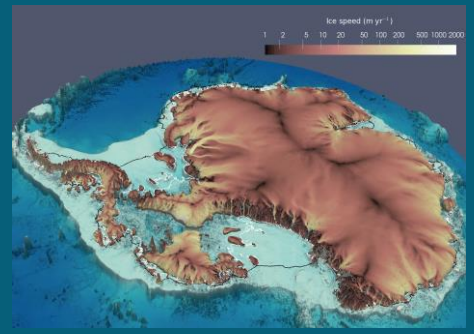
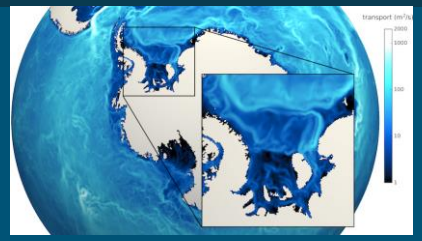
SAND2022-2130C

Sandia National Laboratories

Scalability studies of Albany Land Ice: a performance portable, ice sheet solver





February 25th, 2022

PRESENTED BY

Jerry Watkins, Max Carlson, Carolyn Kao, Irina Tezaur, Ray Tuminaro

SIAM Conference on Parallel Processing for Scientific Computing

SAND

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

1) Motivation - Why are we interested in scalability?
2) Albany, Trilinos and Kokkos
3) Linear solver in Albany Land Ice
4) Numerical results
5) Conclusions

Motivation

Why are we interested in scalability?

Motivation

- “The top priority today is the continued progress to exascale” – DOE Office of Science HPC Initiative
- Current scientific software must adapt to changing HPC architectures
- New scientific software must be designed to mitigate issues from changing HPC architectures

OLCF Summit – IBM POWER9 CPU + NVIDIA V100 GPU

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

OLCF Frontier (2022, >1.5 EF) – AMD EPYC CPU + AMD GPUs

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

GPUs in open-science are here, need efficient access to computational power

Albany, Trilinos and Kokkos

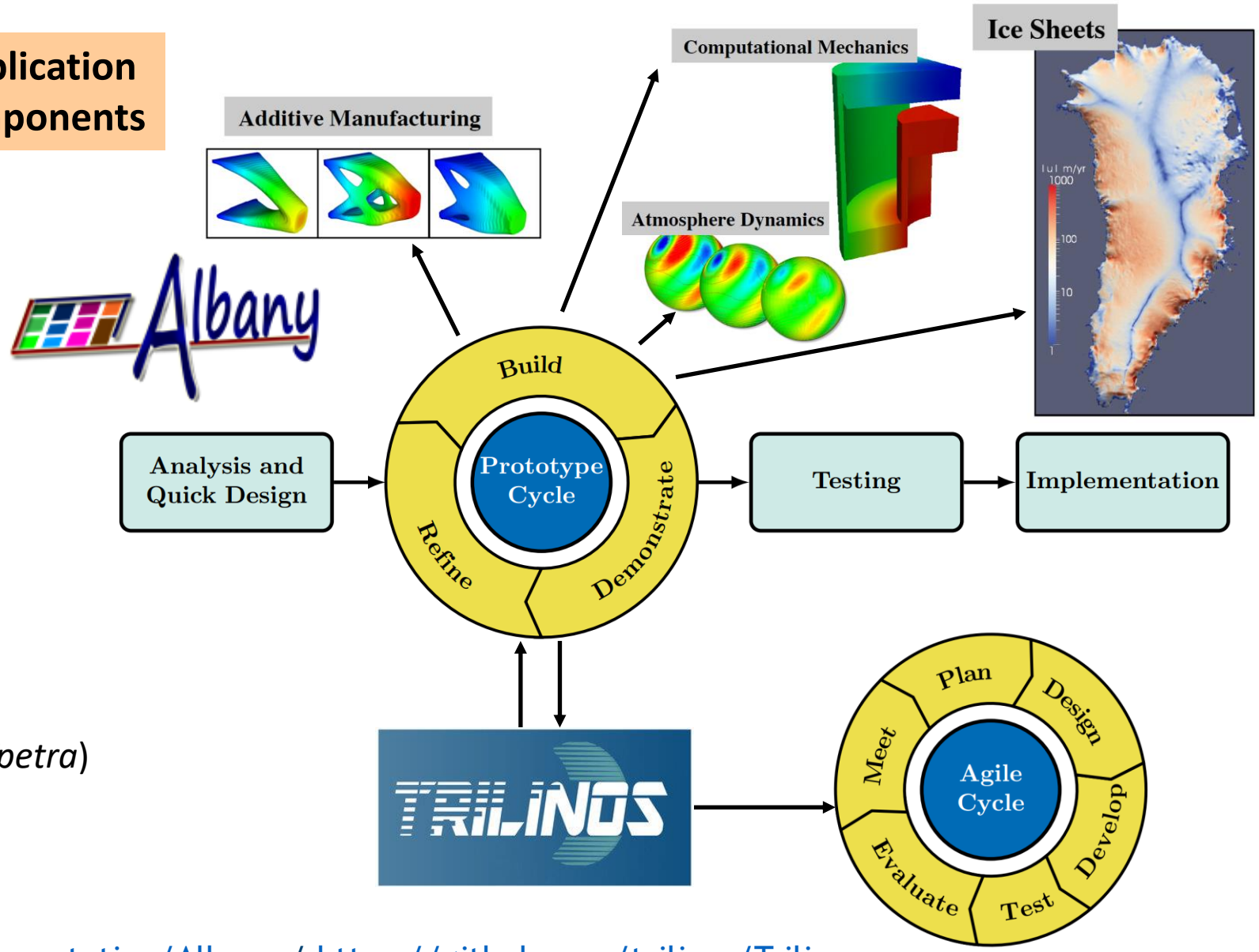
What software tools are we using?

