Sandia National Laboratories

Exceptional service in the national interest

Automatic Performance Tuning for Albany Land-Ice

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

[1] Sandia National Laboratories, Livermore, CA, USA.

ESCO 2024, Pilsen, Czechia.

June 11, 2024

SAND2024-07296C

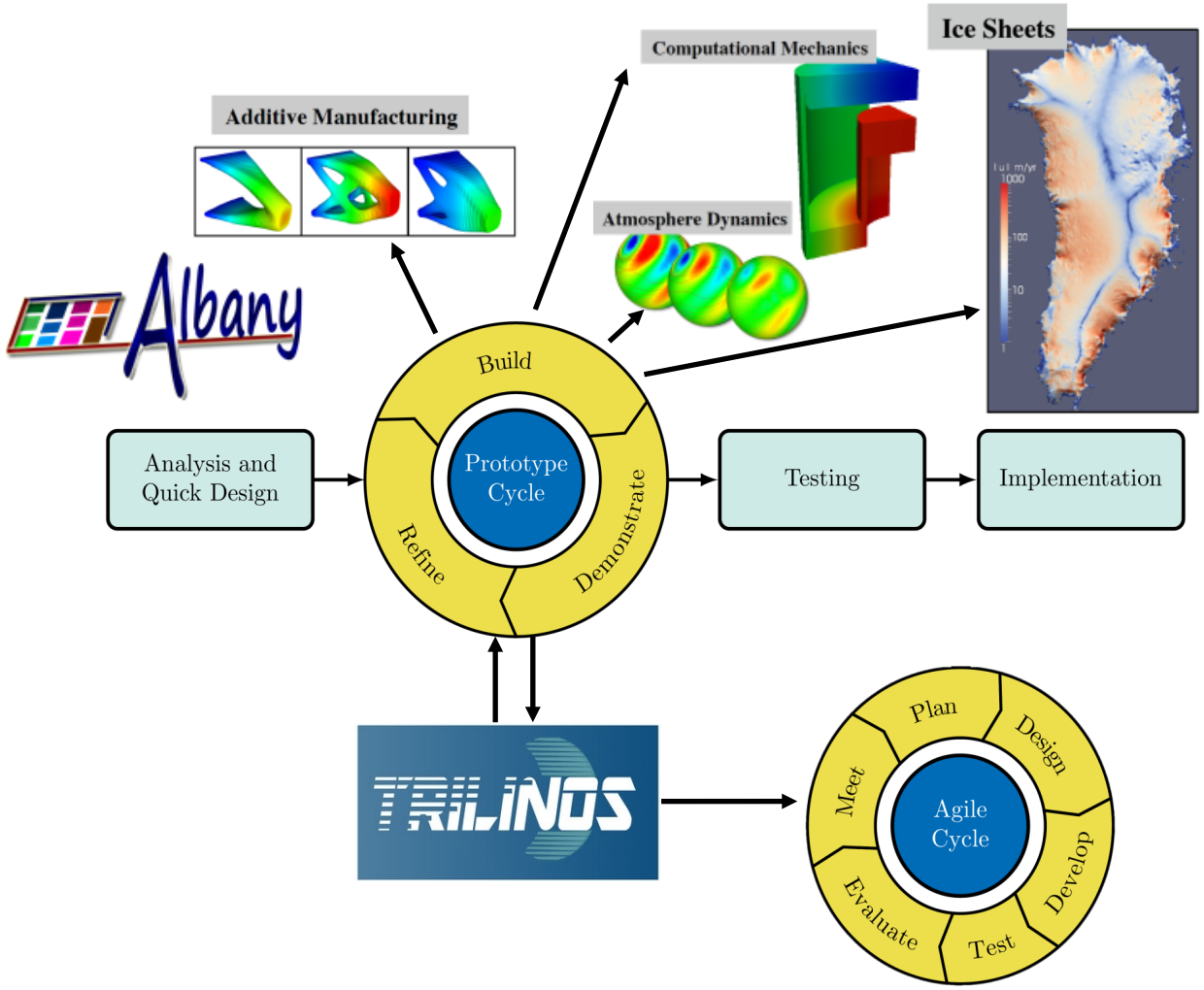
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

U.S. DEPARTMENT OF ENERGY

NNSA

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*:

- Designed to facilitate prototyping of scientific models and analysis tools
- Albany-Land Ice is a model that evolved from prototype to full-fledged production software
- Close collaboration with Trilinos developers facilitates efforts to maintain Albany's **scalability** and **portability**

