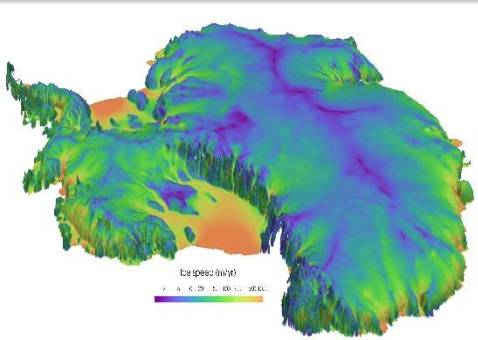
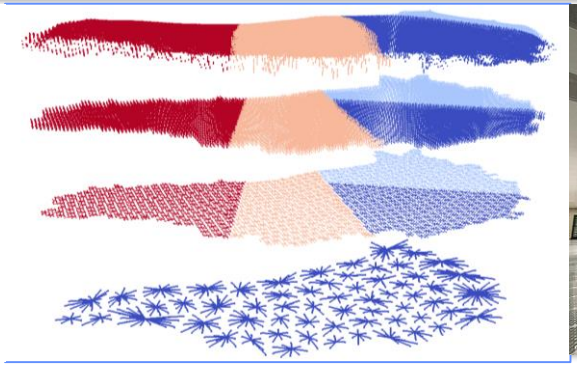
*Exceptional service in the national interest*

Sandia National Laboratories





# Towards Performance-Portability of the Albany Land-Ice Solver to New and Emerging Architectures Using Kokkos


Jerry Watkins[1], Irina Tezaur[1], Irina Demeshko[2]

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

[2] Los Alamos National Laboratory, Los Alamos, NM, USA.


ESCO 2018 Pilsen, Czech Republic June 4-8, 2018


U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-mission 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.

SAND2018-4907 C

# Outline

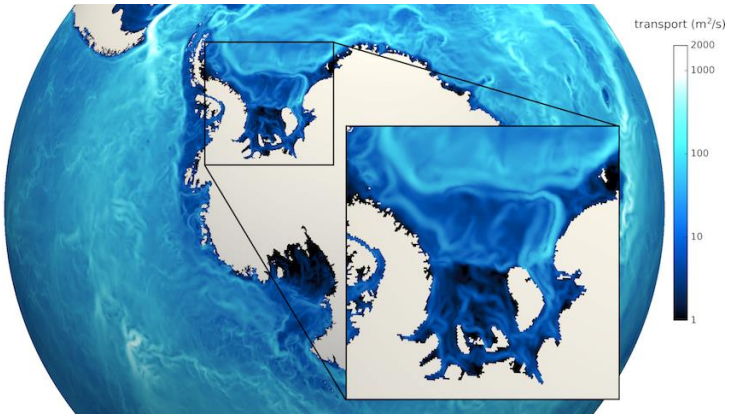
1. Background
   - Motivation
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. Performance-Portability of Finite Element Assembly via Kokkos
3. Performance Results
   - Architecture Comparison
   - Scaling Study
   - Preliminary Results on Volta GPU
4. Summary & Future Work.

# Outline

1. **Background**
   - **Motivation**
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. Performance-Portability of Finite Element Assembly via Kokkos
3. Performance Results
   - Architecture Comparison
   - Scaling Study
   - Preliminary Results on Volta GPU
4. Summary & Future Work.

# Motivation

- Scientific models (e.g., climate models) need more ***computational power*** to achieve ***higher resolutions***.
- High performance computing (HPC) architectures are becoming increasingly more ***heterogeneous*** in a move towards ***exascale***.
- Climate models need to adapt to execute ***correctly*** & ***efficiently*** on new HPC architectures with drastically ***different memory models***.

# MPI+X Programming Model

- HPC architectures are rapidly changing, but ***trends*** remain the same.
  - ***Computations*** are ***cheap***, ***memory transfer*** is ***expensive***.
  - ***Single core cycle*** time has improved but stagnated.
  - Increased ***computational power*** achieved through ***manycore architectures***.

→ MPI-only is not enough to exploit emerging massively parallel architectures.

**<u>Approach:</u>** MPI+X Programming Model

| Year | Memory Access Time | Single Core Cycle Time |
|---|---|---|
| 1980s | ~100 ns | ~100 ns |
| Today | ~50-100 ns | ~1 ns |

- MPI: inter-node parallelism.
- X: intra-node parallelism.

→ *Examples:* X = OpenMP, CUDA, Pthreads, etc.

# Outline

1. **Background**
   - Motivation
   - **ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver**
2. Performance-Portability of Finite Element Assembly via Kokkos
3. Performance Results
   - Architecture Comparison
   - Scaling Study
   - Preliminary Results on Volta GPU
4. Summary & Future Work.

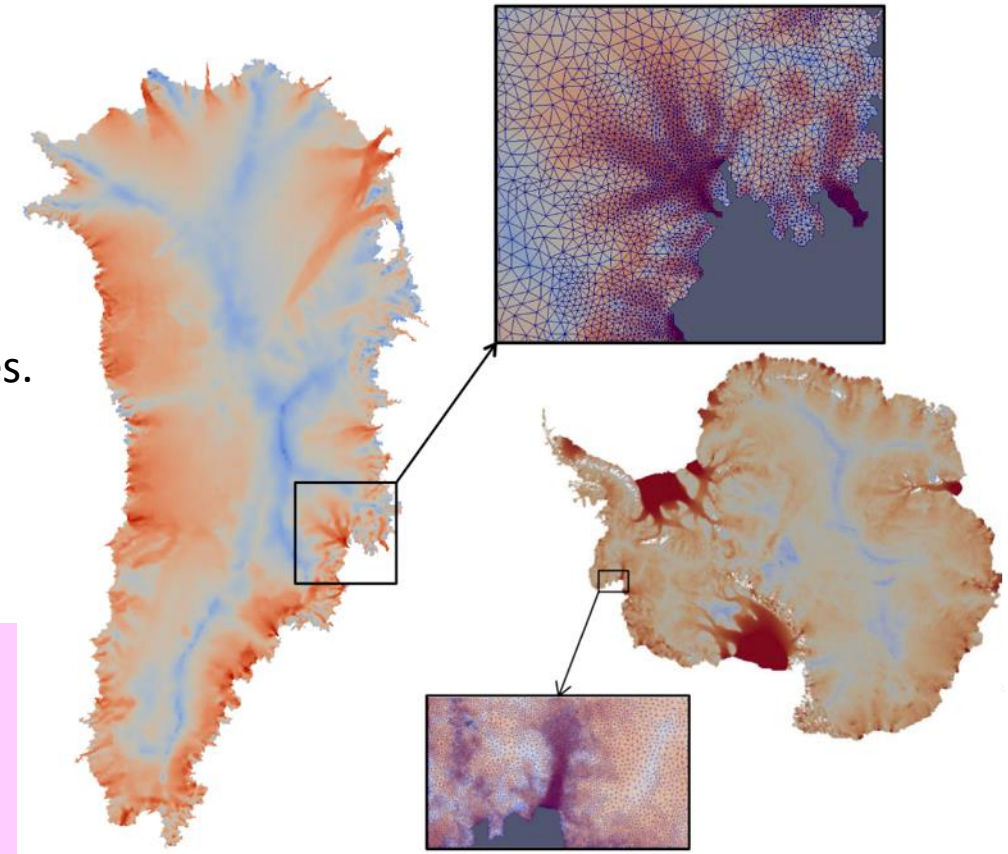
# ProSPect Project for Land-Ice Modeling

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

**Sandia's Role in the ProSPect Project:** to **develop** and **support** a robust and scalable land ice solver based on the "First-Order" (FO) Stokes equations → *Albany Land-Ice*

**Requirements for *Albany Land-Ice*:**

- ***Unstructured grid*** meshes.
- ***Scalable, fast*** and ***robust.***
- ***Verified*** and ***validated.***
- ***Portable*** to new architecture machines.
- ***Advanced analysis*** capabilities: deterministic inversion, calibration, uncertainty quantification.



