SAND2022-7840C

Sandia National Laboratories

Performance Portable Modernizations for Albany Land-Ice

PRESENTED BY

Max Carlson, Jerry Watkins, Irina Tezaur

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

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.

Albany Land-Ice

Model and Software

Motivation

- “The top priority today is the continued progress to exascale” – DOE Office of Science HPC Initiative
- **Next Generation Architecture:** a new computing architecture that requires a very different programming model to fully utilize
- GPUs in open science are here – and they’re not going anywhere

ORNL Summit (200 PF) – 2 IBM POWER9 CPU + 6 NVIDIA V100 GPUs

ANL Aurora (2021, >1 EF) – Intel Xeon CPU + Intel Xe GPU

ORNL Frontier (2021, >1.5 EF) – 1 AMD EPYC CPU + 4 AMD Radeon Instinct GPUs

NERSC Cori (30 PF) – 2 Intel Xeon “Haswell”, 1 Intel Xeon Phi “KNL”

NERSC Perlmutter (2021) – AMD EPYC CPU-only, CPU + NVIDIA GPUs

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: $\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$
- 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: $A(T) = A_0 e^{-\frac{Q}{RT}}$

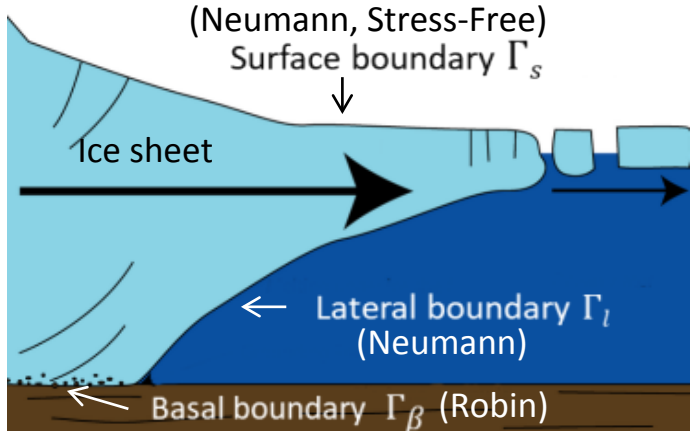
Stokes$(\boldsymbol{u}, p)$ in $\Omega \in \mathbb{R}^3$

↓

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 μ.
- Valid for both **Greenland** and **Antarctica** and used in **continental scale** simulations.

Temperature Model

- Cold ice temperature equation: $$\rho c \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) + \rho c \boldsymbol{u} \cdot \nabla T = \tau : \dot{\epsilon}$$

Because of the internal dissipation heat, there can be internal melting and the ice becomes temperate (mixture of ice and water)

It is convenient to model the temperature and porosity (water content) equations in terms of the enthalpy defined as *h*

$$h := \rho c (T - T_0) + \rho_w L \, \phi$$

	Cold ice $h < h_m$	Temperate ice $h \geq h_m$
T	$T = T_0 + \frac{1}{\rho c} h$	$T = T_m$
ϕ	0	$\frac{1}{\rho_w L}(h - h_m)$

- Steady-state Enthalpy equation reads: $$\nabla \cdot \boldsymbol{q}(h) + \boldsymbol{u} \cdot \nabla h = \tau : \dot{\epsilon}$$

Total enthalpy flux

$$\boldsymbol{q} = -\frac{k}{\rho c} \nabla h \quad \text{If cold}$$

$$\boldsymbol{q} = -\frac{k}{\rho c} \nabla h_m + \rho_w L \boldsymbol{j}(h) \quad \text{If temperate}$$

Gravity driven water flux

$$\boldsymbol{j}(h) = \frac{1}{\eta_w} k_0 \left(\frac{h - h_m}{\rho c} \right)^{\gamma} (\rho_w - \rho) \boldsymbol{g}$$

Stefan's condition at the bed

$$m = G + \tau_b \cdot \boldsymbol{u} - k \nabla T \cdot \boldsymbol{n}$$

At surface elevation:

$$T = T_{\text{air}}$$

- *A. Aschwaden et al., An Enthalpy formulation for glaciers and ice sheets, Journal of Glaciology, 2012*
- *I. Hewitt, and C. Schoof: Models for polythermal ice sheets and glaciers, The Cryosphere, 2016*
- *C. Schoof and I. J. Hewitt, A model for polythermal ice incorporating gravity-driven moisture transport, Journal of Fluid Mechanics, 2016*