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

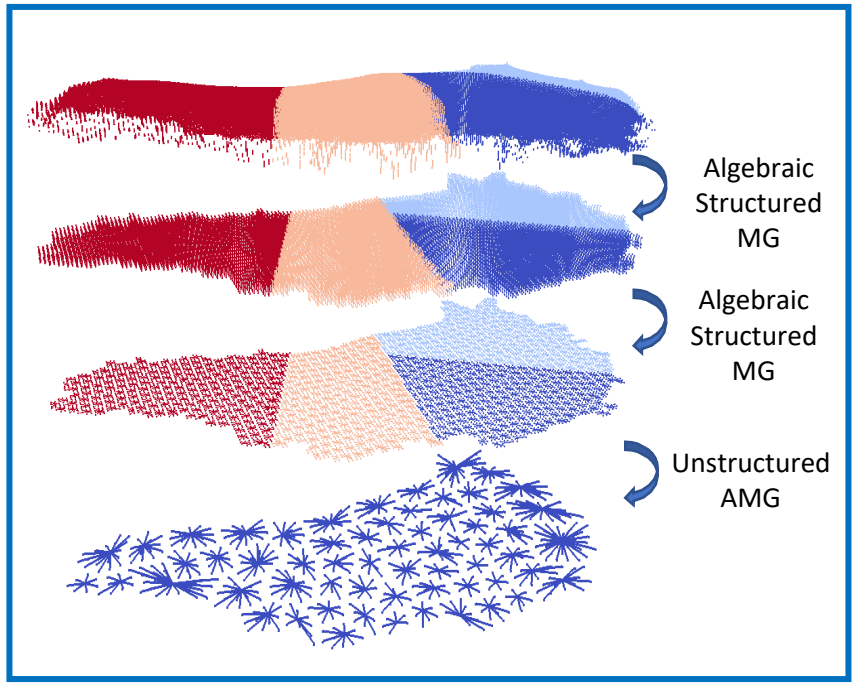
Motivation - Automated Performance Tuning

Problem Description

- Find a **robust** set of parameters for optimal **performance** and **accuracy**
- Often many runtime parameters to choose from (e.g. discretization, solver)
- Optimal parameters are not necessarily unique across architectures or problem inputs
  - Optimal parameters can also shift in time due to code changes, algorithmic optimization, and deployment on new compilers or architectures

Putting Automated Performance Tuning to Work

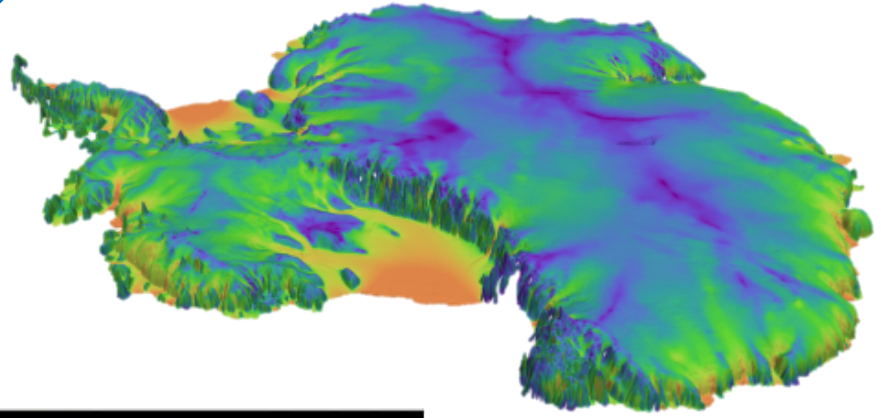
- We consider the production-level land-ice simulation software **Albany Land-Ice (ALI)**
- Effort is ongoing towards improving the performance portability of **ALI** for use on current and next-gen computing platforms
- Current targets for performance improvements include multigrid preconditioner with many run-time parameters



MPAS-Albany Land Ice (MALI)

What is MALI?

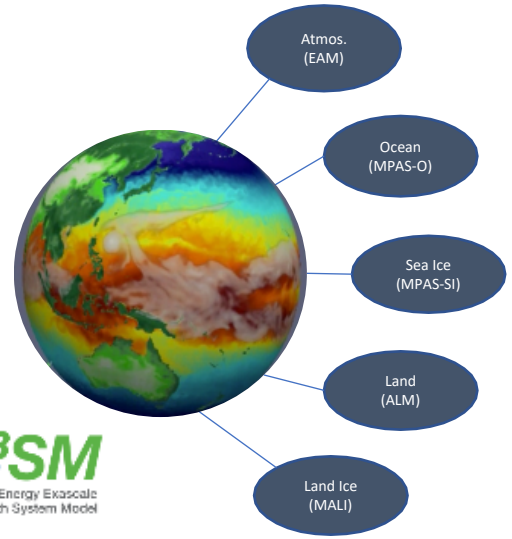
- U.S. DOE SciDAC-funded project land-ice modeling project, **FAnSSIE (Framework for Antarctic System Science in E3SM, FY23-FY27)**
- Albany Land Ice is the velocity solver of **MALI (MPAS-Albany Land Ice)**, the land-ice component of the U.S. DOE's Energy Exascale Earth System Model (E3SM)
- Portable performance is critical to target new and upcoming computing platforms such as NERSC's **Perlmutter** supercomputer



NERSC PERLMUTTER

FASTMATH

RAPIDS



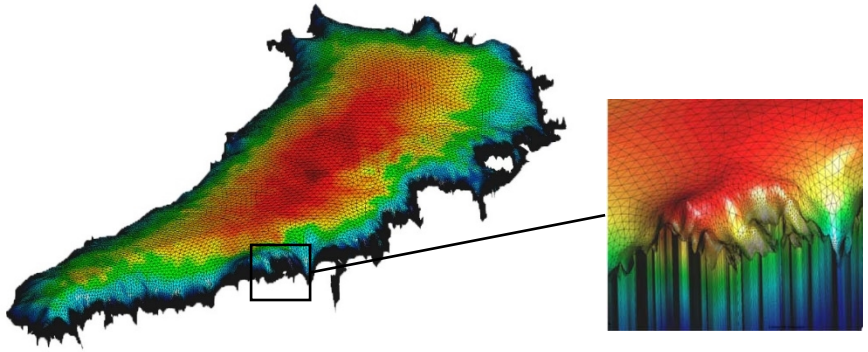
E3SM Energy Exascale Earth System Model

https://github.com/MALI-Dev/E3SM

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.