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

* Energy Exascale Earth System Model.

# ProSPect Project for Land-Ice Modeling

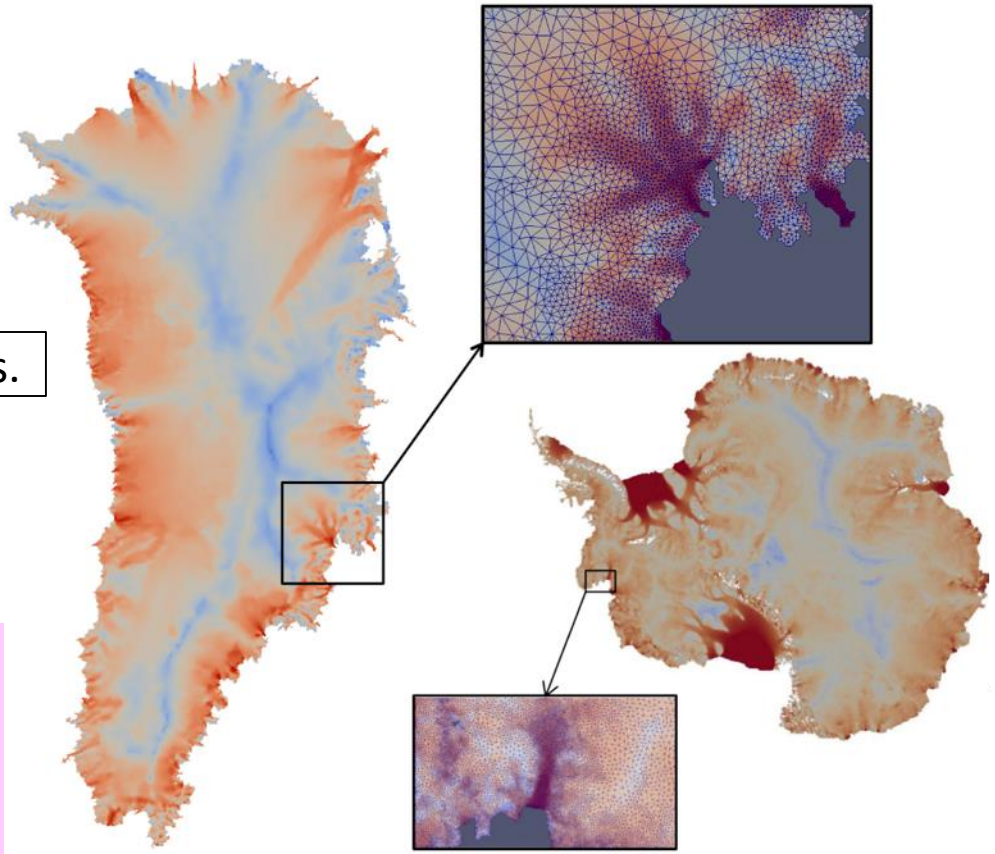
**"ProSPect"** = Probabilistic Sea Level Projections from Ice Sheet and Earth System Models
*5 year SciDAC4 project (2017-2022).*

**Sandia's Role in the ProSPect Project:** to **develop** and **support** a robust and scalable land ice solver based on the "First-Order" (FO) Stokes equations → *Albany Land-Ice*

**Requirements for *Albany Land-Ice*:**

- ***Unstructured grid*** meshes.
- ***Scalable, fast*** and ***robust.***
- ***Verified*** and ***validated.***
- ***Portable*** to new architecture machines.
- ***Advanced analysis*** capabilities: deterministic inversion, calibration, uncertainty quantification.

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



* Energy Exascale Earth System Model.

# First-Order (FO) Stokes Model

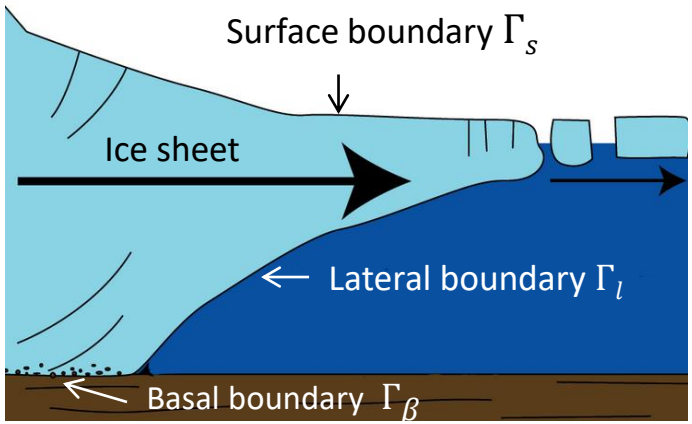
- Ice behaves like a very ***viscous shear-thinning fluid*** (similar to lava flow).
- ***Quasi-static*** model with ***momentum balance*** given by ***"First-Order" Stokes PDEs***: "nice" elliptic approximation* to Stokes' flow equations.

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

$$\dot{\boldsymbol{\epsilon}}_1{}^T = (2\dot{\epsilon}_{11} + \dot{\epsilon}_{22}, \dot{\epsilon}_{12}, \dot{\epsilon}_{13})$$
$$\dot{\boldsymbol{\epsilon}}_2{}^T = (2\dot{\epsilon}_{12}, \dot{\epsilon}_{11} + 2\dot{\epsilon}_{22}, \dot{\epsilon}_{23})$$
$$\dot{\epsilon}_{\mathrm{ij}} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

- Viscosity $\mu$ is nonlinear function given by "***Glen's law***":

$$\mu = \frac{1}{2} A(T)^{-\frac{1}{n}} \left(\frac{1}{2}\sum_{ij} \dot{\boldsymbol{\epsilon}}_{ij}{}^2\right)^{\left(\frac{1}{2n}-\frac{1}{2}\right)} \qquad (n = 3)$$