Albany Strategy – finite element codebase in C++

Albany is built primarily for **Rapid Application Development** from **Trilinos Agile Components**

Component Examples (*package name*)

- 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…*



https://github.com/SNLComputation/Albany / https://github.com/trilinos/Trilinos

Albany Overview

Albany provides the "Glue" – connects components

Ex: Finite element assembly (FEA)

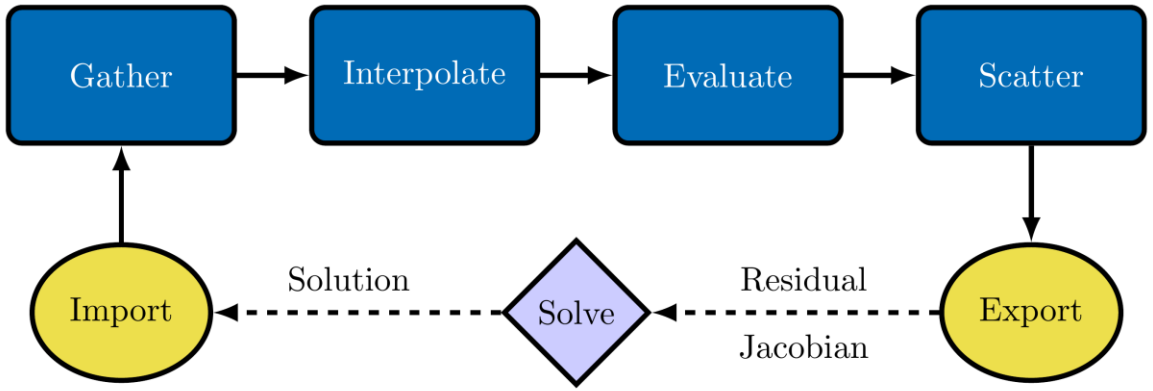
- **Tpetra** manages **distributed** memory linear algebra (**MPI+X**)
- **Phalanx** manages **shared** memory computations (**X**)
  - **Gather** fills element local solution
  - **Interpolate** solution/gradient to quad points
  - **Evaluate** residual/Jacobian
  - **Scatter** fills global residual/Jacobian

New PDEs only require new code for **Evaluate**

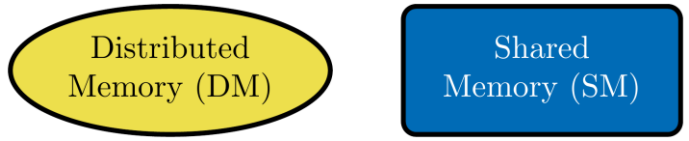
- Leverage existing tools for rapid development

Trilinos Packages

FEA Overview



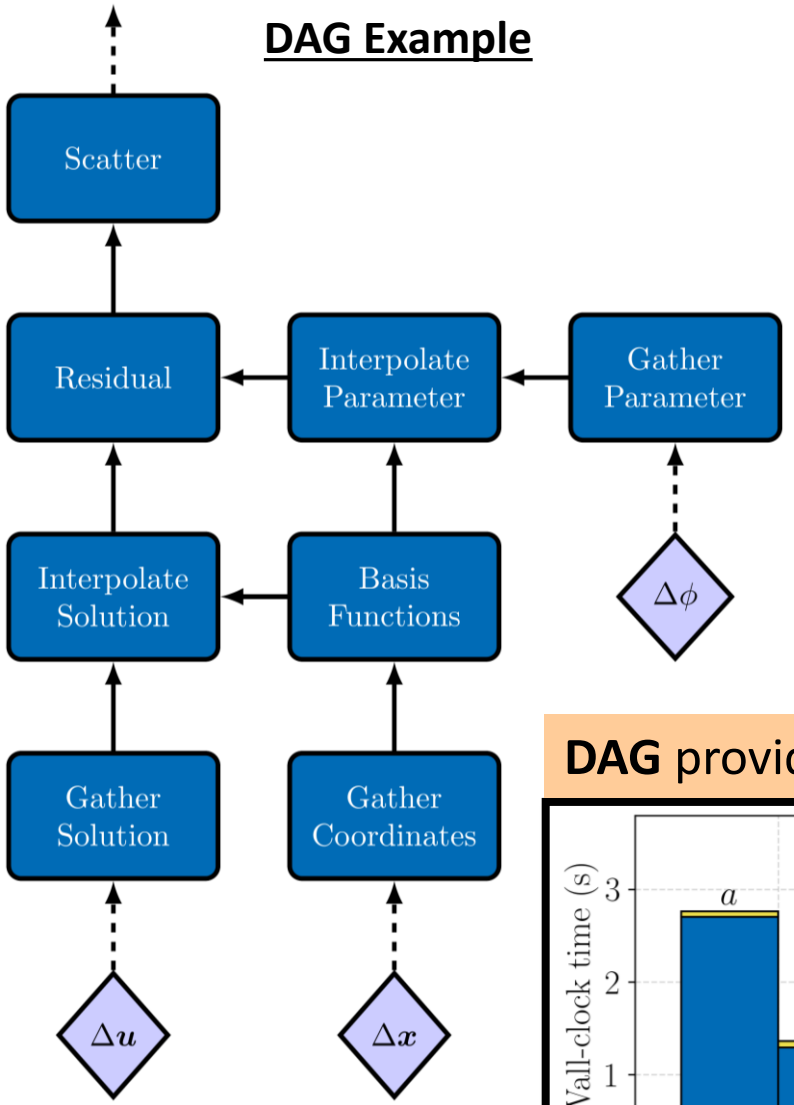
Memory Model



https://github.com/SNLComputation/Albany / https://github.com/trilinos/Trilinos

Phalanx – directed acyclic graph (DAG)

DAG Example



Advantages:

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

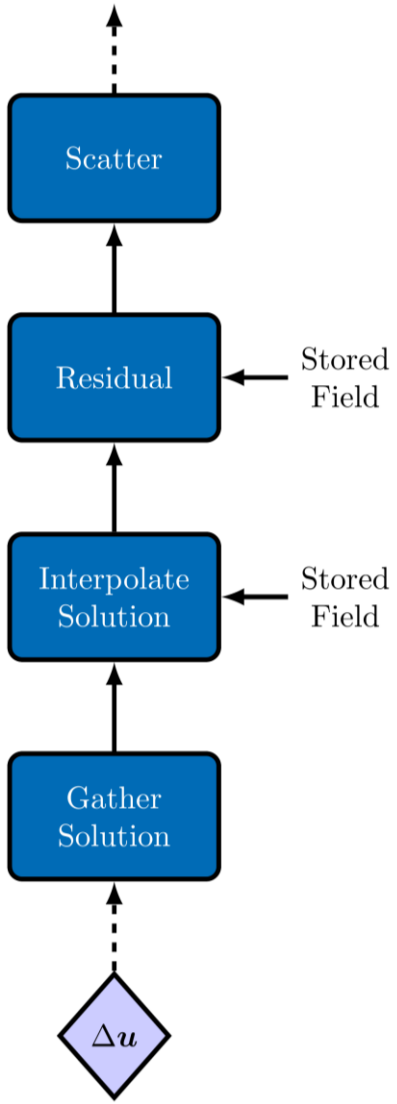
Extension:

