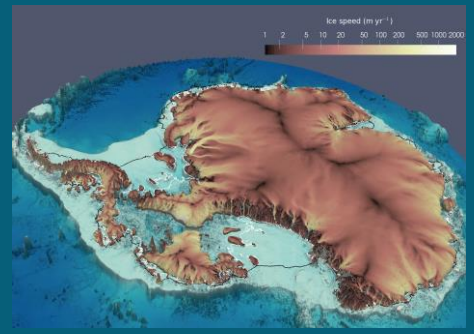
This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

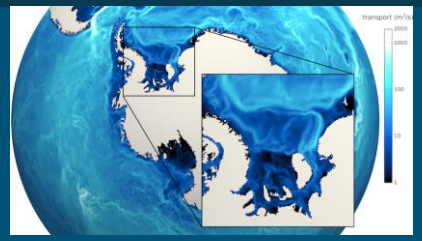
SAND2022-12674C

Sandia National Laboratories

# Performance, portability and productivity in ice-sheet modeling using MALI









September 20th, 2022

PRESENTED BY

Jerry Watkins, Max Carlson, Irina Tezaur, Mauro Perego

Earth Science Performance, Productivity, Reliability Engineering

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 performance, portability and productivity?
2) MALI software
3) Performance, portability and productivity
4) Conclusions

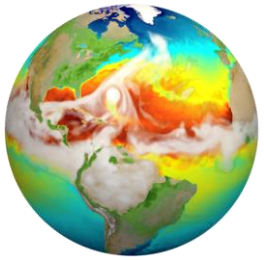
# Motivation

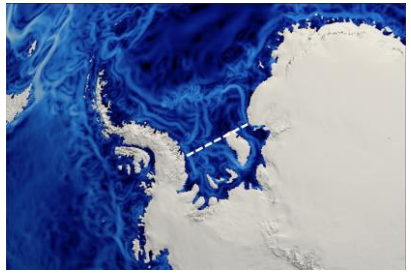
**Why are we interested in performance, portability and productivity?**

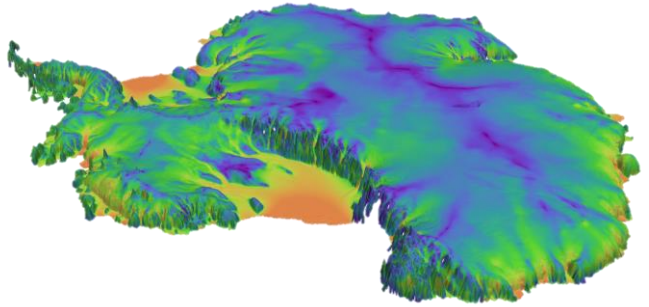
# Motivation

**High-fidelity simulations of the DOE E3SM's ice sheet model, MALI, on exascale systems**

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








OLCF Summit
NVIDIA V100 GPU

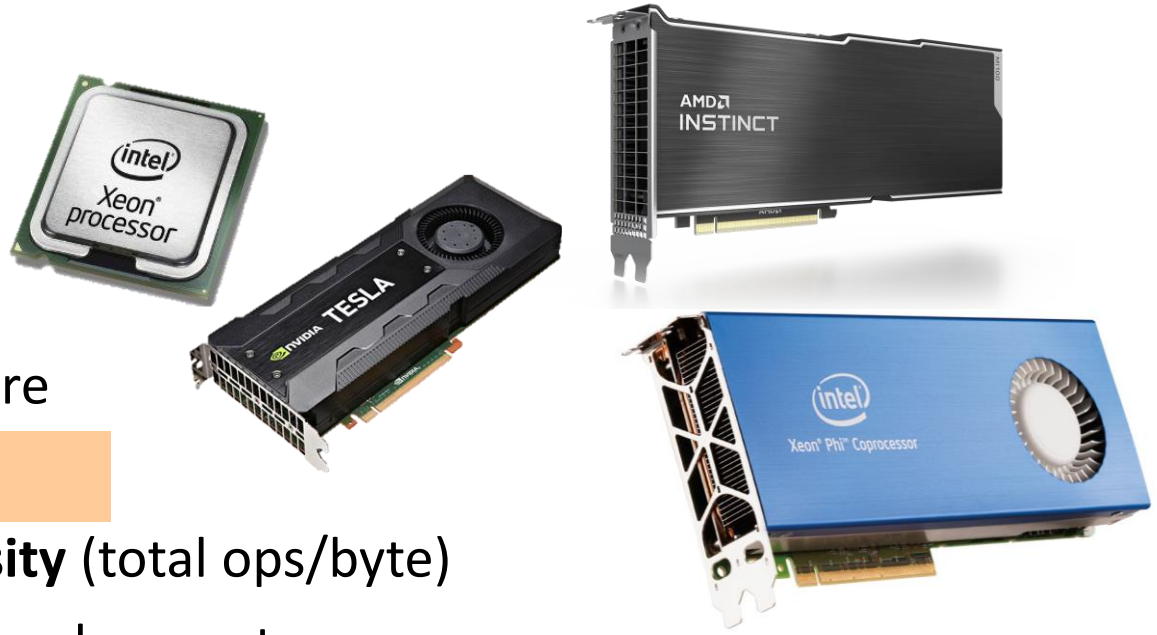

ALCF Aurora
Intel Xe GPU


OLCF Frontier
AMD Instinct GPU


NERSC Perlmutter
NVIDIA A100 GPU

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

# Performance portability for exascale computing




**Challenges:**

- Diverse set of HPC vendors and architectures
  - Intel, AMD, NVIDIA, IBM, ARM-based
  - CPUs with vector processing; GPUs
- Software life cycle is much longer than hardware

**Different architectures, trend remains the same**

- Need algorithms with **higher arithmetic intensity** (total ops/byte)
- Need fundamental **abstractions** during code development

**Performance portability:** A reasonable level of performance is achieved across a wide variety of computing architectures with the same source code.

**Approaches:**

- **Libraries** – High-level abstractions with specified input/output (e.g. BLAS)
- **Task-based** – Data-centric abstractions for mapping tasks to resources (e.g. Legion)
- **MPI+X** – Algorithmic-level abstractions for distributed (MPI) and shared (X) memory parallelism (e.g. **Directives**: OpenMP, OpenACC; **Frameworks**: Kokkos, RAJA, OCCA)

# MALI software

What software tools are we using?