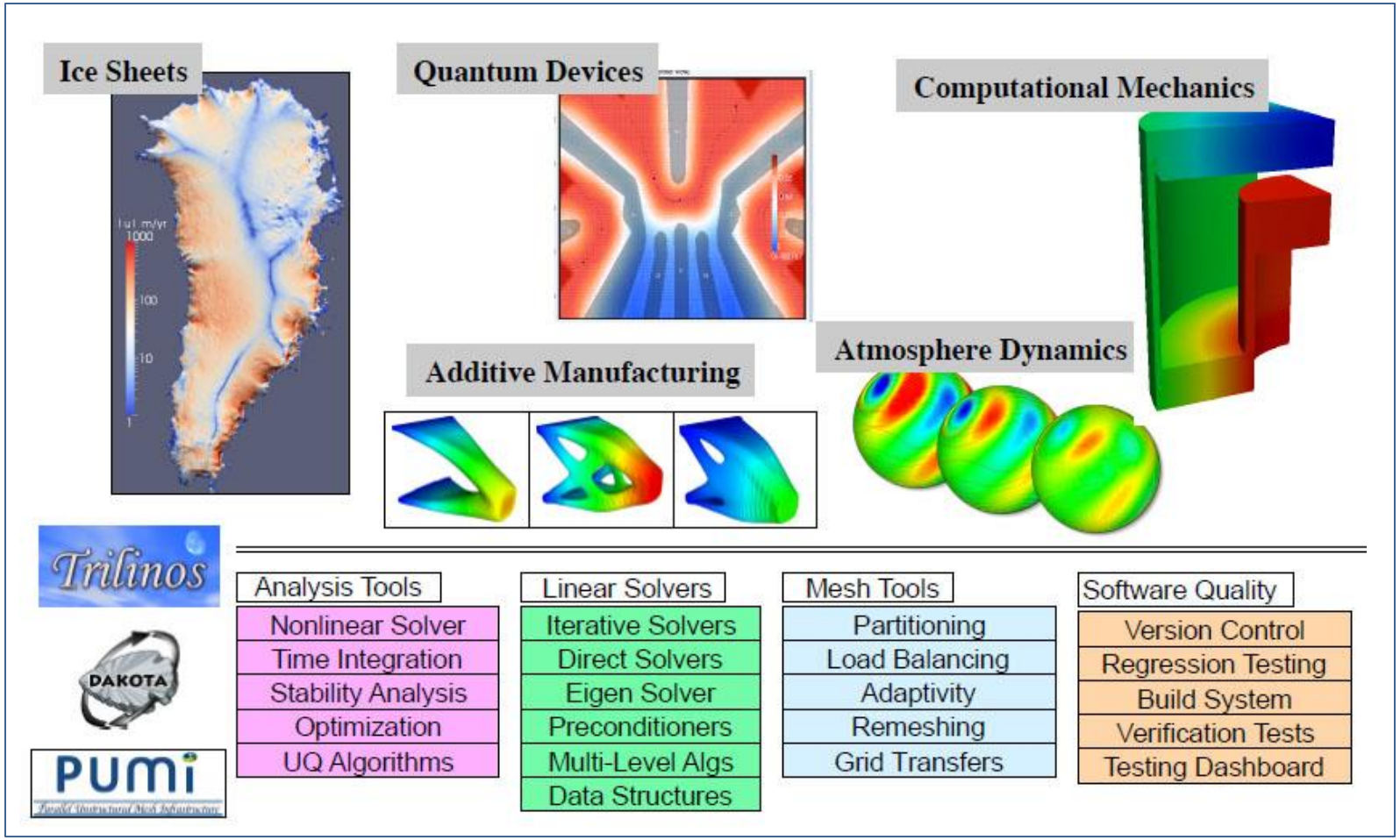

- Relevant boundary conditions:
  - ***Stress-free BC:*** $2\mu \dot{\boldsymbol{\epsilon}}_i \cdot \boldsymbol{n} = 0$, on $\Gamma_s$
  - ***Floating ice BC:*** $2\mu \dot{\boldsymbol{\epsilon}}_i \cdot \boldsymbol{n} = \begin{cases} \rho g z \boldsymbol{n}, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases}$, on $\Gamma_l$
  - ***Basal sliding BC:*** $2\mu \dot{\boldsymbol{\epsilon}}_i \cdot \boldsymbol{n} + \beta(x,y) u_i = 0$, on $\Gamma_\beta$

$\beta(x,y) =$ basal sliding coefficient

***Assumption:** aspect ratio $\delta$ is small and normals to upper/lower surfaces are almost vertical.

# Albany Code Base and Albany Land-Ice Solver

**Albany Land-Ice** implemented in open-source* C++ multi-physics finite element Trilinos-based code: Albany




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

# Outline

1. Background
   - Motivation
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. **Performance-Portability of Finite Element Assembly via Kokkos**
3. Performance Results
   - Architecture Comparison
   - Scaling Study
   - Preliminary Results on Volta GPU
4. Summary & Future Work.

# Performance-portability via *Kokkos*

We need to be able to run climate models on ***new architecture machines*** (hybrid systems) and ***manycore devices*** (multi-core CPU, NVIDIA GPU, Intel Xeon Phi, etc.) .

- In Albany Land-Ice, we achieve performance-portability via ***Kokkos***.
  - ***Kokkos****:* C++ library and programming model that provides performance portability across multiple computing architectures.

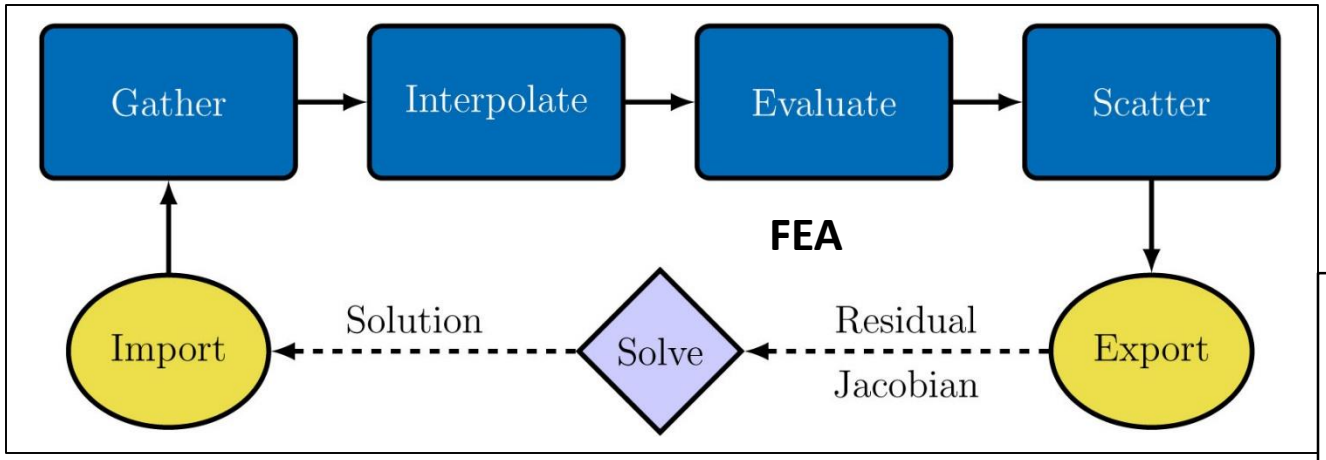
    → *Examples:* Multicore CPU, NVIDIA GPU, Intel Xeon Phi, and more.
  - Provides ***automatic access*** to OpenMP, CUDA, Pthreads, etc.
  - Designed to work with the ***MPI+X*** programming model.
  - Abstracts ***data layouts*** for optimal performance ("array of strucs" vs. struct of arrays", locality).

With ***Kokkos***, you write an algorithm once, and just change a template parameter to get the optimal data layout for your hardware.

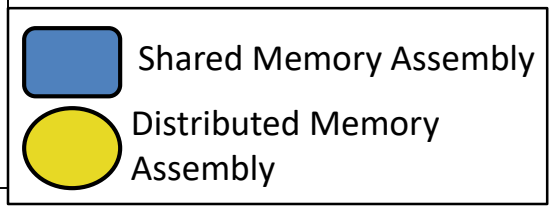
→ Allows researcher to focus on ***application development*** for large heterogeneous architectures.

# Albany Finite Element Assembly (FEA)

***Performance-portability*** work has focused on Finite Element Assembly (FEA).



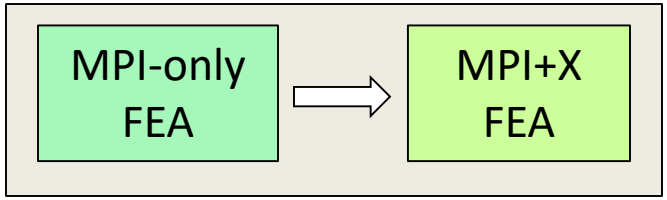

| Problem Type | % CPU time for FEA |
|---|---|
| Implicit | 50% |
| Explicit | 99% |




- ***Import***: imports global solution from nonoverlapping data structure → gives each rank access to relevant data for further communication.
- ***Gather:*** gathers solution values from overlapping data structure to element local data structure indexed to element/local node.
- ***Interpolate:*** interpolated solution/gradient from nodal to quadrature points.
- ***Evaluate:*** evaluates the residual, Jacobian, source terms (templated using AD*).
- ***Scatter:*** scatters residual/Jacobian values from element local to global data structures.
- ***Export:*** exports residual/Jacobian from overlapping to nonoverlapping data structure, where info is updated across MPI ranks.

* Automatic differentiation.