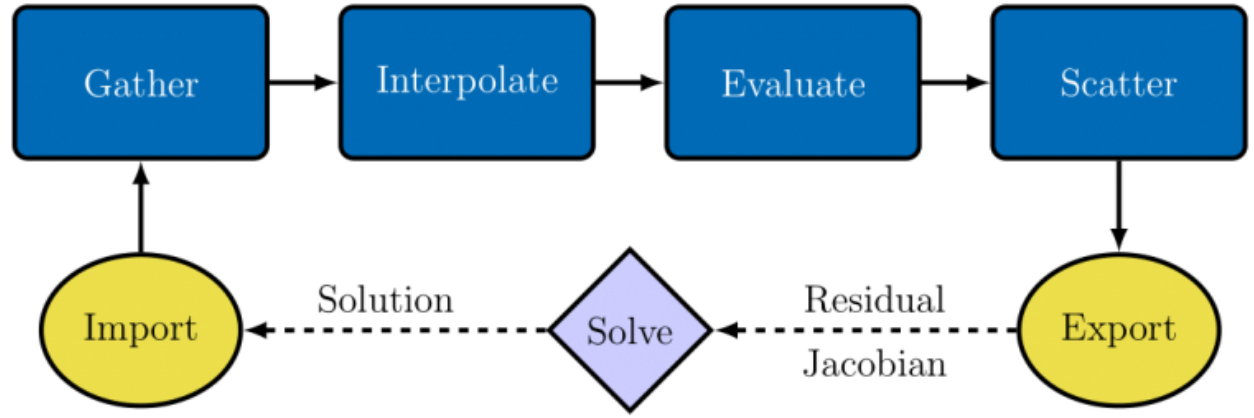
Albany

- Albany is an object-oriented, parallel, C++ code for discretizing and solving PDEs

- Finite element discretization on unstructured grids

- Utilizes a number of libraries from the Trilinos project

- Albany constructs a system and then hands it to Trilinos to solve



Performance Portable Modernization

Kokkos Refactor

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 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.

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

Volume Refactor

- Intermediate data structures converted to Kokkos views for accessibility on device
- Evaluators are parallelized across number of cells in a workset
- Volume evaluators are either completely data parallel, or at least data parallel with respect to a cell
- Readability is preserved for ease of future implementation

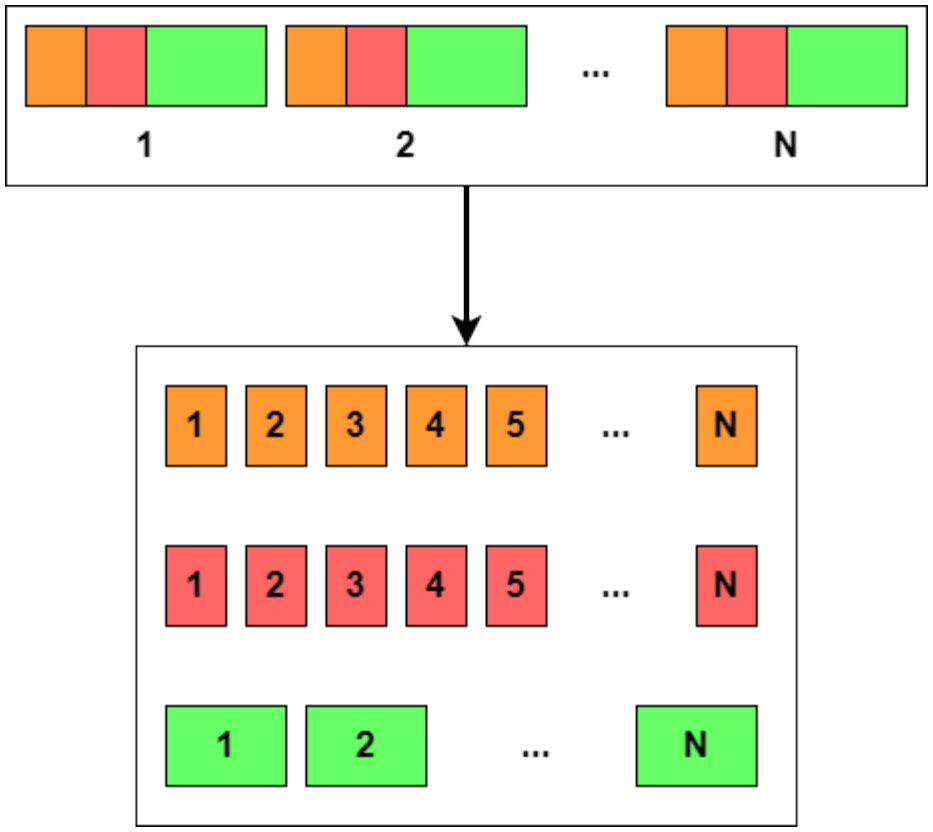
```
void Dissipation<EvalT,Traits>::
evaluateFields(typename Traits::EvalData workset) {
    for (std::size_t cell = 0; cell < workset.numCells; ++cell)
        for (std::size_t qp = 0; qp < numQPs; ++qp)
            diss(cell,qp) = 1.0/scyr * 4.0 * mu(cell,qp) * epsilonSq(cell,qp);
}
```

```
KOKKOS_INLINE_FUNCTION
void Dissipation<EvalT,Traits>::
operator() (const int &cell) const {
  for (int qp = 0; qp < numQPs; ++qp) {
    diss(cell,qp) = 1.0/scyr * 4.0 * mu(cell,qp) * epsilonSq(cell,qp);
  }
}

void Dissipation<EvalT,Traits>::
evaluateFields(typename Traits::EvalData workset) {
  Kokkos::parallel_for(Dissipation_Policy(0, workset.numCells), *this);
}
```

