dg}{dp} = \frac{\partial g}{\partial x}^T \frac{\partial f}{\partial x}^{-1} \frac{\partial f}{\partial p} + \frac{\partial g}{\partial p}$$



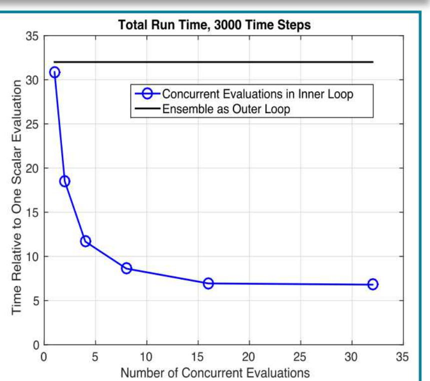
Mesh Adaptivity

- Mesh adaptation can be essential for efficiency and robustness
- Cube geometry subjected to large deformation (elasticity and J2 plasticity results shown)



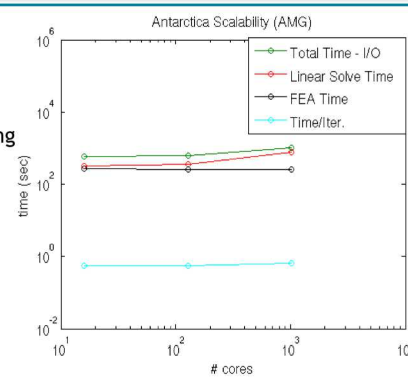
Embedded UQ

- New *Ensemble* data type in Sacado package
- Vectorization of kernels over ensembles
- Contiguous memory access in arrays



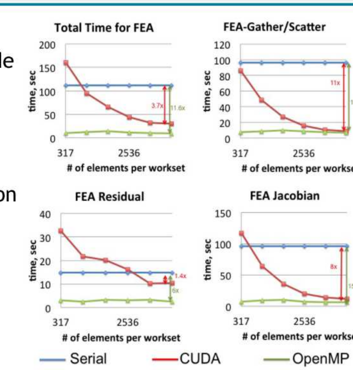
Scalable Linear Algebra

- Scalability of simulations requires effective preconditioning
- Multi-level solves are essential for the largest problems



Performance Portability

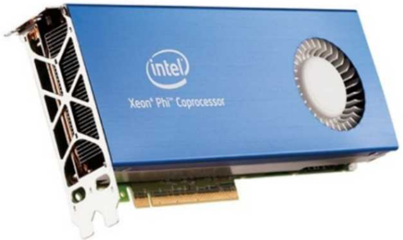
- The Kokkos programming mode supports performance portability of kernels.
- Kokkos' abstraction layer allows code to be tailored for specific devices



Performance portability using *Kokkos*

Strategy to extend *Albany* to provide good performance *on **new architecture machines*** (hybrid systems) and ***manycore devices*** (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.)