ALI's Multigrid Preconditioner

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

Performance of multigrid preconditioners depend on many **run-time parameters**

Run-time parameters:

- Number of levels in the multigrid hierarchy
- Types of smoothing algorithms at fine/coarse levels:
  - **Multi-threaded Gauss-Seidel**
  - **Two-stage Gauss-Seidel**
  - **Chebyshev**
  - **Damped Block-Jacobi**
- Smoother-specific parameters such as:
  - **Multi-threaded/Two-stage Gauss-Seidel:**
    - Number of sweeps
    - Damping factor
  - **Damped Block-Jacobi:**
    - Number of sweeps
    - Damping factor
  - **Chebyshev:**
    - Eigenvalue ratio
    - Chebyshev expansion degree
    - Maximum number of iterations
- Aggregation parameters
- …and more!

Approach to Performance Tuning

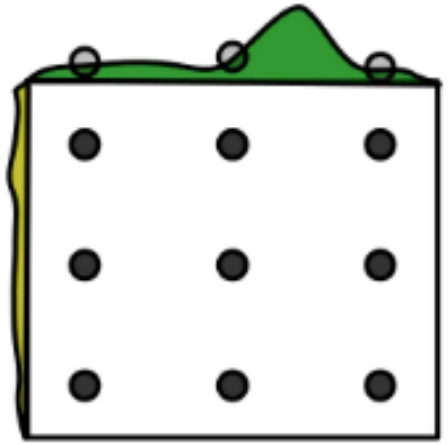
Goal: We want to find the optimal parameters to **minimize** solve time of ALI

- Gradient is not available for this optimization problem, treated as **blackbox optimization**
- Naïve methods for **blackbox optimization:** exhaustive/grid search and random search
- **Bayesian** search works by fitting a Gaussian model to performance data to allow for a more directed approach to exploring parameter space

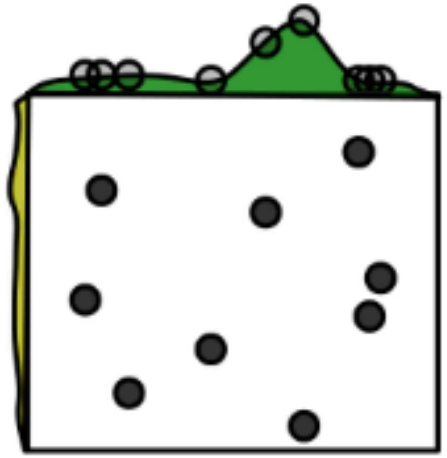
Offline vs **Online** Performance Tuning

- **Online** tuning evaluates candidate parameters on-the-fly during practical execution
- **Offline** tuning evaluates candidate parameters via trial execution and then optimal parameters are used for practical execution

Grid Search

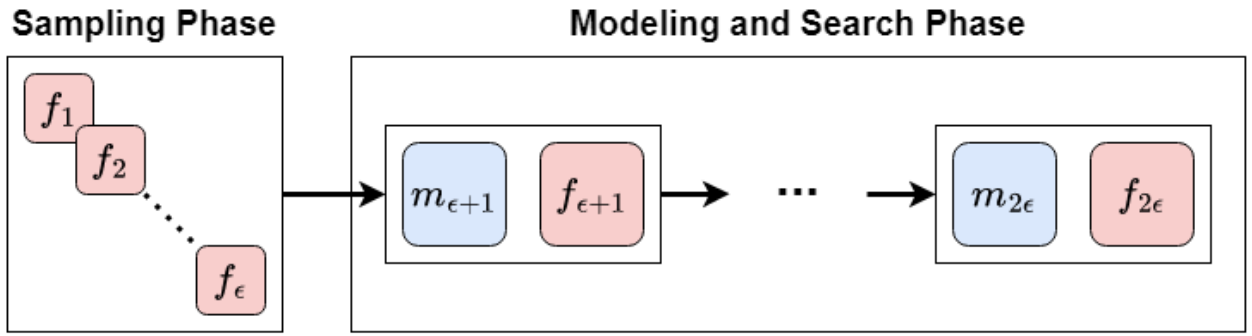


Random Search



Approach to Performance Tuning (GPTune)

- **GPTune** is an autotuning software library with a Python interface that relies on multitask and transfer learning using Bayesian optimization methodologies for blackbox optimization.
- Provides a **reverse communication interface** for Bayesian optimization
  - Noninvasive, no instrumentation of Albany required
- Supports **transfer learning** for leveraging available performance data to potentially lower cost of future/larger tuning tasks



https://github.com/gptune/GPTune

Workflow Management: Automated Tuning

Tasks are assigned to running jobs on a computing cluster, minimizing time jobs have to wait in queue