# MALI (MPAS-Albany Land Ice) software

**MPAS:**

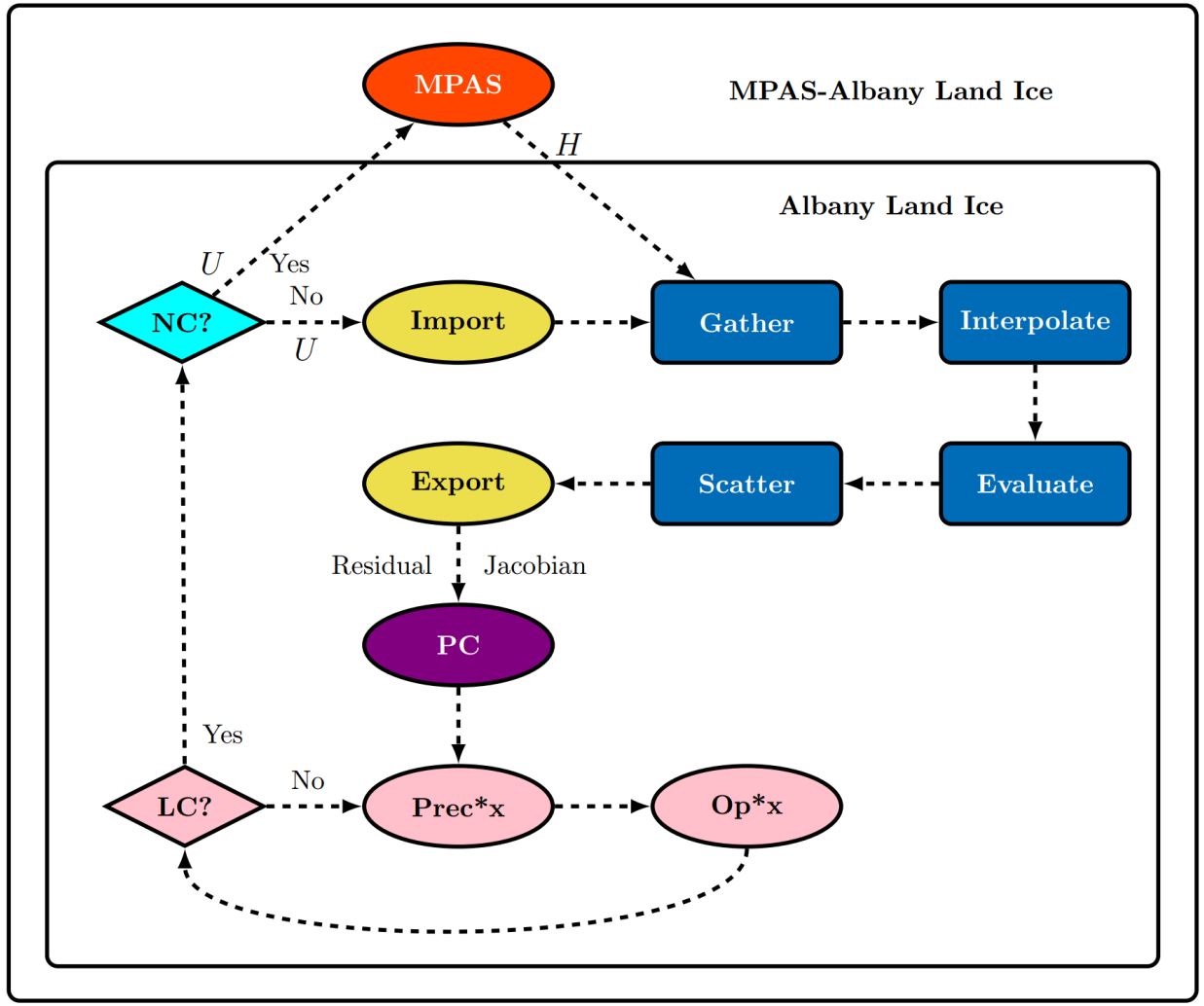
- Thickness/Temperature evolution

**Albany Land Ice:**

- **First-order Stokes velocity solver**

**Trilinos:**

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




https://github.com/MALI-Dev/E3SM / https://github.com/sandialabs/Albany / https://github.com/trilinos/Trilinos

# First Order (FO) Stokes/Blatter-Pattyn Model

Ice behaves like a **very viscous non-Newtonian shear-thinning fluid** (like lava flow) and is modeled **quasi-statically** using **nonlinear incompressible Stokes equations**.

$$\begin{cases} -\nabla \cdot \boldsymbol{\tau} + \nabla p = \rho \boldsymbol{g} \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases}, \quad \text{in } \Omega$$

- Fluid velocity vector: $\boldsymbol{u} = (u_1, u_2, u_3)$
- Isotropic ice pressure: $p$
- Deviatoric stress tensor: $\boldsymbol{\tau} = 2\mu\boldsymbol{\epsilon}$

- Strain rate tensor: $\epsilon_{\text{ij}} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$
- Glen's Law Viscosity*: $\mu = \frac{1}{2}A(T)^{-\frac{1}{n}}\left(\frac{1}{2}\sum_{ij}\boldsymbol{\epsilon}_{ij}{}^2\right)^{\left(\frac{1}{2n}-\frac{1}{2}\right)}$
- Flow factor: $A(T) = A_0 e^{-\frac{Q}{RT}}$

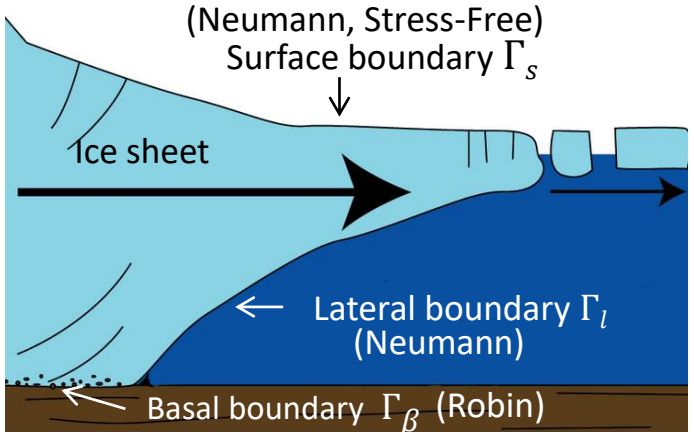
Stokes$(\boldsymbol{u}, p)$ in $\Omega \in \mathbb{R}^3$

↓

FO Stokes$(u, v)$ in $\Omega \in \mathbb{R}^3$

$$\begin{aligned} -\nabla \cdot (2\mu\dot{\boldsymbol{\epsilon}}_1) &= -\rho g \frac{\partial s}{\partial x} \\ -\nabla \cdot (2\mu\dot{\boldsymbol{\epsilon}}_2) &= -\rho g \frac{\partial s}{\partial y} \end{aligned}, \quad \text{in } \Omega$$