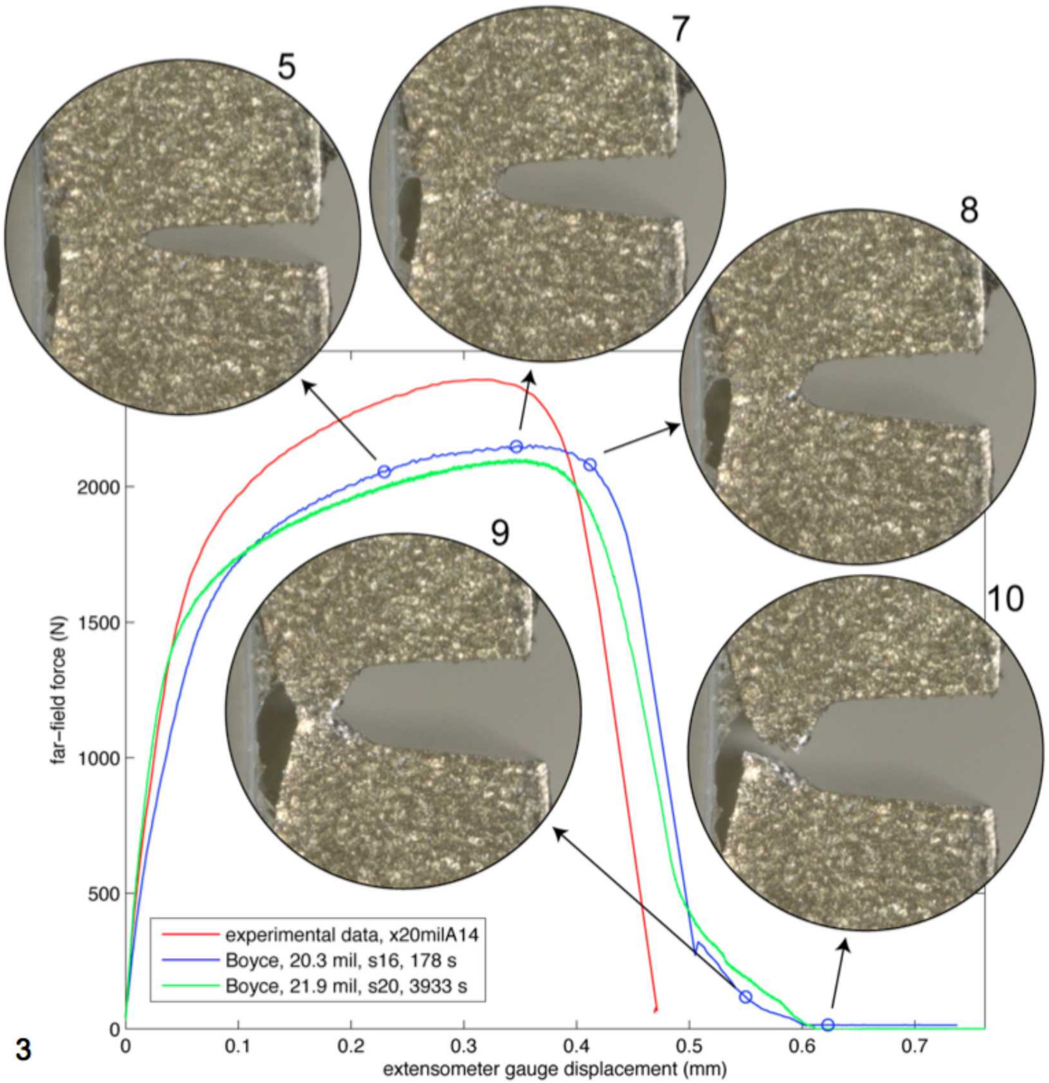
- **Kokkos**: *Trilinos* package with the goal of providing performance portability across diverse devices having different memory models.
- The *Kokkos* strategy: write an algorithm once, and change a template parameter to get the optimal data layout for the target hardware.

