Sandia National Laboratories

Exceptional service in the national interest

Performance-portability of the Albany multi-physics finite element code on the road to exascale

Jerry Watkins[1], Max Carlson[1,2], **Irina Tezaur**[1]

[1] Sandia National Laboratories, Livermore, CA.
[2] University of Utah, Salt Lake City, UT.

Platform for Advanced Scientific Computing (PASC) 2021

Wednesday, July 7, 2021

SAND2021-7742PE

Outline

1. Motivation

2. Albany and its supporting tools

3. Case study: Albany Land Ice (ALI)
   - Overview of ALI model
   - Performance study
   - Automated performance testing
   - Automated parameter/performance tuning

4. Summary & future work

Outline

1. **Motivation**

2. Albany and its supporting tools

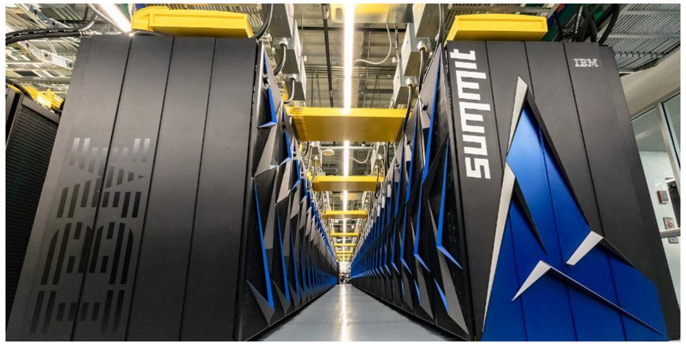
3. Case study: Albany Land Ice (ALI)
    - Overview of ALI model
    - Performance study
    - Automated performance testing
    - Automated parameter/performance tuning

4. Summary & future work

Motivation

"The **top priority** today is the continued progress to **exascale.**" – U.S. DOE Office of Science HPC Initiative

- The future is **GPUs:**

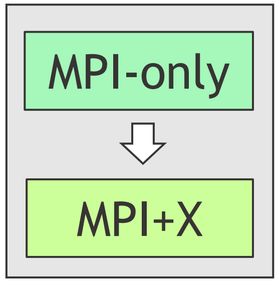


OLCF Summit – IBM POWER9 CPU + NVIDIA V100 GPU

ALCF Aurora (2022, >1 EF) – Intel Xeon CPU + Intel Xe GPU

OLCF Frontier (2021, >1.5 EF) – AMD EPYC CPU + AMD GPUs

NERSC Perlmutter (2021) – AMD EPYC CPU + NVIDIA A100 GPU



- Current scientific software must **adapt** to changing HPC architectures
- New scientific software must be designed to **mitigate issues** from **changing HPC architectures**

Performance portability: achieving a reasonable level of **performance** across a **wide range** of **architectures** using the **same** code base.

Outline

1. Motivation

2. **Albany and its supporting tools**

3. Case study: Albany Land Ice (ALI)
   - Overview of ALI model
   - Performance study
   - Automated performance testing
   - Automated parameter/performance tuning

4. Summary & future work

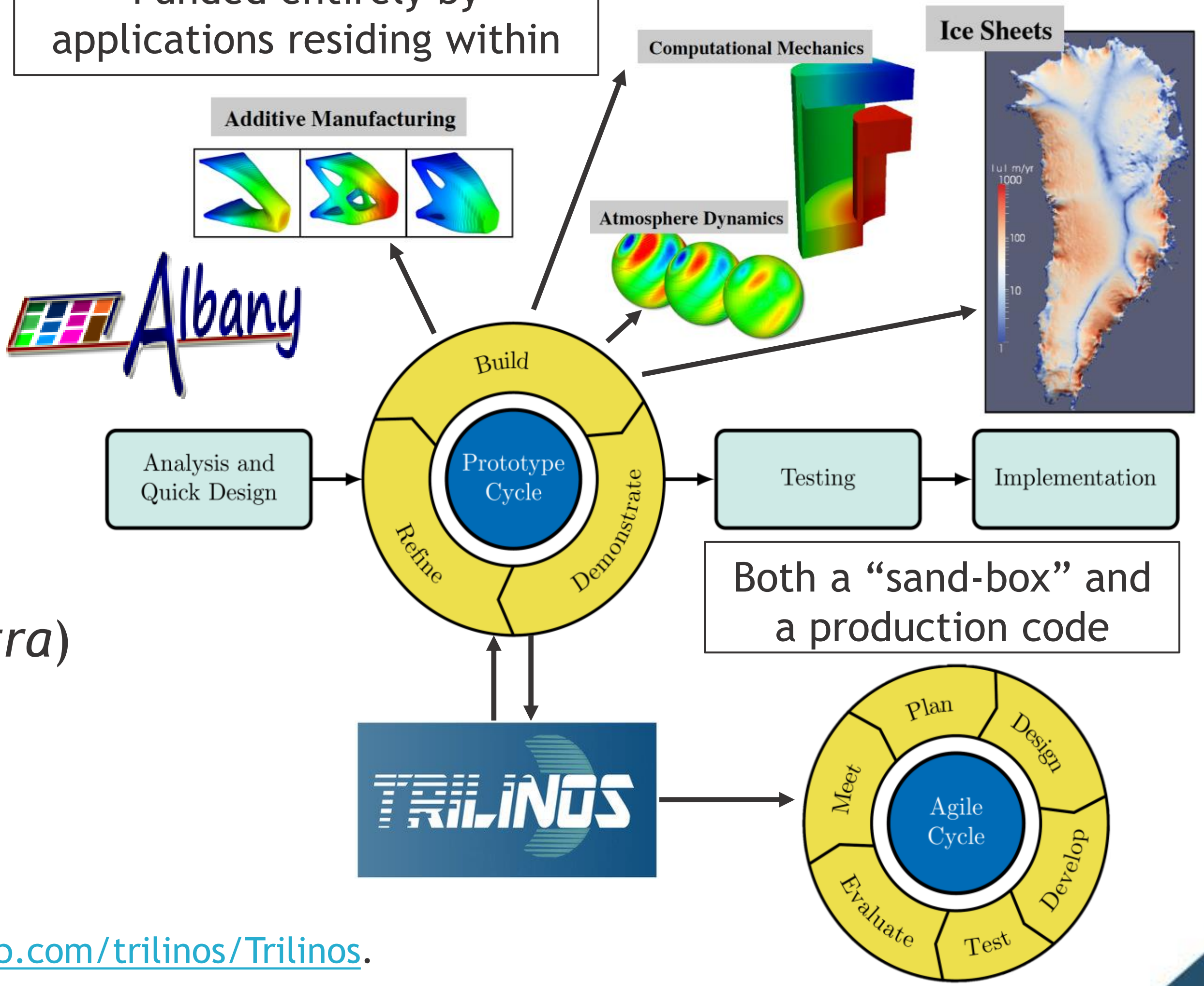
The Albany code base

Albany: open-source[1], parallel, C++, unstructured-grid multi-physics finite element code built for **rapid application development** from **Trilinos[2] Agile Components**

<u>Component Examples (*package name*)</u>

- Mesh tools (*STK*)
- Discretization tools (*Intrepid2*)
- Nonlinear solver (*NOX*)
- Preconditioners (*Ifpack2*)
- Linear solver (*Belos*)
- Field DAG (*Phalanx*)
- Automatic differentiation (*Sacado*)
- Distributed memory linear algebra (*Tpetra*)
- **Shared memory parallelism (*Kokkos*)**
- *Many more…*

Funded entirely by applications residing within



Both a "sand-box" and a production code

[1] https://github.com/SNLComputation/Albany. [2] https://github.com/trilinos/Trilinos.

The Albany code base (cont'd)

Albany provides the "**glue**" that connects components (via abstract interfaces).