***Hydrostatic approximation** + **scaling argument** based on the fact that ice sheets are thin and normals are almost vertical*




**Discussion:**

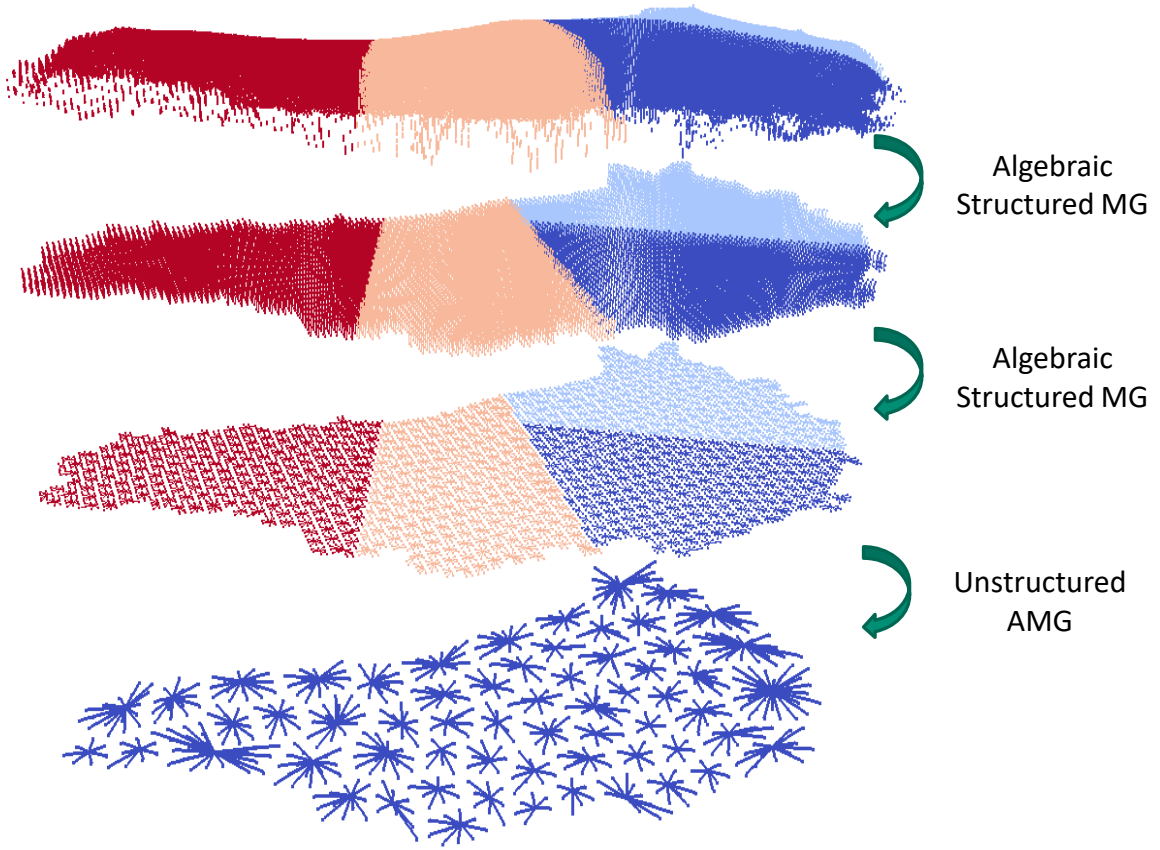
- Nice "**elliptic**" approximation to full Stokes.
- 3D model for two unknowns $(u, v)$ with nonlinear $\mu$.
- Valid for both **Greenland** and **Antarctica** and used in **continental scale** simulations.

# MueLu/Belos – preconditioned iterative solver

**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



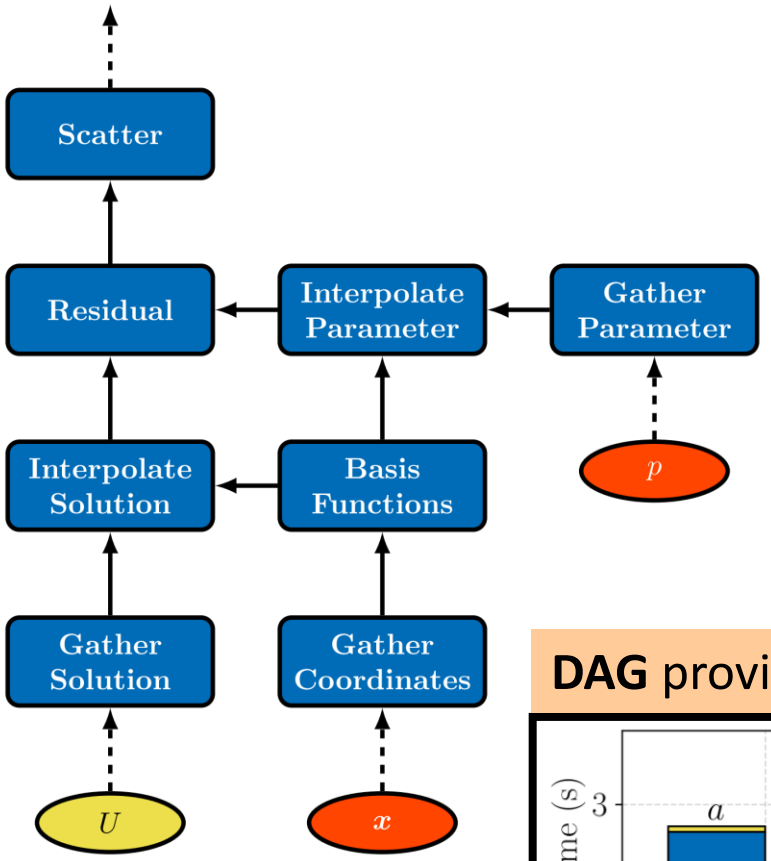

**Solver:** Preconditioned Newton-Krylov

- 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)