# Performance-portability of FEA via *Kokkos*





- **MPI-only FEA:**
  - Each MPI process has workset of cells, computes nested parallel for loops.
- **MPI+X FEA:**
  - Each MPI process has workset of cells.
  - Multi-dimensional parallelism with +X (X=OpenMP, CUDA) for nested parallel for loops.
- **Parallelism** over all **elements** (element local data structure)
  - Kokkos::parallel_for
- **Multidimensional parallelism** for nested **for loops**
  - Kokkos::Experimental::md_parallel_for
- **Atomics** used to scatter local data to global data structures
  - Kokkos::atomic_fetch_add (Tpetra)
- **Data transfer** from host to device handled by CUDA UVM*


* Unified Virtual Memory.

# MPI+X FEA via *Kokkos*

- ***MPI-only*** nested for loop:

```
for (int cell=0; cell<numCells; ++cell)
   for (int node=0; node<numNodes; ++node)
      for (int qp=0; qp<numQPs; ++qp)
         compute A;
```

MPI process *n*

# MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
   for (int node=0; node<numNodes; ++node)
      for (int qp=0; qp<numQPs; ++qp)
         compute A;
```

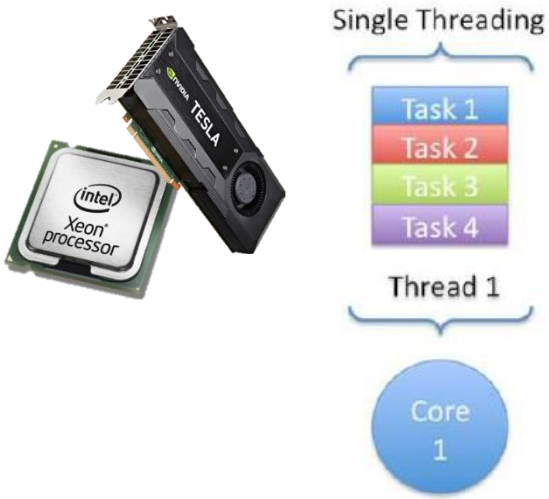
MPI process *n*

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)



# MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;
```

MPI process *n*

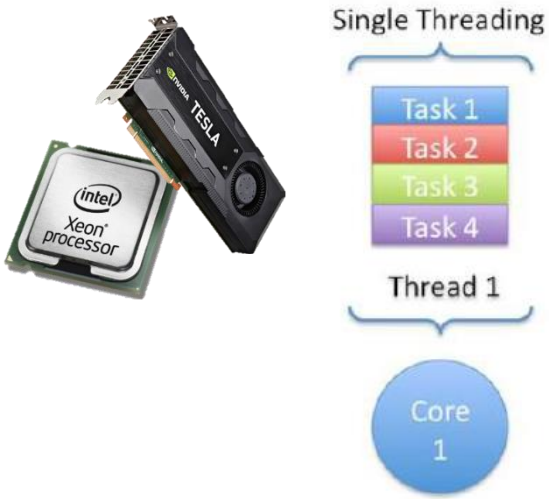
Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)

```
computeA_Policy range({0,0,0},{(int)numCells,(int)numNodes,(int)numQPs});
Kokkos::Experimental::md_parallel_for<ExecutionSpace>(range,*this);
```




* Unified Virtual Memory.

# MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;
```

MPI process *n*

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)

```
computeA_Policy range({0,0,0},{(int)numCells,(int)numNodes,(int)numQPs});
Kokkos::Experimental::md_parallel_for<ExecutionSpace>(range,*this);
```

- ExecutionSpace defined at ***compile time***, e.g.

```
typedef Kokkos::OpenMP ExecutionSpace; //MPI+OpenMP
typedef Kokkos::CUDA ExecutionSpace; //MPI+CUDA
typedef Kokkos::Serial ExecutionSpace; //MPI-only
```


# MPI+X FEA via *Kokkos*

- ***Multi-dimensional parallelism*** for nested for loops via *Kokkos:*

```
for (int cell=0; cell<numCells; ++cell)
  for (int node=0; node<numNodes; ++node)
    for (int qp=0; qp<numQPs; ++qp)
      compute A;
```

MPI process *n*

Thread *1* computes A for (cell,node,qp)=(0,0,0)

Thread *2* computes A for (cell,node,qp)=(0,0,1)

⋮

Thread *N* computes A for (cell,node,qp)=(numCells,numNodes,numQPs)

```
computeA_Policy range({0,0,0},{(int)numCells,(int)numNodes,(int)numQPs});
Kokkos::Experimental::md_parallel_for<ExecutionSpace>(range,*this);
```

- ExecutionSpace defined at ***compile time***, e.g.