With a single Python script, we can ensure parallel efficiency for a variety of automated performance tuning tasks

Problem:

- A single evaluation of the objective function requires solving a problem at scale with Albany Land-Ice
- Samples of the performance model can be evaluated in parallel
- Need to take advantage of as much computing power as is available

Solution:

- Using the Python workflow management tool **Parsl**, the tuning workflow can be automated with **parallel efficiency** in mind

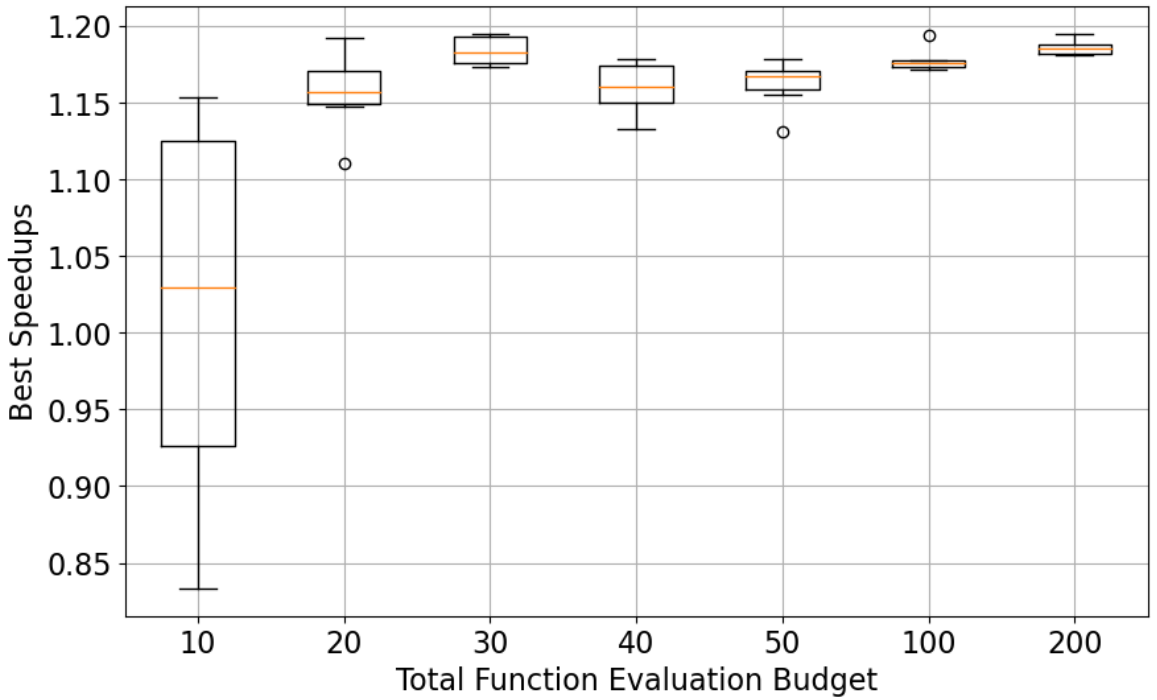
```python
# 1) get number of function evaluations
requested_num_evals = get_num_function_evals(...)
num_func_evals = requested_num_evals.result()

while num_func_evals > 0:

    # 2) create function evaluation configuration files
    requested_eval_configs = get_function_eval_configs(...)

    # 3) do function evaluations
    function_evaluated = []
    for k in range(num_func_evals):
        function_evaluated.append(evaluate_function(...))

    # 4) get function evaluations
    evaluation_results = []
    for k in range(num_func_evals):
        evaluation_results.append(get_evaluation_result(...))

    # 5) write function evaluations to database
    database_updated = update_database(...)

    # 6) get remaining number of function evaluations
    requested_num_evals = get_num_function_evals(...)
    num_func_evals = requested_num_evals.result()
```

