Sandia National Laboratories

Exceptional service in the national interest

Achieving and maintaining performance and performance portability within the Albany multi-physics code: perspectives & tools

Irina Tezaur[1], Max Carlson[1], Jerry Watkins[1], Mauro Perego[1], Kyle Shan[2], Carolyn Kao[3], Kim Liegeois[1]

[1] Sandia National Laboratories, Livermore, CA, USA.
[2] Micron Technology, Boise, ID, USA.
[3] TSMC, Hsinchu City, Taiwan.

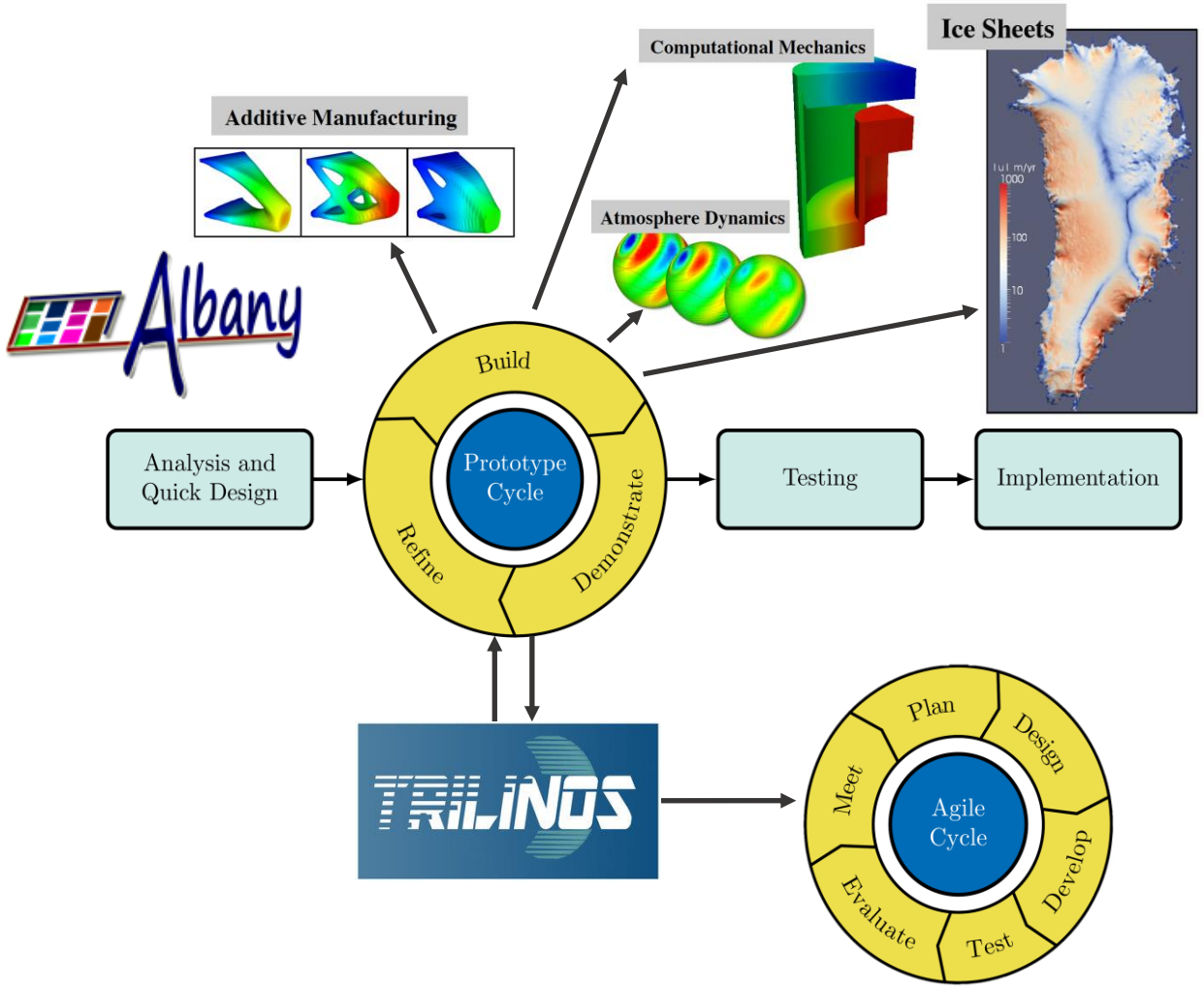
SIAM CS&E 2023, Amsterdam, Netherlands.

February 26 – March 3, 2023

SAND2023-12634C

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**



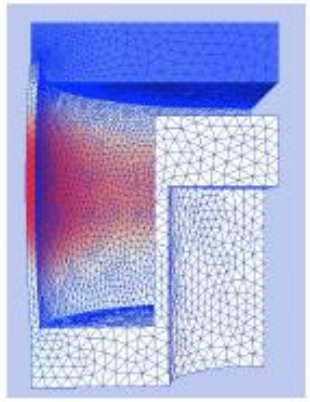
Distinguishing features of *Albany*:

- **Funded** entirely **by applications** residing within
- Both a **"sand-box"** for prototyping **new approaches** and a **production** code
- **Algorithms/software** are developed & matured directly on **applications**
- Applications are "**born**" scalable, fast, robust, performance-portable, and…
- Equipped with **embedded advanced analysis capabilities**: sensitivities, bifurcation analysis, adjoint-based inversion, …

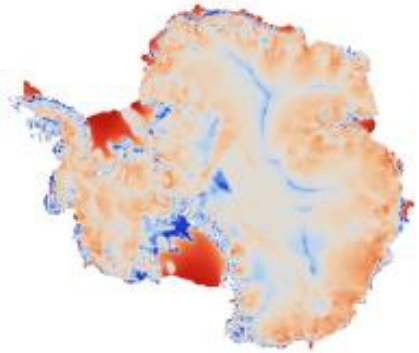
[1] https://github.com/sandialabs/Albany
[2] https://github.com/trilinos/Trilinos

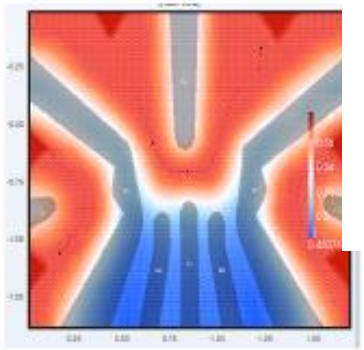
History of Albany

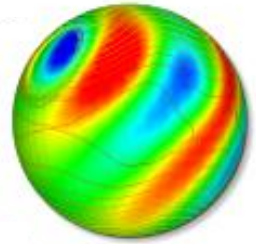
Since its creation ~2009, Albany has housed many **diverse algorithmic projects** & **applications***:

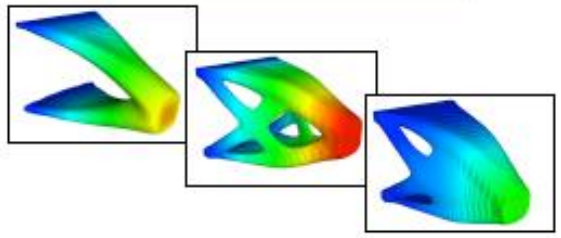


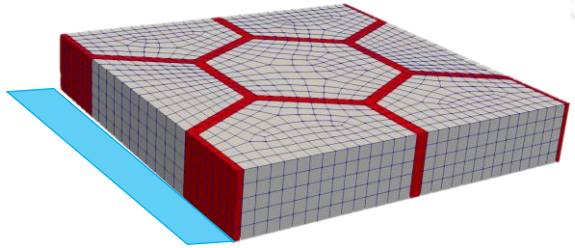
- Demo PDEs
- Quantum Devices (QCAD)
- Ice Sheets (Albany Land-Ice)
- Mechanics (LCM)
- Atmosphere Dynamics (Aeras)
- Particle-continuum coupling (Peridigm-LCM)
- Additive Manufacturing Design (ATO)
- Additive Manufacturing Processing (AMP)
- Arctic Coastal Erosion (ACE)
- Coupled Geomechanics (Albotran)











Andy Salinger
"Father" of Albany

Albany

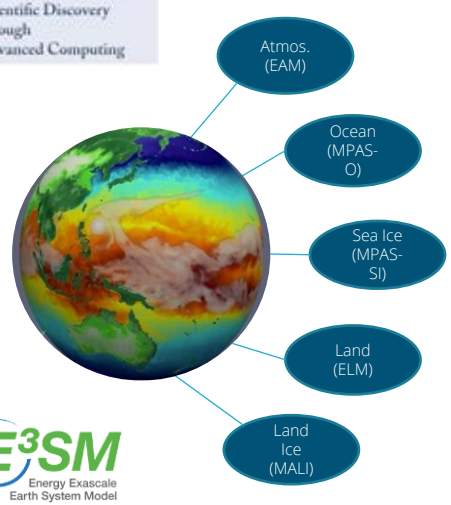
* **I. Tezaur** et al. "Albany: a Trilinos-based multi-physics partial differential equation research tool created using the Agile Components code development strategy," *SIAM CS&E 2019 (featured MS: "Multiphysics: Extensible, Composable Algorithms and Software")*, Spokane, WA.

Albany and its Requirements Today

SciDAC Scientific Discovery through Advanced Computing

Primary customer/funder: U.S. DOE SciDAC-funded project land-ice modeling project, **FAnSSIE (Framework for Antarctic System Science in E3SM, FY23-FY27)**

- Albany houses the velocity solver of **MALI (MPAS-Albany Land Ice)**, the land-ice component of the U.S. DOE's Energy Exascale Earth System Model (E3SM)



Required capabilities in Albany for science using MALI:

- Easy way to add **new physics/PDEs**
- **Performance portability** to advanced heterogeneous platforms
- Fast, scalable and robust **linear solves** across different architectures
- **PDE-constrained optimization** capabilities for ice sheet inversion
- **Adaptive mesh refinement (AMR)** of extruded tetrahedral meshes

Requirements for software quality:

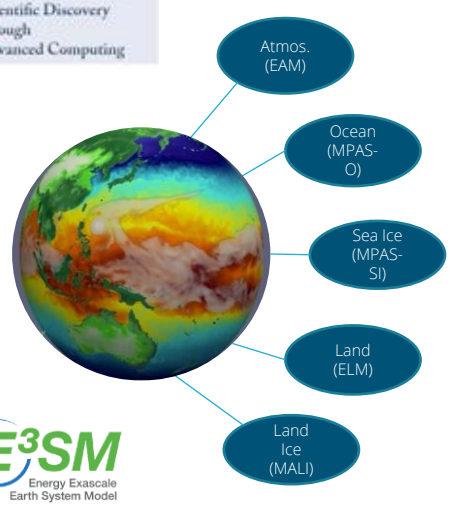
- **Pruning** of unfunded applications/unsupported code
- **Version control**
- **Automated parameter tuning** for various architectures
- Automated **nightly regression** and **performance testing** using **VOTD** Albany & Trilinos

See **MS284** and **MS318** on Thur. Mar. 2 for more on MALI and **ice-sheet modeling**.