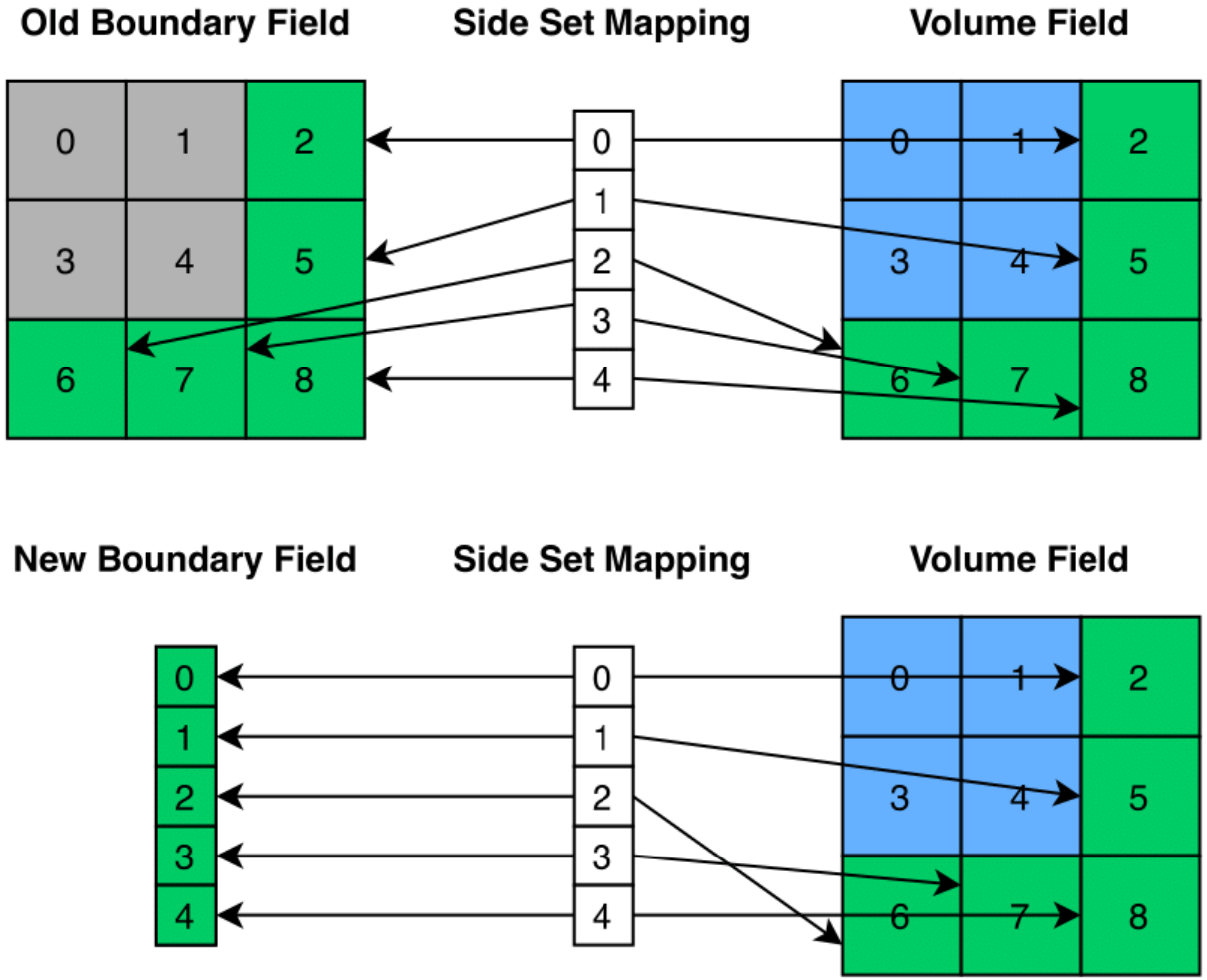
Boundary Refactor

- Boundary index information originally stored as array of structures
- Replaced with structure of Kokkos views for device access
- This structure enables coalesced access of boundary information
- Boundary fields are not coalesced



Boundary Layout Optimization

- Boundary fields had the same layout as volume fields and were accessed using the side set mapping
- Matching the boundary field layout to side set mapping layout results in coalesced access for both
- Access to the volume field is not coalesced but happens only once instead of many times