Parsl

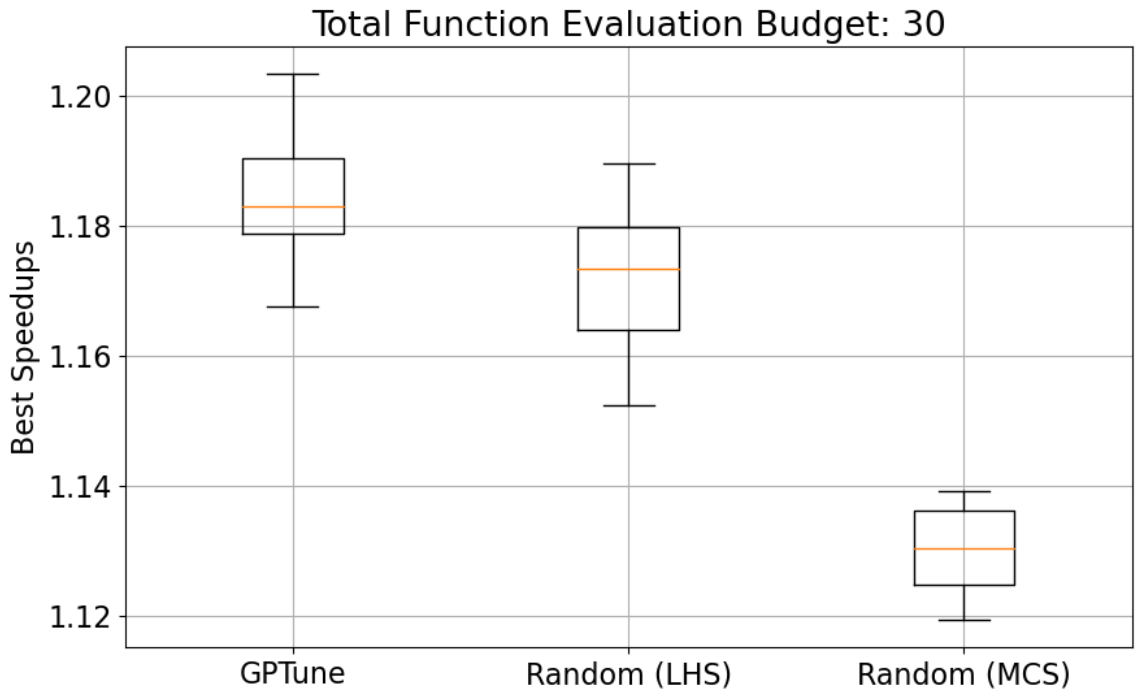
https://github.com/Parsl/parsl

GPTune Pre-tuning Analysis

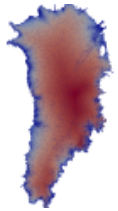
Total function evaluation budget refers to the number of times ALI is allowed to run during the tuning process



Extreme outliers can be found rarely with speedups up to **1.5x**



Good parameters can be found in relatively few runs of Albany



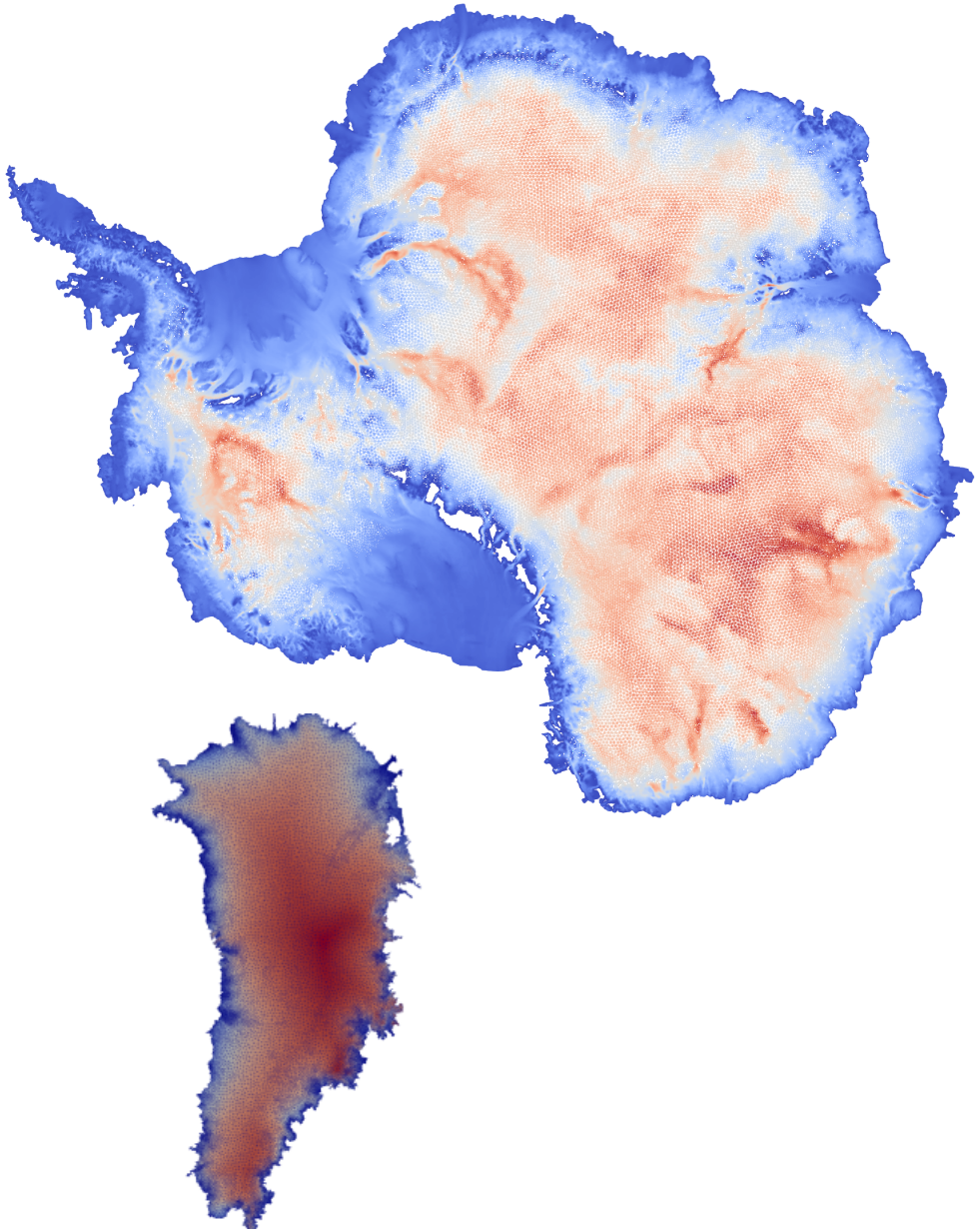
GPTune more reliably produces optimal parameters over random search