Albany and its Requirements Today

Primary customer/funder: U.S. DOE SciDAC-funded project land-ice modeling project, **FAnSSIE (Framework for Antarctic System Science in E3SM, FY23-FY27)**

- Albany houses the velocity solver of **MALI (MPAS-Albany Land Ice)**, the land-ice component of the U.S. DOE's Energy Exascale Earth System Model (E3SM)



Required capabilities in Albany for science using MALI:

- Easy way to add **new physics/PDEs**
- **Performance portability** to advanced heterogeneous platforms
- Fast, scalable and robust **linear solves** across different architectures
- **PDE-constrained optimization** capabilities for ice sheet inversion
- **Adaptive mesh refinement (AMR)** of extruded tetrahedral meshes

This talk: highlight **capabilities** for **achieving/maintaining performance** & **performance portability** in Albany, with focus on **pros/cons** and **lessons learned.**

Requirements for software quality:

- **Pruning** of unfunded applications/unsupported code
- **Version control**
- **Automated parameter tuning** for various architectures
- Automated **nightly regression** and **performance testing** using **VOTD** Albany & Trilinos

Outline

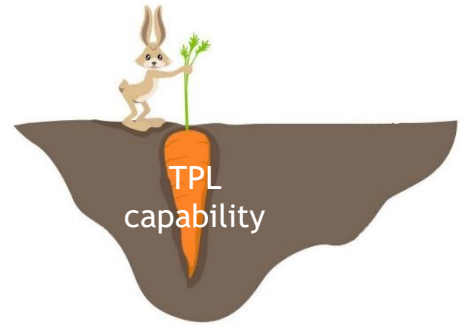
- Albany capabilities & supporting tools
  - Components effort
  - Albany under the hood
  - Kokkos for performance portability
  - Phalanx for template-based evaluators and DAG-based finite element assembly
  - Sacado for automatic differentiation (AD)
  - Some performance & performance portability results
- Maintaining software quality of Albany
  - Code pruning
  - PyAlbany: a Python interface to Albany
  - Nightly regression testing
  - Automated performance testing
  - Automated performance tuning
- Summary & perspectives

Outline

- **Albany capabilities & supporting tools**
  - Components effort
  - Albany under the hood
  - Kokkos for performance portability
  - Phalanx for template-based evaluators and DAG-based finite element assembly
  - Sacado for automatic differentiation (AD)
  - Some performance & performance portability results
- Maintaining software quality of Albany
  - Code pruning
  - PyAlbany: a Python interface to Albany
  - Nightly regression testing
  - Automated performance testing
  - Automated performance tuning
- Summary & perspectives

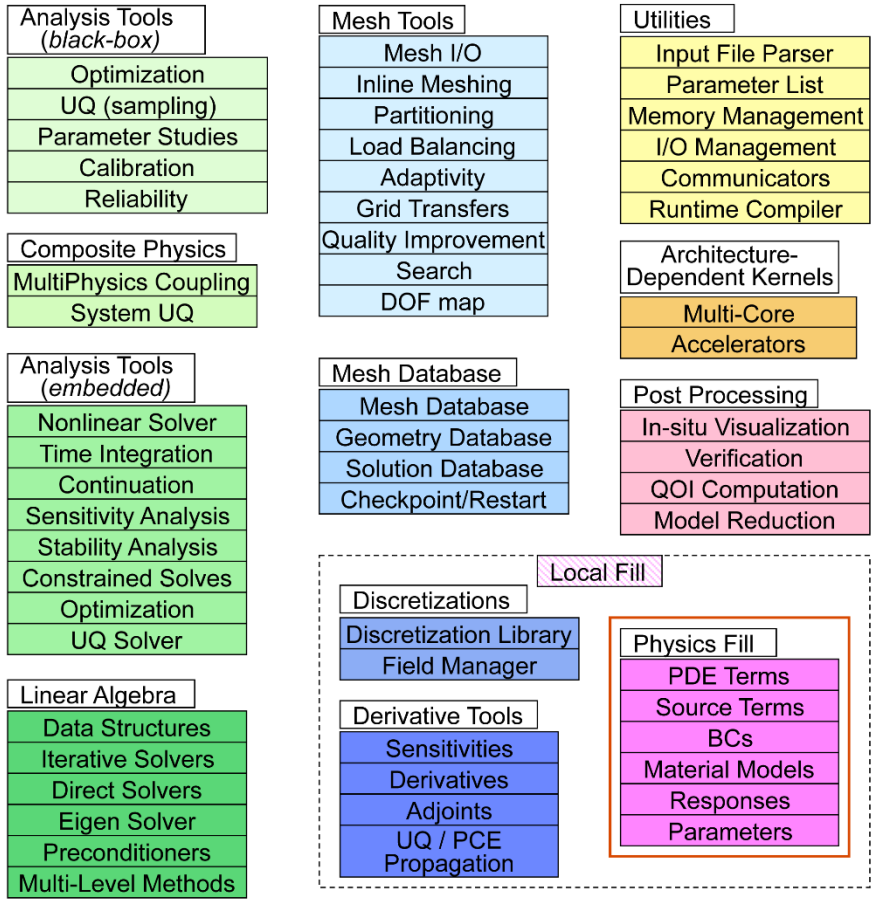
The Components in Albany

Components in Albany = cutting-edge technology from Trilinos, SierraToolKit, DAKOTA, FASTMath, Kitware, etc.



Many components are **Trilinos* packages:**

- 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…*



Trilinos 40+ packages; 120+ libraries

Albany developers work with **Version of the Day (VOTD) Trilinos**, which has pros and cons.

Pro: code can inherit unexpected enhancements "for free"

Con: code might also inherit bugs/regressions which can be frustrating to track down and get fixed to maintain clean dashboard

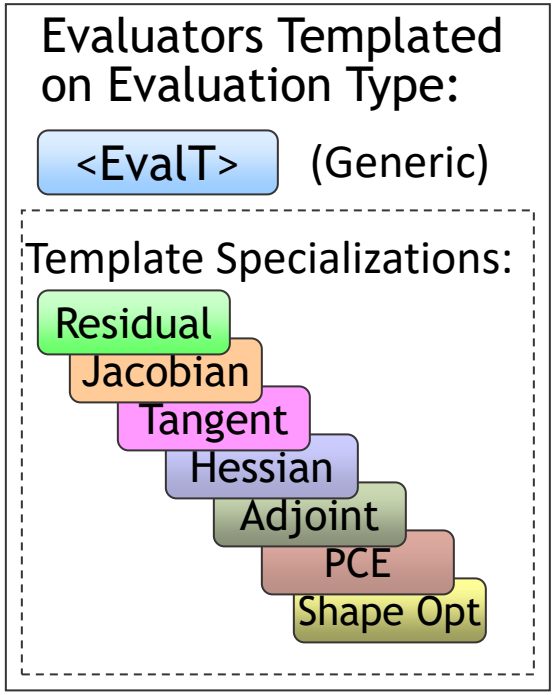
* https://github.com/trilinos/Trilinos

Albany Under the Hood

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

Four Key Ingredients:

1. MPI+X programming model for performance portability
   - Handled by ***Kokkos*** package
2. Template-based generic programming (TBGP)
   - E.g., ***Phalanx*** evaluators templated on evaluation type (right)
3. Graph-based finite element assembly (FEA)
   - ➢ Handled by ***Phalanx*** package
4. Templated-based automatic differentiation
   - ➢ Handled by ***Sacado*** package



For **more details on these ingredients**, please see talk by **Jerry Watkins** (MS284: Thurs. Mar. 2).

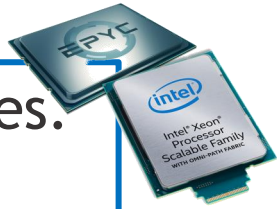
Albany Supporting Tools: Kokkos – Performance Portability

- **Kokkos**[1] is a C++ library that provides **performance portability** across multiple **shared memory** computing architectures via **MPI+X programming model**
  - ***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**

Pros:
- Same code runs on diff. platforms
- Code is "future-proof"
- Forces you to write better code

Cons: better performance may be possible with architecture-specific optimizations

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)

Pros:

- Easy to implement new physics

```
template<typename EvalT, typename Traits>

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)

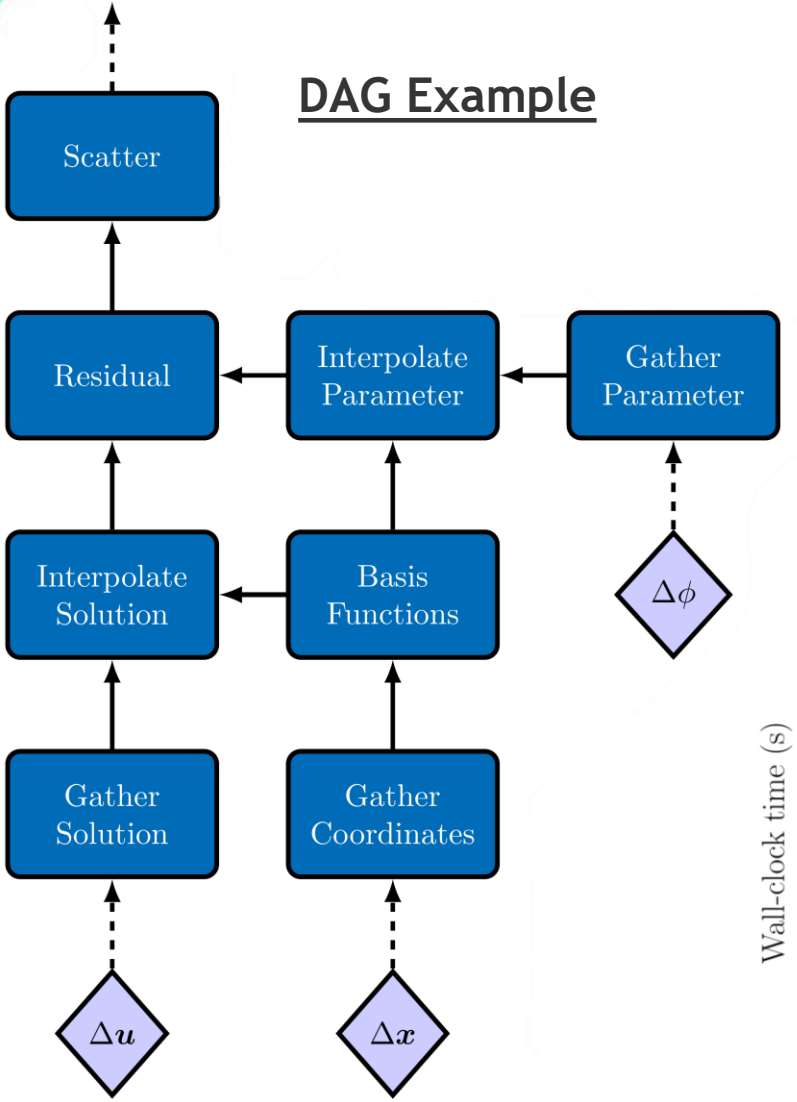
Pros:

- Easy to implement new physics
- Easy to Kokkos-ize

```
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

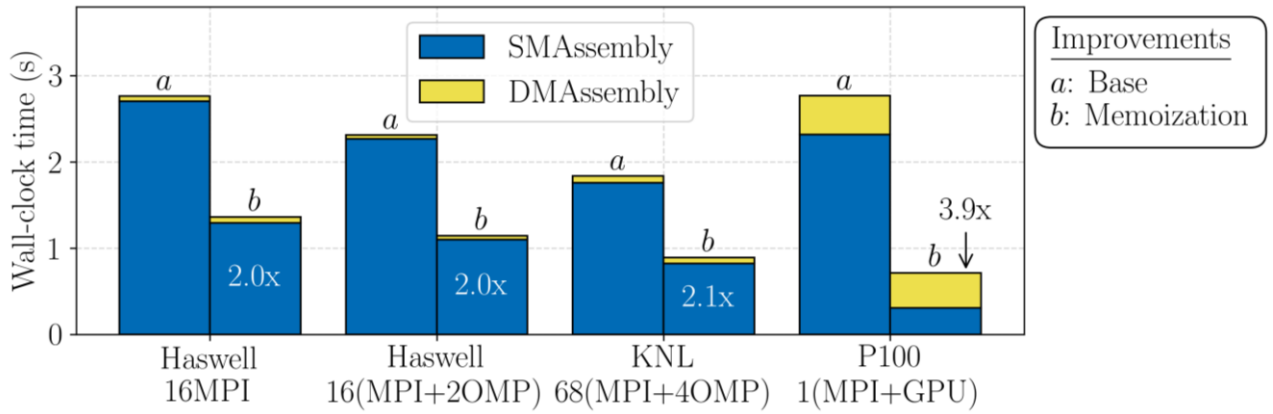


Pros:

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

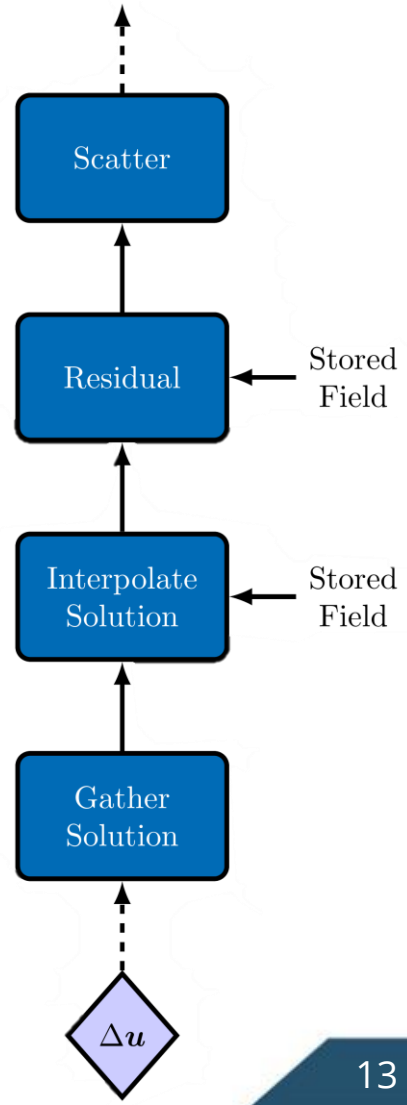
Cons: Performance loss through fragmentation

- **Mitigation:** introduction of memoization



Single CPU socket or GPU

DAG Example (memoization)



Albany Supporting Tools: Sacado – Automatic Differentiation (AD)

Pros:

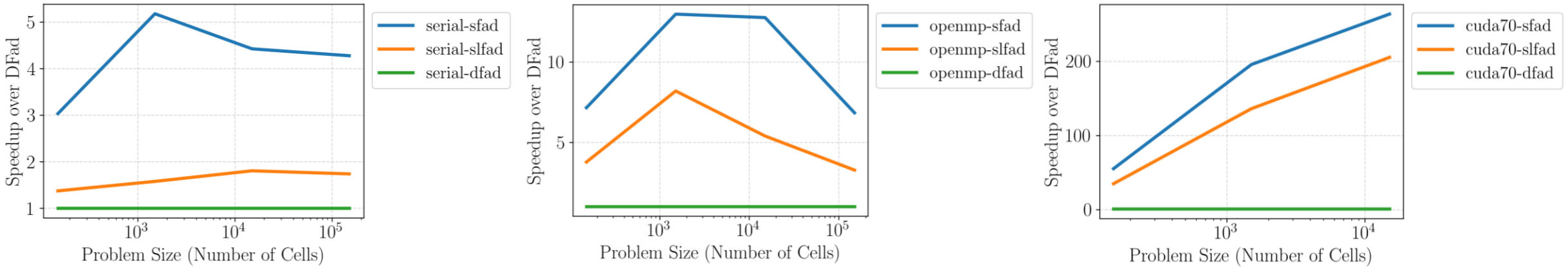
- AD provides **exact** derivatives — no Jacobian derivation or hand-coding required
- Allows for **advanced analysis** capabilities – easily construct any derivative, Hessian
  - ➢ Examples: optimization, sensitivity analysis

Con: AD has some overhead

- **Mitigation:** specify Sacado **data types** for deriv. 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

Significant speedups possible when deriv. 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

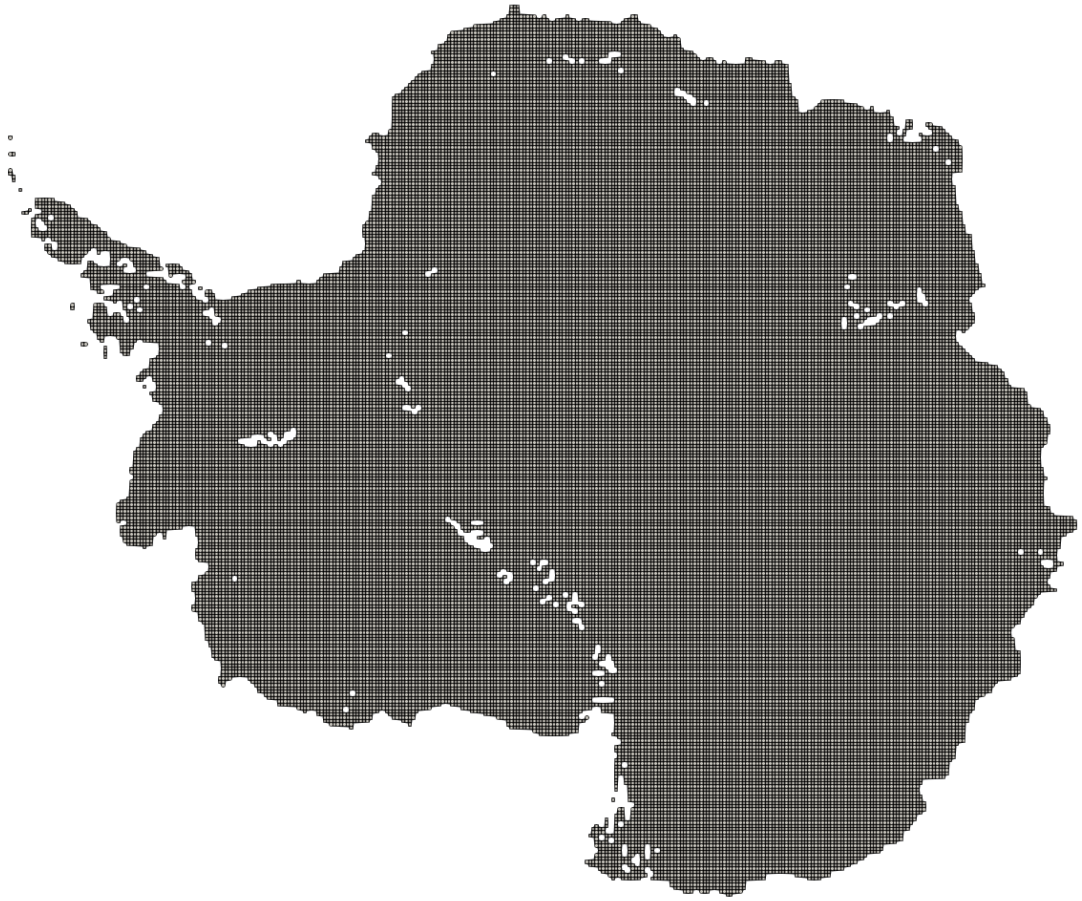
Performance Portability Demonstration: Antarctica Weak Scalability Study

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 solve in ALI
- Structured hexahedral element mesh
- 16 to 1km structured Antarctica meshes, 20 layers
- Scaled up from 1 to 256 compute nodes



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

Benchmark used to assess **performance** & **performance portability**.

Performance on Cori and Summit

Setup:

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

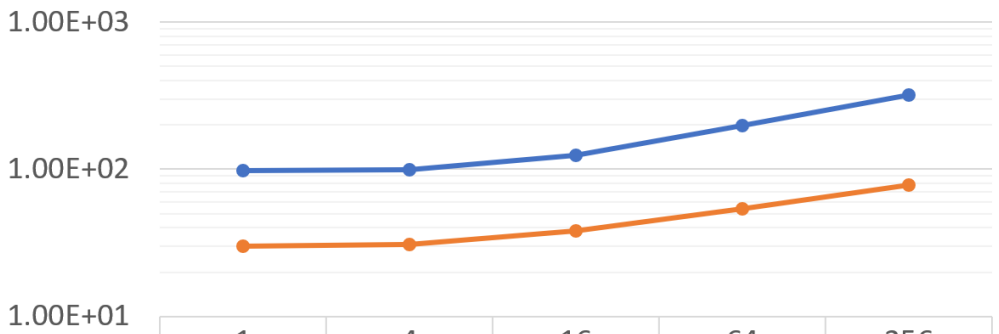
Results:

- 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

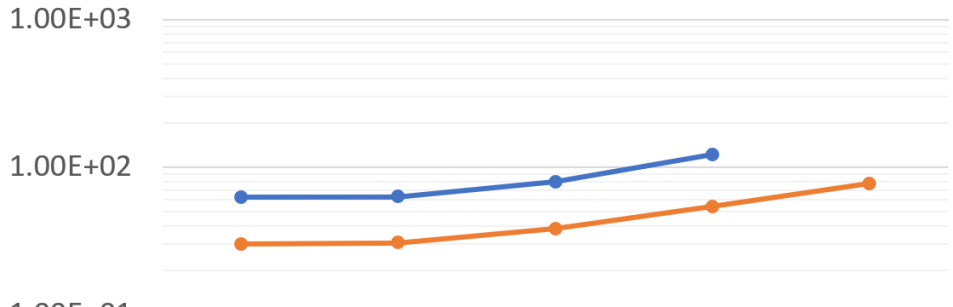
Speedup achieved over MPI-only simulations **without architecture specific tuning!**

For **additional results/analysis**, please see talks by **Jerry Watkins** (MS284: Thurs. Mar. 2) and **Mauro Perego** (MS72: Tues. Feb. 28).

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 Highlights

*From **architecture-agnostic** improvements to Albany FEA.*

Major improvements to finite element assembly time:

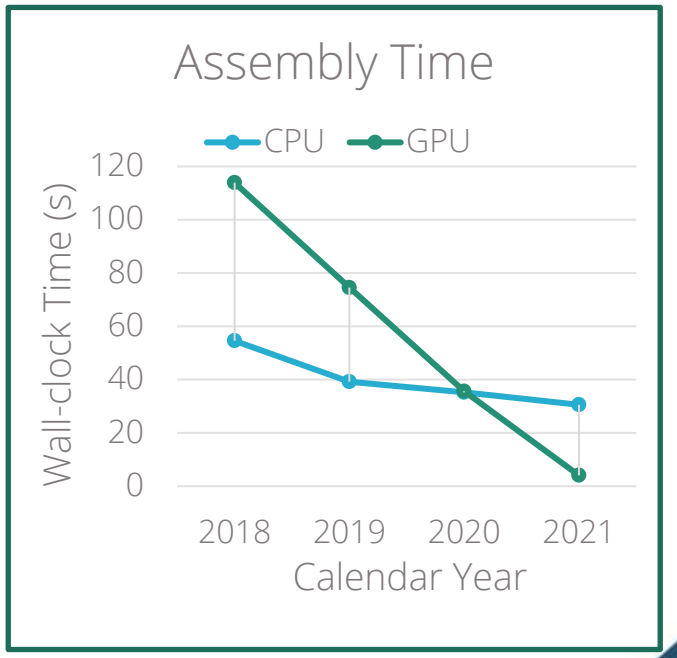
- Memoization to avoid unnecessary data movement and computation
- Boundary condition refactor to reduce memory footprint and data movement
- Tpetra::FECrsMatrix refactor to reduce memory footprint and data movement

*From **Trilinos enhancements**, which Albany inherited.*

Solver portability on Cori and Summit:

- MueLu SemiCoarsen refactor using Kokkos
- Ifpack2 portable smoothers tuned to GPU hardware

Albany **performance improves** in time due to Albany's **development model** and **maintenance workflows.**



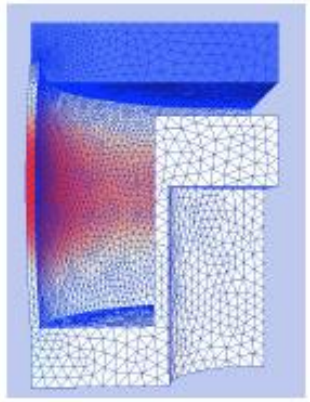
J. Watkins, M. Carlson, K. Shan, I. Tezaur, M. Perego, L. Bertagna, C. Kao. *et al*. "Performance portable ice-sheet modeling with MALI." (Submitted to IJHPCA, 2022) https://arxiv.org/abs/2204.04321

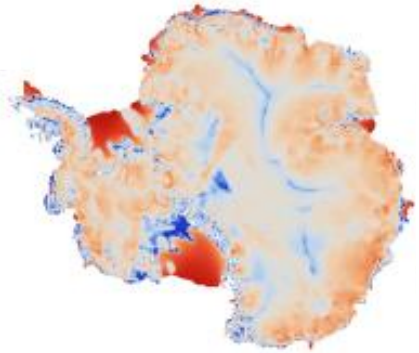
Outline

- Albany capabilities & supporting tools
  - Components effort
  - Albany under the hood
  - Kokkos for performance portability
  - Phalanx for template-based evaluators and DAG-based finite element assembly
  - Sacado for automatic differentiation (AD)
  - Some performance & performance portability results
- **Maintaining software quality of Albany**
  - Code pruning
  - PyAlbany: a Python interface to Albany
  - Nightly regression testing
  - Automated performance testing
  - Automated performance tuning
- Summary & perspectives