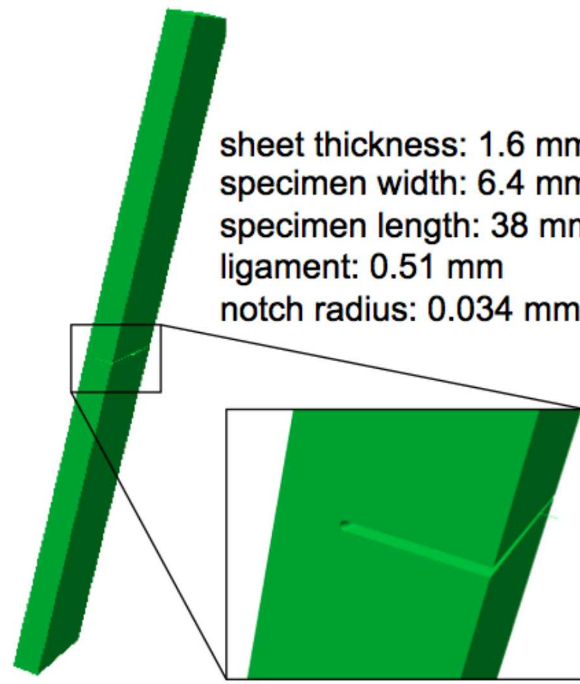


Albany Example I

https://github.com/gahansen/Albany/wiki/PAALS-Tutorial-2014

- Weld Failure: Motivation for the necessity of mesh adaptation



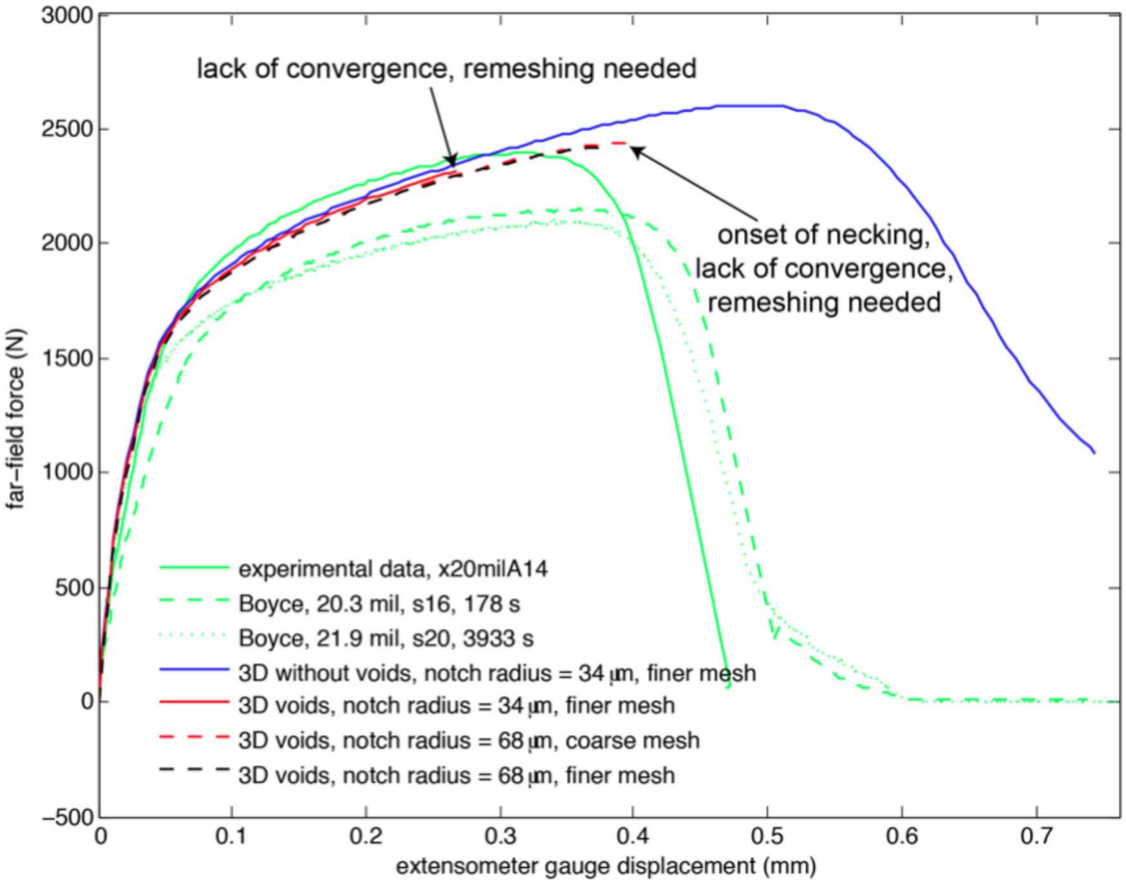


- *Predict the peak load and the rapid drop in the load-bearing capacity of the weld.*

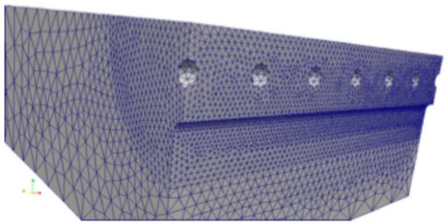
c.f., J. Foulk, etal; USNCCM 13 Presentation, July 30, 2015, SAND2015-9289C

Albany Example I

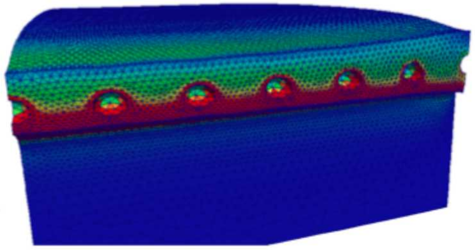
- Comparison of the computed load-displacement curve vs. experiment.