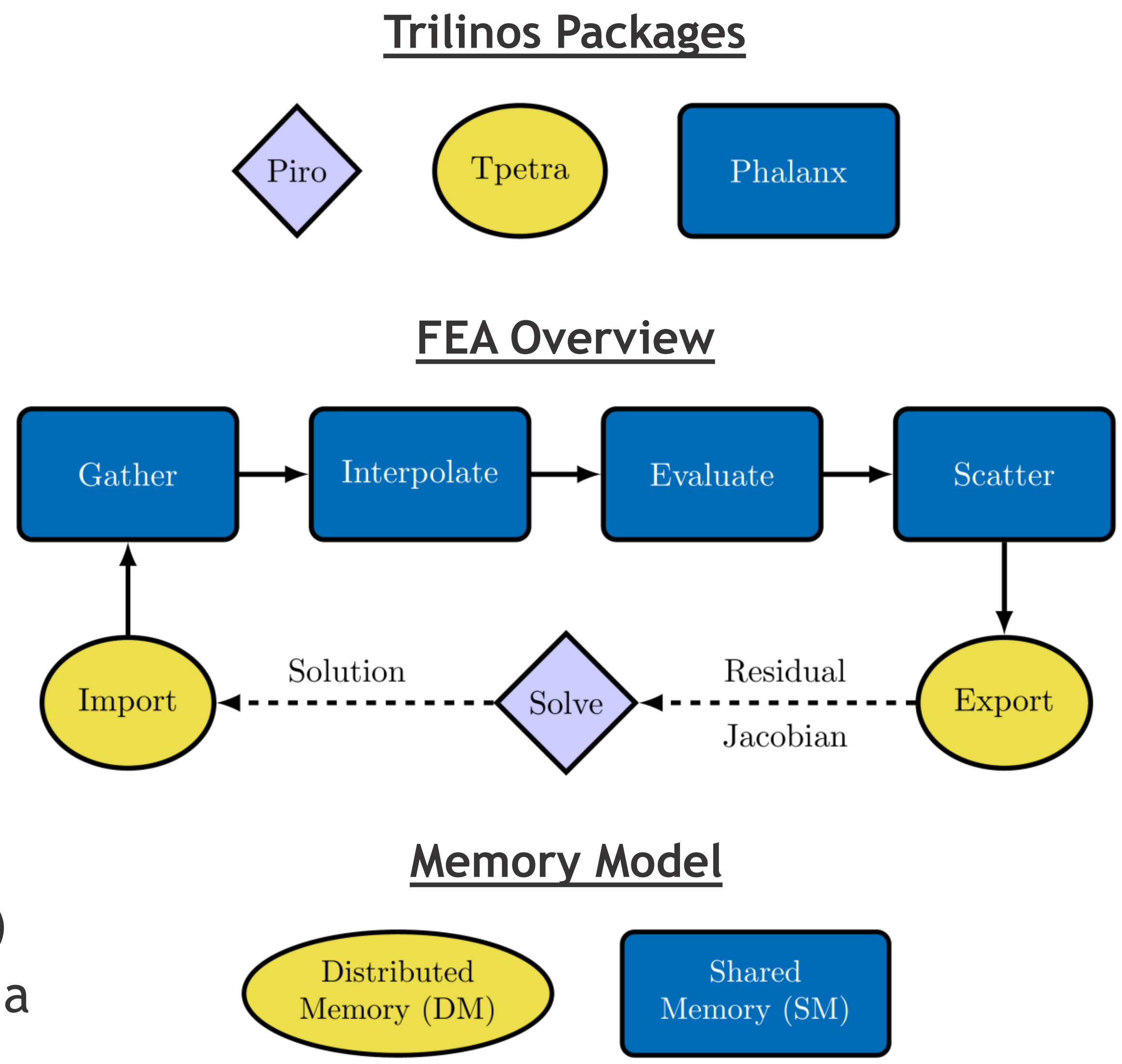
Albany finite element assembly (FEA):

- **Tpetra** manages **distributed** memory linear algebra (MPI+X)
- **Phalanx** manages **shared** memory computations (X)
  - ➢ **Gather** fills element local solution
  - ➢ **Interpolate** solution/gradient to quad points
  - ➢ **Evaluate** residual/Jacobian
  - ➢ **Scatter** fills global residual/Jacobian

Adding new PDEs requires just implementing a new **Evaluate** routine

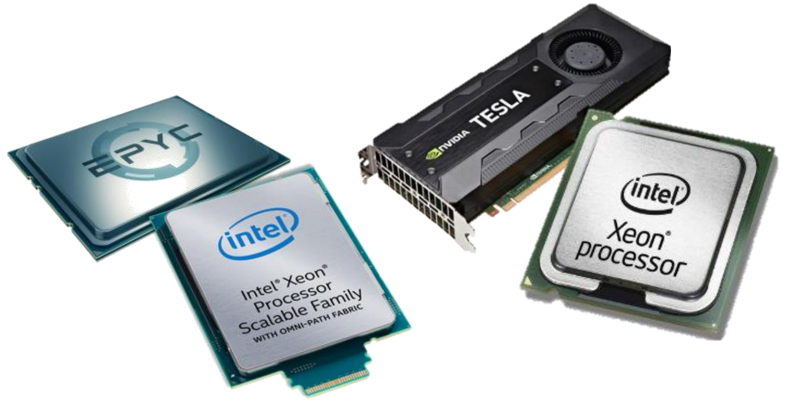
Albany design highlights:

- **Piro** manages the solve (e.g., Newton, time-stepper)
- Jacobians (+ sensitivities, Hessians, etc.) obtained via **automatic differentiation (Sacado)**
- **Kokkos** achieves performance portability using **MPI+X**



Albany supporting tools: Kokkos – performance portability

- **Kokkos**[1] is a C++ library that provides **performance portability** across multiple **shared memory** computing architectures
  - ➢ ***Examples:*** Multicore CPU, NVIDIA GPU, Intel KNL and much more…
- Abstract **data layouts** and **hardware features** for optimal performance on **current** and **future** architectures
- Allows researchers to focus on **application** or **algorithmic development** instead of **architecture specific programming**



With Kokkos, you write an algorithm **once** for multiple hardware architectures.
Template parameters are used to get **hardware specific features**.

[1]https://github.com/kokkos/kokkos.

Albany supporting tools: Phalanx Evaluator – templated Phalanx node

- A Phalanx node (**evaluator**) is constructed as a C++ class
- Each evaluator is templated on an **evaluation type** (e.g., residual, Jacobian)
- The evaluation type is used to determine the **data type** (e.g., double, Sacado data types)

```
template<typename EvalT, typename Traits>
KOKKOS_INLINE_FUNCTION
void StokesFOResid<EvalT, Traits>::
operator() (const int& cell) const{
  for (int cell=0; cell < numCells; cell++) {
    for (int node=0; node < numNodes; ++node){
      Residual(cell,node,0)=0.;
    }
  }
  for (int cell=0; cell < numCells; cell++) {
    for (int node=0; node < numNodes; ++node) {
      for (int qp=0; qp < numQPs; ++qp) {
        Residual(cell,node,0) +=
            Ugrad(cell,qp,0,0)*wGradBF(cell,node,qp,0) +
            Ugrad(cell,qp,0,1)*wGradBF(cell,node,qp,1) +
            force(cell,qp,0)*wBF(cell,node,qp);
      }
    }
  }
}
```

Albany supporting tools: Phalanx Evaluator – templated Phalanx node

- A Phalanx node (**evaluator**) is constructed as a C++ class
- Each evaluator is templated on an **evaluation type** (e.g., residual, Jacobian)
- The evaluation type is used to determine the **data type** (e.g., double, Sacado data types)
- Kokkos **RangePolicy** is used to parallelize over **cells** over an **Execution Space** (e.g., Serial, OpenMP, CUDA)
- Inline **functors** are used as kernels
- MDField data layouts
  - ➢ Serial/OpenMP – **LayoutRight** (row-major)
  - ➢ CUDA – **LayoutLeft** (col-major)