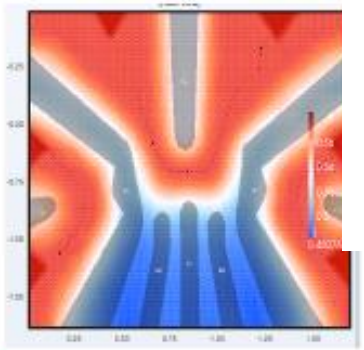
Albany in February 2019

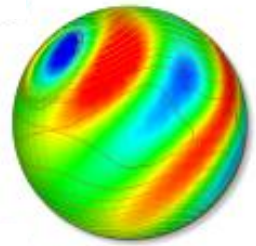
https://github.com/sandialabs/Albany

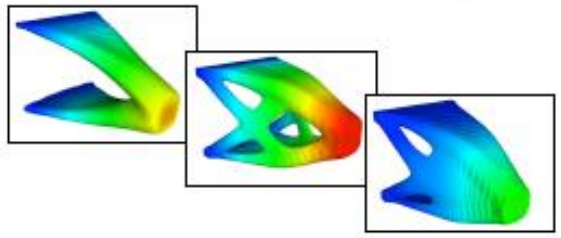


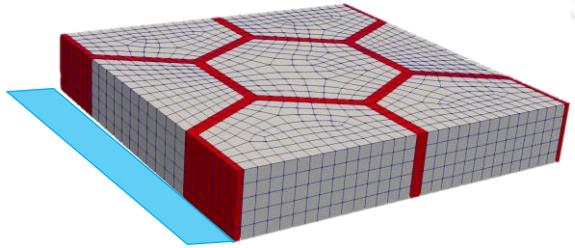


- Demo PDEs
- Quantum Devices (QCAD)
- Ice Sheets (Albany Land-Ice)
- Mechanics (LCM)
- Atmosphere Dynamics (Aeras)
- Particle-continuum coupling (Peridigm-LCM)
- Additive Manufacturing Design (ATO)
- Additive Manufacturing Processing (AMP)
- Arctic Coastal Erosion (ACE)
- Coupled Geomechanics (Albotran)









* **I. Tezaur** et al. "Albany: a Trilinos-based multi-physics partial differential equation research tool created using the Agile Components code development strategy," *SIAM CS&E 2019 (featured MS: "Multiphysics: Extensible, Composable Algorithms and Software")*, Spokane, WA.

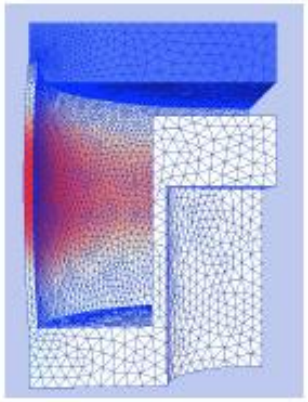
Albany Today

***Main Albany repo**:* https://github.com/sandialabs/Albany

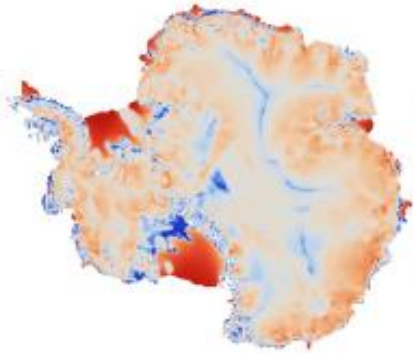
Tags: https://github.com/sandialabs/Albany/tags

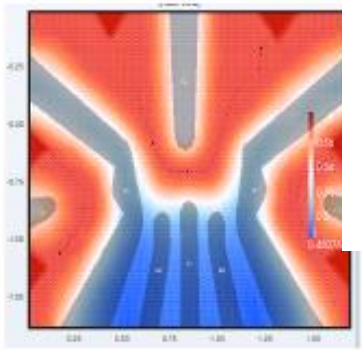
Albany-LCM repo*: https://github.com/sandialabs/LCM

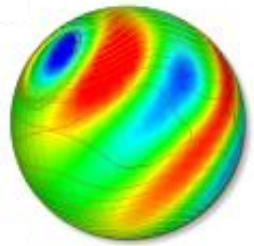
Albany-RPI repo: https://github.com/scorec/albany

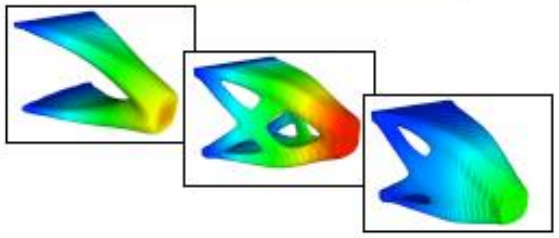


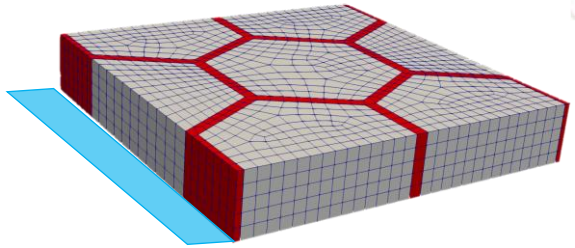
- ***Demo PDEs***
- Quantum Devices (QCAD)
- ***Ice Sheets (MPAS-Albany Land Ice)***
- Mechanics (LCM)
- Atmosphere Dynamics (Aeras)
- Particle-continuum coupling (Peridigm-LCM)
- Additive Manufacturing Design (ATO)
- Additive Manufacturing Processing (AMP)
- Arctic Coastal Erosion (ACE)
- Coupled Geomechanics (Albotran)











* Maintained & tested nightly by Sandia teams.

New capability: ***PyAlbany***, a **python wrapper** for running Albany!

Lesson learned: supporting **unfunded capabilities** in large HPC code is not sustainable long-term…

New capability: Albany is know a **spack-age** (https://github.com/E3SM-Project/spack.git)

PyAlbany: A Python Interface to Albany

Lesson learned: PyAlbany allows to easily and quickly…

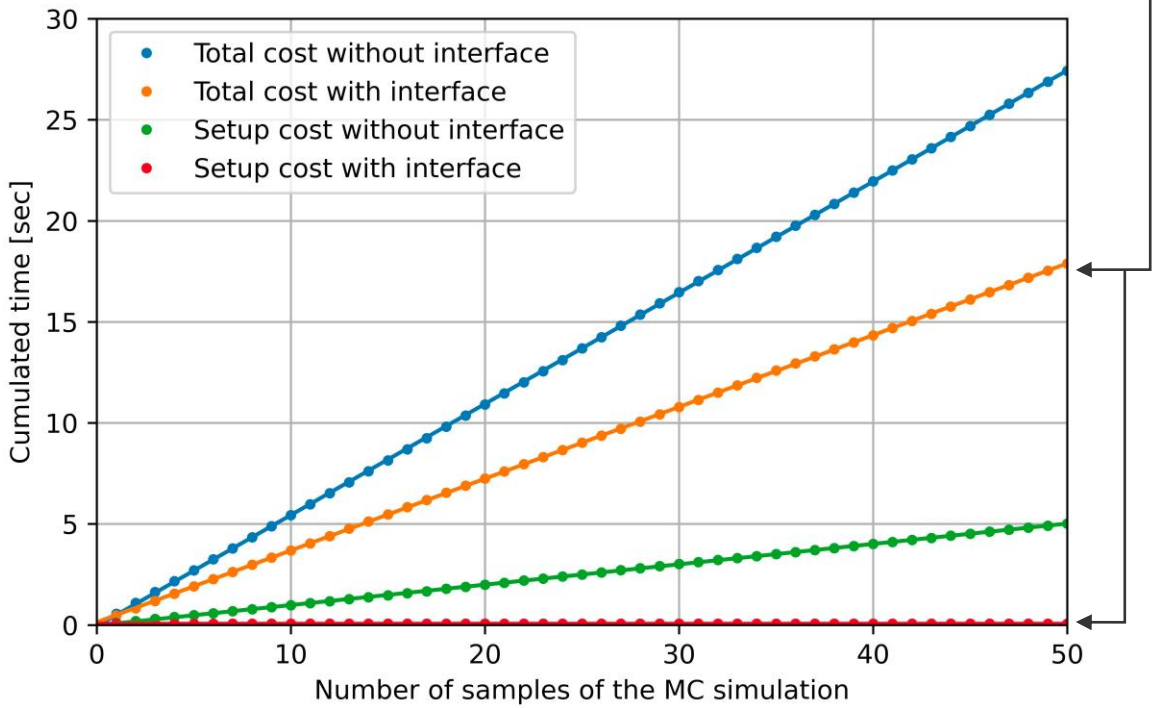
- Use Albany **without C++** or **bash knowledge** (convenient for students)
- Prototype applications that require **multiple Albany solves**
- Enable **fast pre-processing** and **post-processing** in Python
- Use Python as a **glue language** to couple Albany with other software: for UQ methods (PyDakota), machine learning (TensorFlow, Keras, Scikit-learn), plotting (Matplotlib, Paraview), …

Running Albany through **PyAlbany** is **faster** for MCMC analysis! (don't need to go through the setup phase of Albany for every sample)

```python
import numpy as np
from numpy.random import default_rng
from PyAlbany import Utils

parallelEnv = Utils.createDefaultParallelEnv()

# Create an Albany problem:
filename = "input_scalar.yaml"
paramList = Utils.createParameterList(
    filename, parallelEnv
)

problem = Utils.createAlbanyProblem(paramList, parallelEnv)

parameter_map_0 = problem.getParameterMap(0)
parameter_0 = Utils.createVector(parameter_map_0)

parameter_0_view = parameter_0.getLocalView()

N = 200
p_min = -2.
p_max = 2.

# Generate N samples randomly chosen in [p_min, p_max]:
rng = default_rng()
p = rng.uniform(p_min, p_max, N)
QoI = np.empty((N,))

# Loop over the N samples and evaluate the quantity of interest:
for i in range(0, N):
    parameter_0_view[0] = p[i]
    parameter_0.setLocalView(parameter_0_view)
    problem.setParameter(0, parameter_0)

    problem.performSolve()

    response = problem.getResponse(0)
    QoI[i] = response.getLocalView()[0]
```

Listing 3: Monte Carlo example.

Please see talk by **Kim Liegeois** (MS284: Thurs. Mar. 2) for more on PyAlbany.

