SAND2014-18203PE

National Laboratories

Albany

On Next-Generation Systems

Albany/Kokkos Integration:

- Irina Demeshko, Carter Edwards, Mike Heroux, Roger Pawlowski, Eric Phipps, Andy Salinger

Albany Project Leads:

- Andy Salinger, Glen Hansen, Jake Ostien, Irina Kalashnikova

Kokkos Developers:

- Carter Edwards, Dan Sunderland, Christian Trott

Humble Speaker:

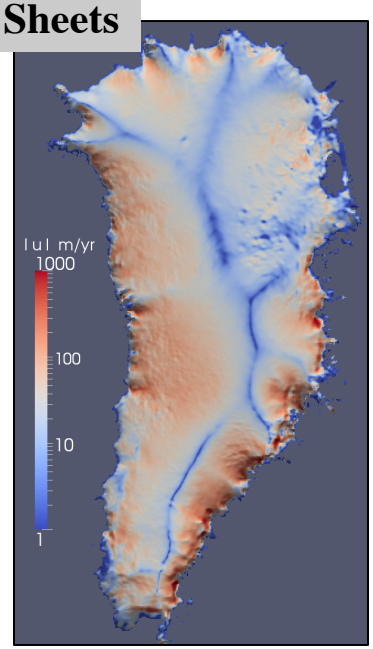
- Karen Devine

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000

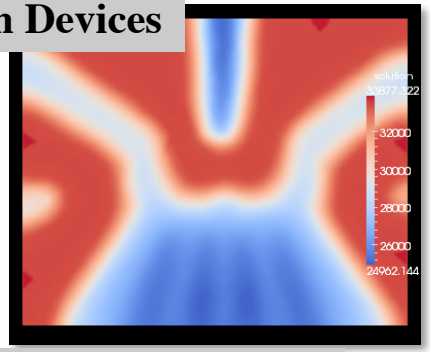
Sandia National Laboratories

Albany is a finite element code built primarily from Trilinos components

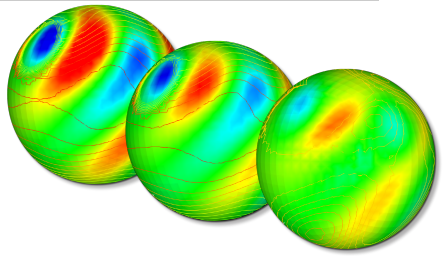
Ice Sheets



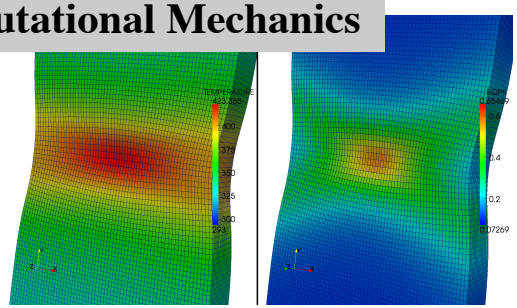
Quantum Devices



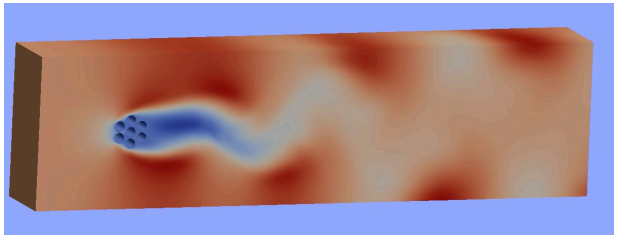
Atmosphere Dynamics



Computational Mechanics



Incompressible Flow



Trilinos

PUMI
Parallel Unstructured Mesh Infrastructure

DAKOTA

Mesh Tools
Partitioning
Load Balancing
Adaptivity
Remeshing
Grid Transfers

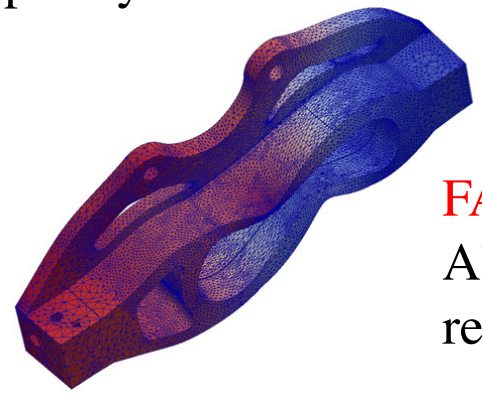
Linear Solvers
Iterative Solvers
Direct Solvers
Eigen Solver
Preconditioners
Multi-Level Algs
Data Structures

Analysis Tools
Nonlinear Solver
Time Integration
Stability Analysis
Optimization
UQ Algorithms

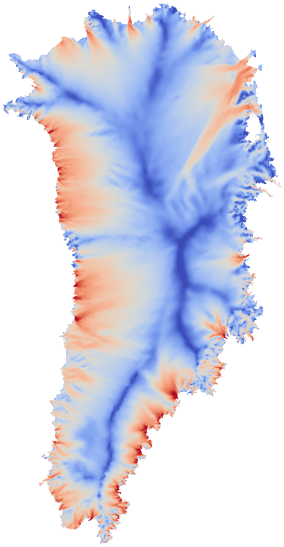
Software Quality
Version Control
Regression Testing
Build System
Verification Tests
Continuous Integration

What is the relationship between Albany and FASTMath?