# Phalanx – directed acyclic graph (DAG)

**DAG Example**




**Advantages:**

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

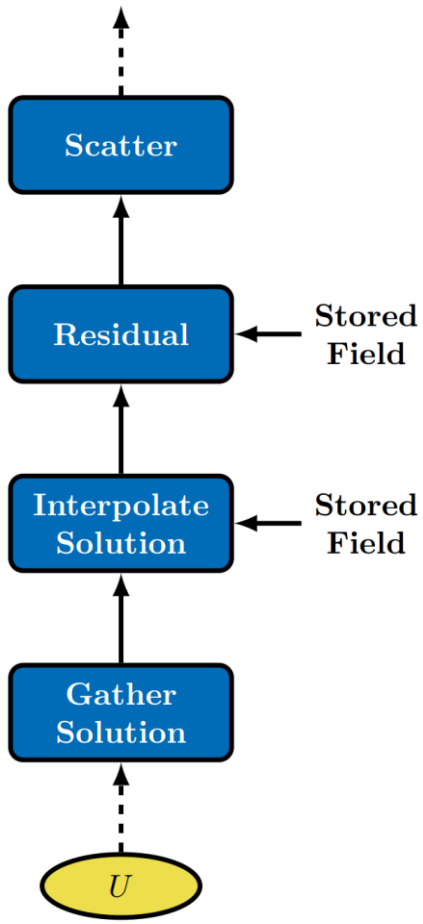
**Disadvantage:**

- Performance loss through fragmentation

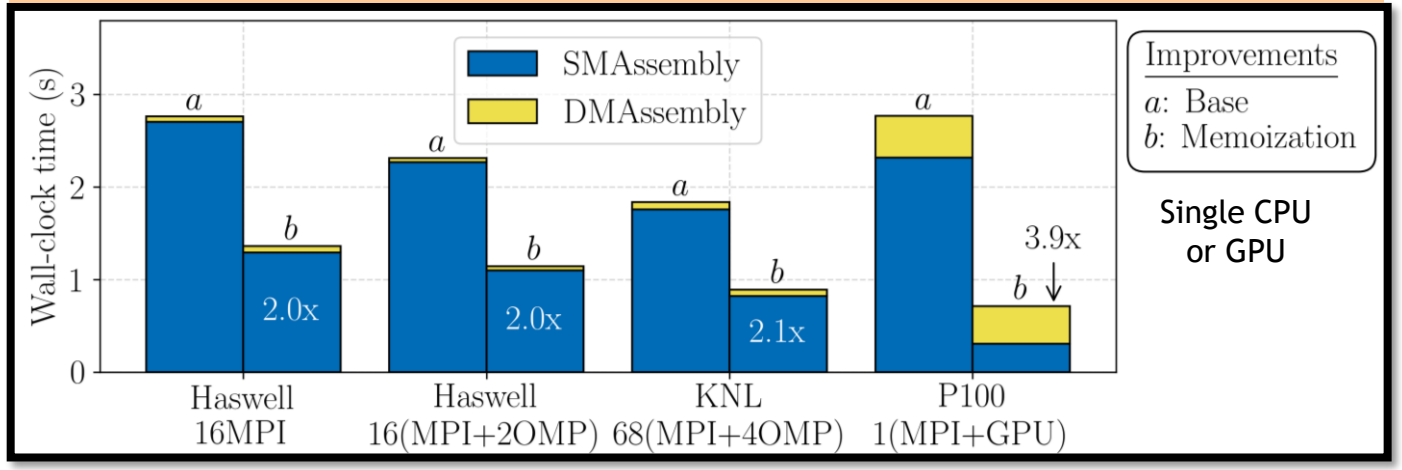
**Extension:**

- Performance gain through memoization

**DAG Example (memoization)**



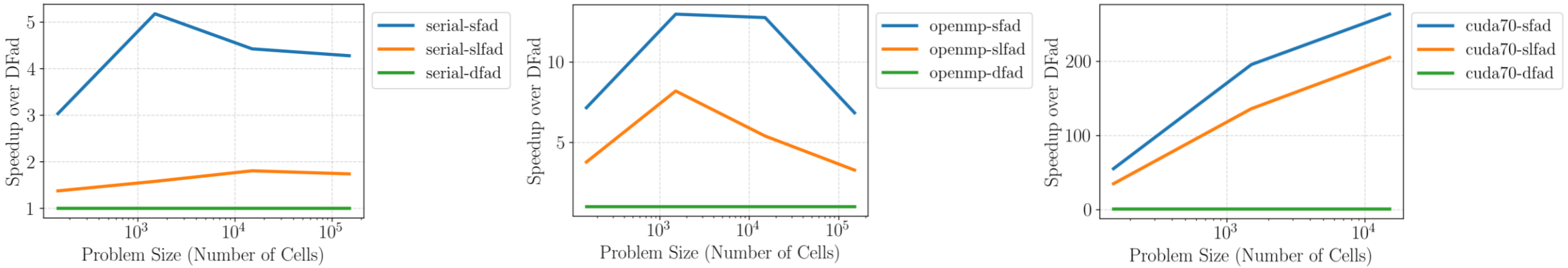

**DAG** provides flexibility; **Memoization** improves performance



# 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

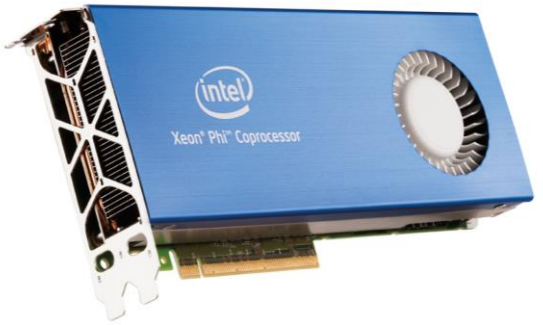
**AD** capability allows for advanced analysis while maintaining performance portability