- Performance gain through memoization

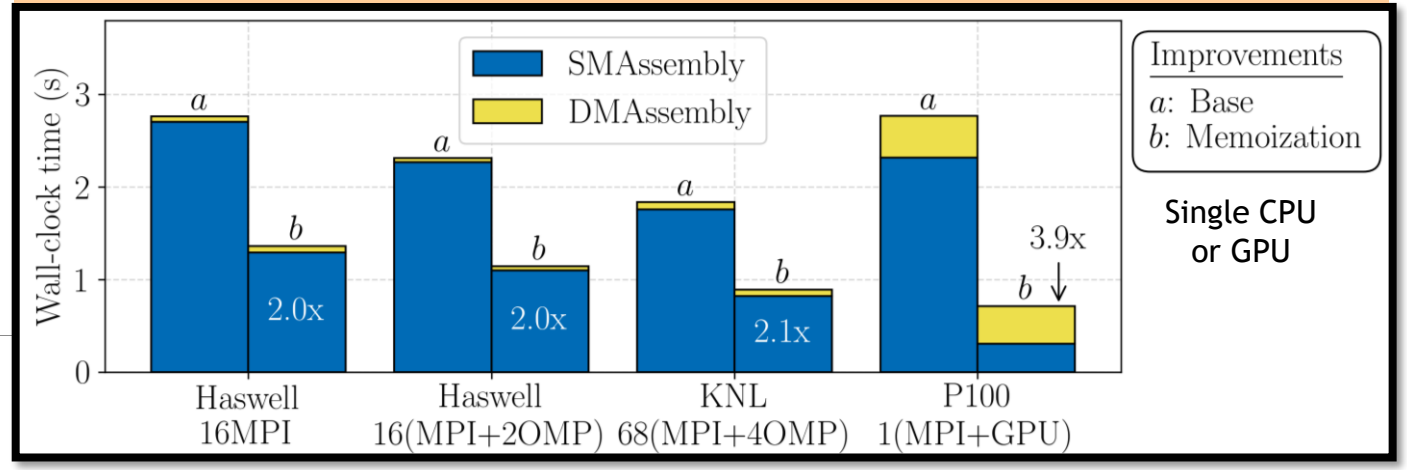
Disadvantage:

- Performance loss through fragmentation

DAG Example (memoization)



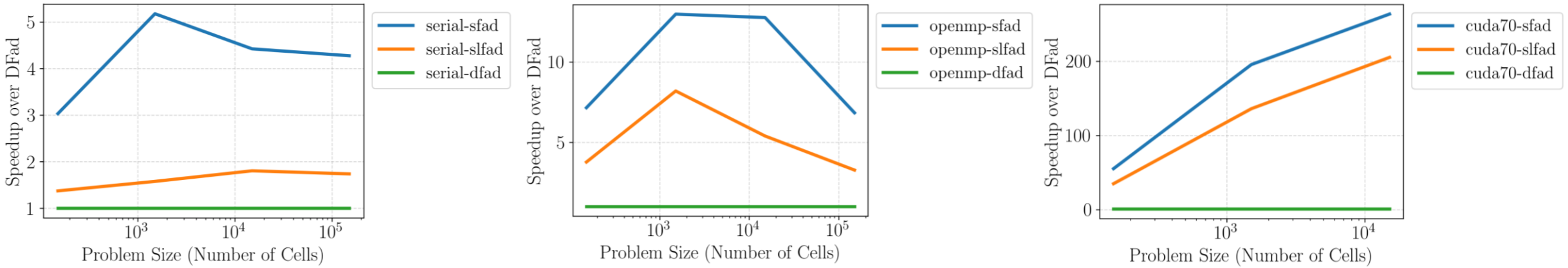
DAG provides flexibility; **Memoization** improves performance



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
  - Ex: 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

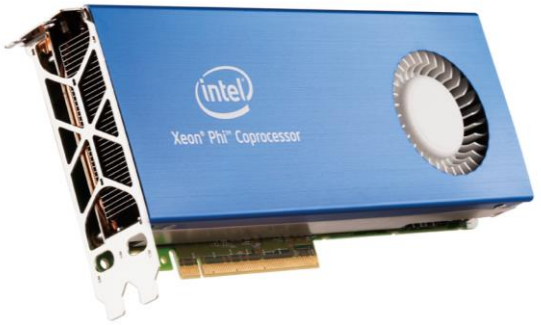
AD capability allows for advanced analysis while maintaining performance portability



Fad Type Comparison: Tetrahedral elements (4 nodes), 2 equations, ND = 4*2 = 8

Kokkos – performance portability

- **Kokkos** 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.

https://github.com/kokkos/kokkos/

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)

```
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 node=0; node<numNodes; ++node){
    Residual(cell,node,0)=0.;
  }
  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);
    }
  }
}
```

Template parameters are used to get hardware specific features.

Linear solver in Albany Land Ice

How are we currently solving the linear system?