notch radius: 68 μm



onset of necking
notch radius = 68 μm
coarse mesh

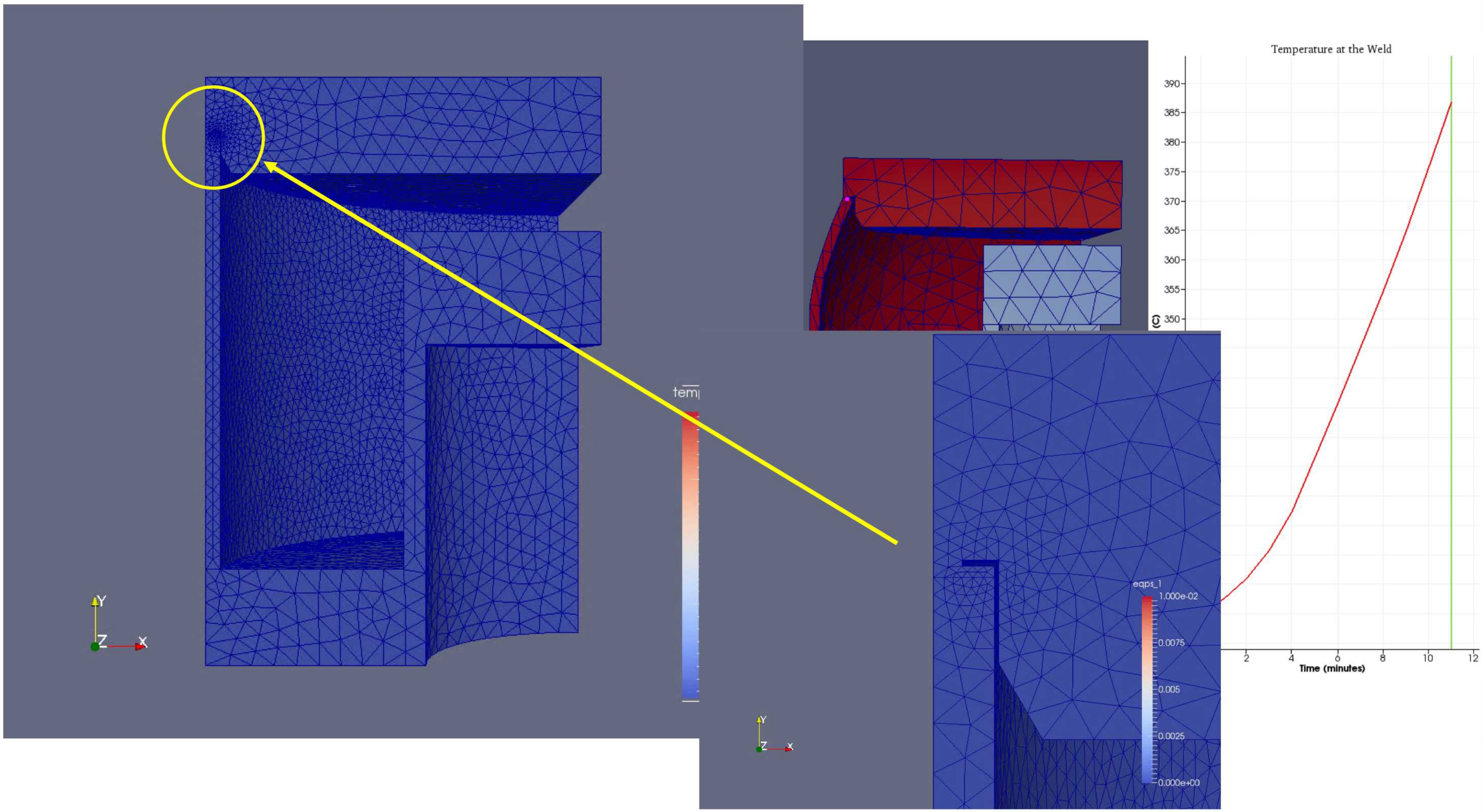


c.f., J. Foulk, etal; USNCCM 13 Presentation, July 30, 2015, SAND2015-9289C

Albany Example II

https://github.com/gahansen/Albany/wiki/PAALS-Tutorial-2015

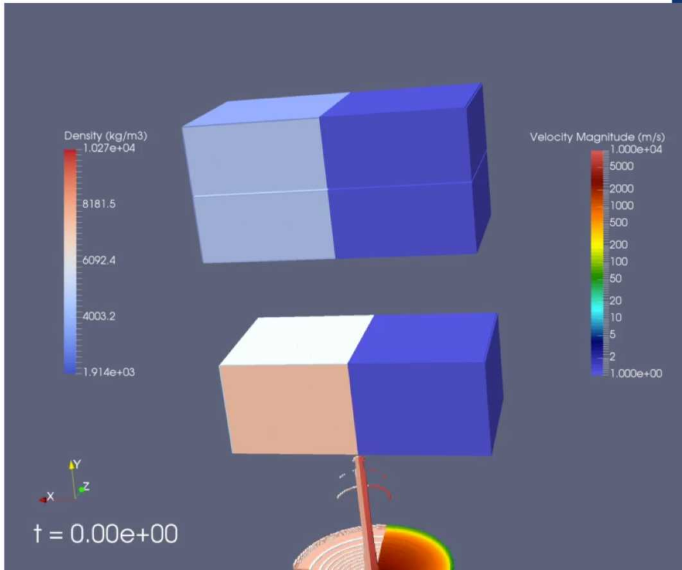
- Coupled thermomechanics, J2 plasticity, temperature dependent material hardening at the weld