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

Tuning Case Overview

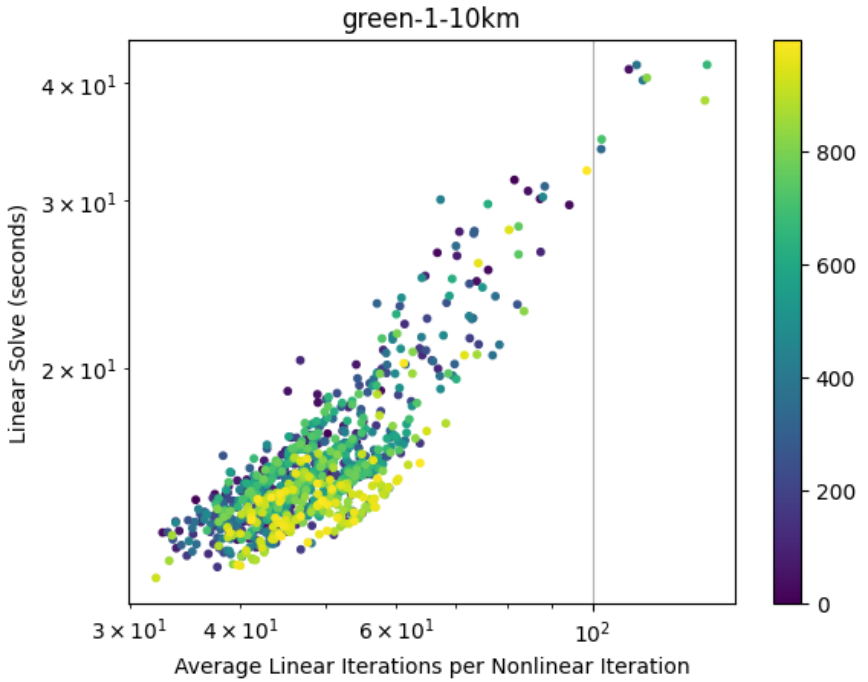
Cases:

- Tuned on a variety of typical land ice meshes:
  - Antarctic Ice Sheet (AIS)
  - Greenland Ice Sheet (GIS)
- Variety of mesh resolutions
  - AIS: 2-10km resolution
  - AIS: 4-20km resolution
  - GIS: 1-10km resolution
- For 4-20km Antarctica mesh, tuned on a single node and on 8 GPU nodes to see impacts of strong scaling



Chebyshev Smoother Tuning

- Followup tuning after tuning on smoother choice
- Tuning resulted in good speedup on top of the speedup from smoother selection tuning
- Optimal parameters resulted in best runtime AND convergence
- Optimal parameters for Antarctica meshes/setups are largely the same