```
typedef Kokkos::OpenMP ExecutionSpace; //MPI+OpenMP
typedef Kokkos::CUDA ExecutionSpace; //MPI+CUDA
typedef Kokkos::Serial ExecutionSpace; //MPI-only
```



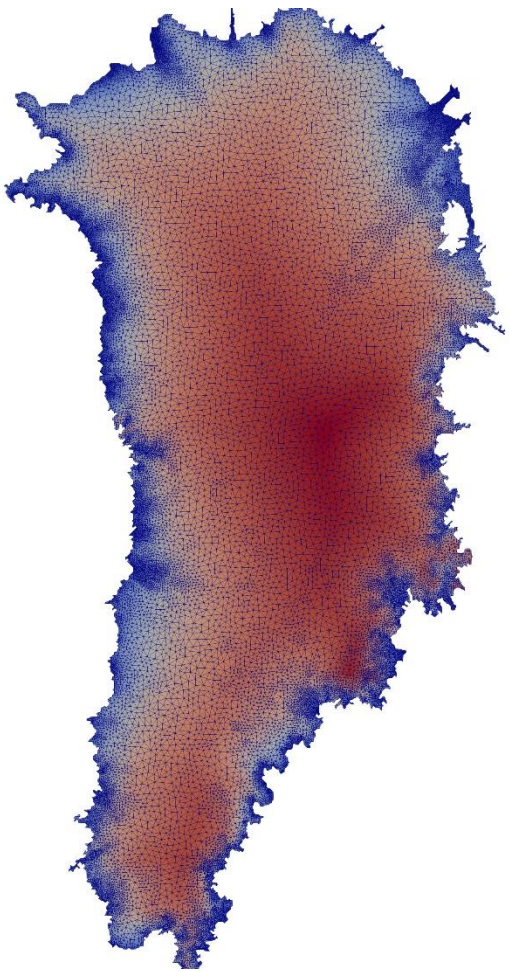
Kokkos parallelization in Albany Land-Ice is only over **cells**.

# Outline

1. Background
   - Motivation
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. Performance-Portability of Finite Element Assembly via Kokkos
3. **Performance Results**
   - Architecture Comparison
   - Scaling Study
   - Preliminary Results on Volta GPU
4. Summary & Future Work.

# Performance Study*: Greenland Ice Sheet (GIS)

| Mesh | Resolution | # Elements |
|---|---|---|
| Case 1 | 4km-20km | 1.51 million |
| Case 2 | 1km-7km | 14.4 million |



- Unstructured **tetrahedral** element meshes
- **Wall-clock time** averaged over 100 global assembly evaluations (residual + Jacobian)
- Performance analysis focuses on **finite element assembly**
- **No-slip boundary condition** at bedrock
- **3 devices considered**: Haswell, KNL, P100 GPU
- **Notation** for performance results:

$$r(\mathrm{MPI} + j\mathrm{X}), \qquad \mathrm{X} \in \{\mathrm{OMP}, \mathrm{GPU}\}$$

$r$ = # MPI ranks
$j$ = # OpenMP threads or GPUs/rank
$\mathrm{X}$ = architecture for shared memory parallelism

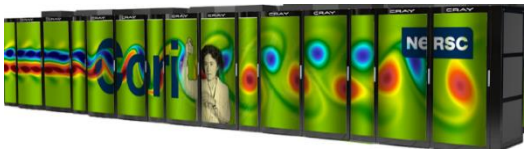
* See J. Watkins, I. Tezaur, I. Demeshko. *Lecture Notes in Computational Science & Engineering,* 2018 (accepted).

# Computer Architectures

Performance-portability of FEA in Albany has been tested across ***multiple architectures***: Intel Sandy Bridge, IBM POWER8, IBM POWER9, Keplar/Pascal/Volta GPUs, KNL Xeon Phi

## *Architectures:*

- **Ride** (SNL): 12 nodes [2 POWER8 (16 cores) + P100 (4 GPUs)]
- **Cori** (NERSC): 2,388 Haswell nodes [2 Haswell (32 cores)]
  9,688 KNL nodes [1 Xeon Phi KNL (68 cores)]
- **Quetzal** (SNL): Workstation [Titan V GV100 (1 GPU)]




## *Models:*

- **3 models tested**: MPI-only, MPI+OpenMP, MPI+CUDA
- **MPI+OpenMP**: MPI ranks are mapped to cores,
  OpenMP threads are mapped to hardware-threads*
  - Each Haswell core has 2 256-bit-wide vector processing units
  - Each KNL core has 2 512-bit-wide vector processing units
- **MPI+GPU**: MPI ranks assigned a single core per GPU
  - CUDA-Aware MPI (direct GPU communication)

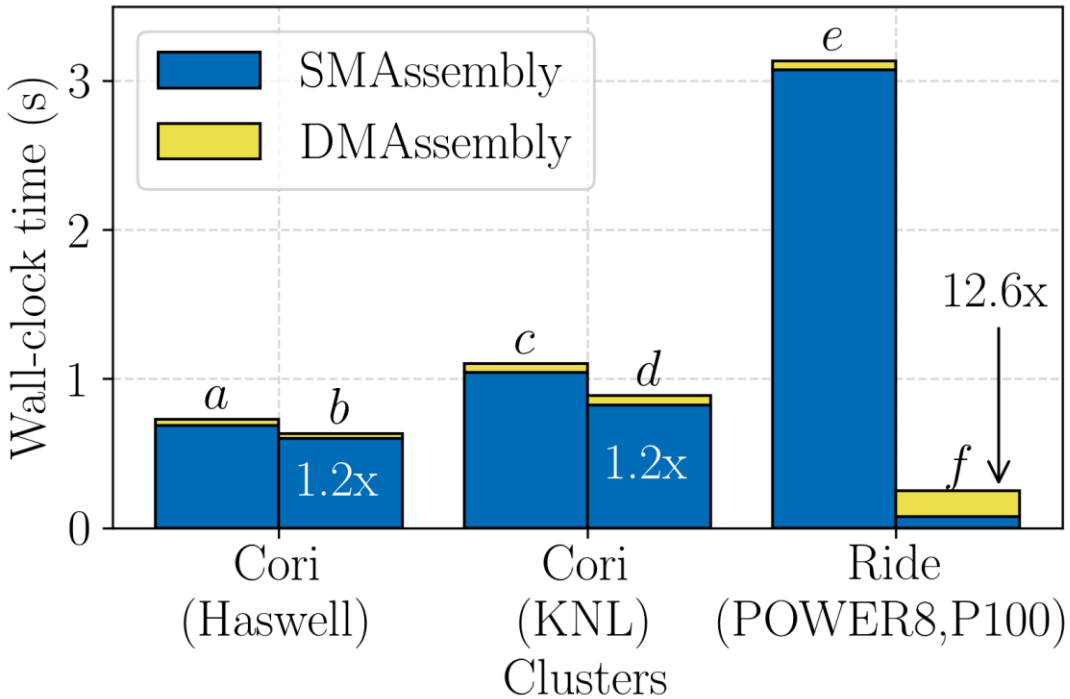

* Alternatives are sub-optimal, e.g., mapping MPI ranks to hardware threads ⇒ slower runs (ranks compete for resources).

# Outline

1. Background
   - Motivation
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. Performance-Portability of Finite Element Assembly via Kokkos
3. **Performance Results**
   - **Architecture Comparison**
   - Scaling Study
   - Preliminary Results on Volta GPU
4. Summary & Future Work.

# Wall-Clock Time on Single Node

Given access to a **single node** of Cori Haswell, Cori KNL, and Ride P100, where should 4km-20km GIS problem be run?




Node Configuration

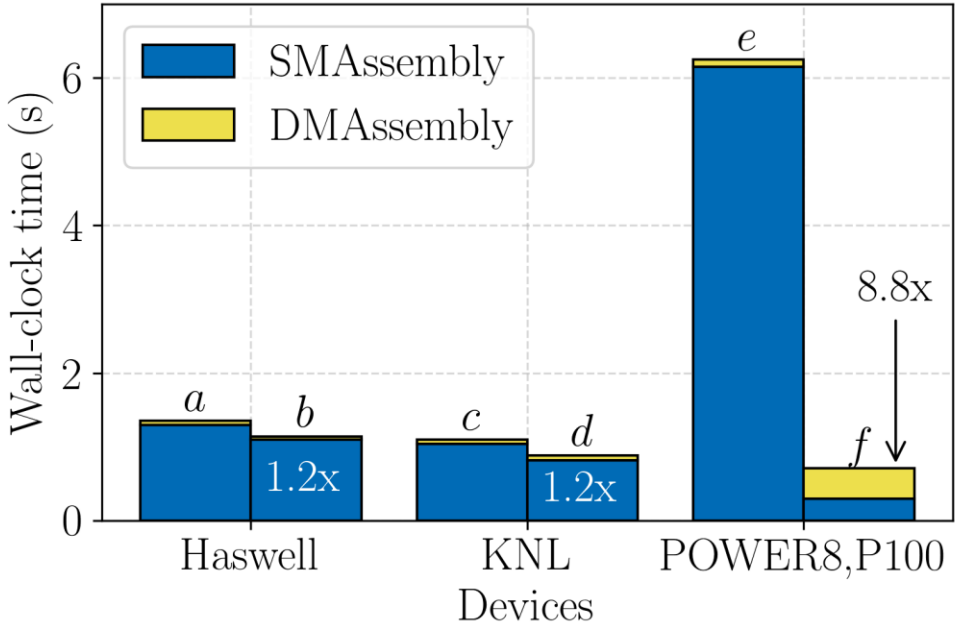
$a$: 32MPI
$b$: 32(MPI+2OMP)
$c$: 68MPI
$d$: 68(MPI+4OMP)
$e$: 16MPI
$f$: 4(MPI+GPU)

SM = Shared Memory
DM = Distributed Memory

- **Speedup** is achieved across all execution spaces.
  - **Ride** node performs best in this case (because of the 4 GPUs)
- **SMAssembly** dominates on CPU
  - ***Possible improvement***: explicit vectorization to utilize VPUs.
- **DMAssembly** is a factor on GPU
  - Tpetra routines (50% CPU time spent in Tpetra) are not fully optimized