**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.

# Performance, portability and productivity

## How well does MALI perform?

# Weak Scalability Study

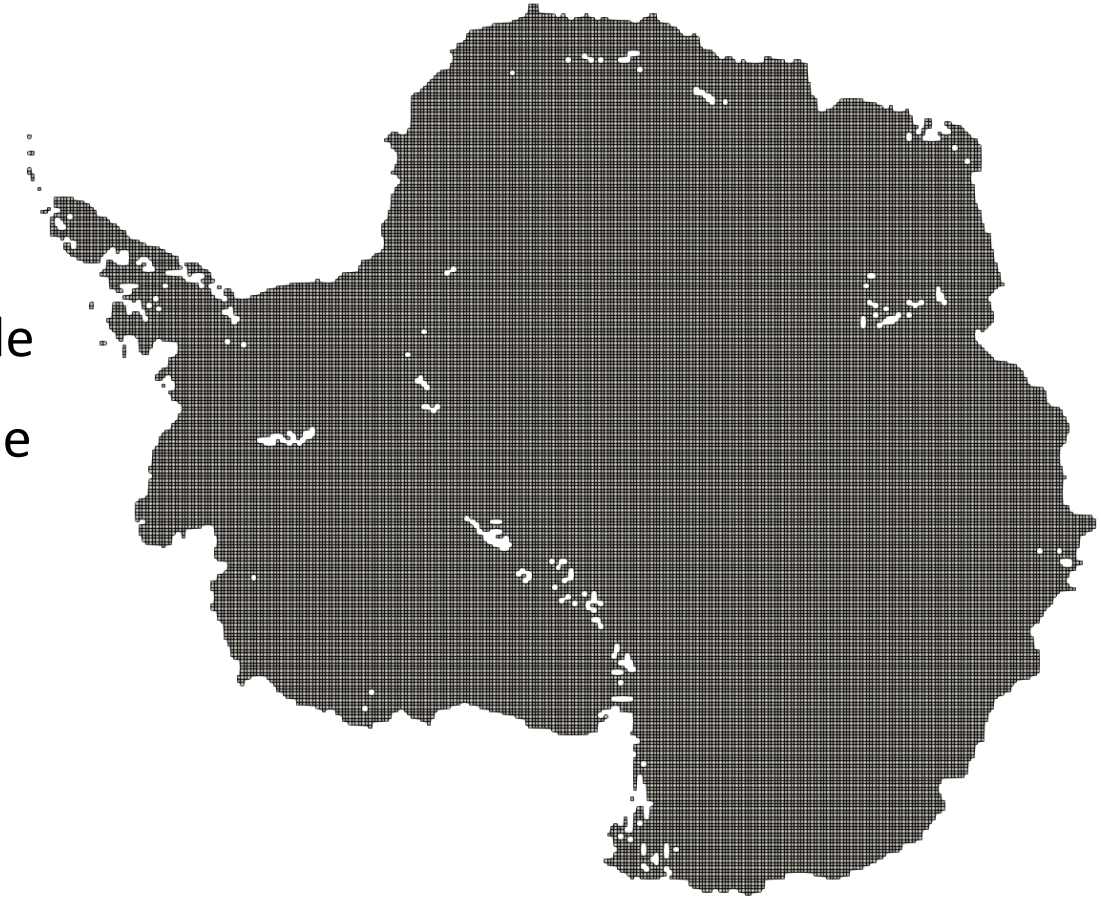
**Architectures:**

- 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:**

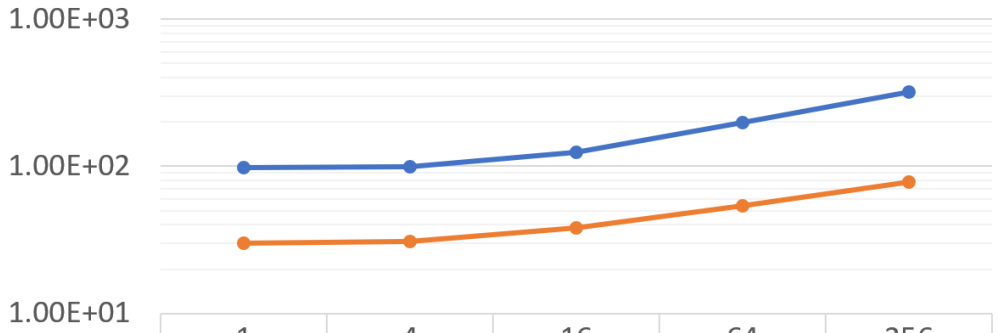
- Same input file for all cases
  - Performance portable point smoothers
  - No architecture specific tuning

**Results:**

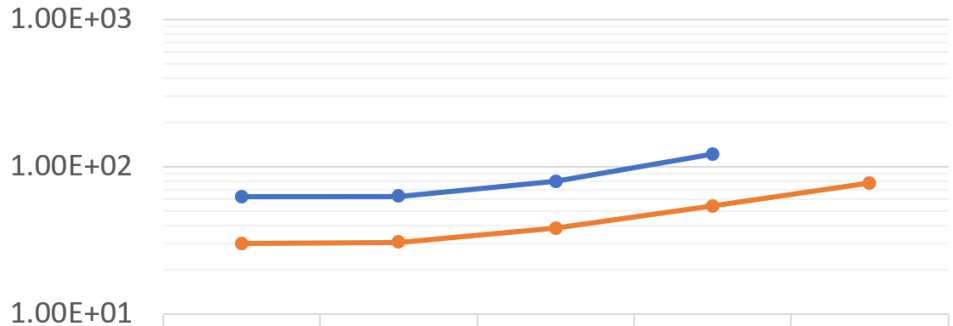
- Performance degrades at higher resolutions
  - (645->1798 total linear iterations)
  - GPU scaling slightly better
- Speedup on GPU
  - 3.2-4.1x speedup Summit over Cori
  - 2.1-2.3x speedup V100 over POWER9

Speedup achieved over MPI-only simulations without architecture specific tuning

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



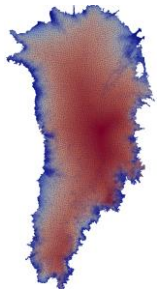

| | 1 | 4 | 16 | 64 | 256 |
|---|---|---|---|---|---|
| Cori (Haswell) | 9.86E+01 | 9.92E+01 | 1.25E+02 | 1.97E+02 | 3.22E+02 |
| Summit (V100) | 3.01E+01 | 3.08E+01 | 3.81E+01 | 5.40E+01 | 7.78E+01 |
| Speedup | 3.27 | 3.22 | 3.27 | 3.65 | 4.13 |
| DOFs/GPU | 367255 | 367773 | 368086 | 368401 | 368566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