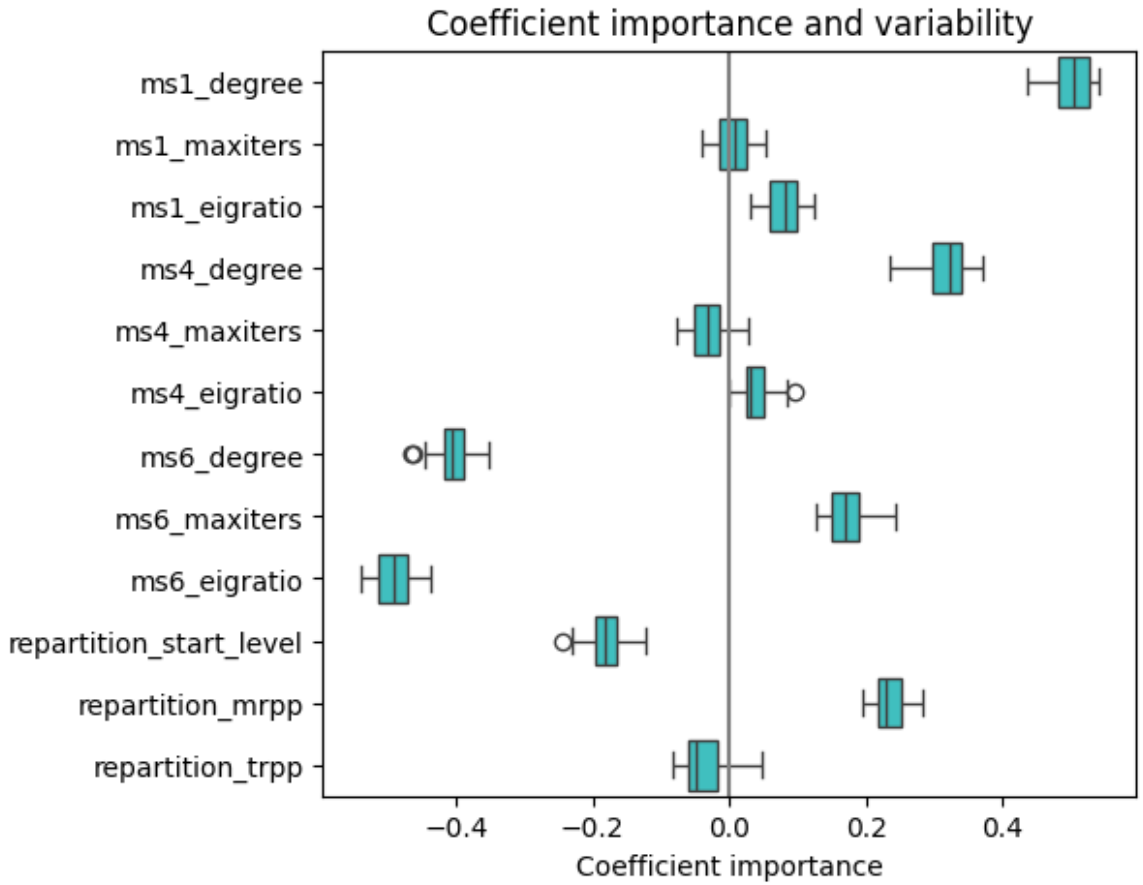


Case Name	Preconditioner Construction	Linear Solve	Total Solve	Tuning Speedup
green-1-10km	15.76s	12.37s	38.24s	**~23% speedup**
ant-4-20km-1node	7.39s	5.71s	18.13s	**~25% speedup**
ant-4-20km-8node	4.19s	3.99s	9.05s	**~36% speedup**
ant-2-10km-2node	15.6s	17.69s	43.9s	**~29% speedup**

Chebyshev Smoother Tuning

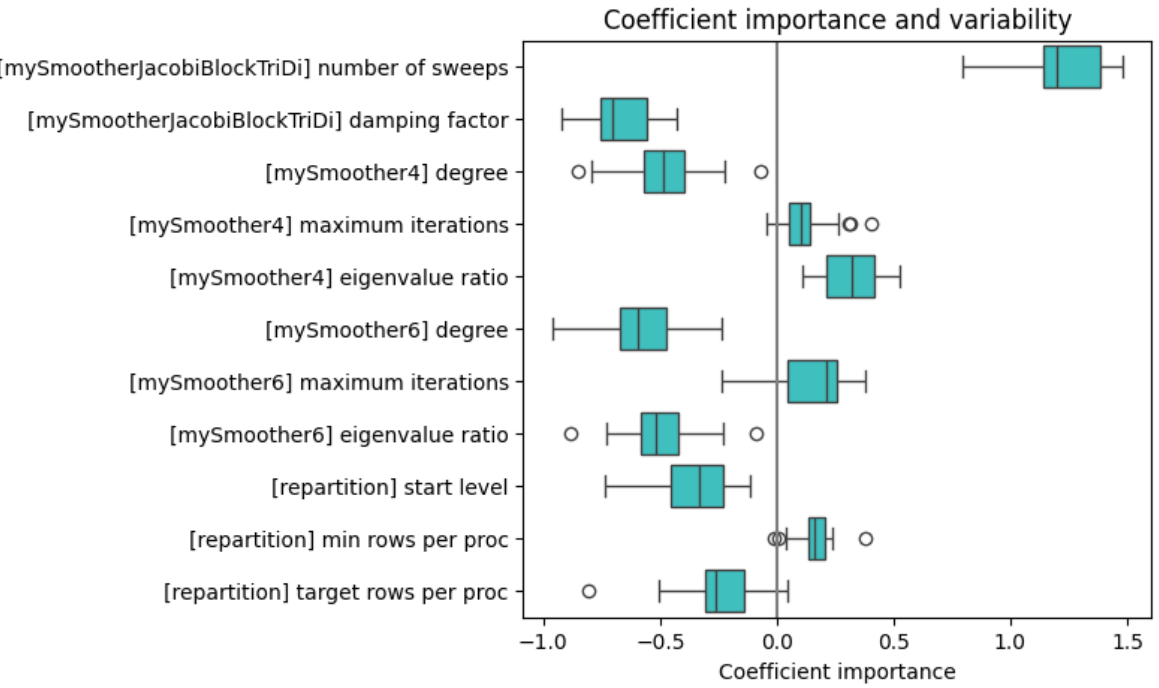
Lessons learned:

- Low polynomial degree should be used for fine smoother and high polynomial degree for coarsest smoother
- Optimal parameters for polynomial degree landed at the extremes of the search space
- Number of levels in multigrid hierarchy has huge impact on solve time and convergence



Block Jacobi Smoother Tuning

- Tuning offered very little speedup for this case
- Linear solve times and convergence are better but preconditioner construction is bottleneck
- Tuning data given to solver developers and solution has been identified