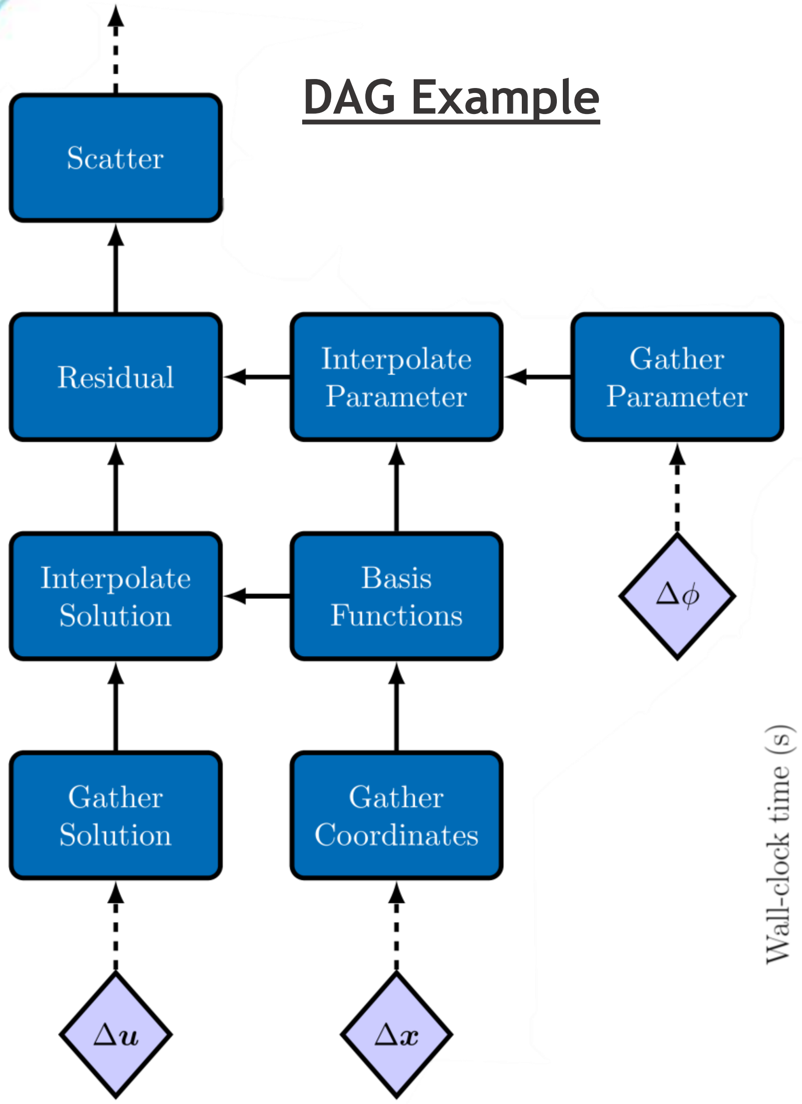
```
typedef Kokkos::CUDA ExeSpace;

template<typename EvalT, typename Traits>
void StokesFOResid<EvalT, Traits>::
evaluateFields(typename Traits::EvalData workset) {
  Kokkos::parallel_for(
      Kokkos::RangePolicy<ExeSpace>(0,workset.numCells),
      *this);
}

template<typename EvalT, typename Traits>
KOKKOS_INLINE_FUNCTION
void StokesFOResid<EvalT, Traits>::
operator() (const int& cell) const{
~~  for (int cell=0; cell < numCells; cell++) {~~
    for (int node=0; node < numNodes; ++node){
      Residual(cell,node,0)=0.;
    }
~~  }~~
~~  for (int cell=0; cell < numCells; cell++) {~~
    for (int node=0; node < numNodes; ++node) {
      for (int qp=0; qp < numQPs; ++qp) {
        Residual(cell,node,0) +=
          Ugrad(cell,qp,0,0)*wGradBF(cell,node,qp,0) +
          Ugrad(cell,qp,0,1)*wGradBF(cell,node,qp,1) +
          force(cell,qp,0)*wBF(cell,node,qp);
      }
    }
  }
~~}~~
```

Albany supporting tools: Phalanx – directed acyclic graph (DAG)

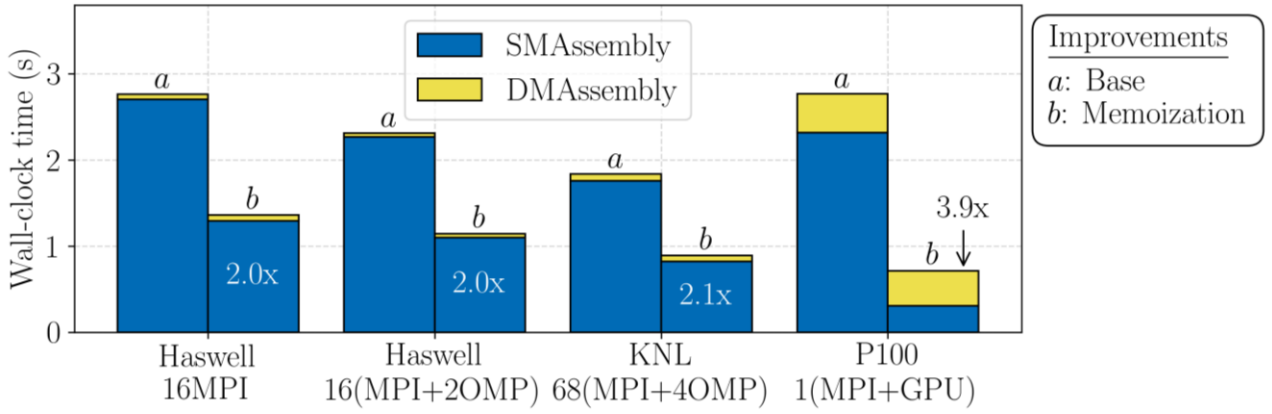
DAG Example



Advantages:
- Increased flexibility, extensibility, usability
- Arbitrary data type support
- Potential for task parallelism

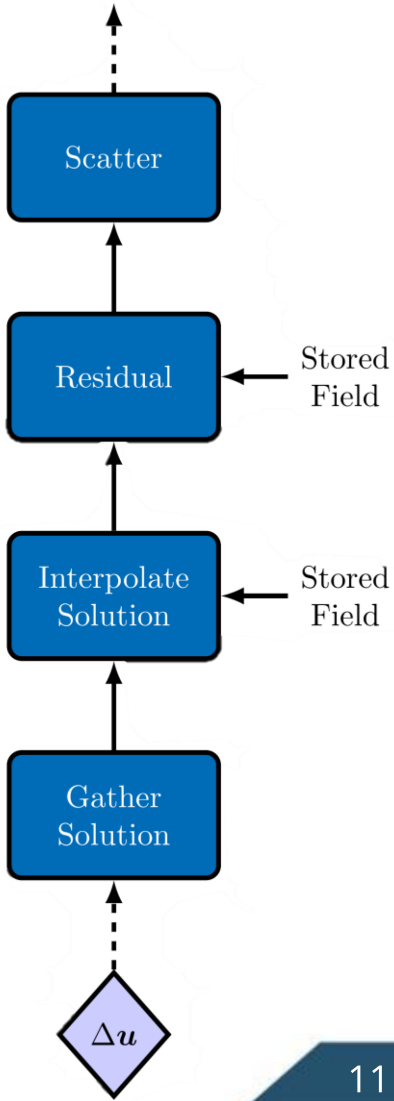
Extension:
- Performance gain through memoization

Disadvantage:
- Performance loss through fragmentation



Single CPU socket or GPU

DAG Example (memoization)