ALEGRA SHOCK & MULTIPHYSICS

http://www.cs.sandia.gov/ALEGRA/Base_Alegra.html

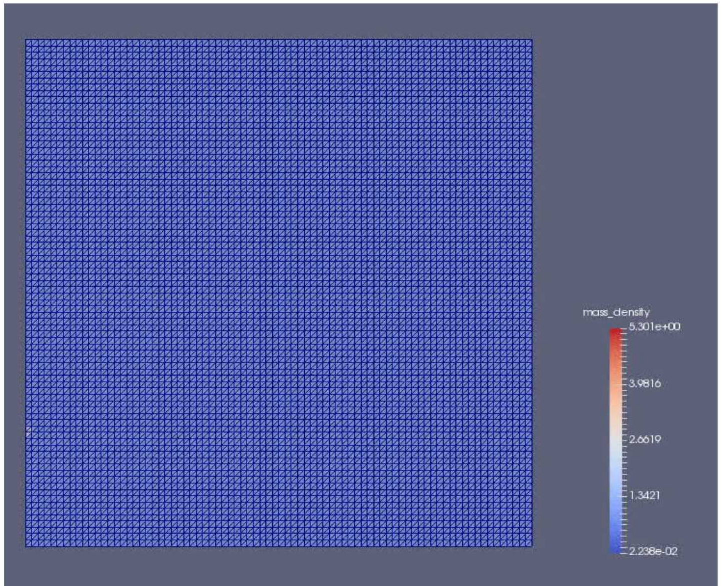
- ALE code -- Arbitrary Lagrangian-Eulerian -- provides flexibility, accuracy and reduced numerical dissipation over a pure Eulerian code; modern remeshing technology allows for robust mesh smoothing and control.
- Hydrodynamic and solid dynamics
- Models large distortions and strong shock propagation in multiple-materials
- Finite element code; descendent of PRONTO and uses some CTH Eulerian technology
- Energy deposition and explosive burn models
- Geometry -- 2D/3D Cartesian, 2D cylindrical

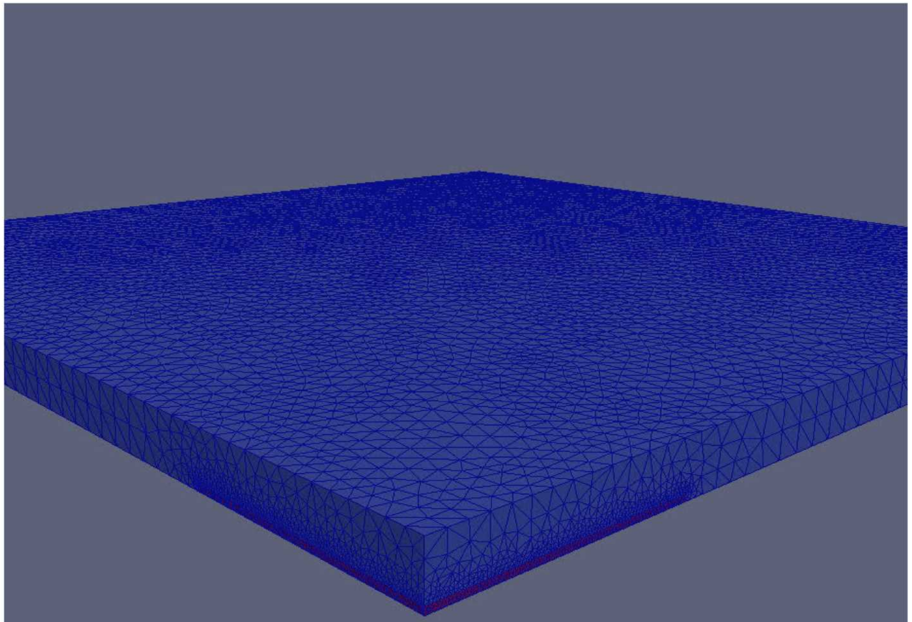


The LGR-TK Code (LaGR is not ALE)