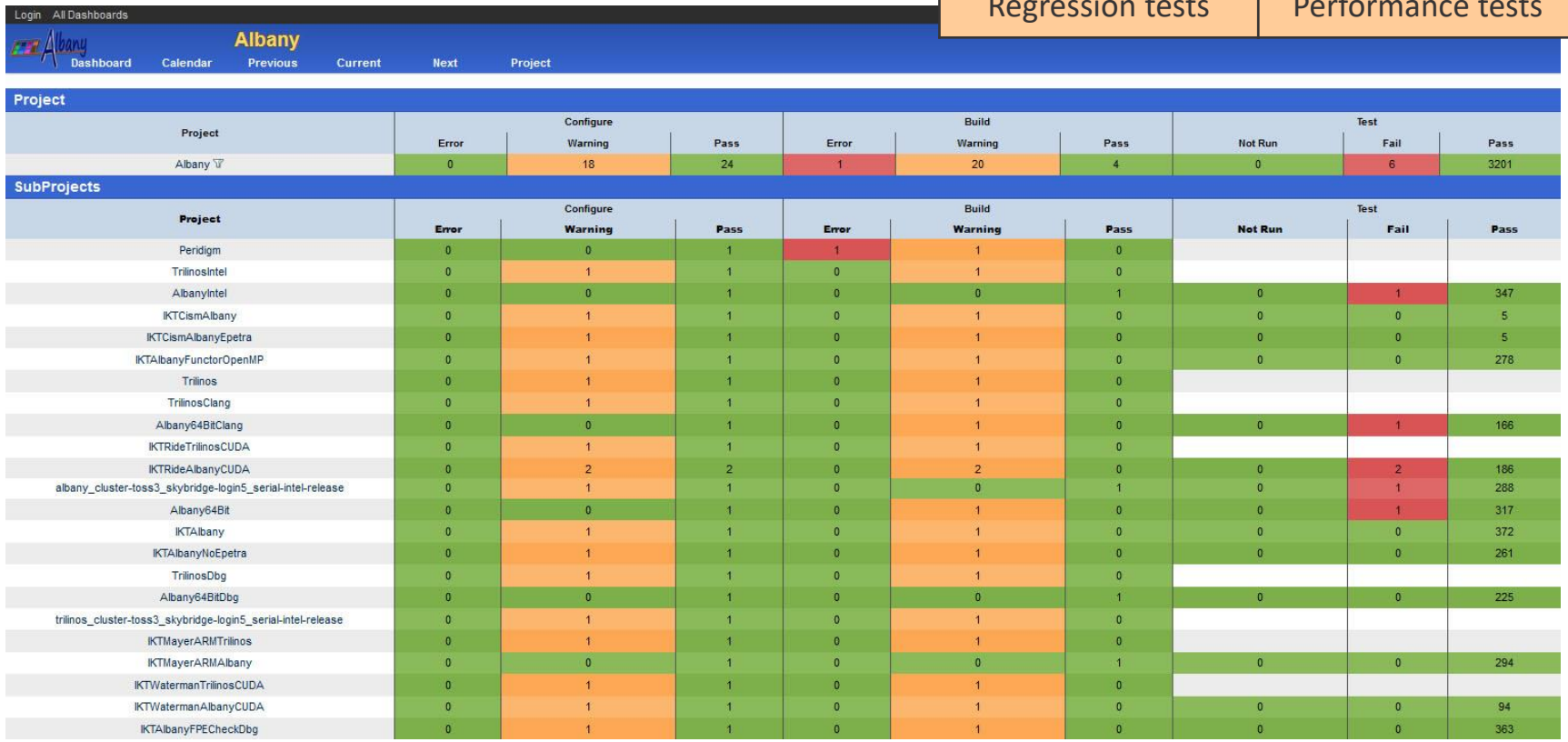


K. Liegeois, M. Perego, T. Hartland, "PyAlbany: A Python interface to the C++ multiphysics solver Albany". Journal of Computational and Applied Mathematics, 425, 115037, 2023.

(Automated Regression) Testing, Testing, Testing

Lesson learned: since MALI uses Trilinos and Albany **Versions of the Day (VOTD)**, **nightly testing** across variety of architectures is **essential** for maintaining **code quality**!

Repository*	Nightly test harness	Mailing lists
Version control	Unit tests	Issue tracking
Build system	Verification tests	Web pages
Config mgmt	Code coverage	Licensing
Regression tests	Performance tests	Release process



Login All Dashboards

Albany

Dashboard Calendar Previous Current Next Project

Project

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
Albany	0	18	24	1	20	4	0	6	3201

SubProjects

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
Peridigm	0	0	1	1	1	0			
TrilinosIntel	0	1	1	0	1	0			
AlbanyIntel	0	0	1	0	0	1	0	1	347
IKTCismAlbany	0	1	1	0	1	0	0	0	5
IKTCismAlbanyEpetra	0	1	1	0	1	0	0	0	5
IKTAlbanyFunctorOpenMP	0	1	1	0	1	0	0	0	278
Trilinos	0	1	1	0	1	0			
TrilinosClang	0	1	1	0	1	0			
Albany64BitClang	0	0	1	0	1	0	0	1	166
IKTRideTrilinosCUDA	0	1	1	0	1	0			
IKTRideAlbanyCUDA	0	2	2	0	2	0	0	2	186
albany_cluster-toss3_skybridge-login5_serial-intel-release	0	1	1	0	0	1	0	1	288
Albany64Bit	0	0	1	0	1	0	0	1	317
IKTAlbany	0	1	1	0	1	0	0	0	372
IKTAlbanyNoEpetra	0	1	1	0	1	0	0	0	261
TrilinosDbg	0	1	1	0	1	0			
Albany64BitDbg	0	0	1	0	0	1	0	0	225
trilinos_cluster-toss3_skybridge-login5_serial-intel-release	0	1	1	0	1	0			
IKTMayerARMTrilinos	0	1	1	0	1	0			
IKTMayerARMAlbany	0	0	1	0	0	1	0	0	294
IKTWatermanTrilinosCUDA	0	1	1	0	1	0			
IKTWatermanAlbanyCUDA	0	1	1	0	1	0	0	0	94
IKTAlbanyFPECheckDbg	0	1	1	0	1	0	0	0	363

CMake

GitHub

* https://github.com/sandialabs/Albany

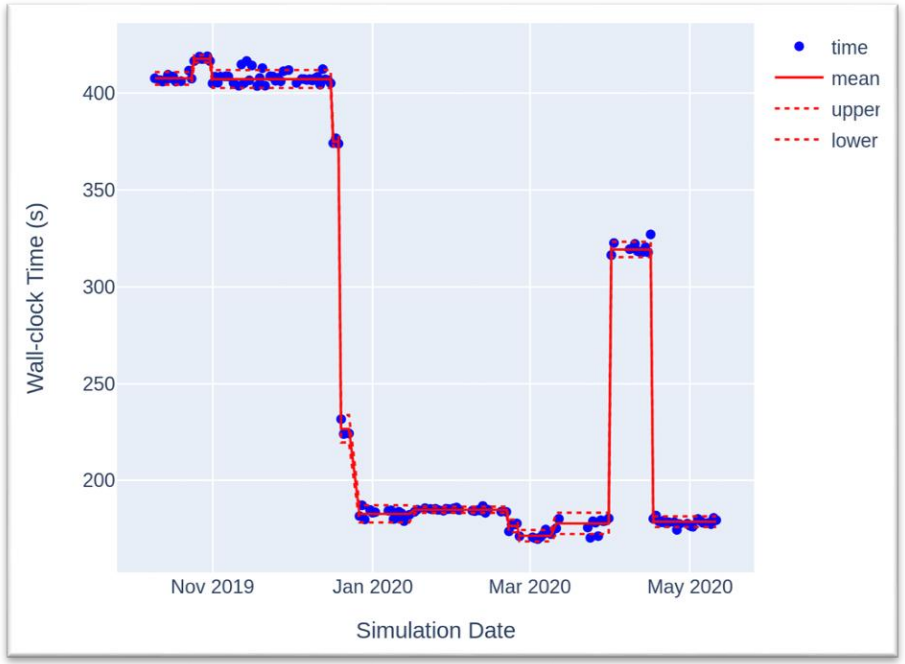
… and More (Automated Performance) Testing!

Lesson learned: nightly **regression testing** is ***not sufficient***! **Performance testing** is also 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

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

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



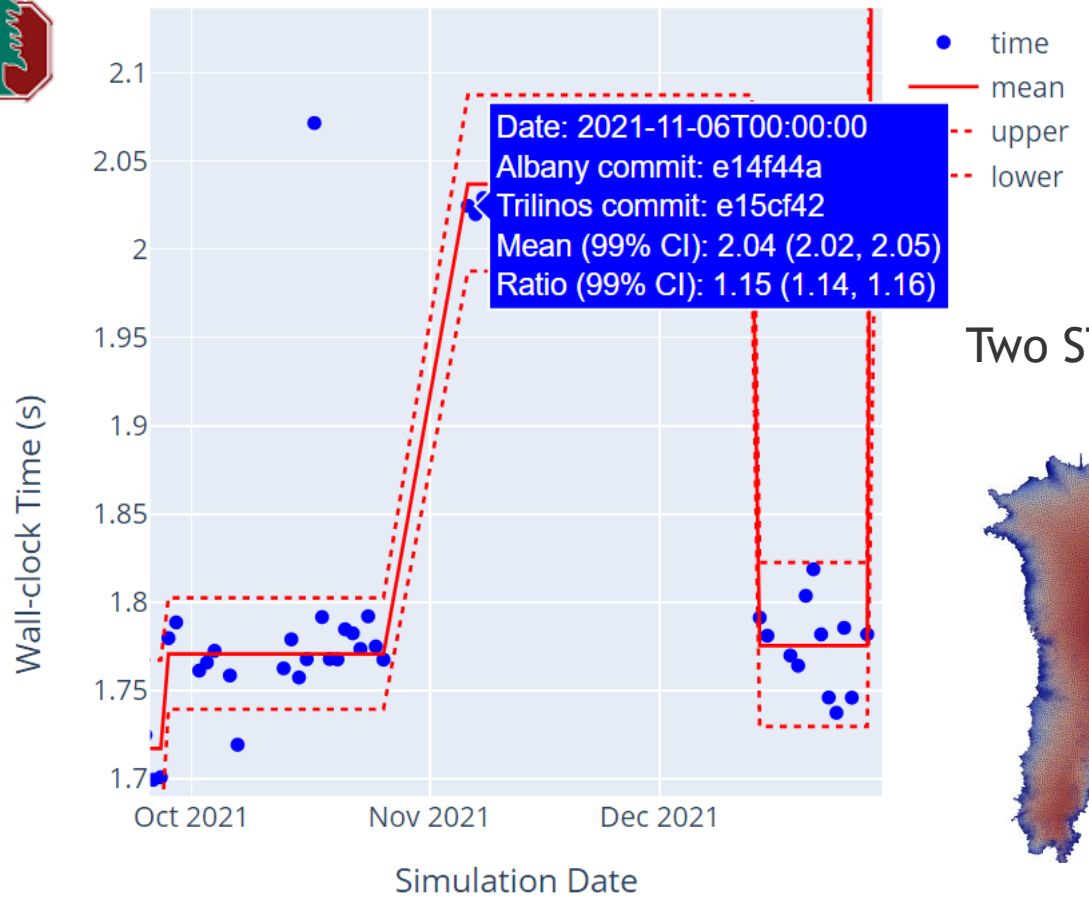
- Infrastructure is provided in a **Jupyter notebook**, exported as html to a website (https://sandialabs.github.io/ali-perf-data)
- Daily **email** provides nightly performance test summary

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

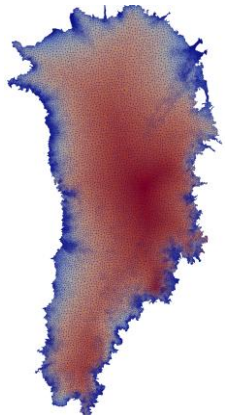
Detecting Performance Regressions/Improvements