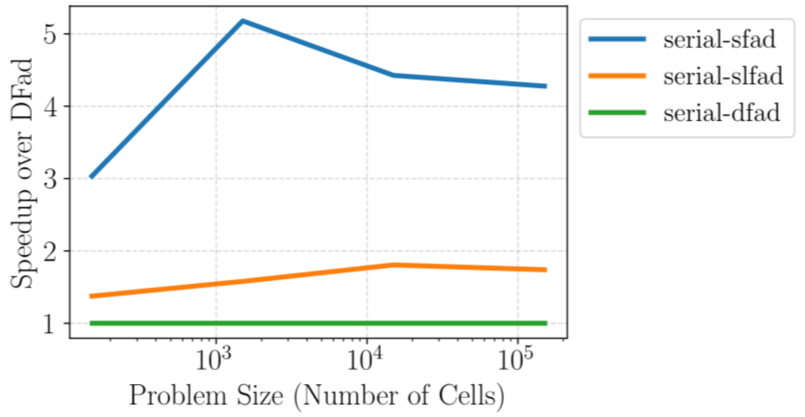


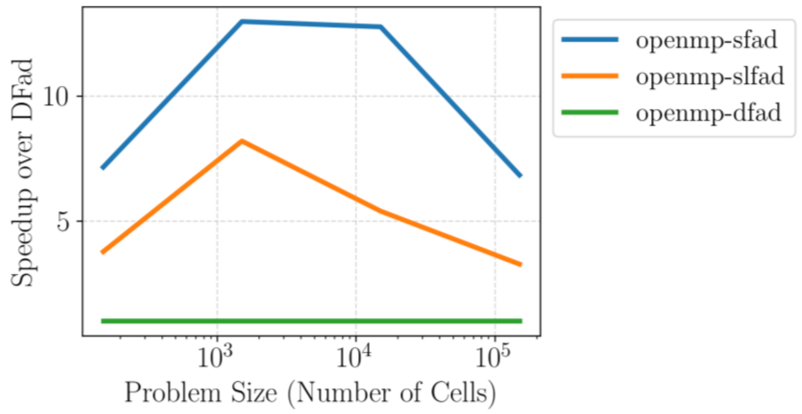
Albany supporting tools: Sacado – automatic differentiation (AD)

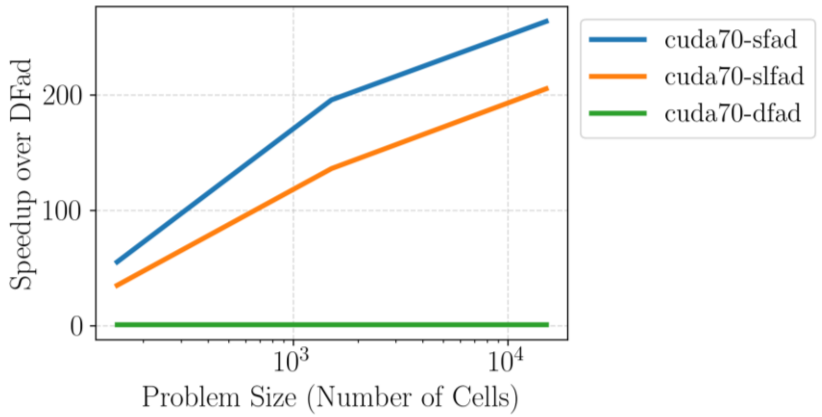
- AD provides **exact** derivatives — no Jacobian derivation or hand-coding required
- Allows for **advanced analysis** capabilities – easily construct any derivative, Hessian
  - ➢ Examples: optimization, sensitivity analysis
- Sacado **data types** are used for derivative components via class **templates**
  - ➢ **DFad** (most flexible) – size set at run-time
  - ➢ **SLFad** (flexible/efficient) – max size set at compile-time
  - ➢ **SFad** (most efficient) – size set at compile-time

There are **significant speedups** when derivative array sizes are known at compile time on **GPU** (50-250x)

Fad Type Comparison (Serial, OpenMP (12 threads), CUDA)







Size Example: Tetrahedral elements (4 nodes), 2 equations, ND = 4*2 = 8

Outline

ProSPect project for land-ice modeling

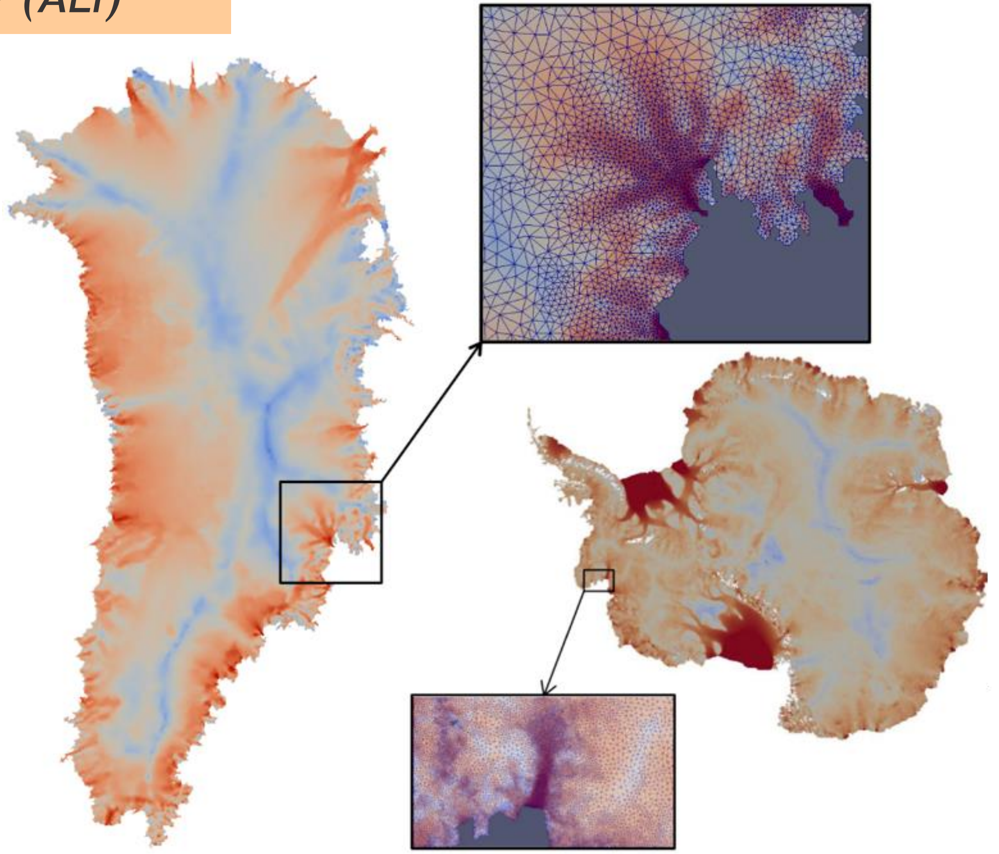
"**ProSPect**[1]" = Probabilistic Sea Level Projections from Ice Sheet & Earth System Models (5 year SciDAC4 project, 2017-2022)

Goal: to **develop** and **support** a robust and scalable land ice solver based on the "First-Order" (FO) Stokes equations → *Albany Land Ice (ALI)*

Requirements for Albany Land Ice (ALI):

- **Unstructured grid** meshes.
- **Scalable, fast** and **robust.**
- **Verified** and **validated.**
- **Portable** to new architecture machines.
- **Advanced analysis** capabilities: deterministic inversion, calibration, uncertainty quantification, sensitivity analysis.

As part of **U.S. DOE E3SM**[2] **Earth System Model**, solver will provide actionable predictions of 21st century sea-level change (including uncertainty bounds).