https://github.com/SNLComputation/lgrtk

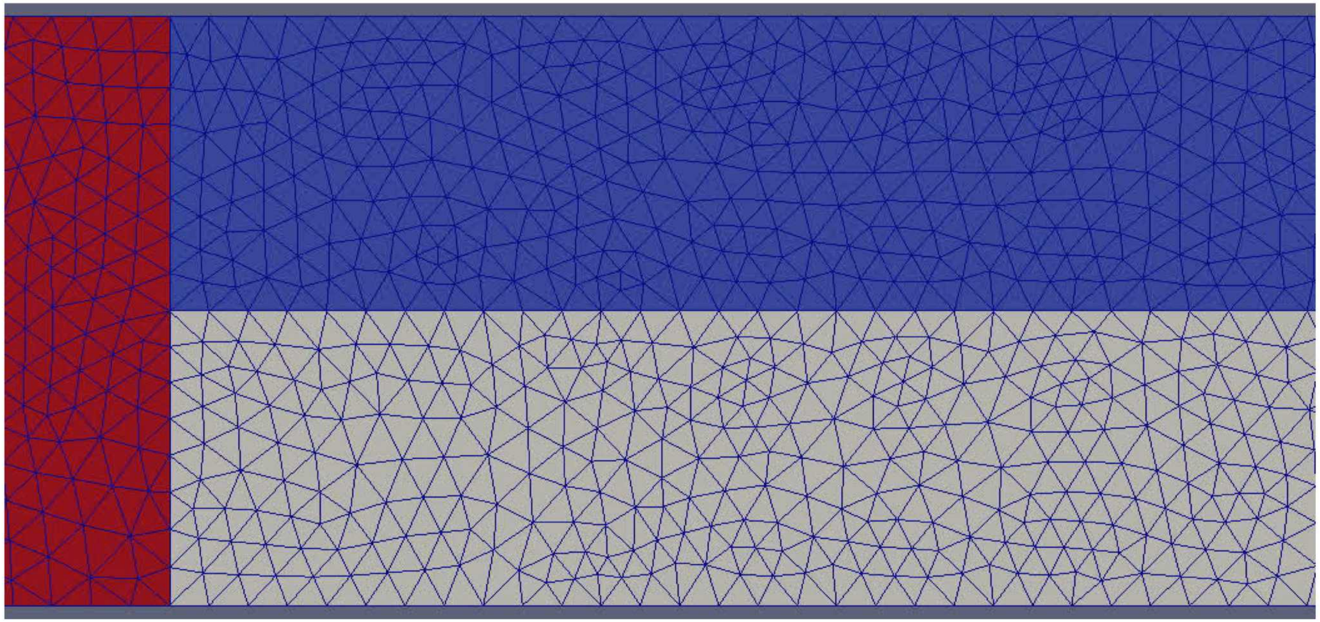
- Portably Performant: runs efficiently on most modern hardware, especially NVIDIA GPUs
- Adaptive Lagrangian: Like ALE (Arbitrary Lagrangian-Eulerian) except no Eulerian component, does adaptation (remeshing) by local modification, maintains single-material elements
- Shock Hydrodynamics: Compressible gases, elastic and plastic solids, and more material models combined to support true multi-material simulations
- All in 3D!