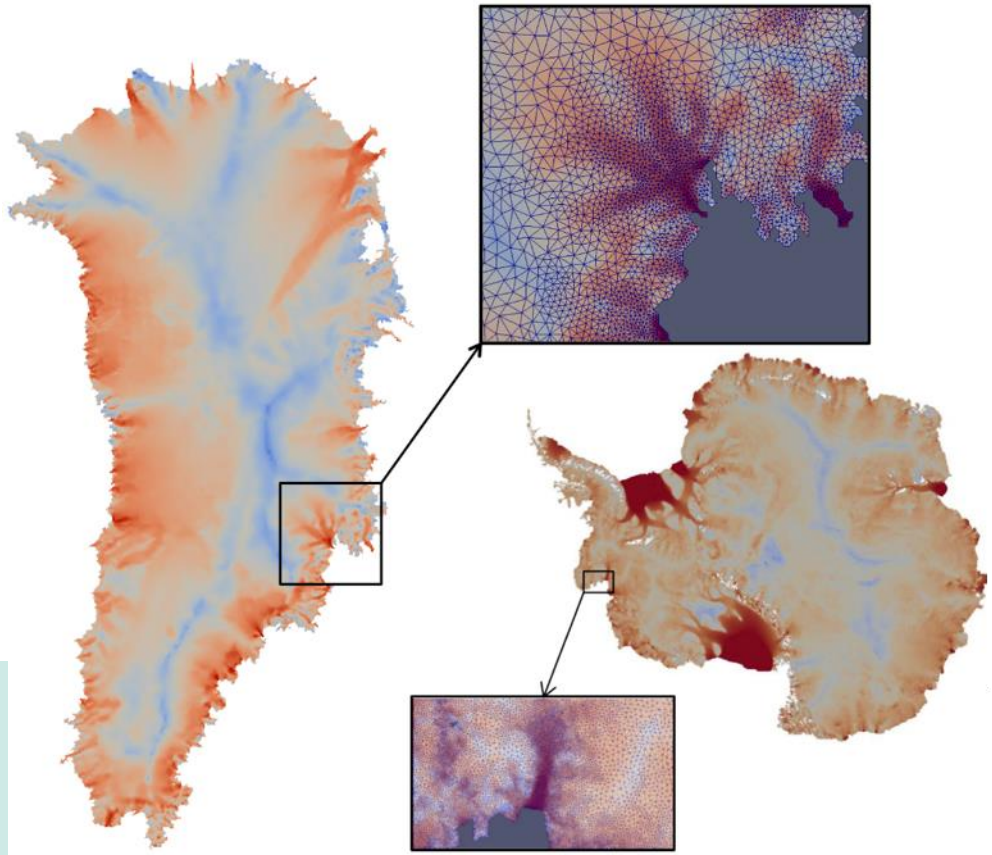
ProSPect – project under SciDAC

ProSPect = Probabilistic Sea Level Projections from Ice Sheet and Earth System Models
5 year SciDAC4 project (2017-2022).

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

Requirements for *Albany Land Ice* (formerly *FELIX*):

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



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

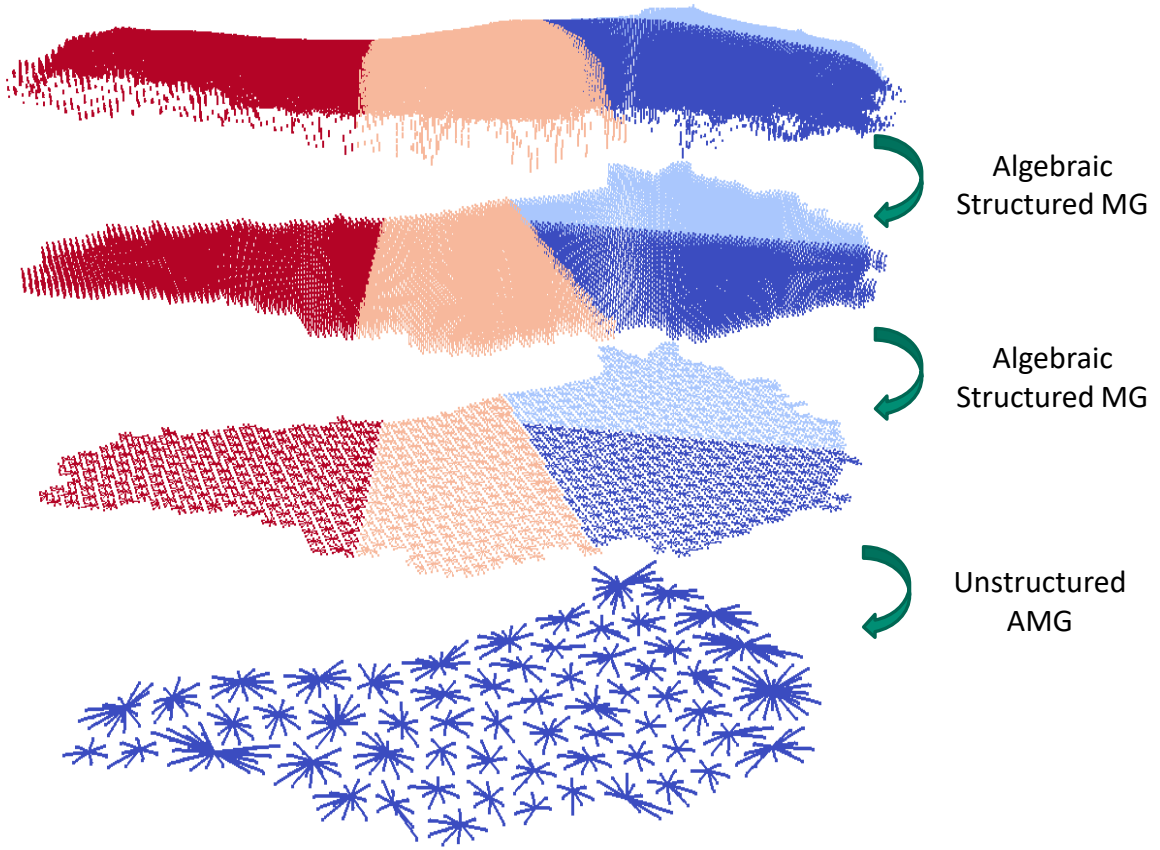
https://doe-prospect.github.io/

Linear solver in Albany Land Ice

Problem: Ice sheet meshes are thin with high aspect ratios

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

- First, matrix-dependent **structured** multigrid to coarsen vertically
- Second, smoothed aggregation **AMG** on single layer
- Implemented in Trilinos – ML/MueLu



Solver: Preconditioned Newton-Krylov

- MDSC-AMG is used as preconditioner for GMRES
- Performance portability through Trilinos/MueLu (multigrid) + Trilinos/Belos (GMRES)