Enthalpy Run-time Improvements

The following performance results are from solving the enthalpy problem on a variable resolution (1km to 10km) mesh of the Greenland ice sheet using

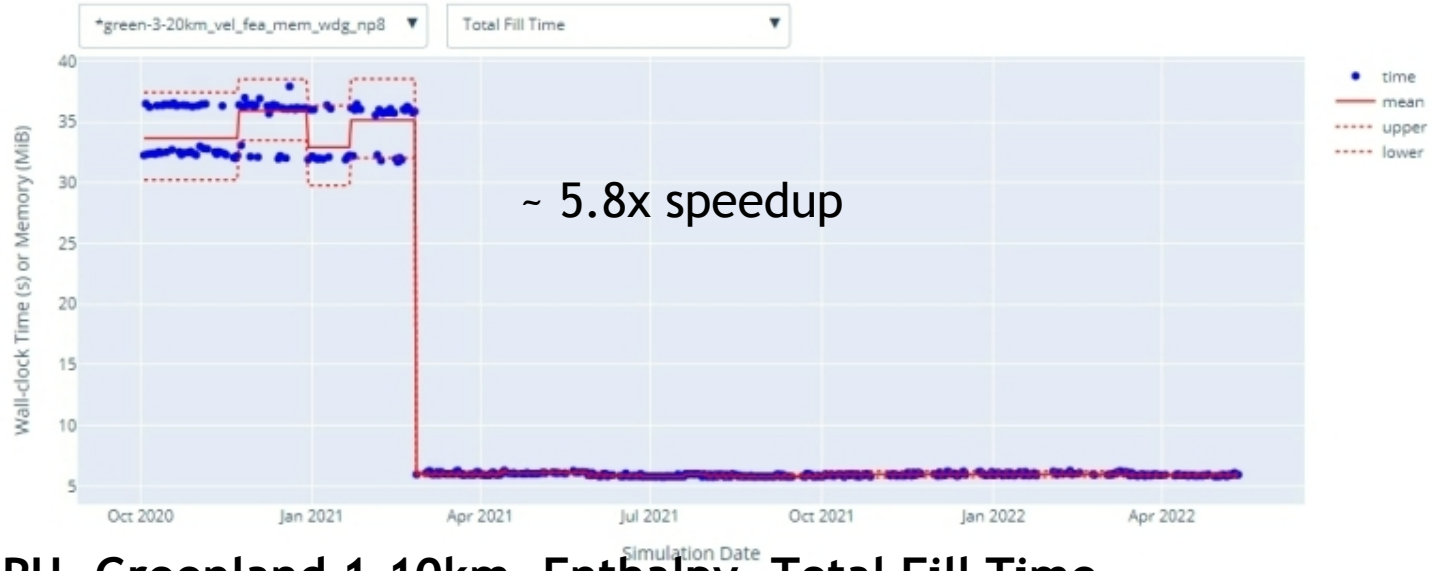
- 4x Nvidia V100 GPUs per node
- Dual-socket POWER9 CPU with 40 cores per node (20 cores per socket)
- Dual-socket Haswell CPU with 32 cores per node (16 cores per socket)
- Single-socket Knight's Landing CPU with 64 cores per node

	PWR9-Serial-original	V100-CUDA-volume	Speedup
Total Fill Time	16.75 seconds	3.8 seconds	4.4x
	PWR9-Serial-original	**V100-CUDA-boundary**	**Speedup**
Total Fill Time	16.75 seconds	2.79 seconds	6x
	KNL-OpenMP-original	**KNL-OpenMP-volume**	**Speedup**
Total Fill Time	28.12 seconds	20.14 seconds	1.4x
	KNL-Serial-original	**KNL-OpenMP-volume**	**Speedup**
Total Fill Time	32.4 seconds	20.14 seconds	1.6x

StokesFO Run-time Improvements

- StokesFO finite element assembly was refactored using the same strategy
- StokesFO and Enthalpy have common evaluators so this refactor also saw a modest run-time improvement for Enthalpy
- With this refactor, all finite element assembly is done on device
- All evaluators now running on device