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

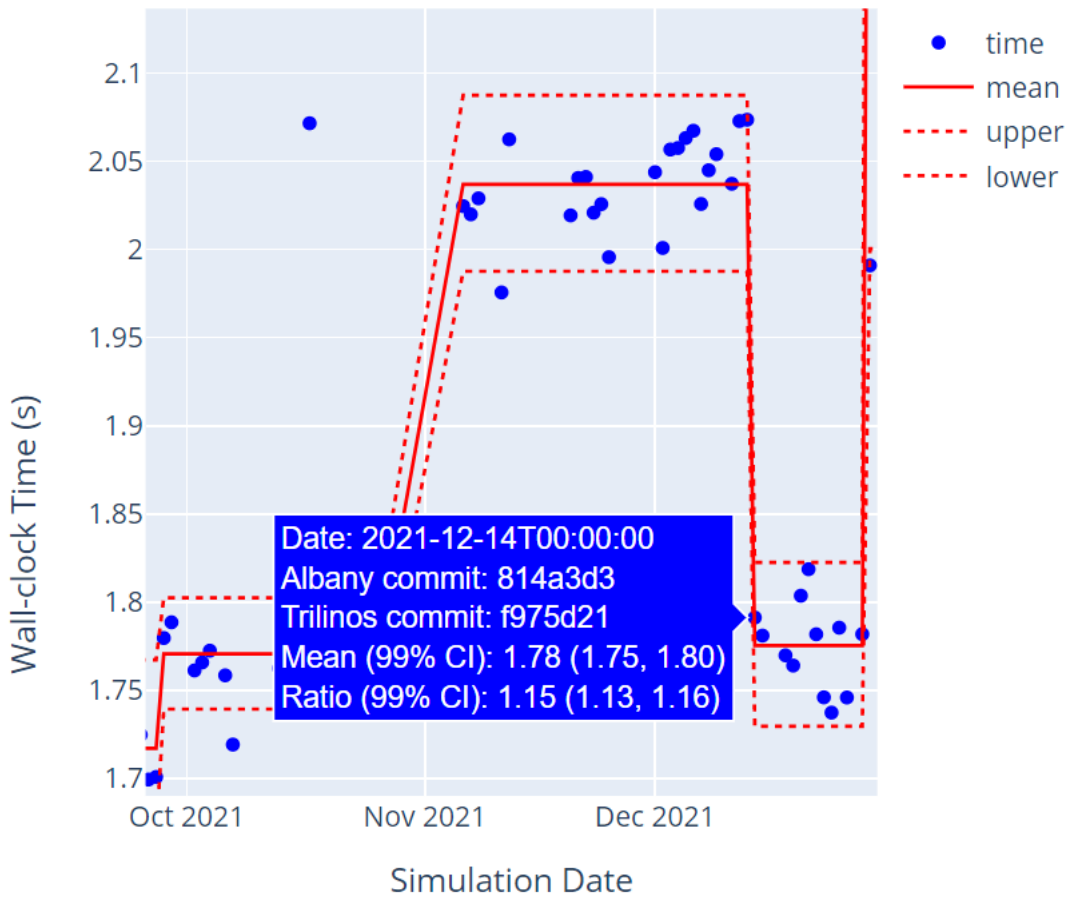
Regression



Two STDs



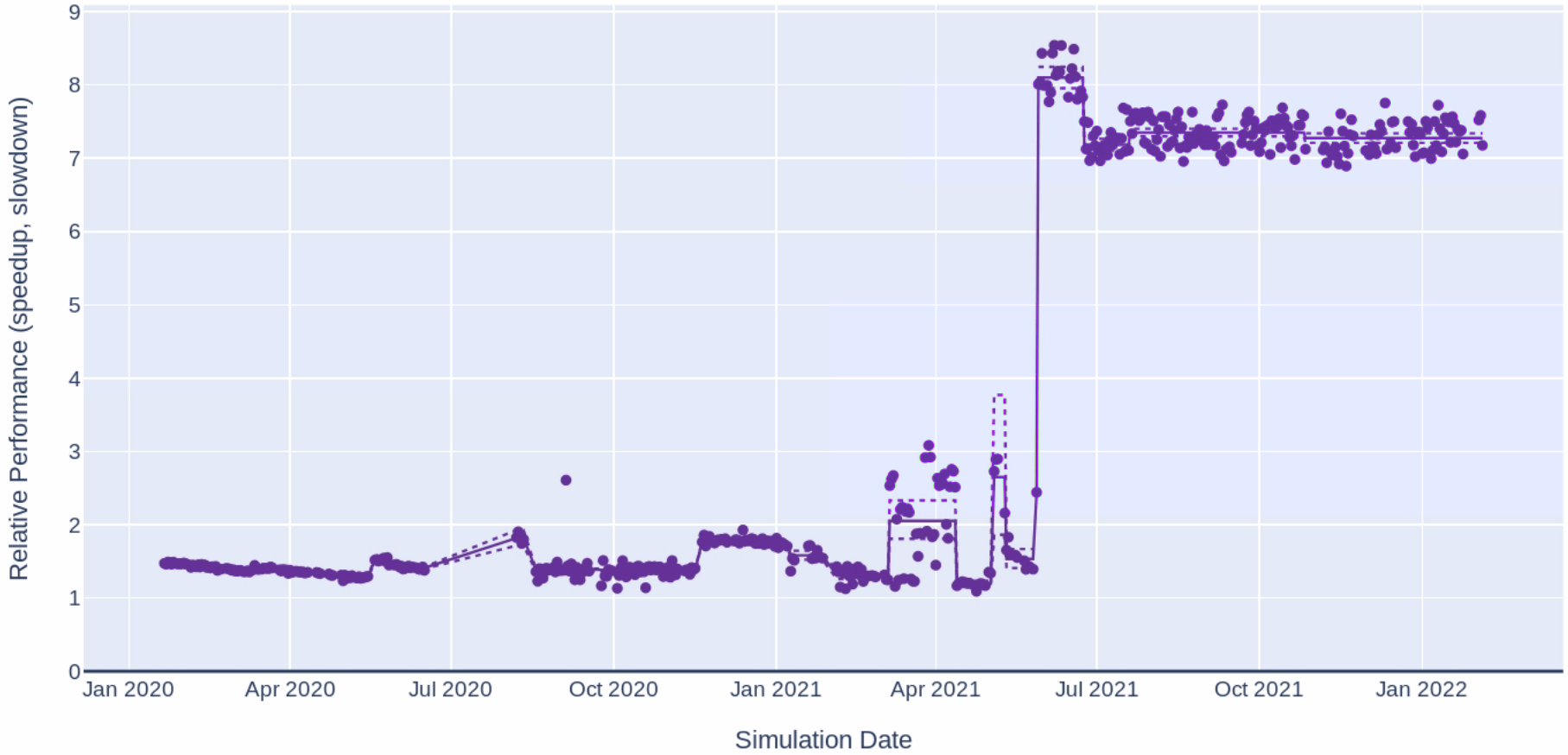
Improvement

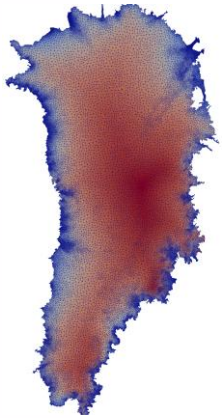


Total Fill time: for a 1-to-7 km resolution Greenland mesh, executed nightly in Albany Land Ice.

Monitoring Performance Comparisons

Example: Memoization comparison (with & without) shows that relative performance has improved



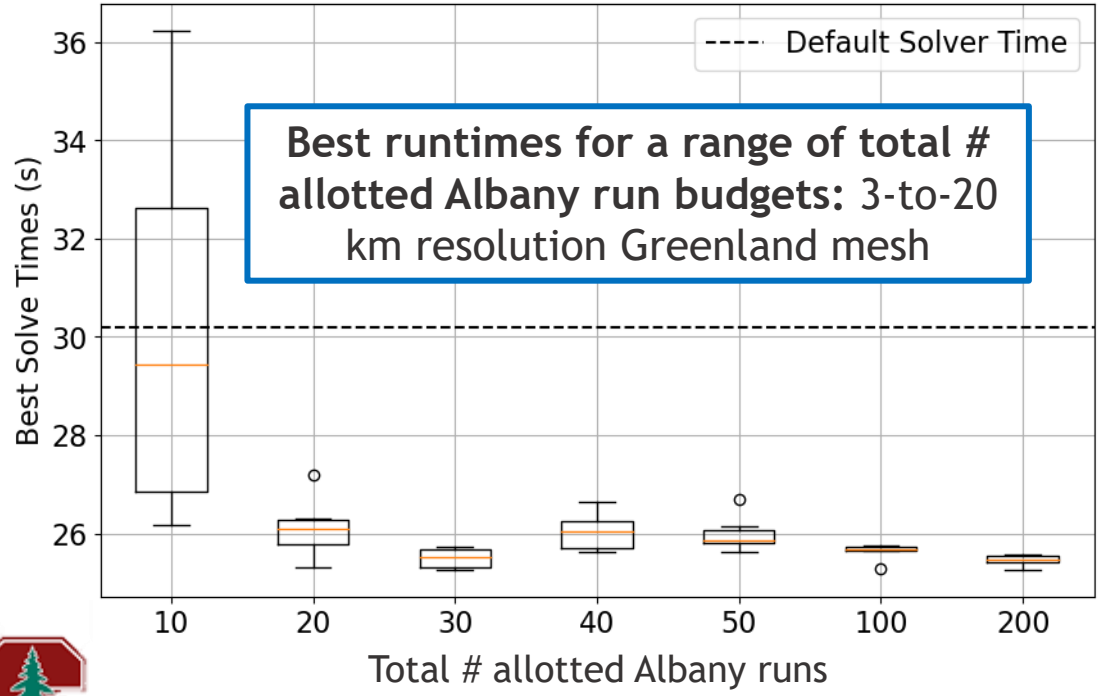


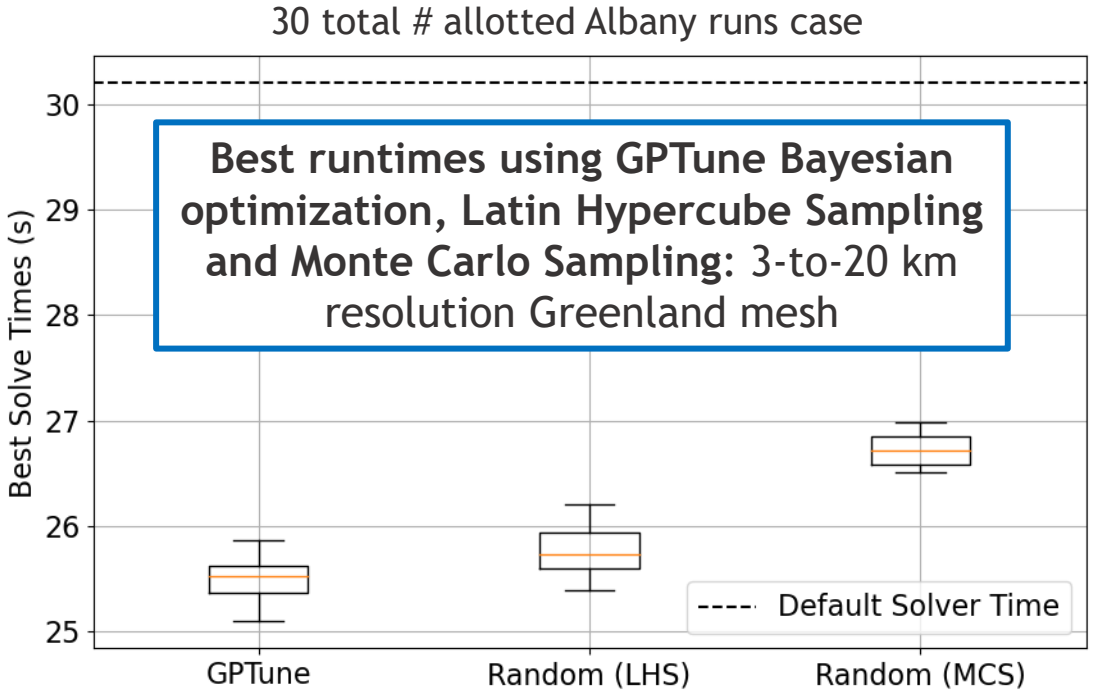
Speedup of Total Fill time: from memoization for a 1-to-7 km resolution Greenland mesh, executed nightly in Albany Land Ice

Automated Parameter/Performance Tuning

Lesson learned: **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.