| | 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 |

# 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

# Performance on Cori and Summit

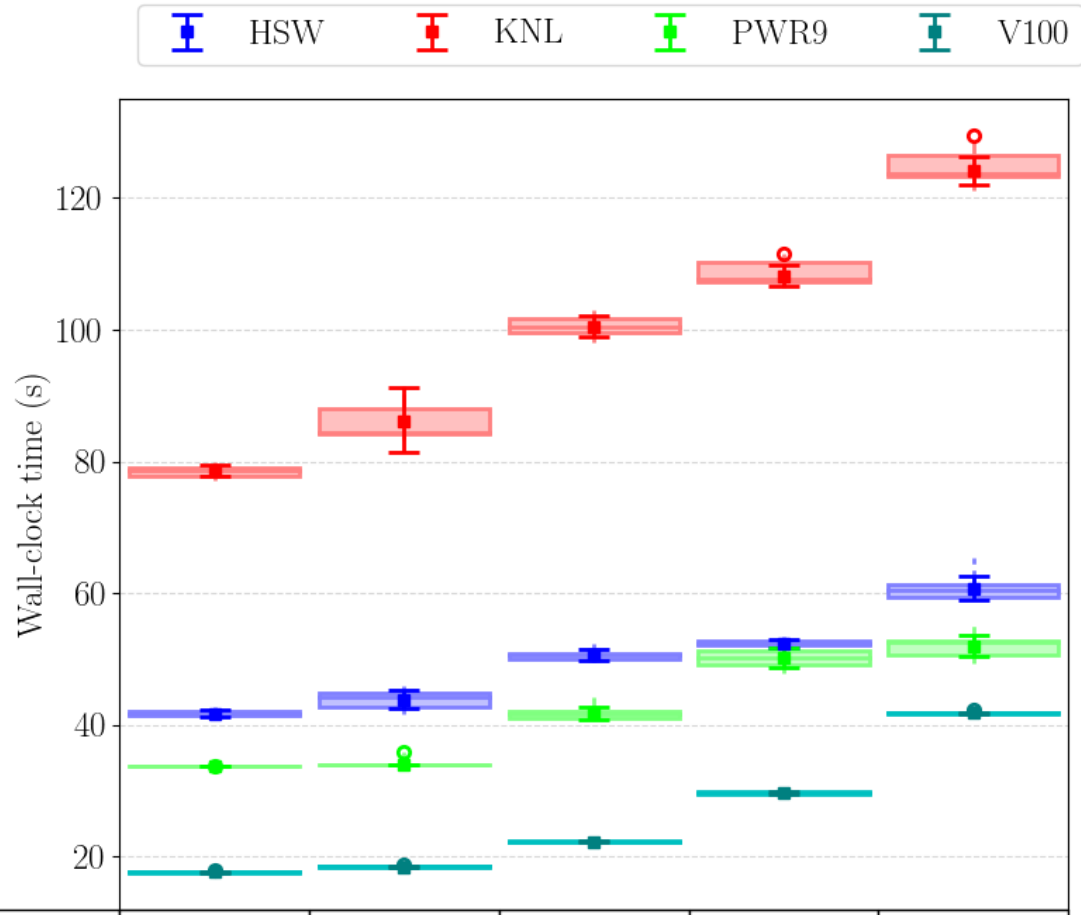
**Setup:**

- 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:

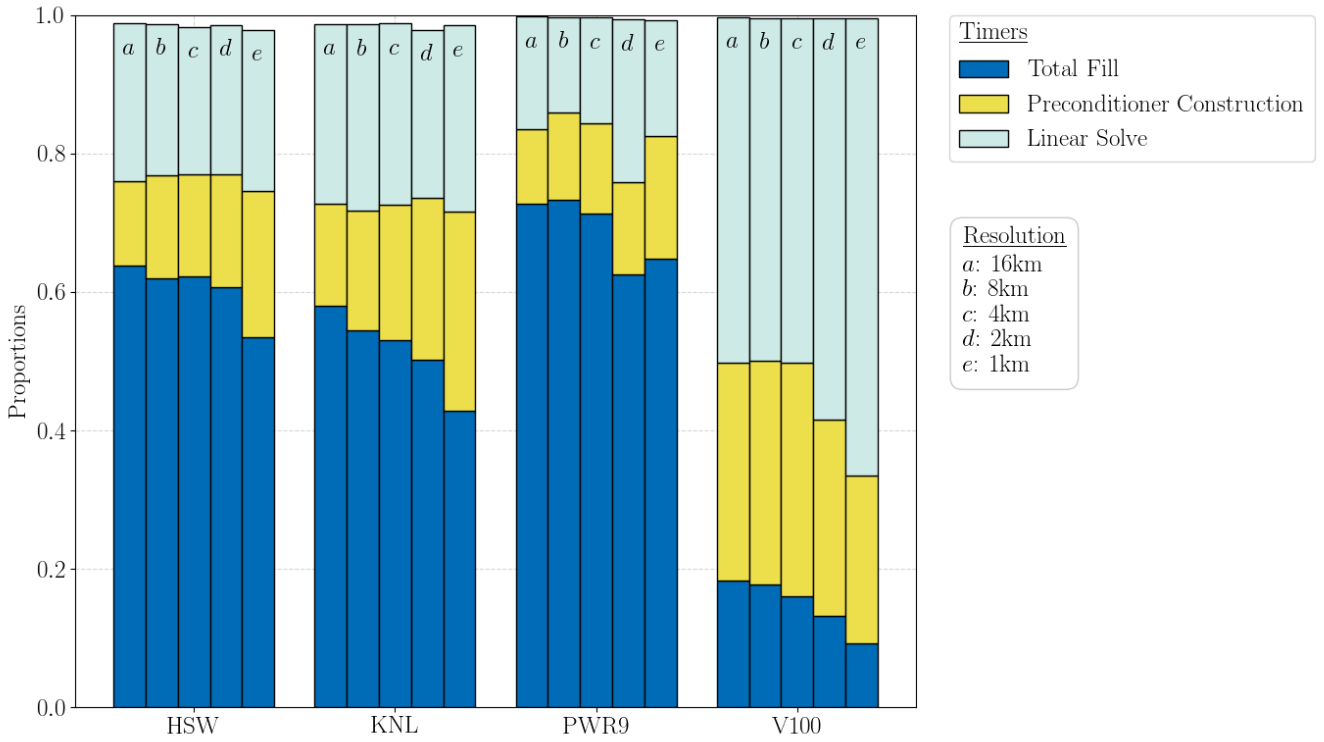
- Higher is better
- Areas of improvement
  - CPU/GPU preconditioner construction
  - GPU linear solve (better precond.)