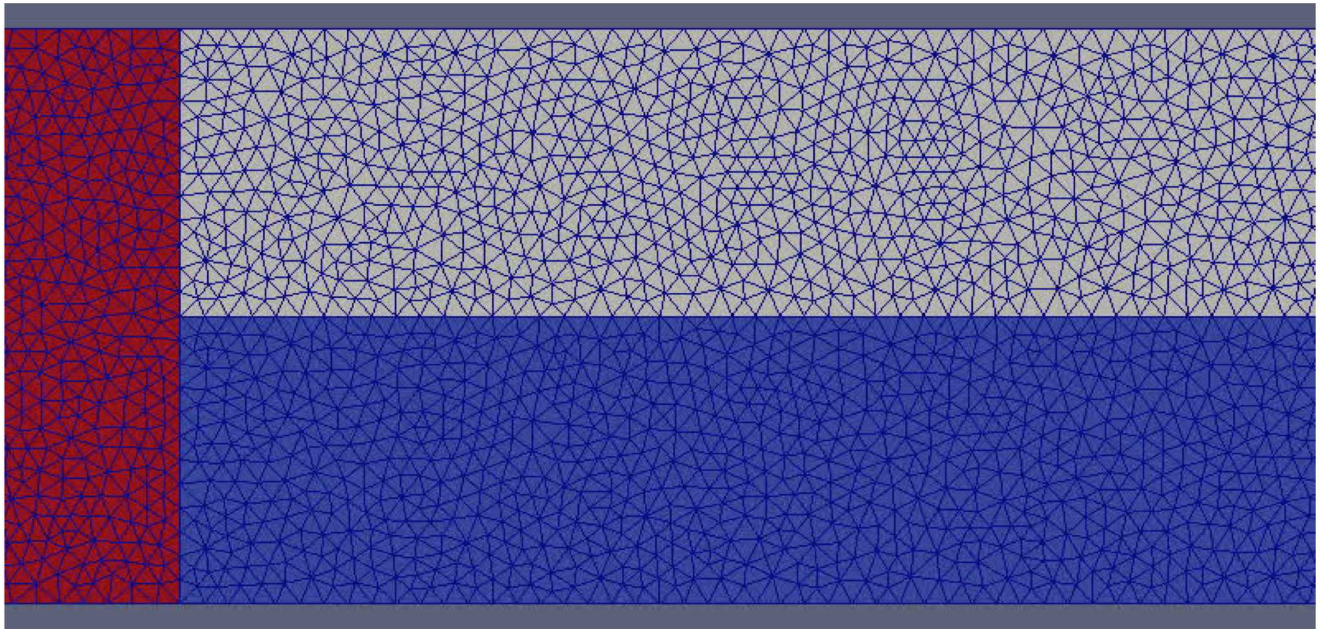


Triple Point Example

- Lagrangian

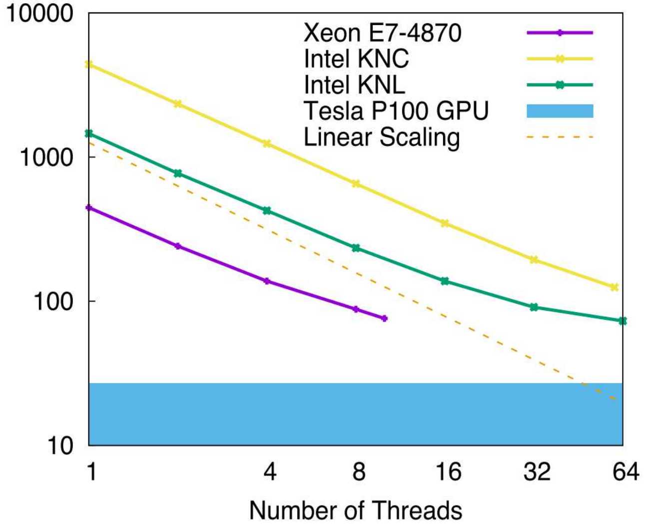


- Adaptive Lagrangian

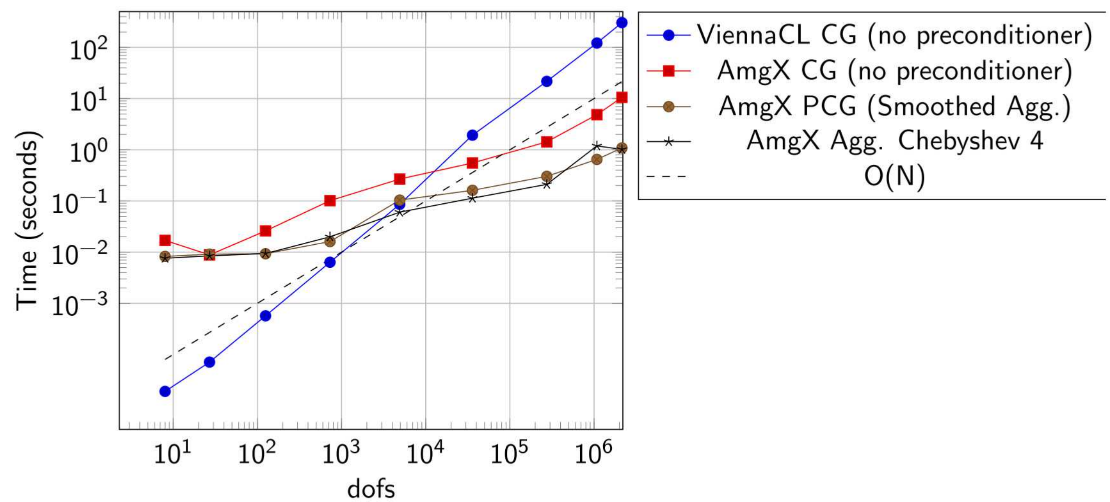


LGR-TK Performance Portability

- Kokkos is used for on-node parallelism
- Good performance across Intel Xeon Phi and NVIDIA cards



White P100 3D FEM (higher-contrast Poisson), performance comparison



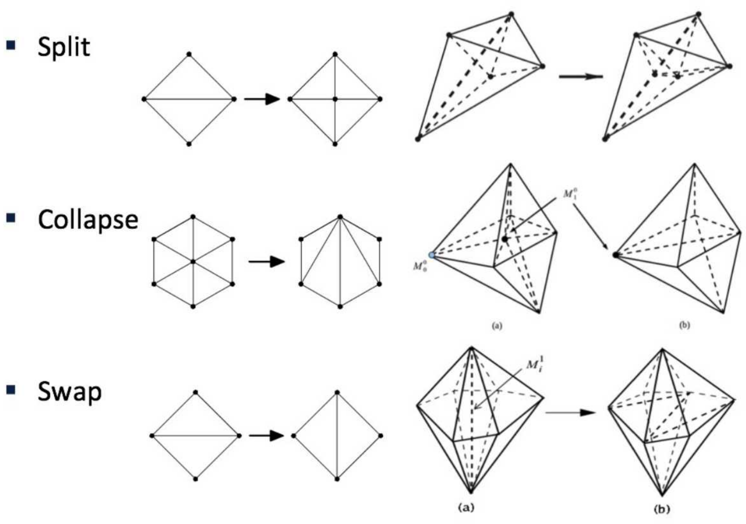
Convergence criterion: reduce residual by factor of 10^{-10}.