GPU, Greenland 3-20km, StokesFO, Total Fill Time

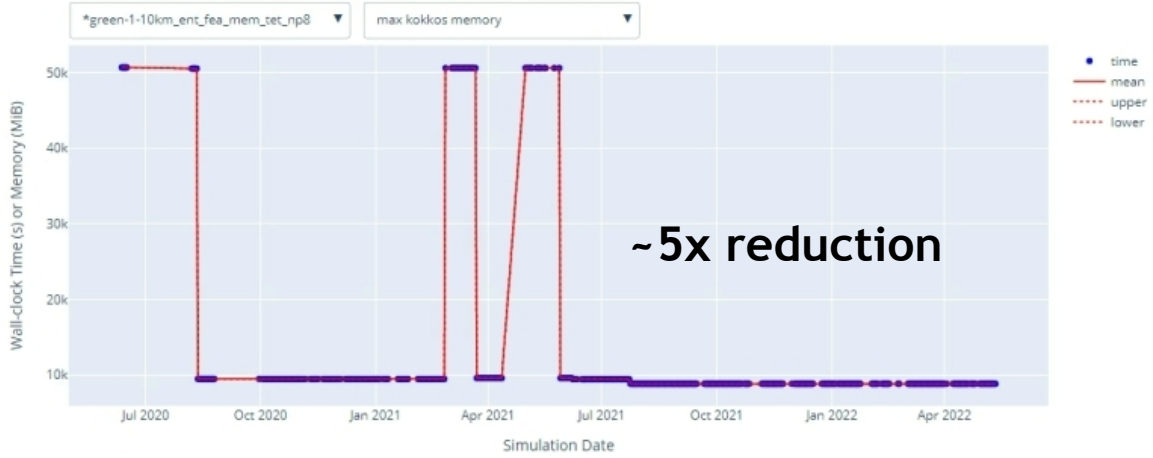


GPU, Greenland 1-10km, Enthalpy, Total Fill Time

Memory Improvements

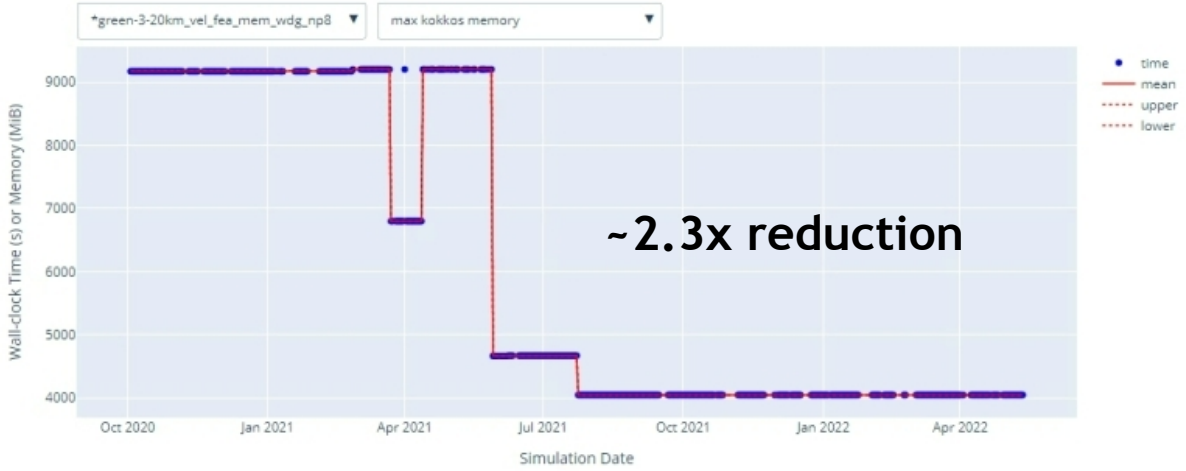
CPU+MPI, Enthalpy, Max Kokkos Memory

GPU, Enthalpy, Max Kokkos Memory



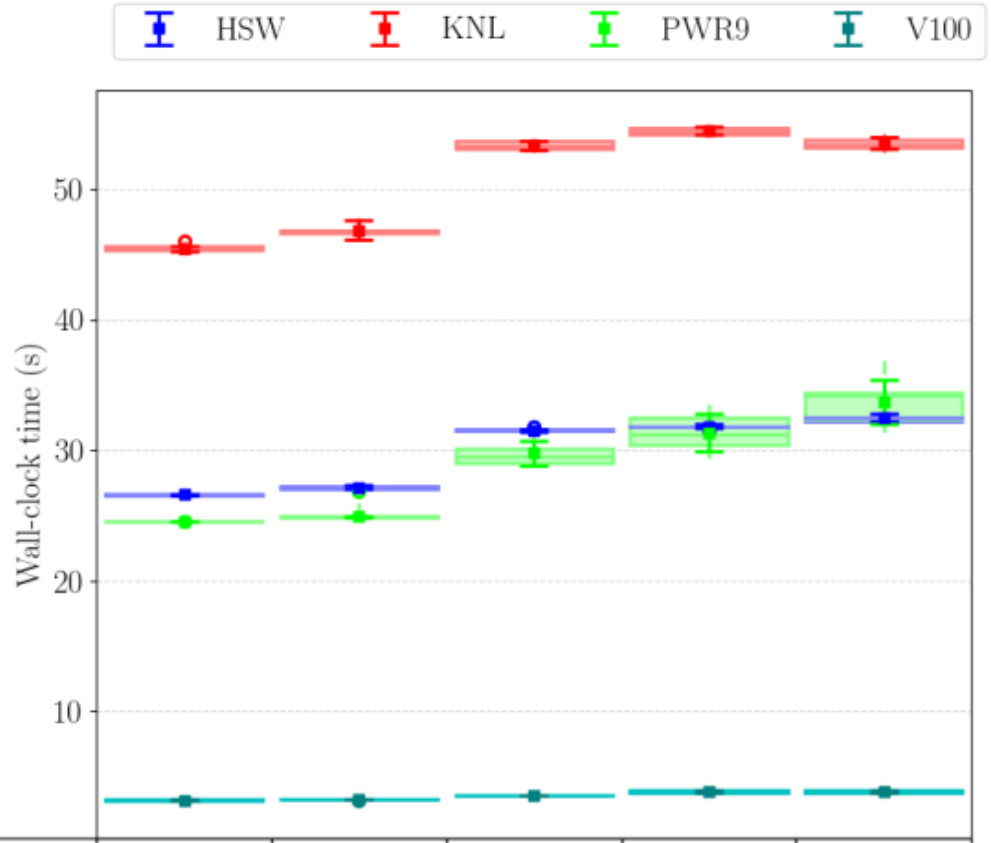
CPU+MPI, StokesFO, Max Kokkos Memory