# Wall-Clock Time on Single Device

What **single device** is best investment (Haswell, KNL, POWER8, P100) for running 4km-20km GIS problem?




Device Configuration

$a$: 16MPI
$b$: 16(MPI+2OMP)
$c$: 68MPI
$d$: 68(MPI+4OMP)
$e$: 8MPI
$f$: 1(MPI+GPU)

SM = Shared Memory
DM = Distributed Memory

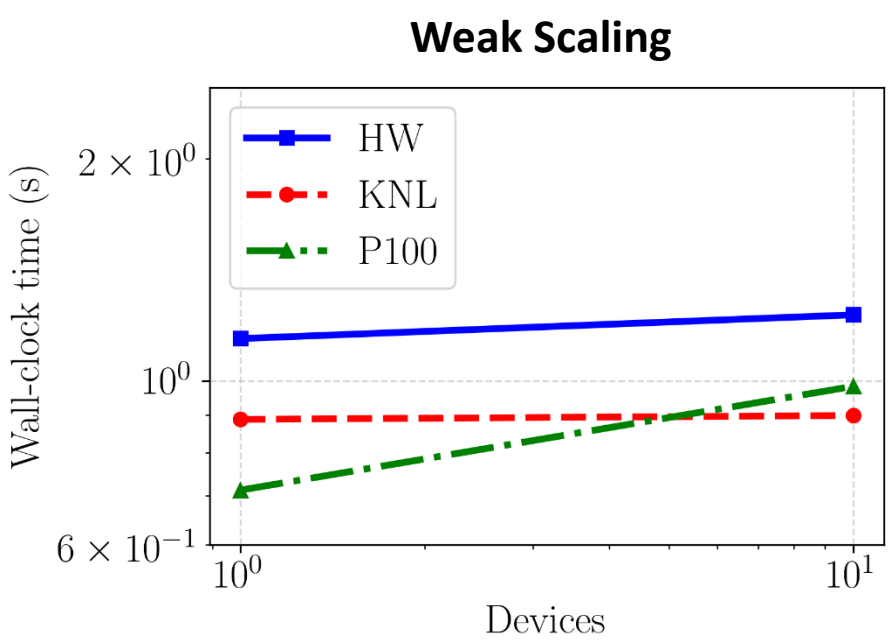
- **Speedup** is achieved across all execution spaces.
  - **P100 GPU** performs best → required some code optimizations (removal of unnecessary recomputations, data movement through memoizer).
- **SMAssembly** dominates on CPU
  - ***Possible improvement***: explicit vectorization to utilize VPUs.
- **DMAssembly** is a factor on GPU
  - Tpetra routines (50% CPU time spent in Tpetra) are not fully optimized

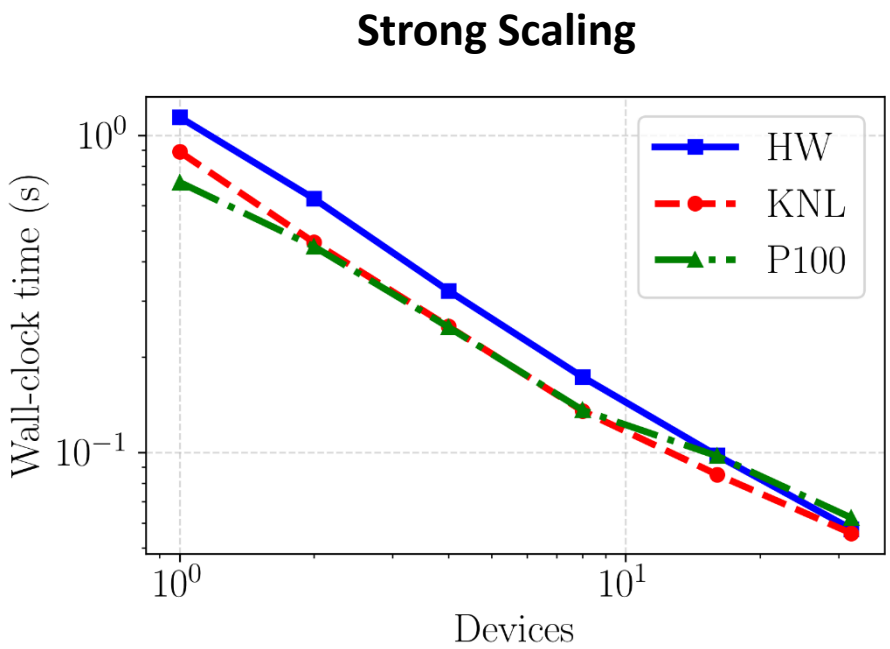
Similar conclusions as for single node.

# Outline

1. Background
   - Motivation
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. Performance-Portability of Finite Element Assembly via Kokkos
3. **Performance Results**
   - Architecture Comparison
   - **Scaling Study**
   - Preliminary Results on Volta GPU
4. Summary & Future Work.

# Scalability Study




- **Reasonable** strong and weak scaling are observed across all devices.
- **Weak scaling (left)**: 4km-20km GIS (1 device), 1km-7km GIS (10 devices).
  - Poor weak scaling on GPU is coming from Tpetra routines (DMAssembly), which are not fully optimized.
- **Strong scaling (right)**: 4km-20km GIS (1-32 devices).
  - Loss of scalability # devices ↑, for KNL and GPU as they are not saturated enough for strong scaling.

# Outline

1. Background
   - Motivation
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. Performance-Portability of Finite Element Assembly via Kokkos
3. **Performance Results**
   - Architecture Comparison
   - Scaling Study
   - **Preliminary Results on Volta GPU**
4. Summary & Future Work.

# Preliminary Results on Volta GPU

**1.51M Elements/GPU:**
bandwidth bound case

| GPUs | Wall Time (s) | Bandwidth (GB/s) | Speedup over previous |
|---|---|---|---|
| K80 | 1.24e-2 | 1.30e2 | – |
| P100 | 3.44e-3 | 4.70e2 | 3.6× |
| Titan V | 3.12e-3 | 5.18e2 | 1.10× |

**1961 Elements/GPU (768 GPUs):**
latency bound case

| GPUs | Wall Time (s) | Bandwidth (GB/s) | Speedup over previous |
|---|---|---|---|
| K80 | 6.22e-5 | 3.38e1 | – |
| P100 | 3.48e-5 | 6.04e1 | 1.78× |
| Titan V | 2.71e-5 | 7.78e1 | 1.29× |

- **Sample residual evaluator** using Kokkos assuming unstructured tets, **single GPU.**
- Results show improvements over **previous GPU generations.**
- Fully saturated bandwidth bound case (left) **expected to perform better** since Titan V has less bandwidth (652.8 GB/s) compared to V100 (900 GB/s) → 1.1× speedup due to increase in bandwidth.
- Speedup is attained on Titan V even in **latency bound strong scaling case** (right).

# Outline

1. Background
   - Motivation
   - ProSPect Project for Land-ice Modeling & Albany Land-Ice Solver
2. Performance-Portability of Finite Element Assembly via Kokkos
3. Performance Results
   - Architecture Comparison
   - Scaling Study
   - Preliminary Results on Volta GPU
4. **Summary & Future Work.**

# Summary & Future Work

Sandia National Laboratories

## *Summary*

- **Performance portability** is achieved across a wide variety of HPC architectures using a single code base through Kokkos
  - Multicore and manycore processors (POWER8, Haswell, KNL)
  - NVIDIA GPUs (P100, Titan V)
- We can use heterogeneous HPC architectures for **climate research** using Albany Land-Ice.
  - **Target:** Cori (Haswell, KNL), Aurora (new Xeon Phi), Summit (POWER9+V100)