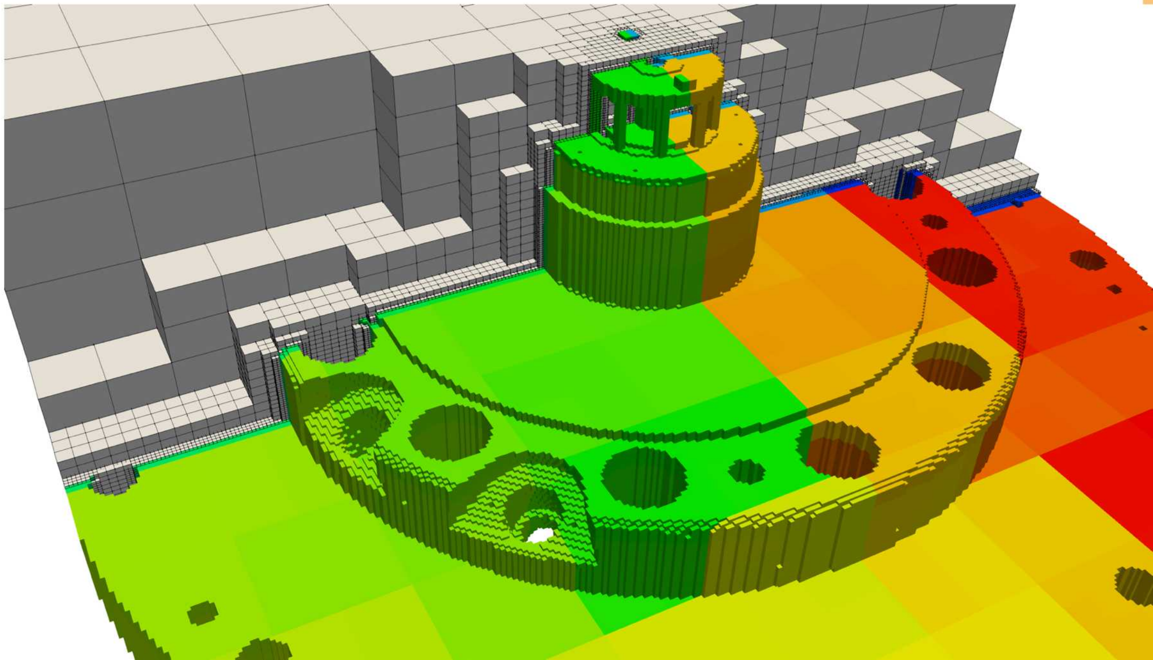
Time (seconds)
to run electrified foil
problem using 1M tets

The Omega_h Code

https://github.com/ibaned/omega_h

- Omega_h is a C++11 library that implements tetrahedron and triangle mesh adaptativity, with a focus on scalable HPC performance using (optionally) MPI, OpenMP, or CUDA. It is intended to provided adaptive functionality to existing simulation codes. Mesh adaptivity allows one to minimize both discretization error and number of degrees of freedom live during the simulation, as well as enabling moving object and evolving geometry simulations. Omega_h will do this for you in a way that is fast, memory-efficient, and portable across many different architectures.





Background Summary of a "Typical" Computational Multiphysics Practitioner

- M.S. or Ph.D. with a computational focus in science or engineering: computational physics or science, computer science, applied math, engineering (mechanical, electrical, chemical, or nuclear), chemistry, etc.
- Scientific programming in C++, computer architectures, software data structures and code architecture, parallel programming, advanced computer science topics (GPUs, team software design & development), software engineering, ...
- Numerical methods, linear algebra, PDEs, real and functional analysis, finite element methods (FEM), finite volume (or difference) methods, discontinuous Galerkin methods, ...
- Subject matter expertise such as computational mechanics, CFD, radiation, neutronics, plasma physics, ...

A Career at Sandia in Software Development

- Apply for an internship at the lab, particularly when you are a post-graduate. A successful internship will help you build your network inside Sandia and have your work become known.
- Watch the Sandia Career's page for interesting openings:

  http://www.sandia.gov/careers/index.html

- Establish a network and keep in touch with them as you approach completion of your degree.
- Work with a professor already collaborating with Sandia.
- Actively participate in appropriate open-source projects.
- Have an attractive LinkedIn, GitHub, ResearchGate, Google Scholar, etc., web presence.