GPU, StokesFO, Max Kokkos Memory



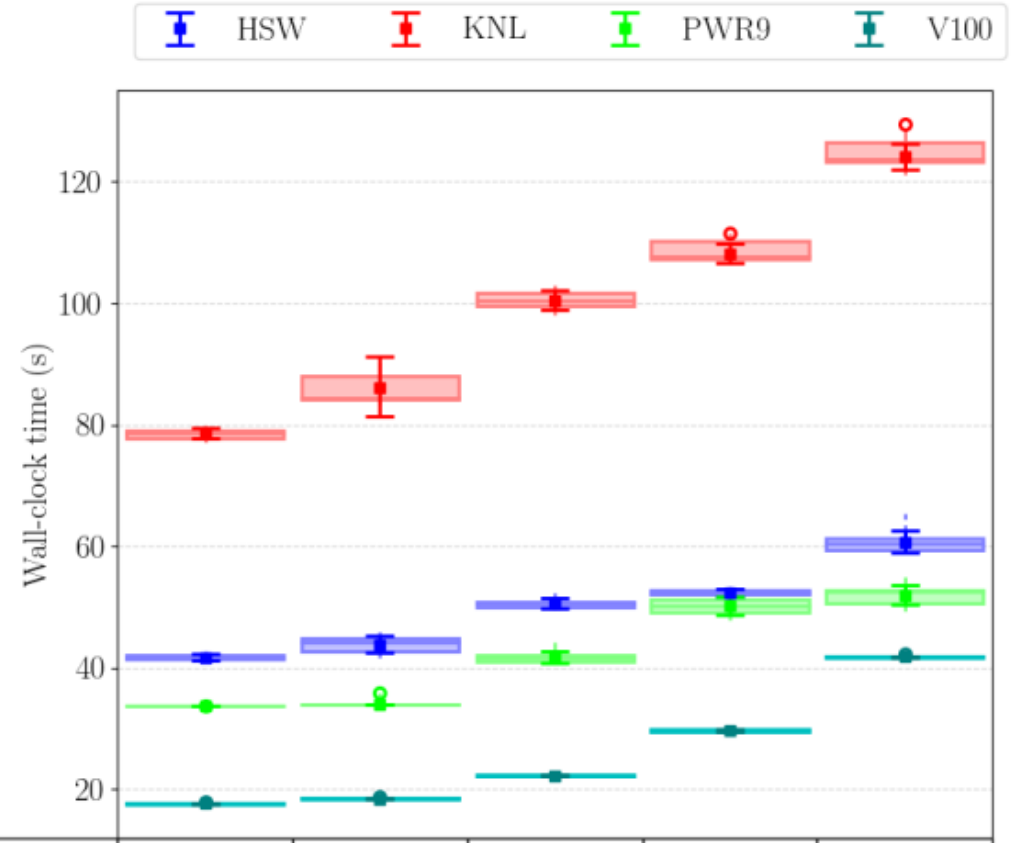
Weak Scalability

Total Fill Time



Resolution	16km	8km	4km	2km	1km
# Nodes	1	4	16	64	256
V100 Speedup	7.63	7.61	8.32	8.00	8.65
99% CI	(7.53, 7.73)	(7.55, 7.67)	(8.08, 8.58)	(7.64, 8.38)	(8.22, 9.10)

Total Time



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)