[1]https://doe-prospect.github.io [2]Energy Exascale Earth System Model

Albany Land Ice (ALI) model

- Ice velocities given by the **"First-Order" Stokes PDEs** with nonlinear viscosity:

$$\begin{cases} -\nabla \cdot (2\mu \dot{\epsilon}_1) = -\rho g \dfrac{\partial s}{\partial x} \\ -\nabla \cdot (2\mu \dot{\epsilon}_2) = -\rho g \dfrac{\partial s}{\partial y} \end{cases}$$

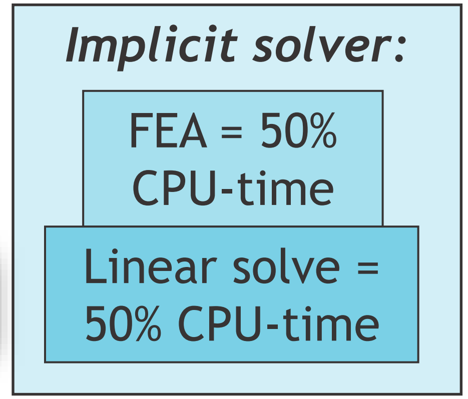
Glen's Flow Law Viscosity

$$\mu = \frac{1}{2} A^{-\frac{1}{n}} \left(\frac{1}{2} \sum_{ij} \dot{\epsilon}_{ij}^{\,2} \right)^{\left(\frac{1}{2n} - \frac{1}{2}\right)}$$

$$\dot{\epsilon}_1^{\,T} = (2\dot{\epsilon}_{11} + \dot{\epsilon}_{22}, \dot{\epsilon}_{12}, \dot{\epsilon}_{13})$$
$$\dot{\epsilon}_2^{\,T} = (2\dot{\epsilon}_{12}, \dot{\epsilon}_{11} + 2\dot{\epsilon}_{22}, \dot{\epsilon}_{23})$$
$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

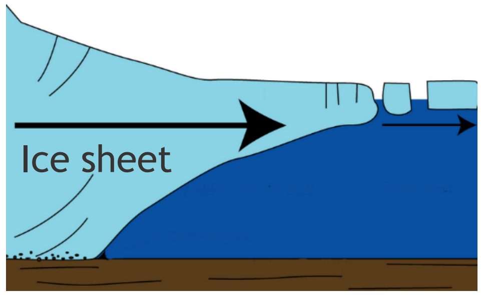
Algorithmic choices for ALI:

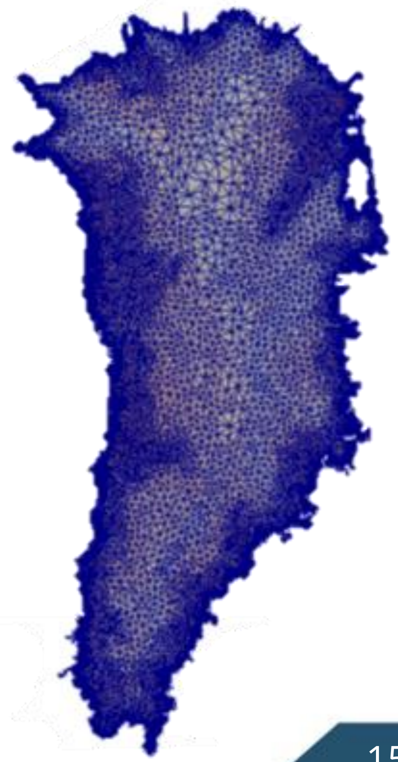
- **3D unstructured grid** FEM discretization (unstructured in x-y, structured in z).
- **Newton method** nonlinear solver with automatic differentiation Jacobians.
- Preconditioned **Krylov iterative linear solvers.**


Implicit solver:

FEA = 50% CPU-time

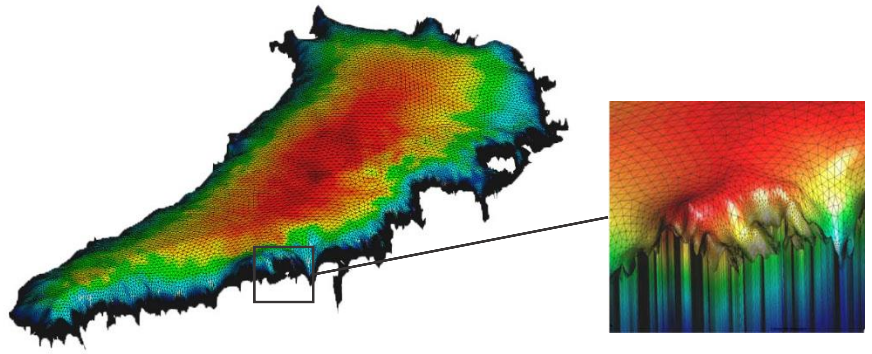
Linear solve = 50% CPU-time





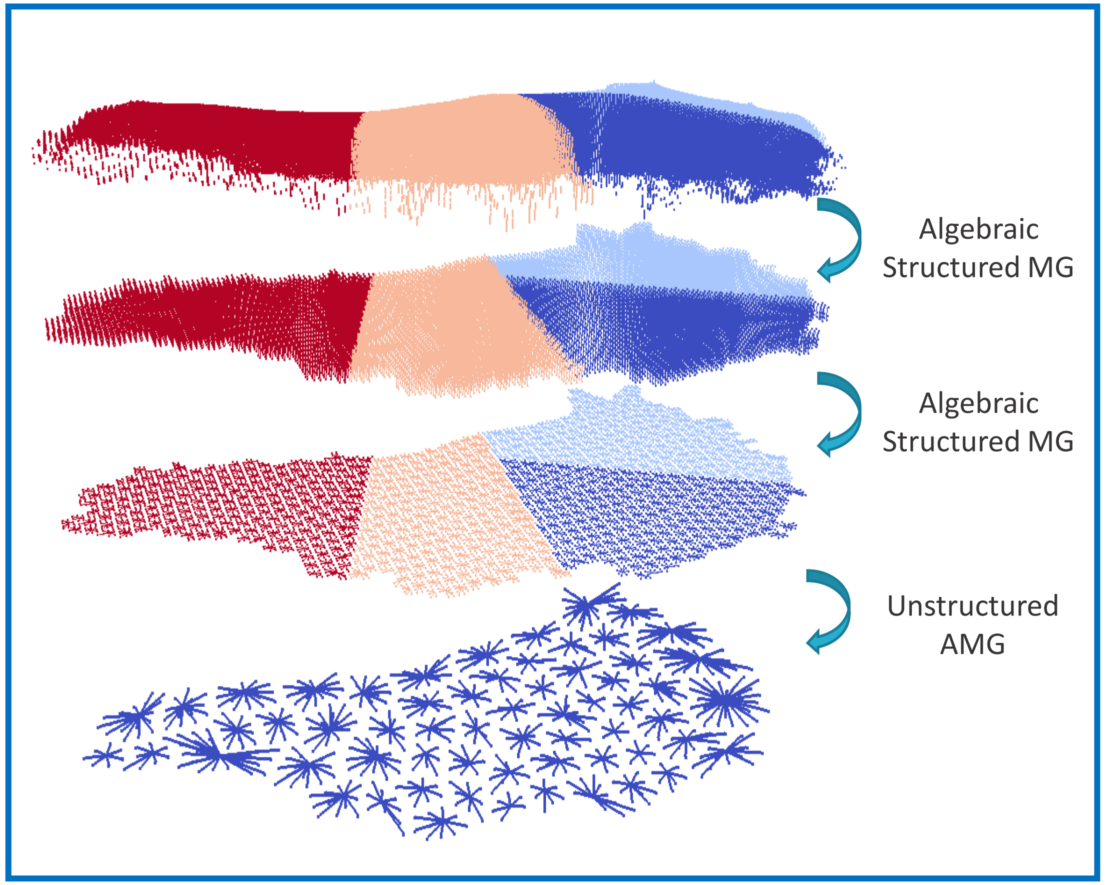
Specialized linear solvers

Problem: Ice sheet meshes are thin with high aspect ratios



Solution: Matrix dependent semi-coarsening algebraic multigrid (MDSC-AMG)[1]

- First, apply algebraic **structured** multigrid to coarsen vertically
- Second, apply **SA-AMG** on single layer



[1] See (Tezaur *et al.*, 2015), (Tuminaro *et al.*, 2016)

Status: Linear solve is **ported to GPU** but not optimized

- Multigrid **tuning** for GPU is work in progress
- Performance-portable **block smoothers** for fine grid coming soon (FY22)

Outline

1. Motivation

2. Albany and its supporting tools

3. **Case study: Albany Land Ice (ALI)**
    - Overview of ALI model
    - **Performance study**
    - Automated performance testing
    - Automated parameter/performance tuning

4. Summary & future work

Performance study: Architectures

Performance-portability of ALI has been tested across ***multiple architectures***: Intel Sandy Bridge, Intel Skylake, IBM POWER8, IBM POWER9, Keplar/Pascal/Volta/Ampere GPUs, KNL Xeon Phi