## 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

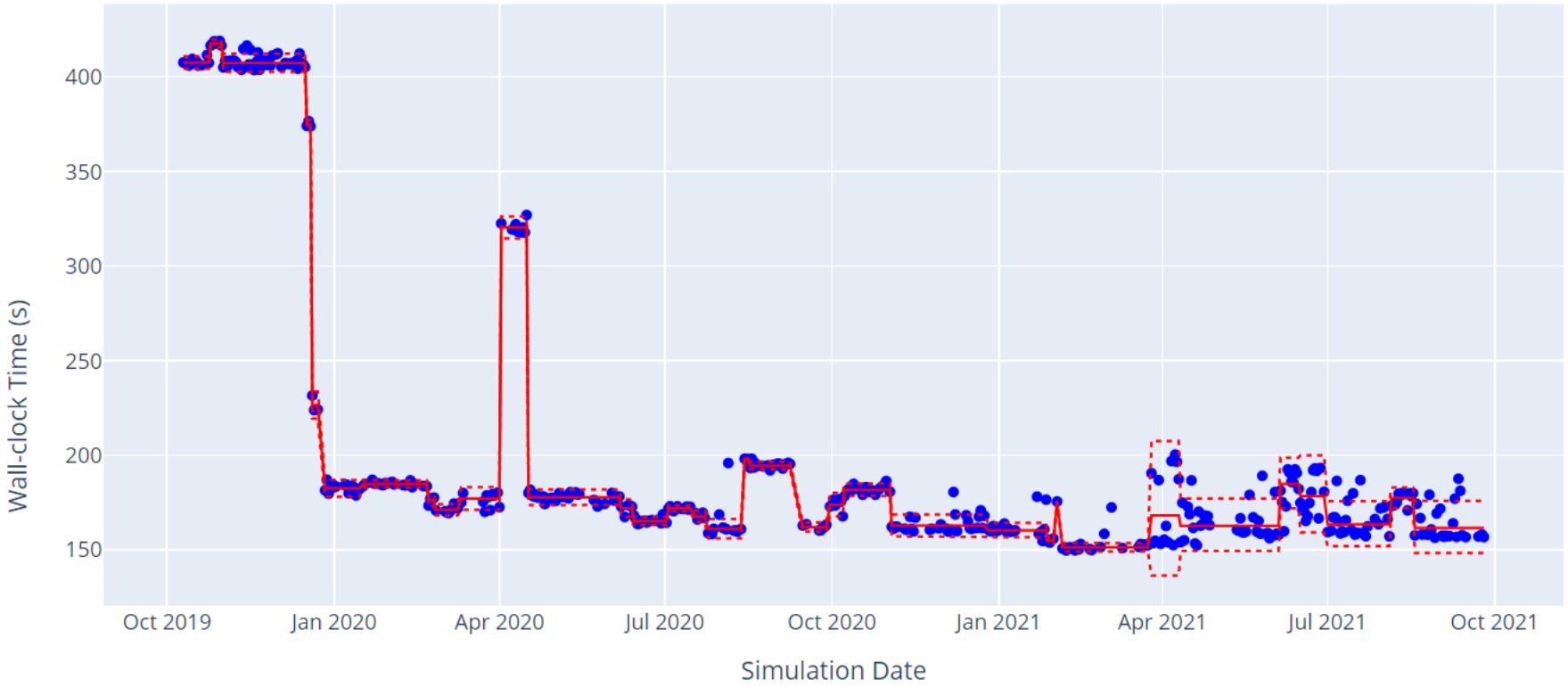
| | Total Solve | Total Fill | Preconditioner Construction | Linear Solve |
|---|---|---|---|---|
| HSW | 68.9% (67.0, 70.9) | 82.2% (81.5, 82.9) | 41.2% (38.2, 44.5) | 67.5% (66.2, 68.8) |
| KNL | 63.5% (62.3, 64.6) | 85.3% (84.5, 86.0) | 33.0% (30.8, 35.5) | 61.1% (60.6, 61.6) |
| PWR9 | 65.1% (63.3, 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) |