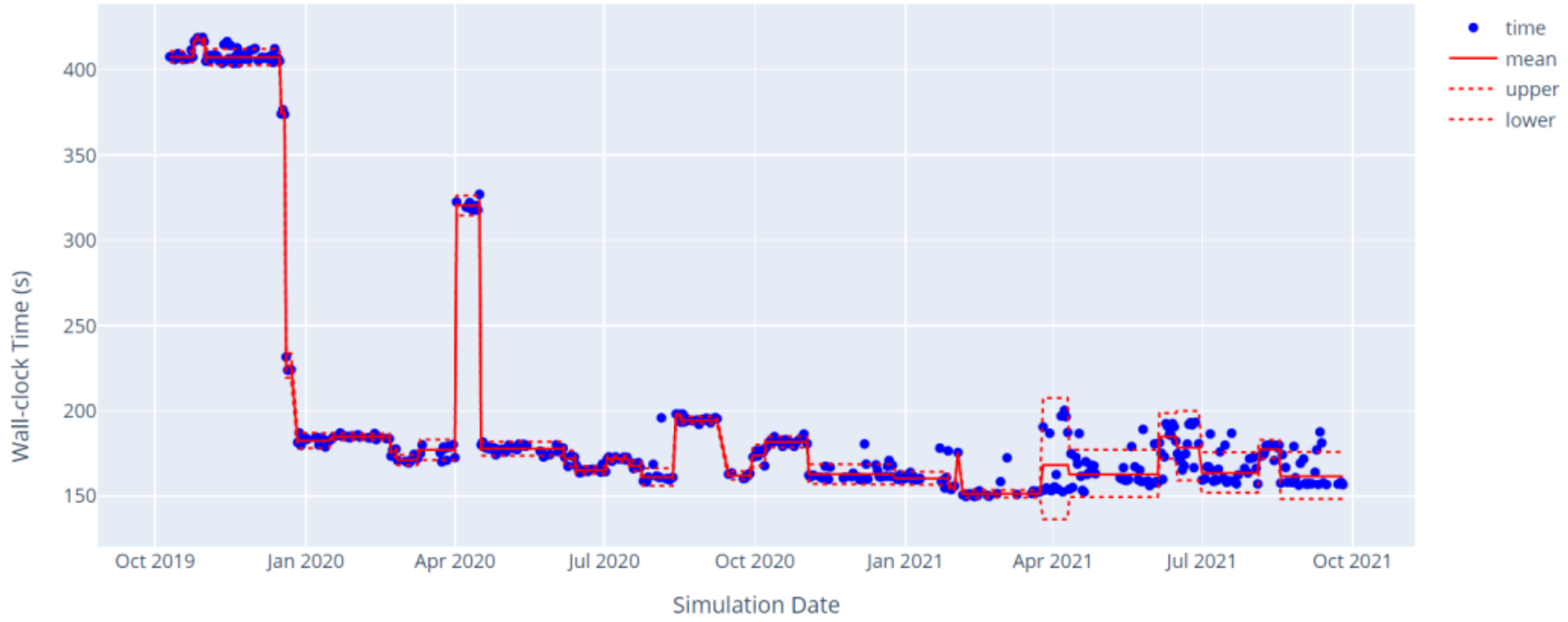
Performance portable ice-sheet modeling with MALI; Jerry Watkins, Max Carlson, et all; **Submitted to IJHPCA, 2022**

Towards Exascale

Automatic Performance Monitoring

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

- Nightly performance testing of run-time and memory continues to be critical for monitoring changes in performance
- **Changepoint detection** for automatic detection of unexpected performance degradation