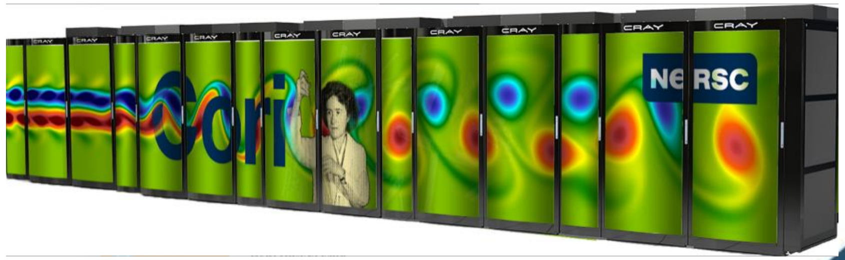
Architectures:

- Cori (NERSC): 2,388 Haswell nodes [2 Haswell (32 cores)]
  9,688 KNL nodes [1 Xeon Phi KNL (68 cores)] (Cray Aries)
- Blake (SNL): 40 nodes [2 Skylake (48 cores)] (Intel OmniPath Gen-1)
- Mayer (SNL): 43 nodes [2 ARM64 Cavium ThunderX2 (56 cores)] (Mx EDR IB)
- Ride (SNL): 12 nodes [2 POWER8 (16 cores) + P100 (4 GPUs)] (Mx C-X4 IB)
- Weaver (SNL): 10 nodes [2 POWER9 (40 cores) + V100 (4 GPUs)] (Mx EDR IB)
- Summit (OLCF): 4600 nodes [2 P9 (22 cores) + V100 (6 GPUs)]
- Kahuna (SNL): 4 nodes [2 Zen2 (64 cores) + A100 (1 GPU)] PCIe Gen 4
- Mutrino (SNL): 100 Haswell nodes [2 Haswell (32 cores)],
  100 KNL nodes [1 KNL (68 cores)]
- Vortex (SNL): 72 nodes [2 POWER9 (44 cores) + V100 (4 GPUs)]

Future Targets: Aurora Intel GPU (ALCF), Frontier AMD GPU (OLCF), Perlmutter (NERSC)

Models:

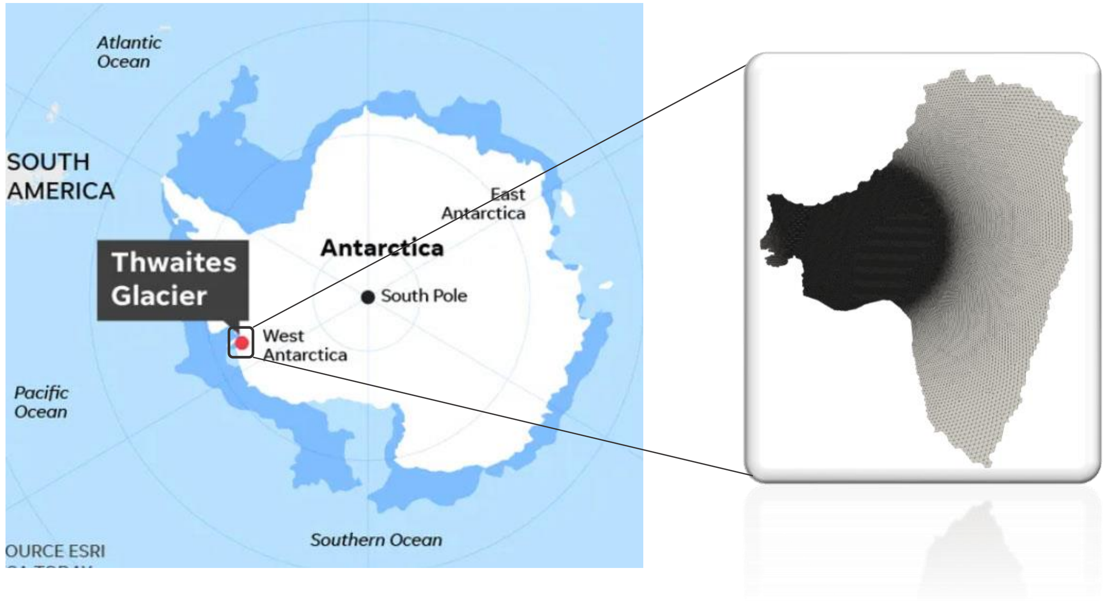
- 2 models: MPI-only, MPI+GPU
  - MPI+GPU: MPI ranks assigned a **single core per GPU**, CUDA UVM used for host to device communication



Performance study: Cases

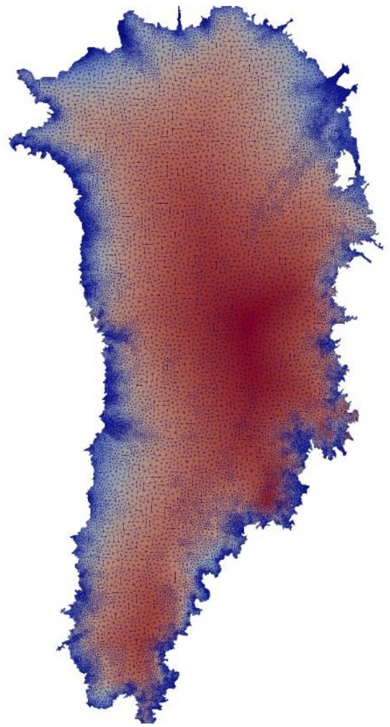
Case 1: Antarctica's Thwaites glacier

- First-Order Stokes equations
- 1-10km variable resolution
- 1,395,680 wedge elements
- 100 evaluations of FEA + linear solve results
- Strong scaling analysis
  - ➢ 32-core Haswell per node
  - ➢ 4 V100 GPUs per node



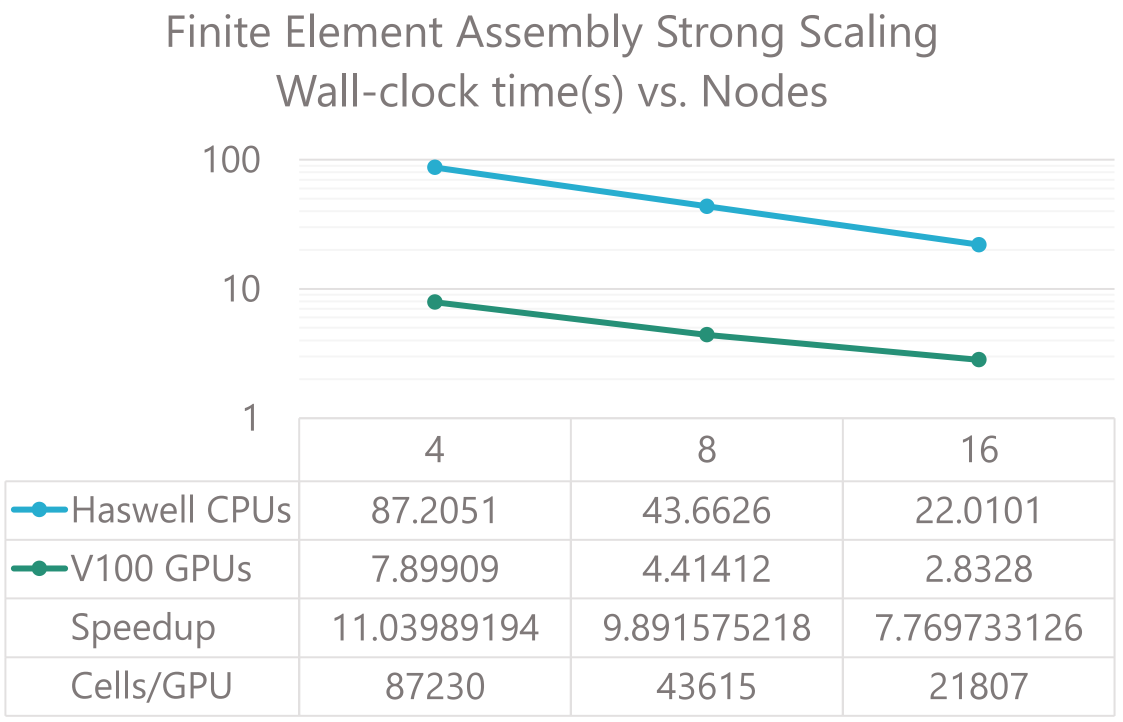
Case 2: Greenland ice sheet

- First-Order Stokes equations (+ Enthalpy equation)
- 1km-7km variable resolution
- 14.4 million tetrahedral elements
- FEA-only results (100 evaluations) highlighting recent performance improvements in Albany