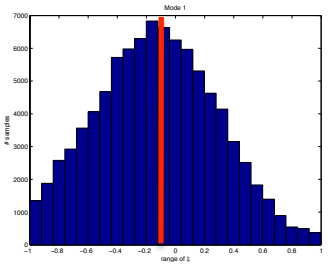
Albany ⇔ PUMI: collaboration is producing unstructured-grid adaptivity workflow



FASTMath ⇔ PISCEES: Albany hosts unstructured grid Ice Sheet model collaboration



FASTMath ⇔ QUEST
All interactions in MidTerm report occurred in Albany



Albany demonstrates, matures, and drives new algorithm development under FASTMath (Trilinos, PUMI):

- Adaptivity (PUMI, PAALS)
- Multilevel preconditioners for thin domains (ML)
- Anderson acceleration (NOX)
- Adjoint-based gradients for inversion (NOX, Sacado)
- New software stack for 64-bit ints (Trilinos)
- New software stack for NextGen architectures (Kokkos)

What algorithms need to perform well on new architectures?

~50% CPU time

Finite Element Integration and Assembly:

- Nested loops over elements, nodes, quadrature points, dimensions, …
- Loading into linear algebra data structures

Work In Progress (see next 3 slides)

~50% CPU time

Linear Solver:

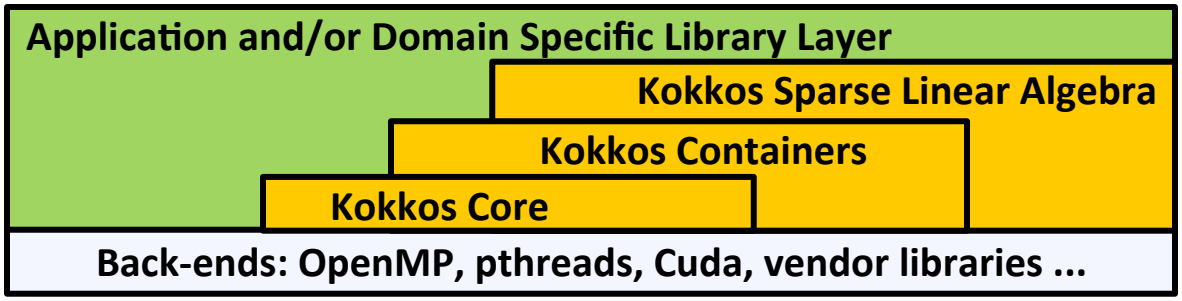
- GMRES/CG
  - MatVec
  - Orthogonalization
- Preconditioners
  - ILU
  - Multi-level

Biggest Research Need

What is our plan? Kokkos programming model for performance portability

Sandia National Laboratories

- Kokkos is *not* funded by FASTMath
- Uses standard C++ template meta-programming to represent device types
  - *Single code can be run on many architectures*



- Multidimensional arrays support *device-specific* memory layout and access (Kokkos::View)
  - a[i][j][k] on CPU
  - a[k][j][i] on GPU

  } are both accessed with a(i,j,k)
- Abstraction layer for node-based parallelism
  - Kokkos::parallel_for; Kokkos::parallel_reduce