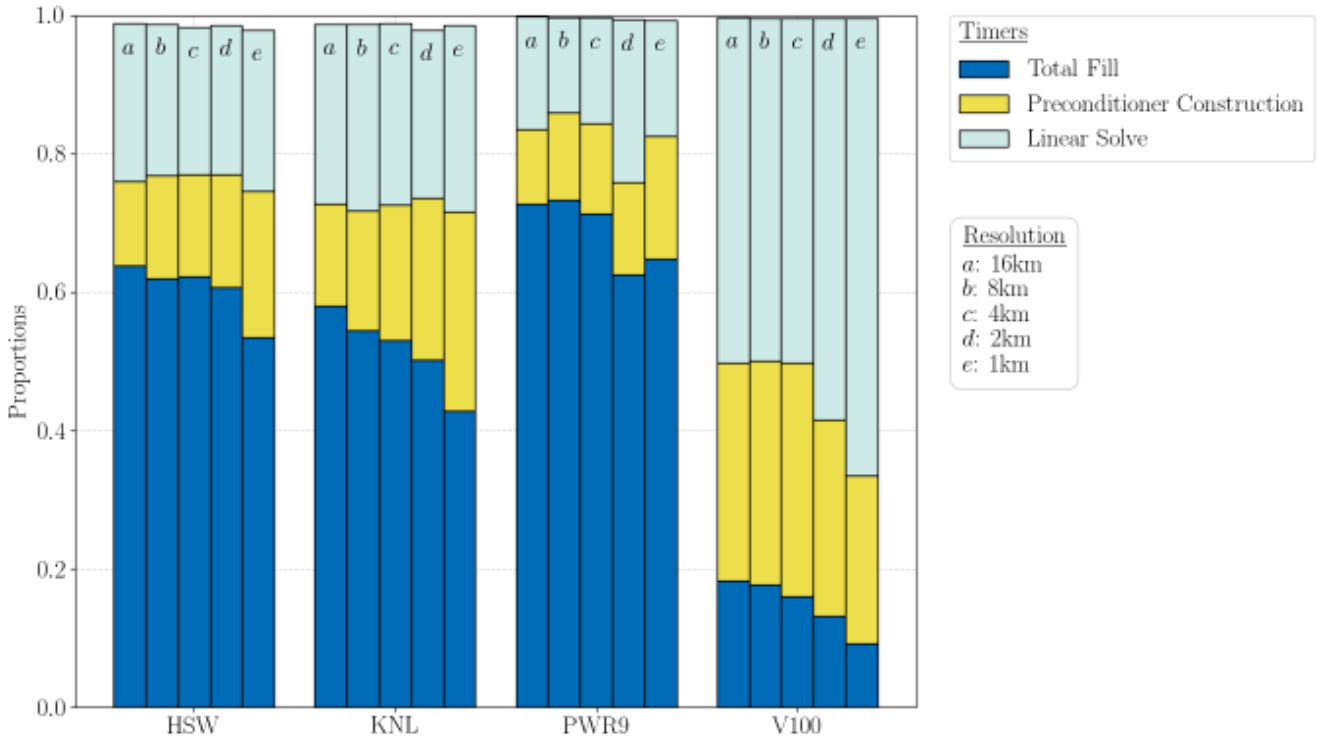


Changepoint detection: Kyle Shan

Areas to Improve

- Memory reductions increased number of degrees of freedom assigned to each GPU
- Increased saturation of device resulted in more efficient computation
- However, scalability is limited on GPU architecture
- GPU-based linear solvers are sensitive to solver parameters

	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)



Automatic Performance Tuning

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

Improving Automatic Performance Tuning

- Tuning can be directed using Bayesian optimization techniques
- Parameters found by random search can be further improved in an directed manner
- We use Albany's python interface PyAlbany as a driver to GPTune autotuning framework
- Currently running experiments to tune Albany Land-Ice on Perlmutter A100 GPUs

Performance portable ice-sheet modeling with MALI

Performance portable ice-sheet modeling with MALI

Jerry Watkins[1,*], Max Carlson[1], Kyle Shan[2,**], Irina Tezaur[1], Mauro Perego[3], Luca Bertagna[3], Carolyn Kao[4,**], Matthew J. Hoffman[5], and Stephen F. Price[5]

[1]Sandia National Laboratories, Quantitative Modeling & Analysis Department, Livermore, CA, USA.
[2]Micron Technology, Boise, ID, USA.
[3]Sandia National Laboratories, Center for Computing Research, Albuquerque, NM, USA.
[4]TSMC, Hsinchu, Taiwan.
[5]Los Alamos National Laboratory, Fluid Dynamics and Solid Mechanics Group, Los Alamos, NM, USA.
[*]Corresponding author; email: jwatkin@sandia.gov.
[**]All work completed as a student at Stanford University.

April 8, 2022

Abstract

High resolution simulations of polar ice-sheets play a crucial role in the ongoing effort to develop more accurate and reliable Earth-system models for probabilistic sea-level projections. These simulations often require a massive amount of memory and computation from large supercomputing clusters to provide sufficient accuracy and resolution. The latest exascale machines poised to come online contain a diverse set of computing architectures. In an effort to avoid architecture specific programming and maintain productivity across platforms, the ice-sheet modeling code known as MALI uses high level abstractions to integrate Trilinos libraries and the Kokkos programming model for performance portable code across a variety of different architectures. In this paper, we analyze the performance portable features of MALI via a performance analysis on current CPU-based and GPU-based supercomputers. The analysis highlights performance portable improvements made in finite element assembly and multigrid preconditioning within MALI with speedups between 1.26–1.82x across CPU and GPU architectures but also identifies the need to further improve performance in software coupling and preconditioning on GPUs. We also perform a weak scalability study and show that simulations on GPU-based machines perform 1.24–1.92x faster when utilizing the GPUs. The best performance is found in finite element assembly which achieved a speedup of up to 8.65x and a weak scaling efficiency of 82.9% with GPUs. We additionally describe an automated performance testing framework developed for this code base using a changepoint detection method. The framework is used to make actionable decisions about performance within MALI. We provide several concrete examples of scenarios in which the framework has identified performance regressions, improvements, and algorithm differences over the course of two years of development.

Available at: https://arxiv.org/abs/2204.04321

Performance portable ice-sheet modeling with MALI; Jerry Watkins, Max Carlson, et all; **Submitted to IJHPCA, 2022;**

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).

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