Example: autotuning used to improve performance of multigrid smoothers on GPU

M. Carlson, J. Watkins, I. Tezaur. "Automatic performance tuning for MPAS-Albany Land Ice." JCAM, 2023.

Outline

- Albany capabilities & supporting tools
  - Components effort
  - Albany under the hood
  - Kokkos for performance portability
  - Phalanx for template-based evaluators and DAG-based finite element assembly
  - Sacado for automatic differentiation (AD)
  - Some performance & performance portability results
- Maintaining software quality of Albany
  - Code pruning
  - PyAlbany: a Python interface to Albany
  - Nightly regression testing
  - Automated performance testing
  - Automated performance tuning
- **Summary & perspectives**

Summary & Perspectives

- In developing HPC codes, there is often a **tradeoff** between **flexibility** and **efficiency**. Libraries like **Kokkos** and **Trilinos** can enable both to a large extent.
  - **Reasonable scalability** and **performance** is obtained across different architectures **without** architecture-specific optimizations within the Albany code base
  - Code promised to be **"future proof"**

- **Maintaining performance** and **portability** is crucial for an active code base
  - Supporting **unfunded capabilities** in large HPC code is not sustainable long-term
  - Regular **regression** and **performance testing** is crucial, especially in presence of every-changing codes
- **Automatic** processes like **testing** and **parameter tuning** can save developers a lot of time
  - A **change-point detection algorithm** can help identify performance variation automatically
  - **Optimal solver parameters** can be determined for a specific architecture **automatically** using black-box optimization algorithms

Bridging the Gap between Sandia & Academia, and Promoting Inclusion & Diversity through Albany

Recently-funded 5-year **Grande CARES MSIPP consortium**[1] will connect students/academics from minority serving institutions in U.S. southwest with Sandia National Laboratories via tools like **Albany**, **Trilinos**, **Kokkos**, …

MSIPP
Minority Serving Institution Partnership Program
Department of Energy Office of Environmental Management

UTEP Receives $1.25M Grant from DOE to Produce Pipeline of Scientists and Engineers [2]

Project will focus on recruiting and training scientists and engineers from underrepresented groups

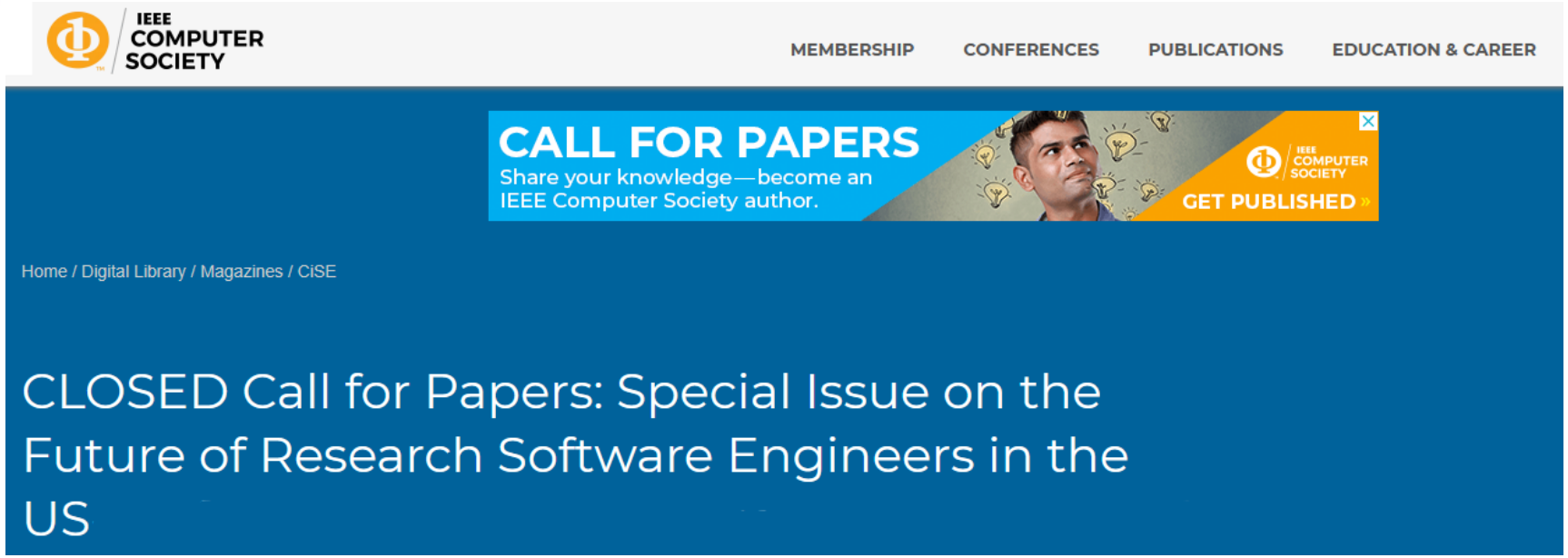
NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH UNIVERSITY

PRAIRIE VIEW A&M UNIVERSITY

NM STATE

[1] https://sites.google.com/view/grande-cares/home
[2] https://www.utep.edu/newsfeed/2023/utep-receives-1.25m-grant-from-doe-to-produce-pipeline-of-scientists-and-engineers.html

Special Issue of CiSE on Research Software Engineers



IEEE

Computing in SCIENCE & ENGINEERING

*"The goal of this special issue is to explore the **future** of **research software engineers** in the U.S., with emphasis on the **cultural**, **educational**, and **professional paradigm shifts** that need to occur."**

Estimated publication of special CiSE issue:
March/April 2024.

* https://www.computer.org/digital-library/magazines/cs/future-research-software-engineer

References

[1] A. Salinger, R. Bartlett, A. Bradley, Q. Chen, I. Demeshko, X. Gao, G. Hansen, A. Mota, R. Muller, E. Nielsen, J. Ostien, R. Pawlowski, M. Perego, E. Phipps, W. Sun, I. Tezaur. "Albany: Using Agile Components to Develop a Flexible, Generic Multiphysics Analysis Code", *Int. J. Multiscale Comput. Engng* 14 (4) 415-438, 2016.

[2] R. Tuminaro, M. Perego, I. Tezaur, A. Salinger, S. Price. "A matrix dependent/algebraic multigrid approach for extruded meshes with applications to ice sheet modeling", *SIAM J. Sci. Comput.* 38 (5) C504-C532, 2016.

[3] S. Price, M. Hoffman, J. Bonin, T. Neumann, I. Howat, J. Guerber, I. Tezaur, J. Saba, J. Lanaerts, D. Chambers, W. Lipscomb, M. Perego, A. Salinger, R. Tuminaro. "An ice sheet model validation framework for the Greenland ice sheet", *Geosci. Model Dev.* 10 255-270, 2017.

[4] I. Demeshko, J. Watkins, I. Tezaur, O. Guba, W. Spotz, A. Salinger, R. Pawlowski, M. Heroux. "Towards performance-portability of the Albany finite element analysis code using the Kokkos library", *J. HPC Appl.* 1-23, 2018.

[5] K. Evans, J. Kennedy, D. Lu, M. Forrester, S. Price, J. Fyke, A. Bennett, M. Hoffman, I. Tezaur, C. Zender, M. Vizcaino. "LIVVkit 2.1: Automated and extensible ice sheet model validation", *Geosci. Model Develop* 12 1067-1086, 2019

[6] M. Hoffman, M. Perego, S. Price, W. Lipscomb, T. Zhang, D. Jacobsen, I. Tezaur, A. Salinger, R. Tuminaro, L. Bertagna, "MPAS-Albany Land Ice (MALI): A variable resolution ice sheet model for Earth system modeling using Voronoi grid", *Geosci. Model Develop* 11 3747-3780, 2018.

[7] J. Watkins, I. Tezaur, I. Demeshko. "A study on the performance portability of the finite element assembly process within the Albany land ice solver", E. van Brummelen, A. Corsini, S. Perotto, G. Rozza, eds. *Numerical Methods for Flows: FEF 2017 Selected Contributions*, Elsevier, 2019.

[8] J. Watkins, M. Carlson, K. Shan, I. Tezaur, M. Perego, L. Bertagna, C. Kao, M. Hoffman, S. Price. “Performance portable ice sheet modeling with MPAS-Albany Land Ice”, submitted to *Int. J. HPC Appl.*

[9] M. Carlson, J. Watkins, I. Tezaur. "Automatic performance tuning for MPAS-Albany Land Ice", submitted to *J. Comput. Appl. Math.*

[10] K. Liegeois, M. Perego, T. Hartland, “PyAlbany: A Python interface to the C++ multiphysics solver Albany”. Journal of Computational and Applied Mathematics, 425, 115037, 2023.

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).

Office of Science
U.S. Department of Energy

Sandia
National
Laboratories

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

Start of backup slides

MALI (MPAS-Albany Land Ice) Software Ecosystem

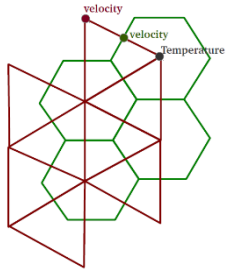
MALI = MPAS Model for Prediction Across Scales + Albany Land Ice

MPAS[1]:

- Thickness & temperature evolution

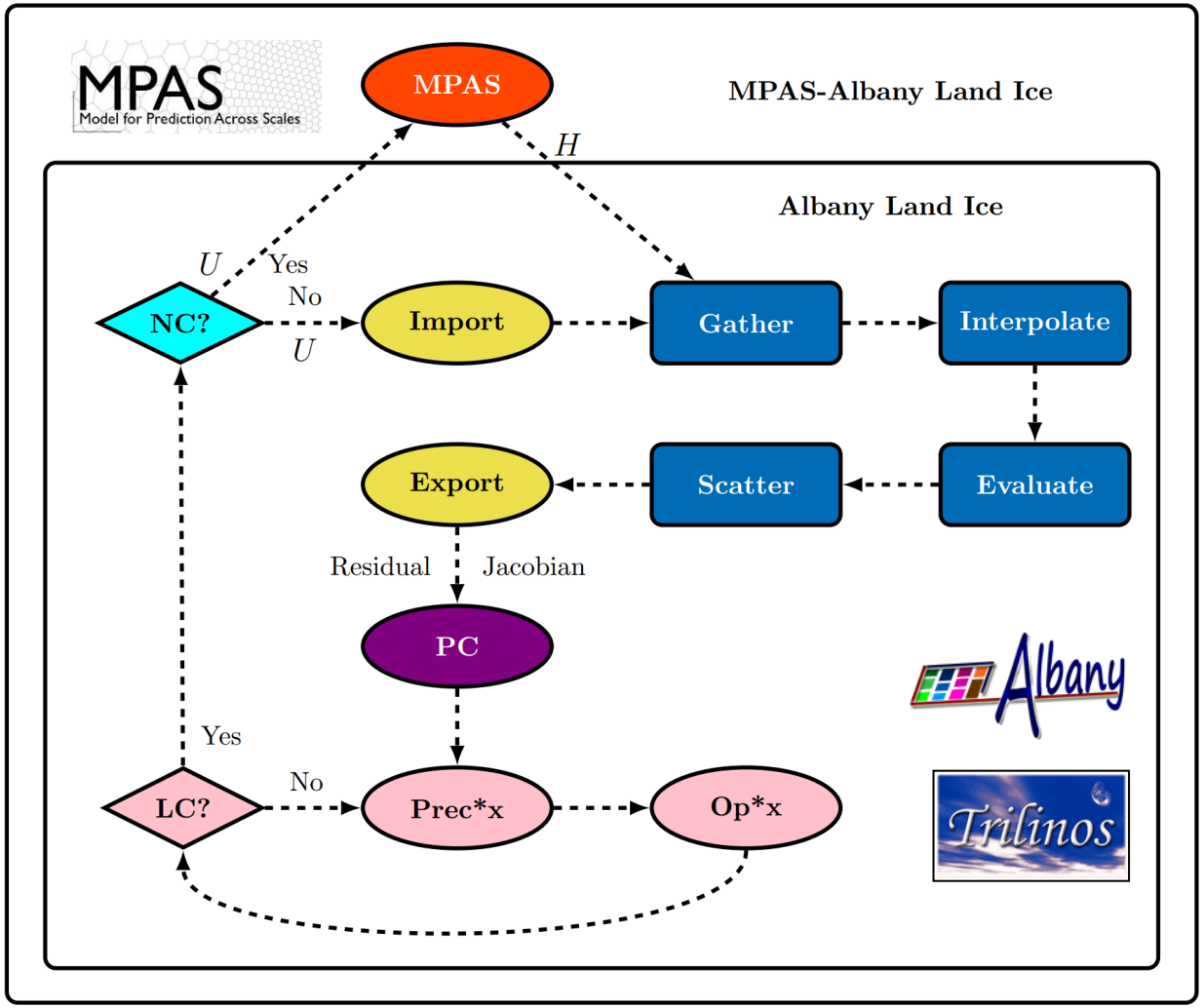
Albany Land Ice[2]:

- First-order Stokes velocity solver



Trilinos[3]:

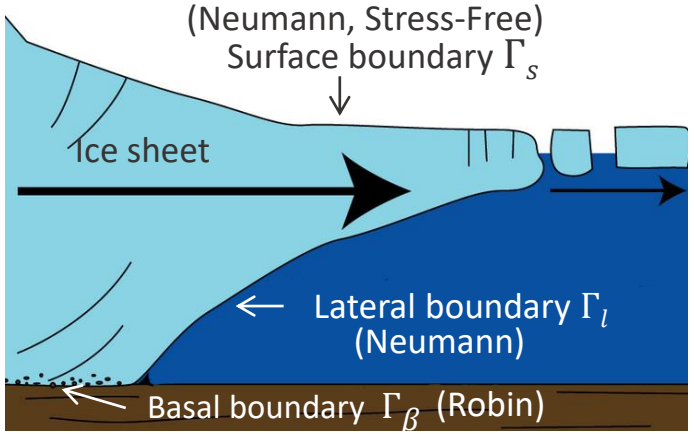
- Mesh tools (*STK*)
- Discretization tools (*Intrepid2*)
- Nonlinear/Linear solver (*NOX/Belos*)
- Distributed memory linear algebra (*Tpetra*)
- Multigrid Preconditioner (*MueLu*)
- Field DAG (*Phalanx*)
- Automatic differentiation (*Sacado*)
- Shared memory parallelism (*Kokkos*)
- Many more…



[1] https://github.com/MALI-Dev/E3SM. [2] https://github.com/sandialabs/Albany. [3] https://github.com/trilinos/Trilinos.