- The **open-source** Albany multi-physics finite element code is available here:
  - https://github.com/gahansen/Albany
- Performance studies show that further **optimization** is needed to fully utilize resources

## *Future Work*

- Code **optimizations** for FEA:
  - Introduction of hierarchical parallelism, GEMMS.
  - Explicit vectorization on CPUs.
  - Multiple CUDA instances on GPUs for better node utilization.
  - Explicit data management to minimize memory transfers.
- Performance portability for **linear solvers** is an ongoing research topic within Trilinos.

# Funding/Acknowledgements

Support for this work was provided through Scientific Discovery through Advanced Computing (**SciDAC**) projects funded by the U.S. Department of Energy, Office of Science (**OSCR**), Advanced Scientific Computing Research and Biological and Environmental Research (**BER**) → **ProSPect SciDAC Application Partnership.**

***ProSPect team members:*** *K. Evans, M. Hoffman, C. Jackson, W. Lipscomb, M. Perego, S. Price, A. Salinger, I. Tezaur, R. Tuminaro, J. Bassis, G. Stadler, M. Eldred, J. Jakeman.*

***Computing resources:*** *NERSC, OLCF.*

# References


[1] M.A. Heroux *et al.* "An overview of the Trilinos project." *ACM Trans. Math. Softw.* **31**(3) (2005).

[2] A. Salinger, *et al.* "Albany: Using Agile Components to Develop a Flexible, Generic Multiphysics Analysis Code", *Int. J. Multiscale Comput. Engng*. 14(4) (2016) 415-438.

[3] **I. Tezaur**, M. Perego, A. Salinger, R. Tuminaro, S. Price. "*Albany/FELIX*: A Parallel, Scalable and Robust Finite Element Higher-Order Stokes Ice Sheet Solver Built for Advanced Analysis", *Geosci. Model Develop.* 8 (2015) 1-24.

[4] C. Edwards, C. Trott, D. Sunderland. "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns", *J. Par. & Distr. Comput.* **74** (12) (2014) 3202-3216.

[5] 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.* (2018) 1-23.

[6] J. Watkins, **I. Tezaur**, I. Demeshko. "A study on the performance portability of the finite element assembly process within the Albany land ice solver", *Lecture Notes in Computational Science and Engineering (accepted)*.

[7] 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 (2017) 255-270

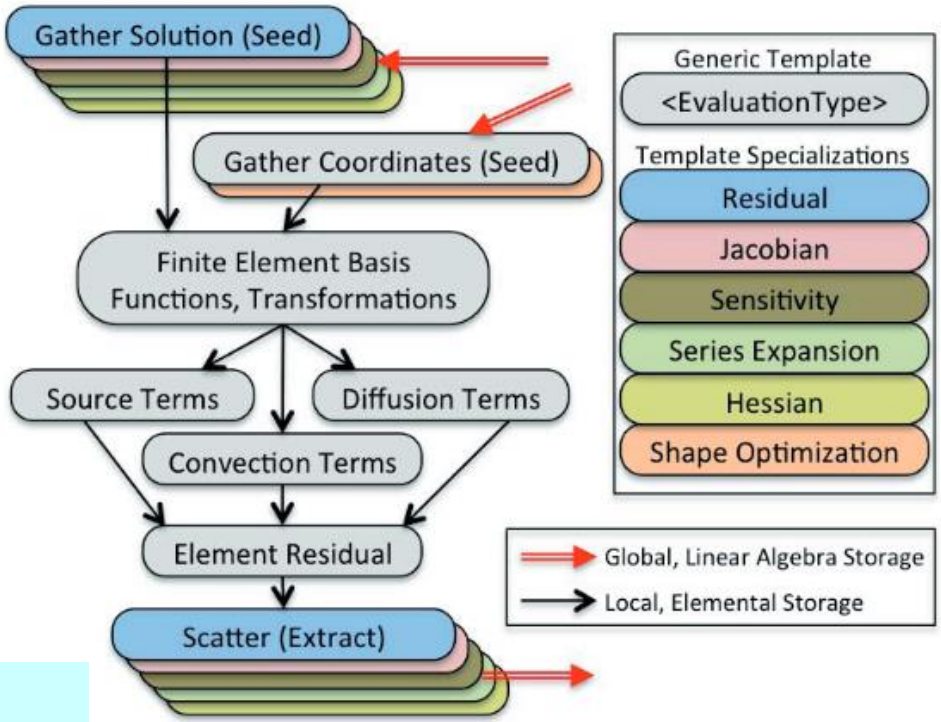
# Appendix: Parallelism on Modern Hardware

| Year | Memory Access Time | Single Core Cycle Time |
|---|---|---|
| 1980s | ~100 ns | ~100 ns |
| Today | ~50-100 ns | ~1 ns |

- *Memory access time* has remained the *same*.
- *Single core* performance has *improved* but *stagnated*.
- *Computations* are *cheap*, *memory transfer* is *expensive*.
- *More performance* from *multicore/manycore* processors.

# Appendix: Albany FEA

- ***Gather operation*** extracts solution values out of global solution vector.
- Physics ***evaluator*** functions operate on ***workset*** of elements, store evaluated quantities in local field arrays.
- FEA relies on ***template based generic programming*** + ***automatic differentiation*** for Jacobians, tangents, etc.
- ***Scatter operation*** adds local residual, Jacobian to global residual, Jacobian.




***Performance-portability***: focus on *FEA*.

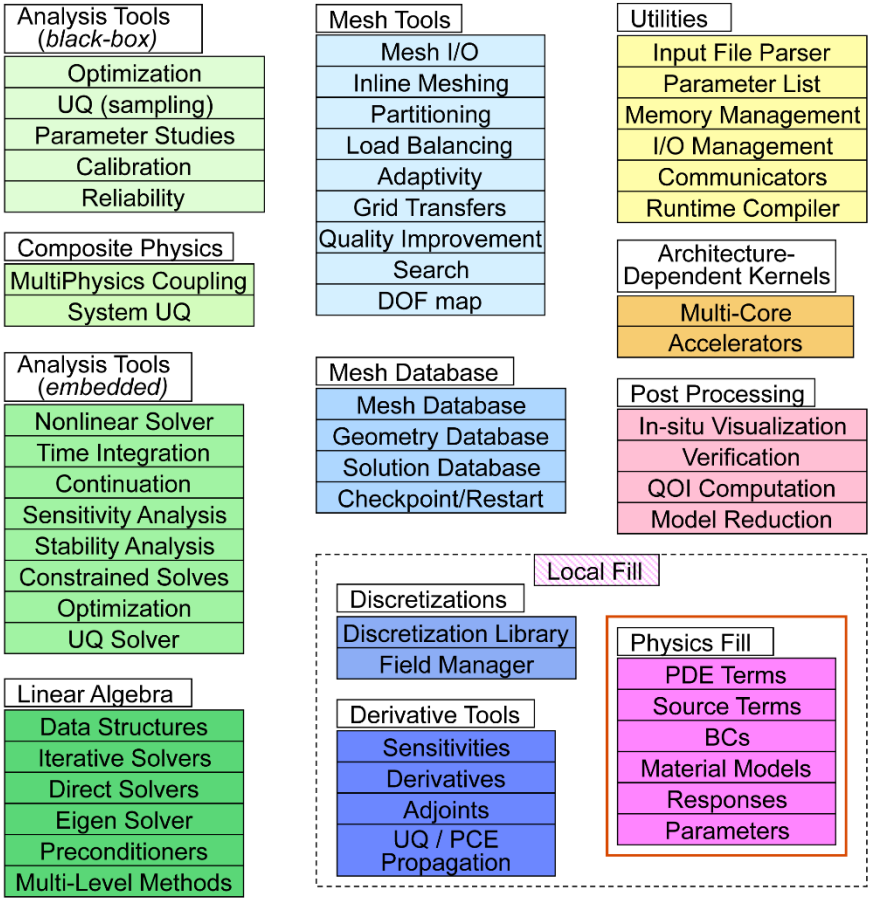
- ***MPI-only FEA:***
  - Each MPI process has workset of cells & computes nested parallel for loops.
- ***MPI+X FEA:***
  - Each MPI process has workset of cells.
  - Multi-dimensional parallelism with +X (X=OpenMP, CUDA) for nested parallel for loops.