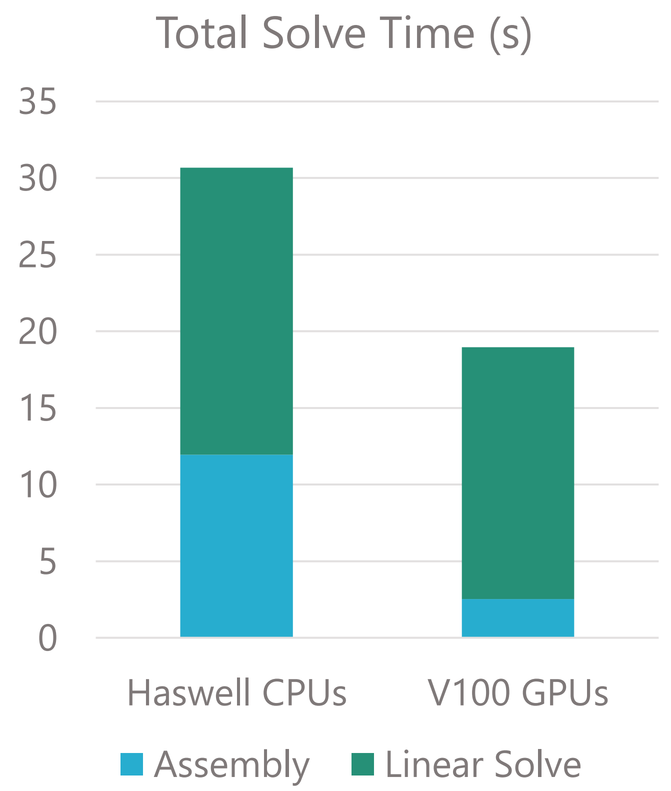


Architectures:
- ➢ Haswell
- ➢ KNL
- ➢ POWER9
- ➢ V100 GPU

Performance study: Thwaites Glacier



	4	8	16
Haswell CPUs	87.2051	43.6626	22.0101
V100 GPUs	7.89909	4.41412	2.8328
Speedup	11.03989194	9.891575218	7.769733126
Cells/GPU	87230	43615	21807



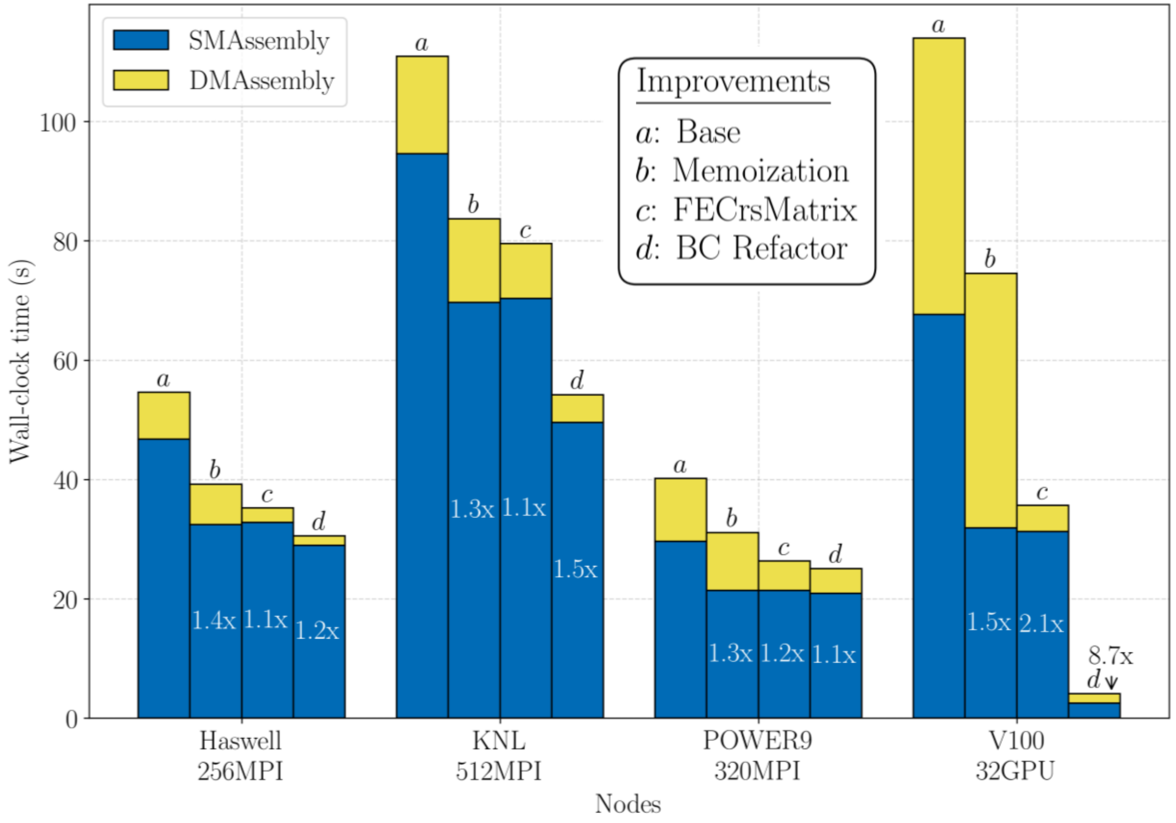
- **11x** speedup V100 over Haswell node
- **WIP:** Reduce memory usage
  - ➢ UVM degrades performance when using more than 87,230 Cells/GPU

- **1.6x** speedup V100 over Haswell (4 nodes)
- **WIP:** Optimize GPU solve
  - ➢ Linear solve is **86%** of total solve time on GPU!
  - ➢ Linear solve needs block relaxation

Performance study: Greenland Perf. Improvements

Implement Tpetra::FECrsMatrix

- Original — on-the-fly host memory allocations
- Addressed by Tpetra via FECrsMatrix (**2.1x**) – single memory allocation during setup



Refactor of boundary conditions

- Boundary data converted to contiguous data structures
- Kernels constructed for boundary conditions
- **8.7x** speedup on GPU relative to original* with boundary kernels on host (StokesFO)

Refactor of Enthalpy equation

- Kernels constructed using Kokkos
- **66x** speedup on GPU relative to original* with all kernels on host

*Original: running on host with device transfers

Tpetra::FECrsMatrix: Luca Bertagna, Mauro Perego, Chris Siefert; **Refactor:** Max Carlson

Outline

1. Motivation

2. Albany and its supporting tools

3. **Case study: Albany Land Ice (ALI)**
   - Overview of ALI model
   - Performance study
   - **Automated performance testing**
   - Automated parameter/performance tuning