Converting a finite-element kernel to Kokkos for portable node-level parallelism

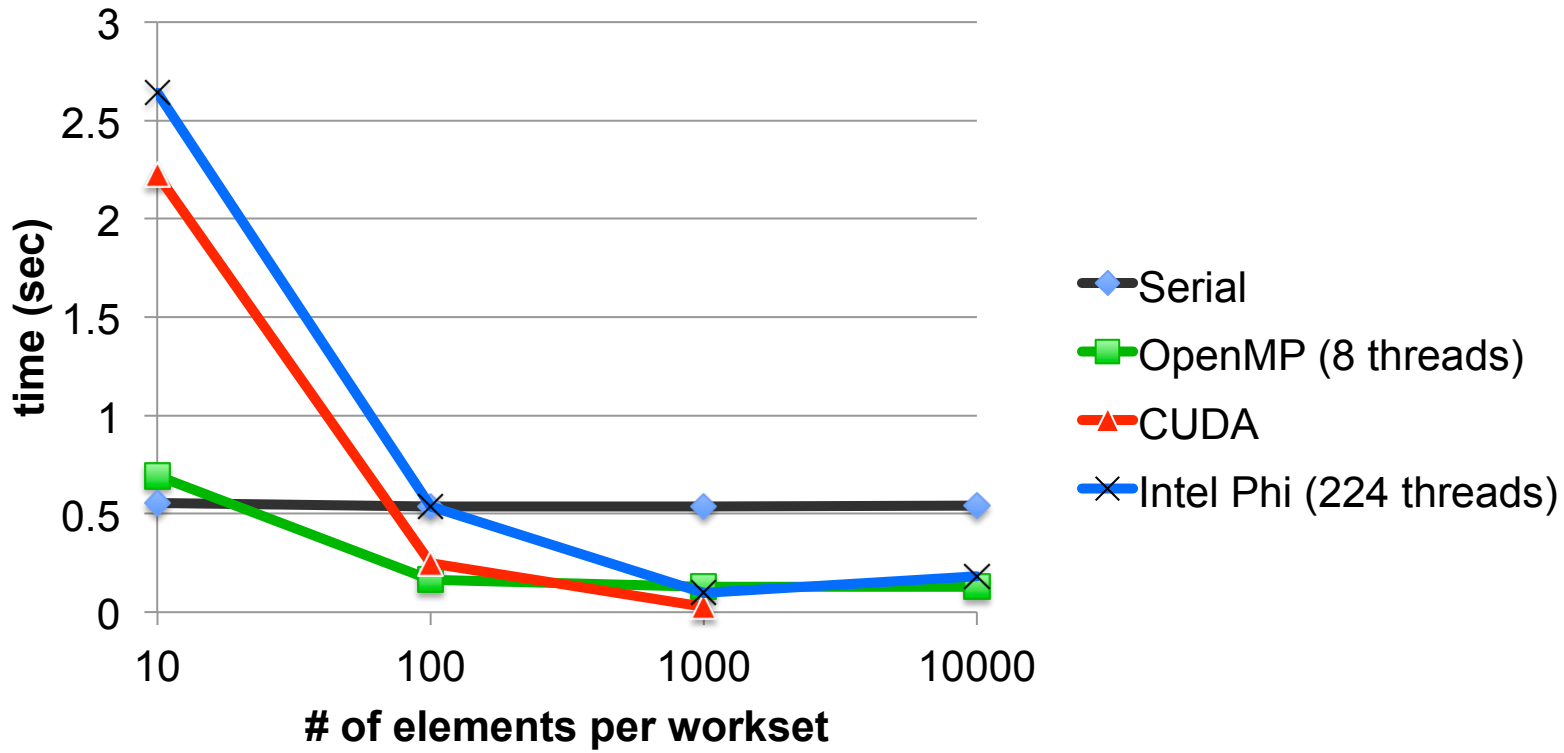
```
template<typename EvalT>
void CoordGrad<EvalT>::evaluateFields()
{
// Outer loop over a Workset ofElements
for(int cell = 0; cell < NumCells; cell++) {
 for(int qp = 0; qp < numQPs; qp++) {
    for(int row = 0; row < numDims; row++){
      for(int col = 0; col < numDims; col++){
        for(int nd = 0; nd < numNodes; nd++){
          coordGrad[cell][qp][row][col] +=
             coordVec[cell][nd][row]
             * basisGrads[nd][qp][col];
        }
      }
    }
  }
} // cell loop
}
```

```
template<typename EvalT>
void CoordGrad<EvalT>::evaluateFields()
{
  // Outer loop over a Workset of Elements
  Kokkos::parallel_for (NumCells, *this);
}
*****************************************
template<typename EvalT>
KOKKOS_INLINE_FUNCTION
void CoordGrad<EvalT>:: operator ()
  (const int cell) const
{
  for(int qp = 0; qp < numQPs; qp++) {
    for(int row = 0; row < numDims; row++){
      for(int col = 0; col < numDims; col++){
        for(int nd = 0; nd < numNodes; nd++){
          coordGrad(cell, qp, row, col) +=
             coordVec(cell, nd, row)
             * basisGrads(nd, qp, col);
        }
      }
    }
  }
}
```

Refactoring follows a simple recipe:

- **Outer loop** moved to `parallel_for(int)`
- **Inner kernel** moved to `operator(int)` functor
- **Arrays** `a[i][j]` converted to `Kokkos::Views a(i,j)`

Finite Element Assembly on *Four* architectures using a *Single* implementation



Timings are for residual vector assembly for Ice Sheet equations, moved from Albany to a stand alone mini-app.

Sandia National Laboratories

Current and Future Efforts

- Move Kokkos versions of Ice Sheet kernels from mini-app back into Albany (debugging)
  - Minimum Cuda version not yet installed on Titan
- Accommodate automatic differentiation data types through the stack
  - Currently works, but needs performance evaluation
- Propagate Kokkos data structures through all underlying Trilinos libraries
  - E.g., the Intrepid FE library
- Make Albany/Trilinos build system flexible to handle CUDA

- Linear Algebra:
  - Albany developers punting this to Trilinos developers