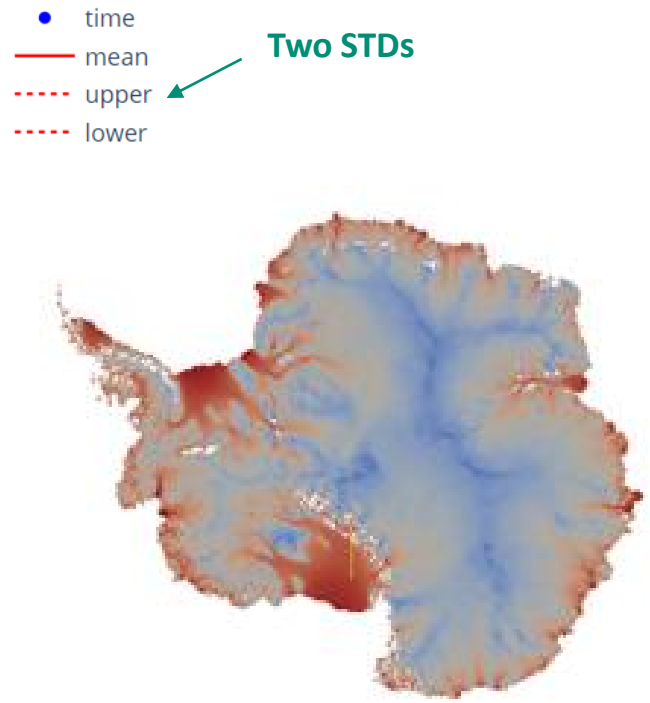


# Changepoint detection for performance testing

**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



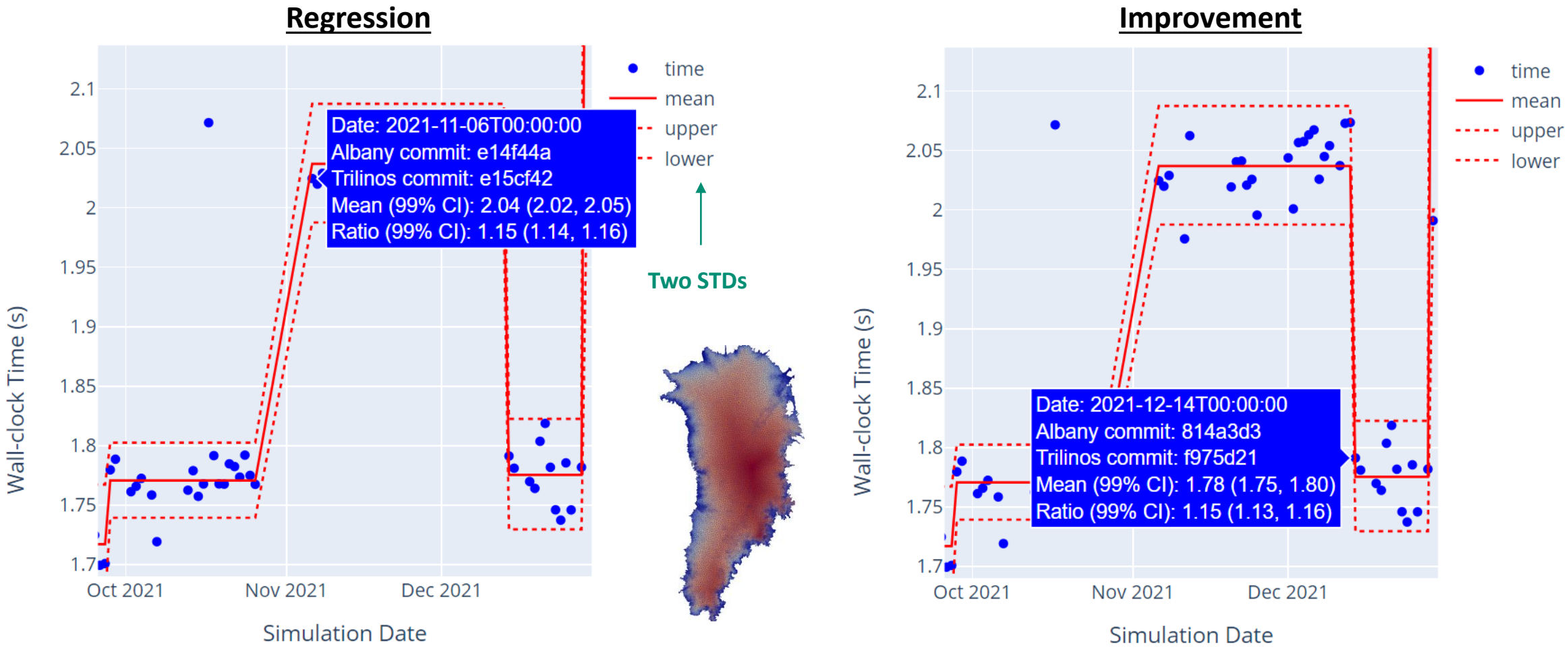




**Total Time** for a 2-to-20 km resolution Antarctica mesh, executed nightly in Albany Land Ice

**Changepoint Detection:** Kyle Shan

# Detecting performance regressions/improvements

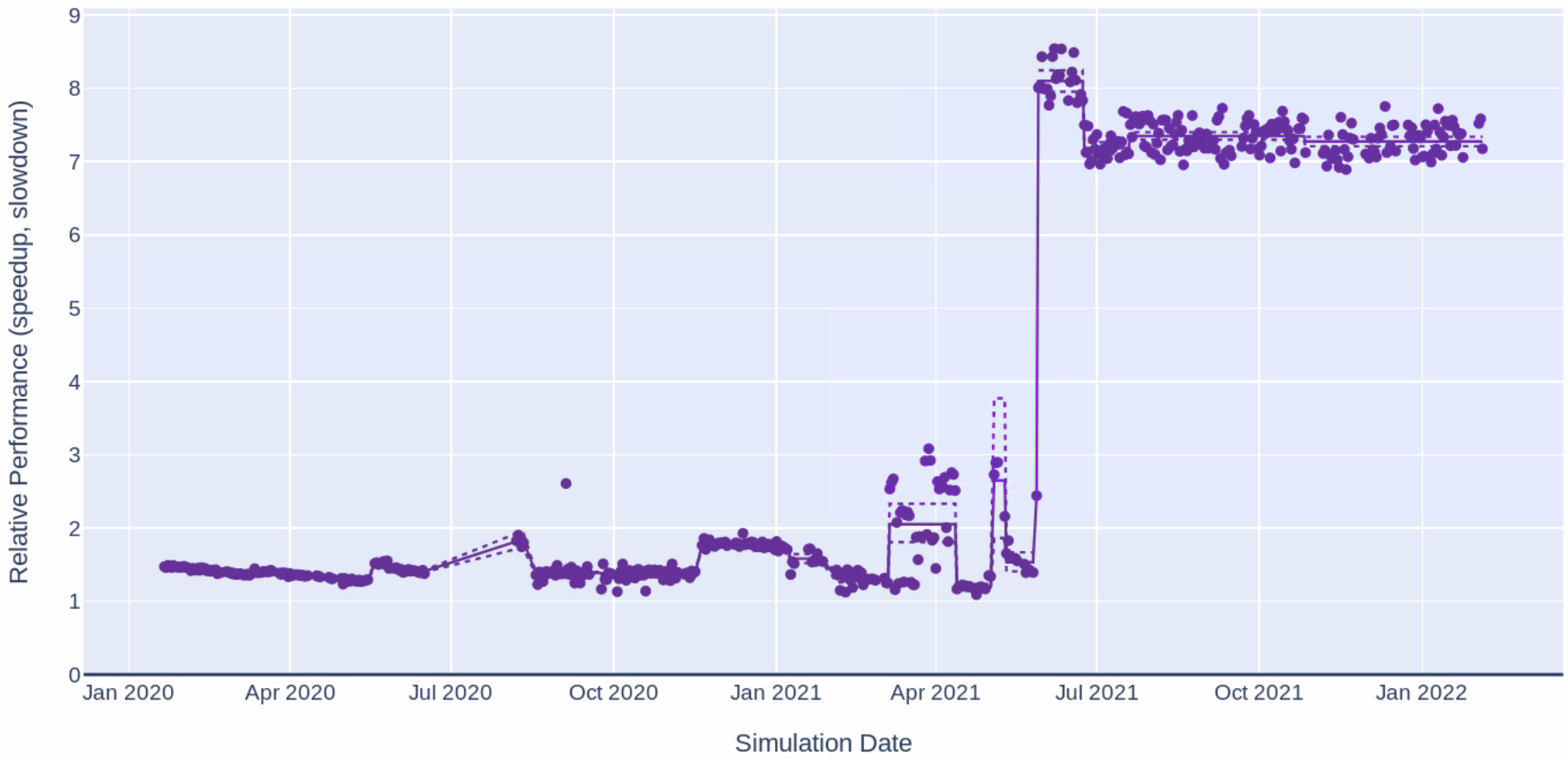
**Example: Transition to Kokkos 3.5.0 caused a performance regression but was soon fixed**



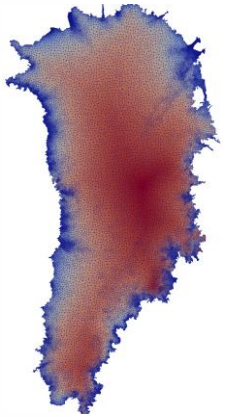

**Total Fill** time for a 1-to-7 km resolution Greenland mesh, executed nightly in Albany Land Ice

# Algorithmic performance comparisons

**Example: Memoization comparison (w. & w.o.) shows that relative performance has increased**




**99% confidence interval for the mean**



**Speedup** of **Total Fill** time from **memoization** for a 1-to-7 km resolution Greenland mesh, executed nightly in Albany Land Ice

# Conclusions

# Conclusions

- **HPC software/hardware is changing rapidly** which poses a significant challenge for open-science
- Multiple **performance portable** features exist in the **MALI** software stack to meet this challenge
- **Performance** on next generation computing architectures is a **work in progress**
  - **1.9x** speedup of V100 node over POWER9 node in total solve time
  - 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
  - **Change-point detection** adds level of confidence to performance regressions/improvements
  - **Autotuning framework** adds level of confidence to optimal parameters for given system

Watkins *et al*. “Performance portable ice-sheet modeling with MALI.” (Submitted, 2022)
https://arxiv.org/abs/2204.04321

# 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