First-Order (FO) Stokes Velocity 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.**



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

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

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

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

- 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: $A(T) = A_0 e^{-\frac{Q}{RT}}$

Implicit solver:

FEA* = 50% CPU-time

Linear solve = 50% CPU-time

- **First-Order Stokes model:** nice **elliptic** approximation to the full Stokes equations, derived using **hydrostatic approximation** & scaling argument based on ice sheets being thin with near-vertical normal
- **3D model** for two unknowns, the (u_1, u_2) ice velocities.
- Highly **nonlinear** rheology with viscosity μ given by **Glen's flow law**
- Valid for both **Greenland** and **Antarctica** continental-scale simulations

Albany

* Finite element assembly

Albany Under the Hood

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

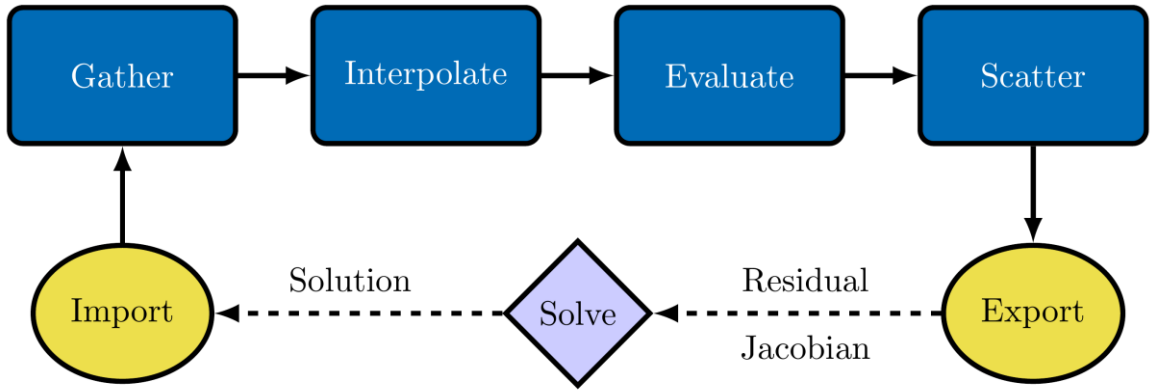
Adding new PDEs requires just implementing a new **Evaluate** routine

Albany design highlights:

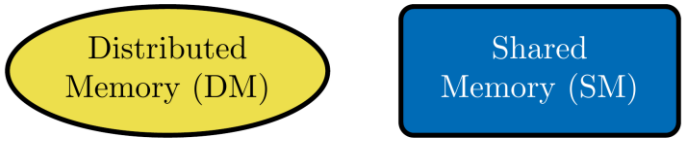
- **Piro** manages the solve (e.g., Newton, time-stepper)
- Jacobians (+ sensitivities, Hessians, etc.) obtained via **automatic differentiation** (**Sacado**)
- **Kokkos** achieves performance portability using **MPI+X**

Trilinos Packages

FEA Overview

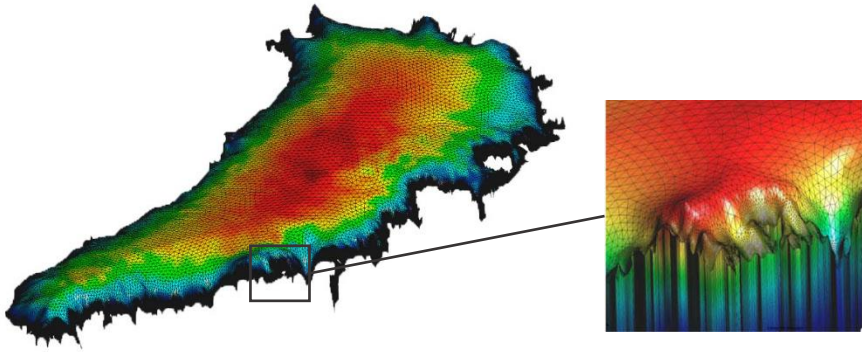


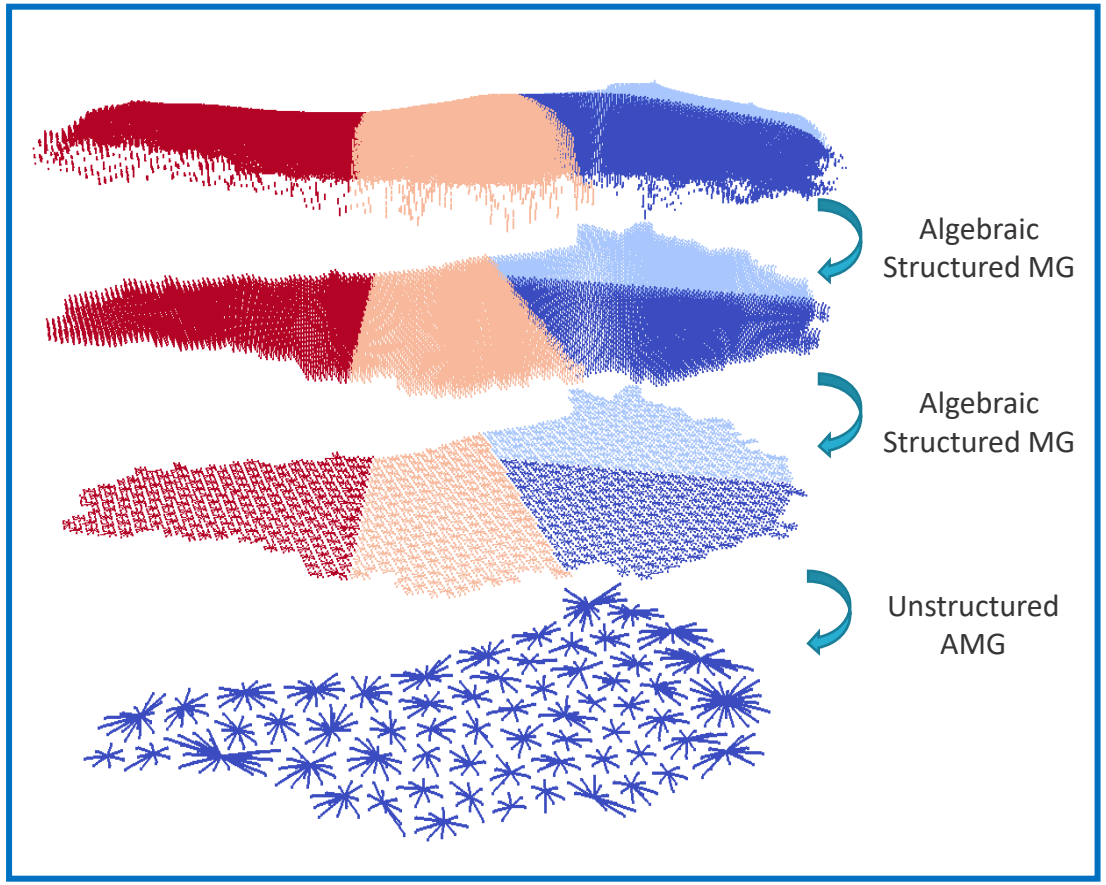
Memory Model



Albany Supporting Tools: Belos/MueLu – Preconditioned Iterative Solver

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)

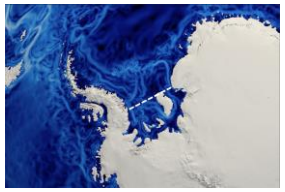
Solver: Preconditioned Newton-Krylov

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

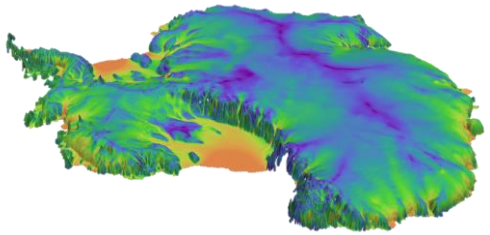
Future Performance Overview

Applications

E3SM + MALI



MALI-standalone



DOE Exascale

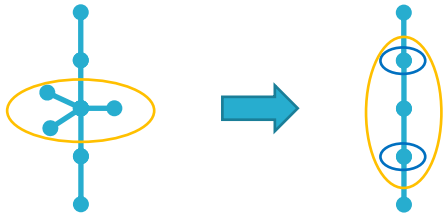
NERSC Perlmutter

OLCF Frontier

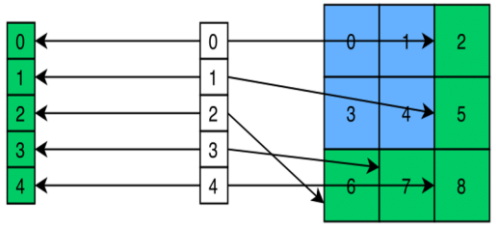
ALCF Aurora

Performance

Algorithmic improvements



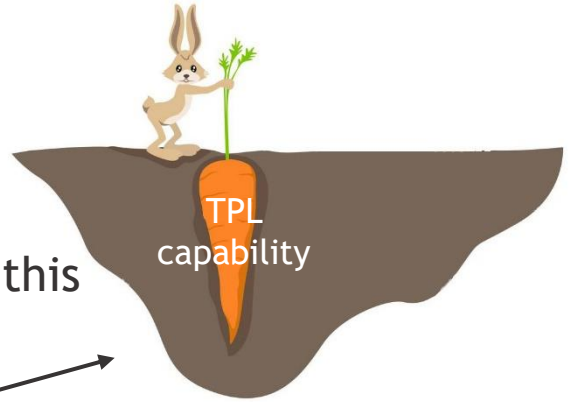
Performance Optimization



FASTMATH

RAPIDS

Summary & Perspectives



- HPC architectures **changing rapidly** poses a significant challenge for mod/sim codes
  - The **Albany, Trilinos** and **Kokkos** software stack offers an efficient way to meet this challenge for large scale **finite element analysis**
  - There are both **pros** and **cons** to using ever-changing TPLs like these
    - **Pro:** code can inherit unexpected enhancements "for free"
    - **Con:** code might also inherit bugs/regressions which can be frustrating to track down and get fixed to maintain clean dashboard
  - **Performance portability** to next-generation architectures including GPUs is enabled via **Kokkos**
    - **Reasonable scalability** and **performance** can be obtained across different architectures **without** architecture-specific optimizations
    - Code promised to be "**future proof**"

- **Maintaining performance** and **portability** is crucial for an active code base
  - Supporting **unfunded capabilities** in large HPC code is not sustainable long-term
  - Regular **regression** and **performance testing** is crucial, especially in presence of every-changing codes
  - A **change-point detection algorithm** can help identify performance variation automatically
- **Optimal solver parameters** can be determined for a specific architecture **automatically** using black-box optimization algorithms