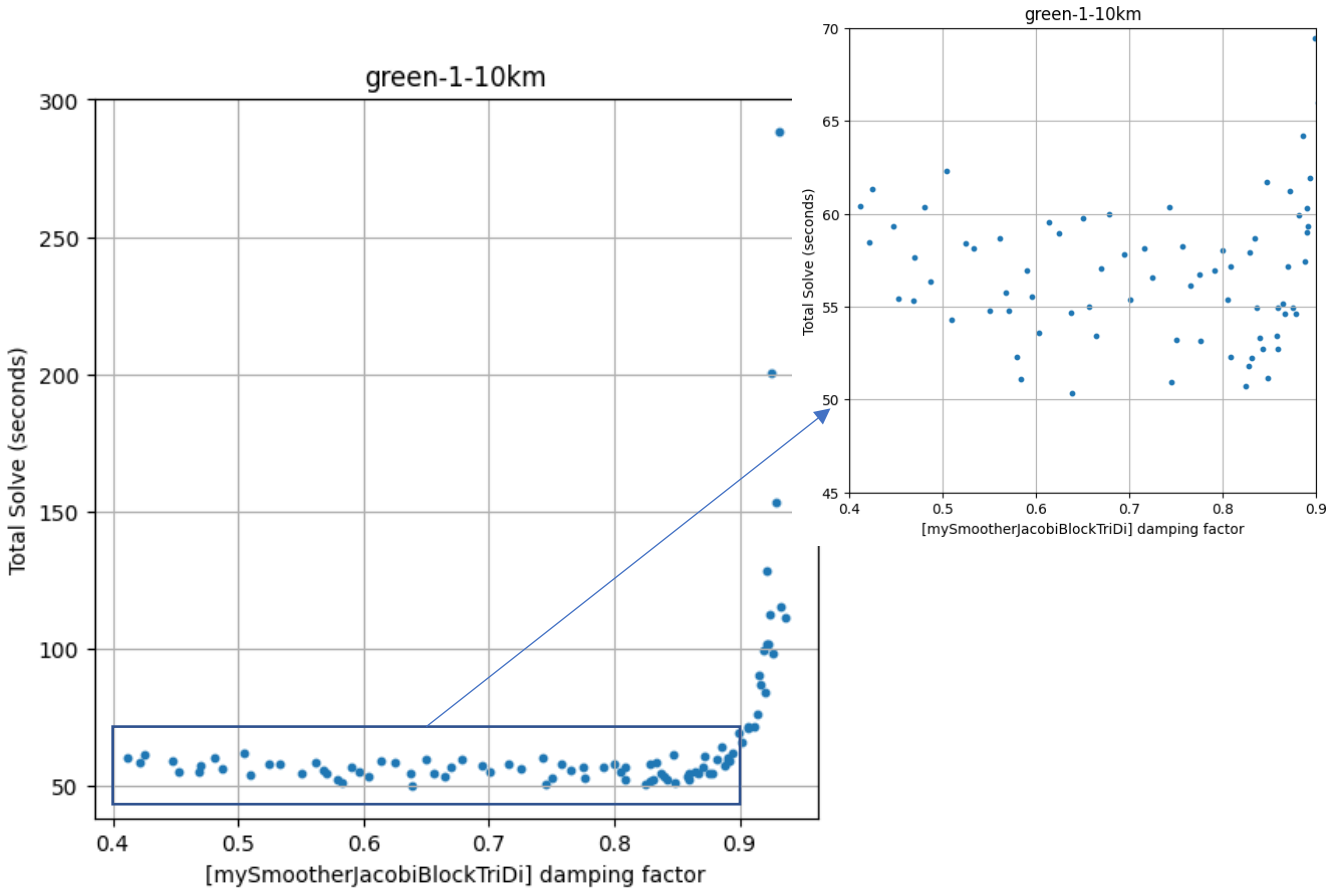
Case Name	Preconditioner Construction	Linear Solve	Total Solve	Tuning Speedup
green-1-10km	**32.2s**	8.2s	50.35s	~1% speedup
ant-4-20km-1node	**13.72s**	4.47s	23.16s	~1% speedup
ant-4-20km-8node	**5.53s**	3.37s	9.99s	~1% speedup
ant-2-10km-4node	**18.47s**	9.61s	33.59s	~2% speedup

Preconditioner construction time dominates total solve time but linear solve is a large improvement over the all-Chebyshev case

Block Jacobi – Damping Factor

Lessons Learned:

- Total solve time has a major bottleneck in preconditioner construction that tuning can't solve
- Solve time depends primarily on block-Jacobi fine smoother, less so on Chebyshev coarse smoothers
- Tuning data is helpful to pass on to solver developers



Optimal damping factor is around 0.63

Looking ahead

Takeaways:

- Automated performance tuning is capable of producing good multigrid parameters for a given problem in relatively few runs of Albany Land-Ice
- Bayesian optimization consistently produces better parameters than random search but could be improved

Future work:

- Integrate with kokkos-tools to enable online tuning of MALI due to ice sheet instabilities over time
- Connect automated tuning framework with nightly performance testing to run tuning tasks when performance changepoints are detected
- Use **transfer learning** to leverage tuning results for small problems (such as Greenland ice sheet) to reduce cost of tuning large problems (such as Antarctica ice sheet)