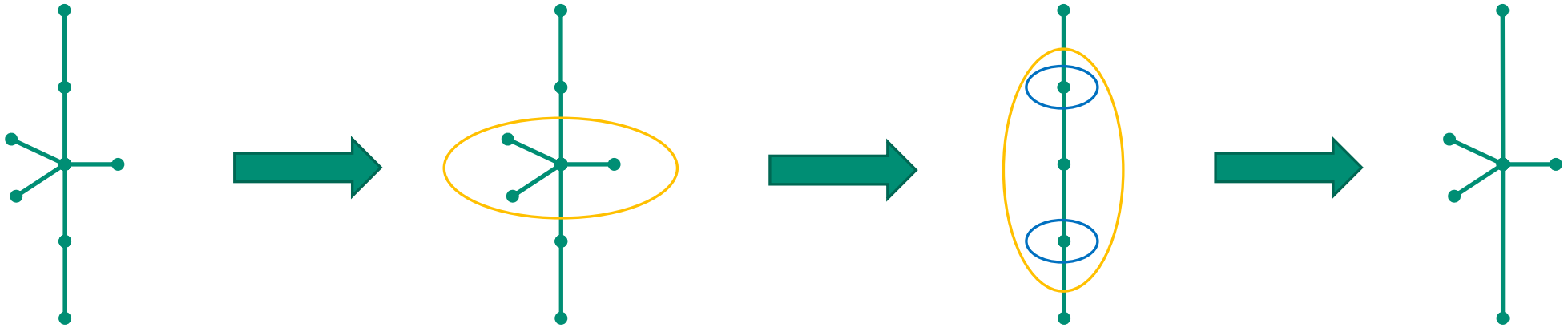
See (Tezaur *et al.,* 2015), (Tuminaro *et al.,* 2016)

Matrix dependent grid transfers

Grid transfers for semicoarsening in the vertical direction (**prolongation matrix assembly**)

- Refactor work needed to ensure matrix assembly is portable

1. Collapse matrix to only contain entries in the vertical direction
   - Sum all values in a plane
   - No horizontal coupling
2. For each point in a coarse layer, solve for the interpolation operator
   - Each thread solves system and fills matrix
   - Kokkos Kernels inline batched LU solve



Prolongation matrix assembly utilizes Kokkos for performance portability

Autotuned performance portable smoothers

Random search used to improve performance of multigrid smoothers on GPU

Smoother parameters:

- Limited to three levels, two smoothers
- Good parameter ranges provided by Trilinos/MueLu team

```
type: RELAXATION
ParameterList:
    'relaxation: type': MT Gauss-Seidel
    'relaxation: sweeps': positive integer
    'relaxation: damping factor': positive real number

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

type: CHEBYSHEV
ParameterList:
    'chebyshev: degree': positive integer
    'chebyshev: ratio eigenvalue': positive real number
    'chebyshev: eigenvalue max iterations': positive integer
```

Results:

- Applied to four cases (Greenland, 3-20km)
  - Different architectures (blake: 8 CPU nodes/weaver: GPU)
  - Different equations (vel: FOStokes/ent: Enthalpy)
- 100 iterations, random search
- Timer: Preconditioner + Linear Solve

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

Cases	#Passed Runs	#Failed Runs	%Failure
blake_vel	70	30	30%
blake_ent	37	63	63%
weaver_vel	71	29	29%
weaver_ent	26	74	74%

Autotuning framework: Carolyn Kao

Numerical results

How well does the solver perform?

Weak Scalability Study

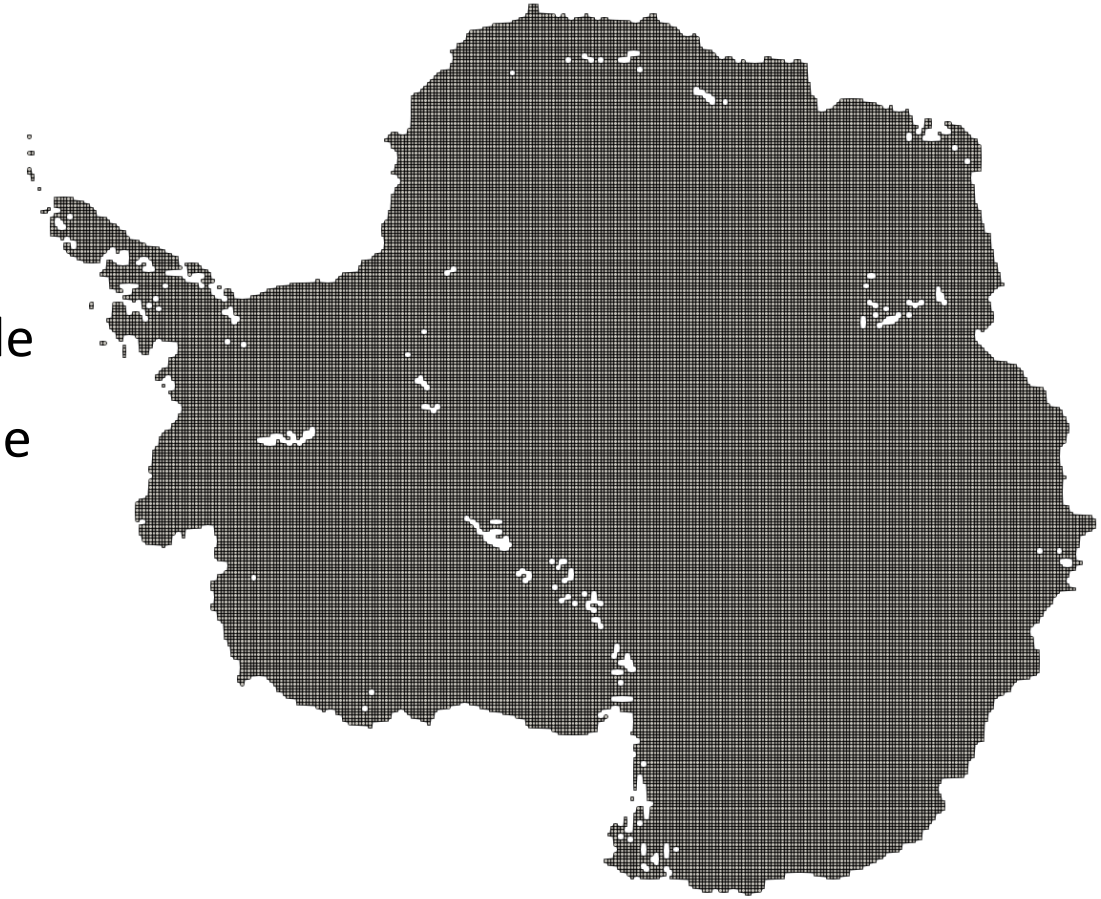
Architectures:

- NERSC Cori-Haswell (**HSW**): 32 cores/node
- NERSC Cori-KNL (**KNL**): 68 cores/node
- OLCF Summit-POWER9-only (**PWR9**): 44 cores/node
- OLCF Summit-POWER9-V100 (**V100**): 44 cores/node + 6 GPU/node

Benchmark:

- First-order Stokes, hexahedral elements
- 16 to 1km structured Antarctica meshes, 20 layers
- 1 to 256 compute nodes

Benchmark used to assess performance



Mesh Example: 16km, structured Antarctica mesh (2.20E6 DOF - 20 layer, 2 equations)

Performance on Cori and Summit

Setup:

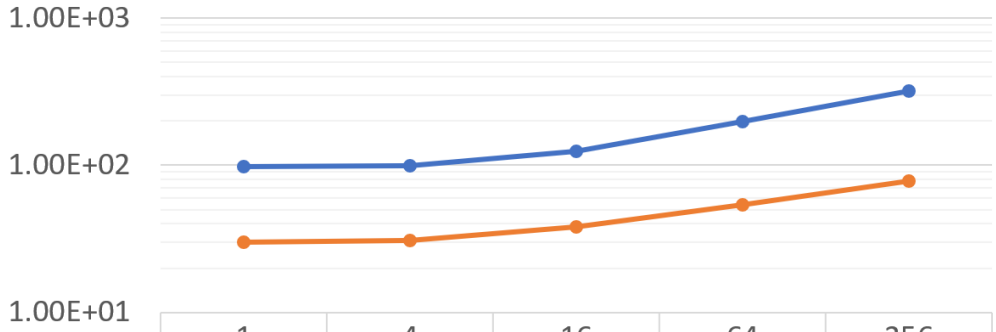
- Same input file for all cases
  - Performance portable point smoothers
  - No architecture specific tuning

Results:

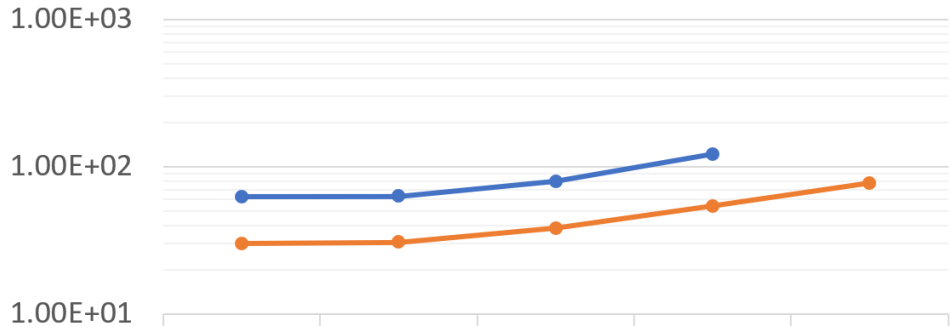
- Performance degrades at higher resolutions
  - (645->1798 total linear iterations)
  - GPU scaling slightly better
- Speedup on GPU
  - 3.2-4.1x speedup Summit over Cori
  - 2.1-2.3x speedup V100 over POWER9

Speedup achieved over MPI-only simulations without architecture specific tuning

Solver Weak Scaling
Wall-clock time (s) vs. Nodes



	1	4	16	64	256
Cori (Haswell)	9.86E+01	9.92E+01	1.25E+02	1.97E+02	3.22E+02
Summit (V100)	3.01E+01	3.08E+01	3.81E+01	5.40E+01	7.78E+01
Speedup	3.27	3.22	3.27	3.65	4.13
DOFs/GPU	367255	367773	368086	368401	368566



	1	4	16	64	256
Summit (POWER9)	6.24E+01	6.31E+01	7.96E+01	1.22E+02	
Summit (V100)	3.01E+01	3.08E+01	3.81E+01	5.40E+01	7.78E+01
Speedup	2.07	2.05	2.09	2.26	
DOFs/GPU	367255	367773	368086	368401	368566

Performance on Cori and Summit

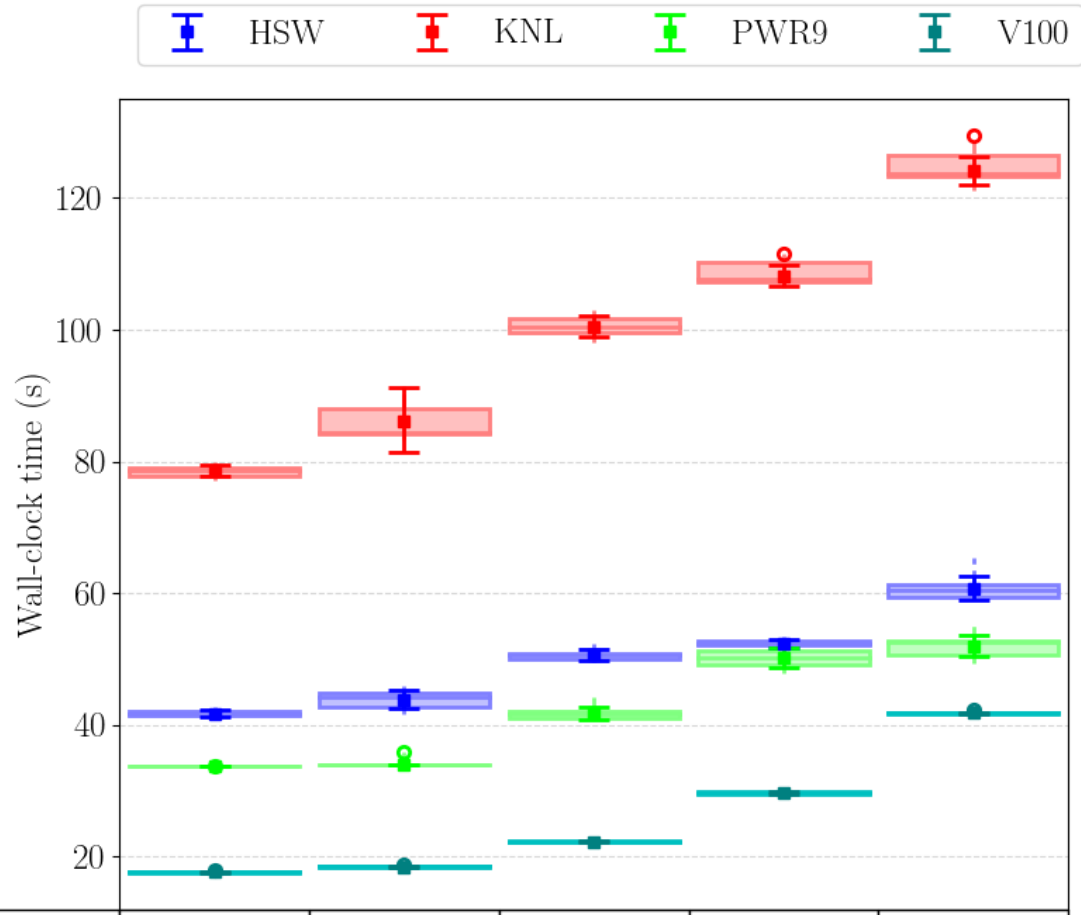
Setup:

- Tuned input files
  - CPU block preconditioner
  - Autotuned GPU point smoothers
- Multiple samples for confidence

Results:

- CPU scales better than GPU
  - 16->18 avg. linear iterations on CPU
  - 88->194 avg. linear iterations on GPU
- Speedup on GPU
  - 1.9->1.2 speedup V100 over POWER9
  - Speedup degrades at higher resolutions

Speedup over MPI-only simulations;
Tuned CPU model scales better



Resolution	16km	8km	4km	2km	1km
# Nodes	1	4	16	64	256
V100 Speedup	1.92	1.85	1.88	1.70	1.24
99% CI	(1.91, 1.92)	(1.84, 1.86)	(1.84, 1.92)	(1.65, 1.74)	(1.21, 1.28)

Areas to improve

Weak Scaling Efficiency:

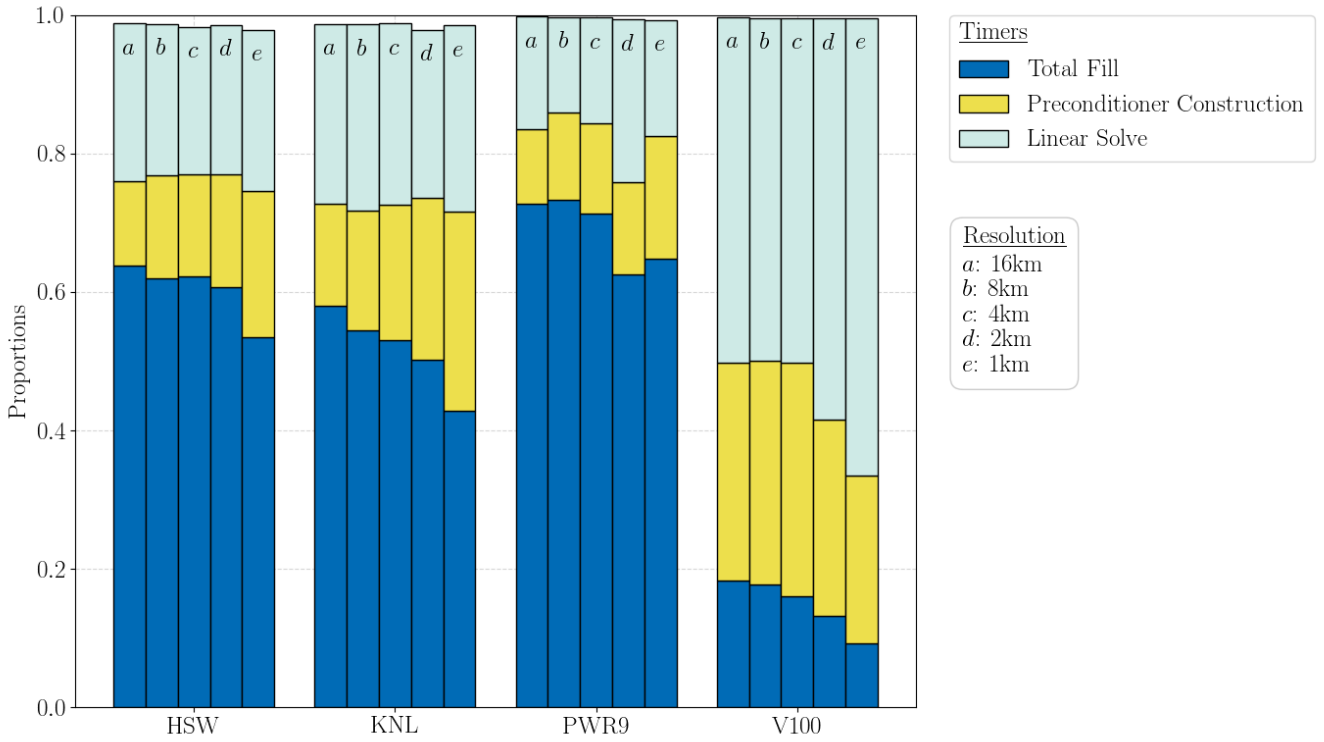
- Higher is better
- Areas of improvement
  - CPU/GPU preconditioner construction
  - GPU linear solve (better precond.)