| Problem Type | % CPU time for FEA |
|---|---|
| Implicit | 50% |
| Explicit | 99% |

# Appendix: Albany Multiphysics Code

FO-Stokes model is implemented within Albany, Sandia open-source* parallel, C++, multi-physics finite element code → ***Albany Land-Ice***.

- ***Component-based*** design for rapid development of new physics & capabilities.
- Extensive use of libraries from the open-source ***Trilinos*** project:
  - Automatic differentiation.
  - Discretizations/meshes, mesh adaptivity.
  - Solvers, preconditioners.
  - Performance-portable kernels.
- ***Advanced analysis*** capabilities:
  - Parameter estimation.
  - Uncertainty quantification (DAKOTA).
  - Optimization.
  - Sensitivity analysis.




40+ packages; 120+ libraries

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

# Appendix: First-Order (FO) Stokes Model

- Ice velocities given by the ***"First-Order" Stokes PDEs*** with nonlinear viscosity:

$$\begin{cases} -\nabla \cdot (2\mu\dot{\epsilon}_1) = -\rho g \dfrac{\partial s}{\partial x} \\ -\nabla \cdot (2\mu\dot{\epsilon}_2) = -\rho g \dfrac{\partial s}{\partial y} \end{cases}$$

$$\mu = \frac{1}{2} A^{-\frac{1}{n}} \left( \frac{1}{2} \sum_{ij} \dot{\epsilon}_{ij}{}^2 \right)^{\left(\frac{1}{2n} - \frac{1}{2}\right)}$$

$$\dot{\epsilon}_1{}^T = (2\dot{\epsilon}_{11} + \dot{\epsilon}_{22}, \dot{\epsilon}_{12}, \dot{\epsilon}_{13})$$
$$\dot{\epsilon}_2{}^T = (2\dot{\epsilon}_{12}, \dot{\epsilon}_{11} + 2\dot{\epsilon}_{22}, \dot{\epsilon}_{23})$$
$$\dot{\epsilon}_{ij} = \frac{1}{2}\left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

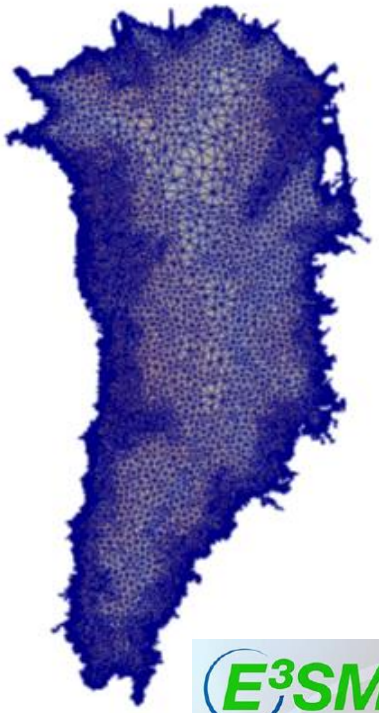
**<u>Algorithmic choices for Albany Land-Ice:</u>**

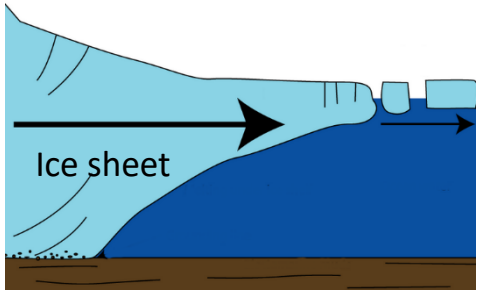
- ***3D unstructured grid*** FEM discretization.
- ***Newton method*** nonlinear solver with automatic differentiation Jacobians.
- Preconditioned ***Krylov iterative linear solvers***.
- ***Advanced analysis capabilities***: deterministic inversion, calibration, UQ.

***Implicit solver:***

FEA** = 50% CPU-time

Linear solve = 50% CPU-time






**Albany Land-Ice** implemented in open-source* multi-physics FE Trilinos-based code: Albany

* https://github.com/gahansen/Albany. **Finite Element Assembly

# Appendix: Ice Sheet Dynamic Equations

- Model for ***evolution of the boundaries*** (thickness evolution equation):

$$\frac{\partial H}{\partial t} = -\nabla \cdot (\bar{\boldsymbol{u}} H) + \dot{b}$$

where $\bar{\boldsymbol{u}}$ = vertically averaged velocity, $\dot{b}$ = surface mass balance (conservation of mass).

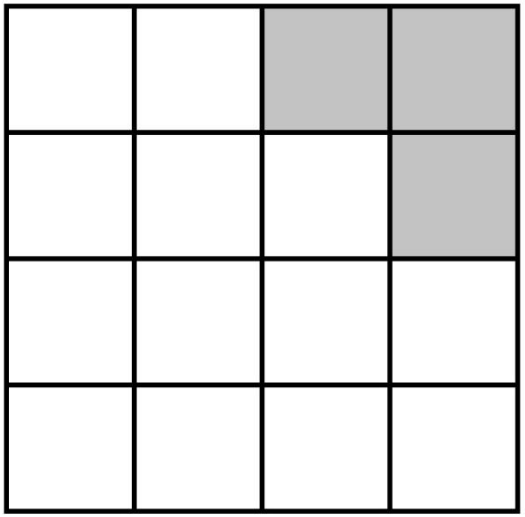
- ***Temperature equation*** (advection-diffusion):

$$\rho c \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) - \rho c \boldsymbol{u} \cdot \nabla T + 2 \dot{\boldsymbol{\epsilon}} \boldsymbol{\sigma}$$

(energy balance).

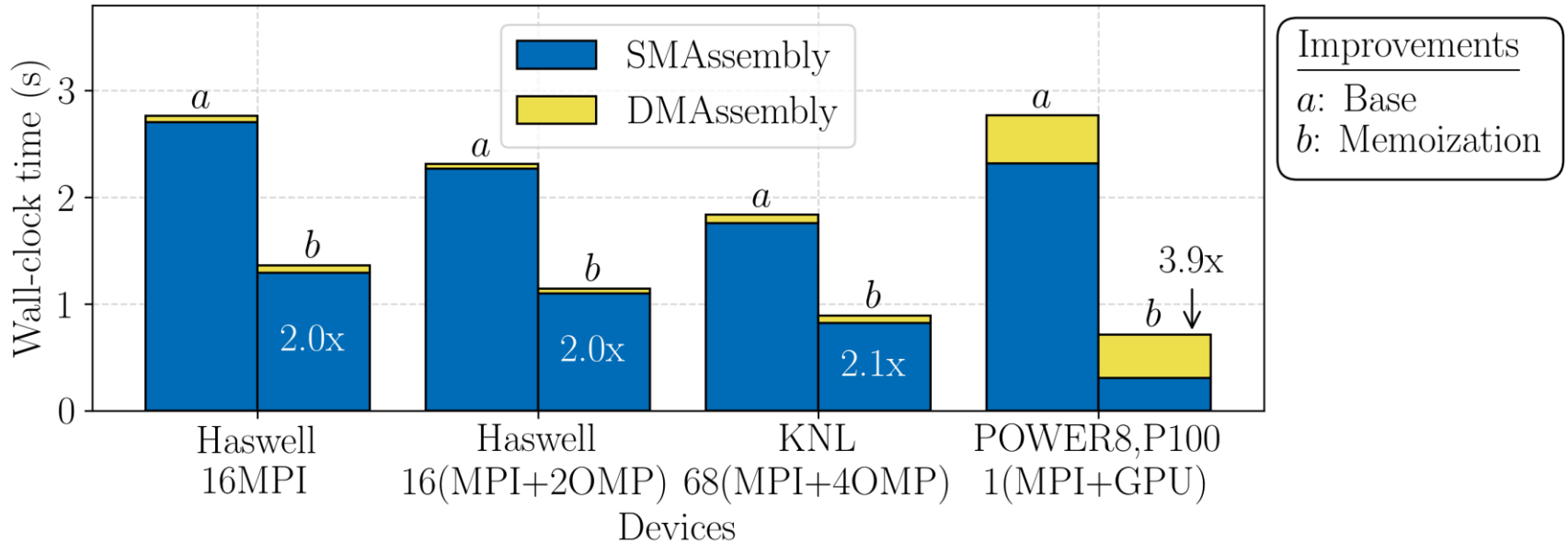
- ***Flow factor*** $A$ in Glen's law depends on temperature $T$: $A = A(T)$.

- Ice sheet ***grows/retreats*** depending on thickness $H$.



Ice-covered ("active") cells shaded in white ($H > H_{min}$)

# Appendix: Performance Improvements from Introduction of Memoizer




- Approximately 2-4× improvement on finite element assembly (FEA) using memorization.

# Single GPU Profile (4km-20km GIS)

- Profile includes **shared memory** local/global assembly (**assembly & computation**)
- Wall-clock time dominated by **Interpolation** and **ALI kernels**
- ***Future Work:***
  - Improve Interpolation and ALI kernels
  - Begin profiling solver performance