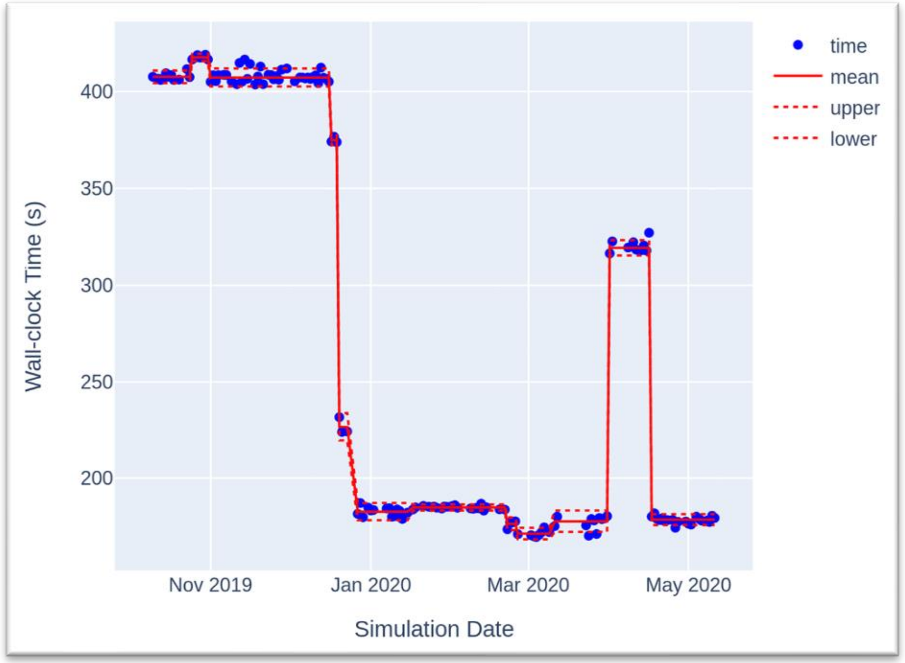
4. Summary & future work

Performance testing: maintaining performance & portability

With Kyle Shan (Micron Technology, formerly Stanford U)

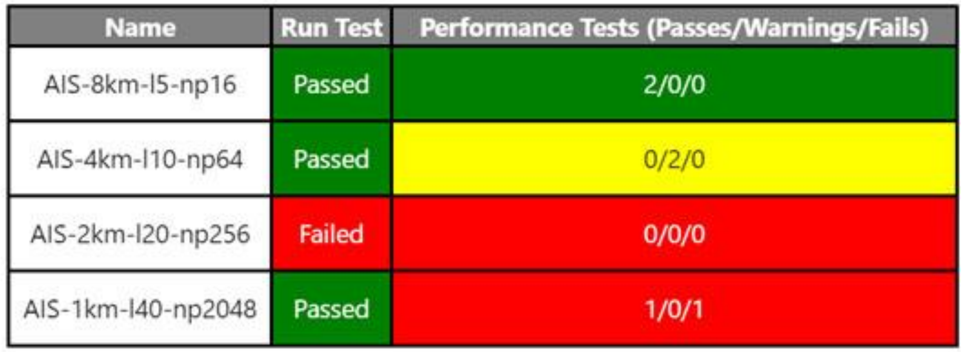
- **Nightly testing** is needed to secure investments in **performance** & **portability** in evolving software
  - ➢ Changes in code base could cause performance deterioration
  - ➢ Performance improvements in one architecture could decrease performance in another
  - ➢ Manual analysis is time consuming and imprecise

Figure below: Total simulation time for a 2-20km resolution Antarctica problem, executed nightly



Solution: changepoint detection algorithm automatically applied to nightly performance test data to identify/flag large changes in performance.

- Infrastructure is provided in a **Jupyter notebook**, exported as html to a website (https://ikalash.github.io)
- Daily **email** provides nightly performance test summary



Name	Run Test	Performance Tests (Passes/Warnings/Fails)
AIS-8km-l5-np16	Passed	2/0/0
AIS-4km-l10-np64	Passed	0/2/0
AIS-2km-l20-np256	Failed	0/0/0
AIS-1km-l40-np2048	Passed	1/0/1

Outline

1. Motivation

2. Albany and its supporting tools

3. **Case study: Albany Land Ice (ALI)**
   - Overview of ALI model
   - Performance study
   - Automated performance testing
   - **Automated parameter/performance tuning**

4. Summary & future work

Automated parameter/performance tuning

Problem: hand-tuning solver parameters can be a long/painful process, does not translate b/w architectures.

Solution: create a **framework** for determining **optimal** parameter values to achieve best performance (smallest CPU time) on HPC systems using offline and real-time data.

Preliminary results: optimization over linear solver parameter space using **random search** for coarse resolution 3-20km mesh **Greenland** problem (100 iterations)

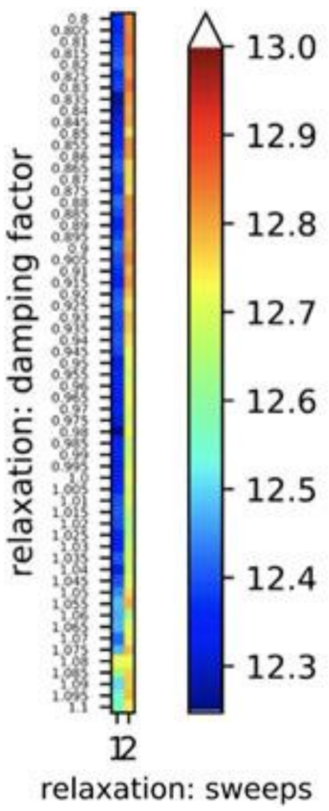
Parameters to optimize

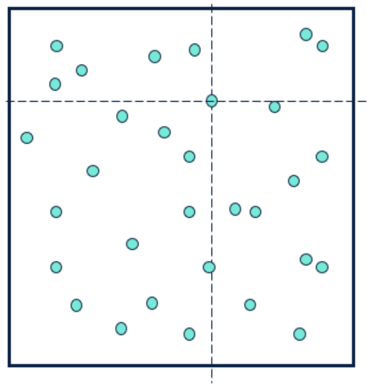
```
type: RELAXATION
ParameterList:
    ’relaxation: type’: Two-stage Gauss-Seidel
    ’relaxation: sweeps’: positive integer
    ’relaxation: inner damping factor’: positive real number
```

	Cases	Manual Tuning (sec.)	Autotuning (sec.)	Speedup
CPU[1]	blake_vel	3.533972	2.658731	1.33x
	blake_ent	3.07725	2.036044	1.51x
GPU[2]	weaver_vel	19.13084	16.30672	1.17x
	weaver_ent	19.76345	15.00014	1.32x

[1] 8 nodes, Skylake CPU
[2] 2 nodes, V100 GPU

With Carolyn Kao (Stanford U)




Random Search

Future work: more sophisticated algorithms for parameter optimization (e.g., **Bayesian optimization**)

Outline

Summary

- HPC architectures are **changing rapidly** which poses a significant challenge for open-science
- The **Albany/Trilinos/Kokkos** software stack offers an efficient way to meet this challenge for large scale **finite element analysis**
- **Albany Land Ice** is currently being used to provide **sea-level change predictions**
- **Performance** on next generation computing architectures is a **work in progress**
  - ➢ 11x speedup of V100 node over Haswell node for finite element assembly
  - ➢ Performance on V100 is ~2x faster than on Haswell node for the full solve
- **Maintaining performance** and **portability** is crucial for an active code base
  - ➢ A change-point detection algorithm can help identify performance variation
- **Optimal solver parameters** can be determined for a specific architecture **automatically** using black-box optimization algorithms
  - ➢ A simple random grid search algorithm can improve performance by up to 1.5x.

Future work

Finite Element Assembly:

- Further **optimization** of FEA (e.g., reduce memory footprint)
- Remove **UVM** (work in progress in Trilinos)

Linear Solver:

- **Block smoother** for linear solver
- Optimization of **linear solver** for GPU
- More sophisticated **auto-tuning algorithms** (e.g., Bayesian optimization)

General:

- **Large-scale performance analysis** on Cori, Summit and Perlmutter

Funding/Acknowledgements

Support for this work was provided by Scientific Discovery through Advanced Computing (SciDAC) projects funded by the U.S. Department of Energy, Office of Science (OS), Advanced Scientific Computing Research (ASCR) and Biological and Environmental Research (BER).

SciDAC Scientific Discovery through Advanced Computing

Office of Science U.S. Department of Energy

FASTMATH

Sandia National Laboratories

NERSC

OAK RIDGE National Laboratory

Computing resources provided by the National Energy Research Scientific Computing Center (NERSC) and Oak Ridge Leadership Computing Facility (OLCF).