Proportions of total solve time:

- Improve assembly on CPU
  - 40-60% of total solve time
- Improve GPU linear solver
  - 80-90% of total solve time

Focus on improving GPU solver

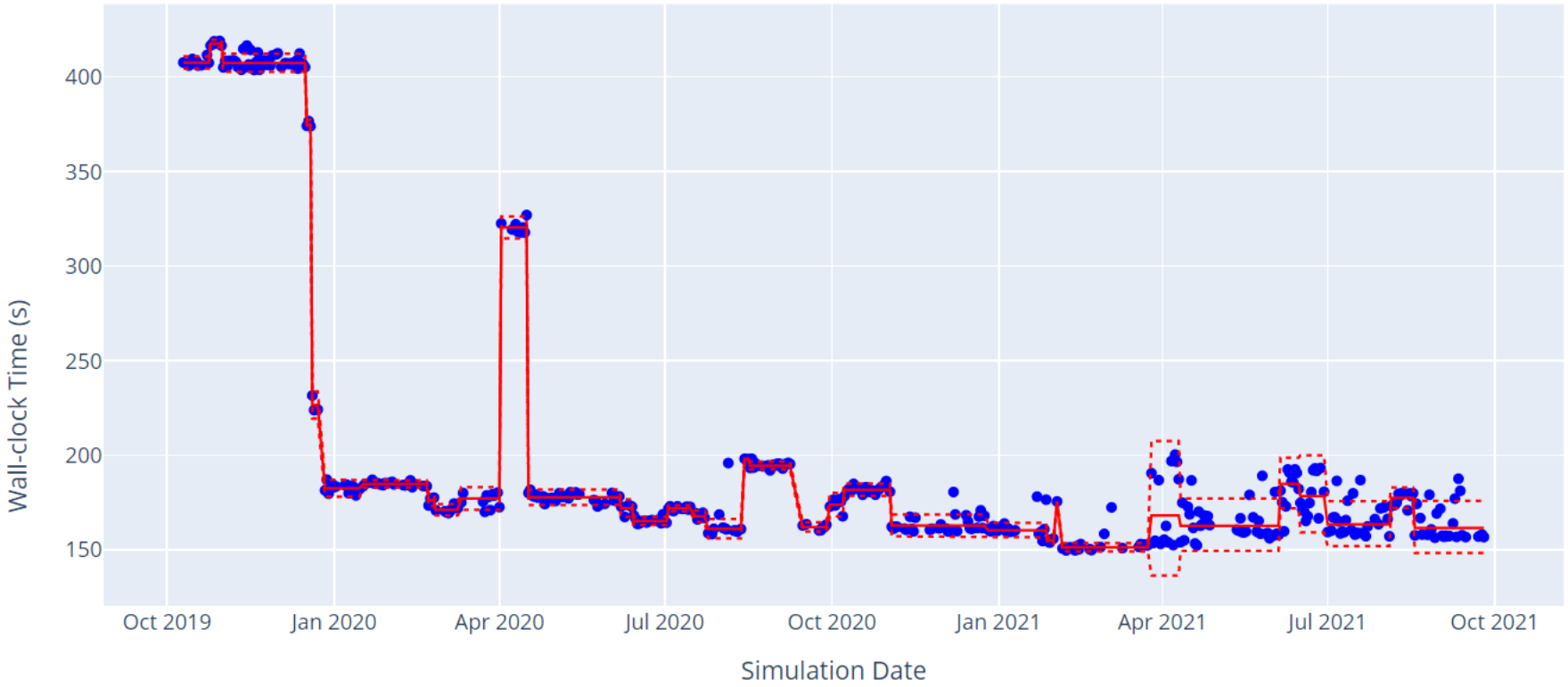
	Total Solve	Total Fill	Preconditioner Construction	Linear Solve
HSW	68.9% (67.0, 70.9)	82.2% (81.5, 82.9)	41.2% (38.2, 44.5)	67.5% (66.2, 68.8)
KNL	63.5% (62.3, 64.6)	85.3% (84.5, 86.0)	33.0% (30.8, 35.5)	61.1% (60.6, 61.6)
PWR9	65.1% (63.3, 66.9)	73.1% (70.0, 76.4)	39.5% (39.0, 40.0)	63.0% (62.9, 63.1)
V100	42.2% (42.0, 42.4)	82.9% (80.5, 85.4)	55.2% (54.7, 55.8)	31.9% (31.6, 32.2)

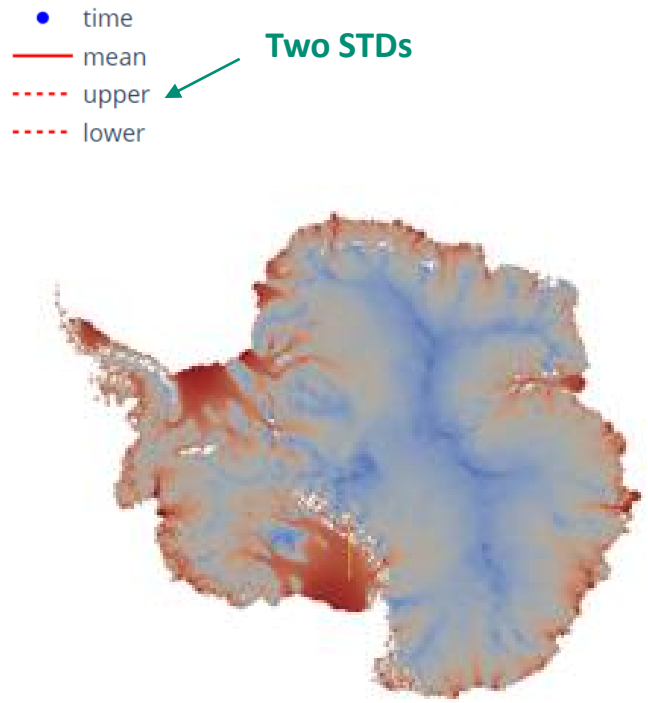


Changepoint detection for performance testing

Maintaining/improving performance and portability in the presence of **active development** is essential

- **Changepoint detection:** process of finding abrupt variations in time series data
- Manual testing and analysis is increasingly infeasible





Total simulation time for a 2-20km resolution Antarctica mesh, executed nightly in Albany Land Ice

Changepoint Detection: Kyle Shan

Conclusions

Conclusions

- 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**
- **Recent improvements** in the **linear solve** of Albany Land Ice has allowed for scalable **performance portable** ice sheet modeling
- **Performance** on next generation computing architectures is a **work in progress**
  - 1.9->1.2x speedup of V100 node over POWER9 node in total solve time (tuned solvers)
  - CPU scales better than GPU using best solvers (65.1% vs. 41.2% weak scaling efficiency)
- Maintaining **performance** and **portability** is crucial for an active code base
  - A change-point detection algorithm can help identify performance variation

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

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

OAK RIDGE National Laboratory