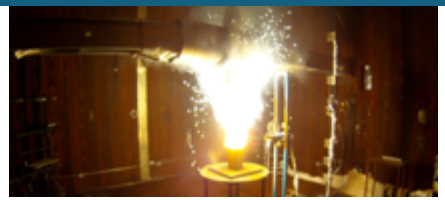
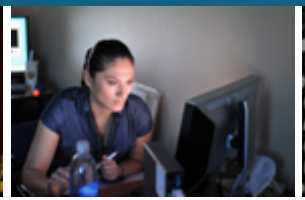
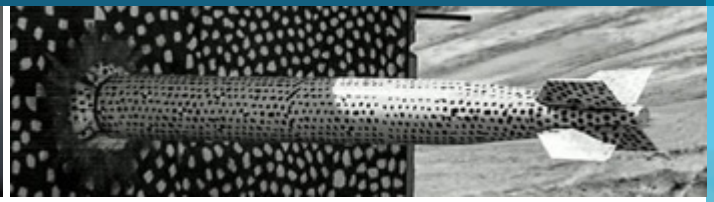
SAND2021-13348PE

Sandia National Laboratories

Enabling Science Simulations with Scalable Computational Frameworks for Scientific Computing







PRESENTED BY

Siva Rajamanickam

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY | NNSA

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.

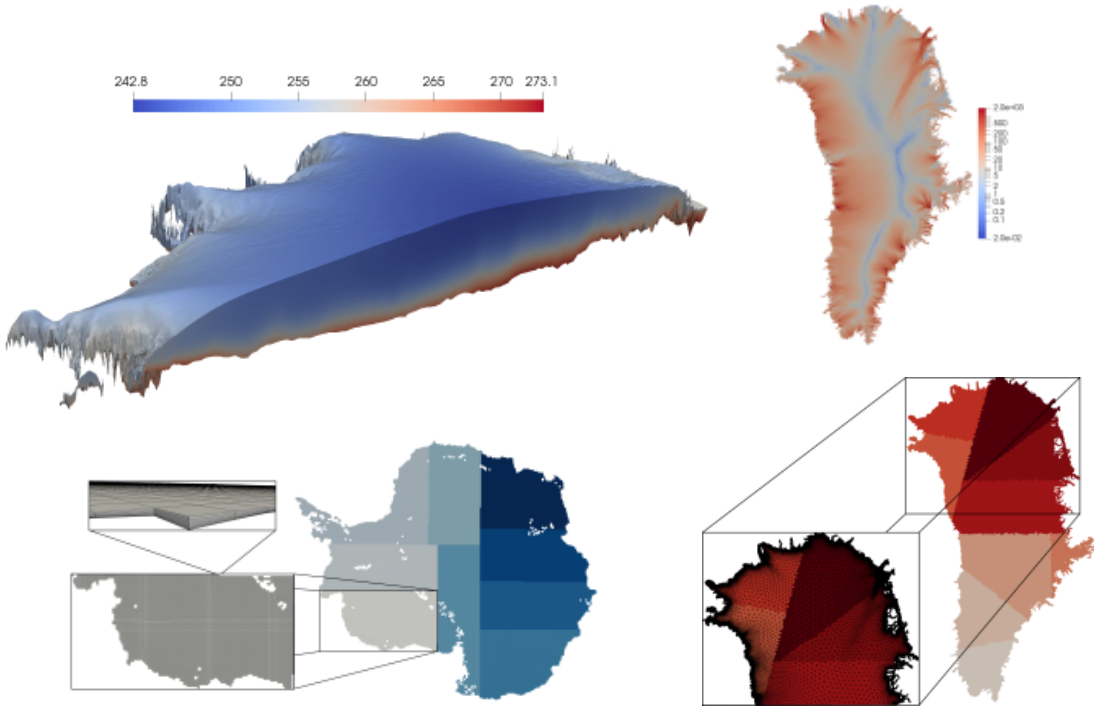
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

- Selected Modeling and Simulation Applications
- Foundational Computational Frameworks
  - Trilinos
  - Kokkos ecosystem
    - Kokkos Core
    - Kokkos Kernels
- Emerging Modeling and Simulation Frameworks
  - MALA : Materials Learning Algorithms

Selected Modeling and Simulation Applications

MALI: Land Ice Simulation of Greenland and Antarctica



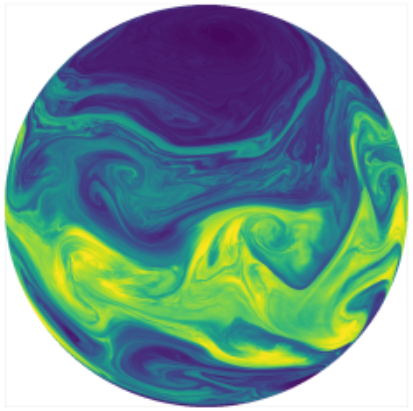
- Unique multigrid/domain decomposition solvers in Trilinos developed over the years

- Antarctic and Greenland ice sheets store most of fresh water on earth
- Mass loss of ice sheets significant source of sea level rise
- Simulation of temperature and velocity of ice sheet give rise to large highly nonlinear system of equations with a strong coupling of the variables
- Unique mesh structure, 2D mesh with extruded for the height dimension
- Unique features of computational frameworks used: Multigrid/ domain decomposition solvers (distributed), graph partitioning techniques, linear algebra data structures and kernels like sparse triangular solver, iterative solvers, preconditioners

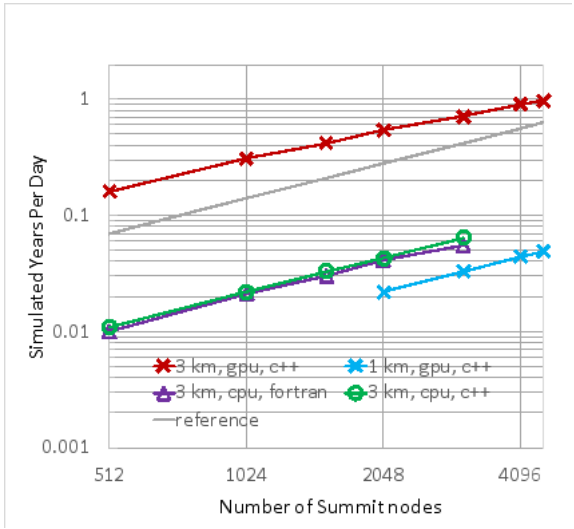
FROSch Preconditioners for Land Ice Simulations of Greenland and Antarctica, Heinlein et al.
"A matrix dependent/algebraic multigrid approach for extruded meshes with applications to ice she modeling.", Tuminaro et al. *SIAM Journal on Scientific Computing* 38, no. 5 (2016): C504-C532.

Trilinos POC: Mauro Perego, Ray Tuminaro, Siva Rajamanickam

HOMMEX-NH: E3SM nonhydrostatic atmosphere dynamical core



Turbulent eddies in water vapor on the 500 hPa model layer from a HOMME-NH 3 km horizontal resolution simulation of baroclinic instability.



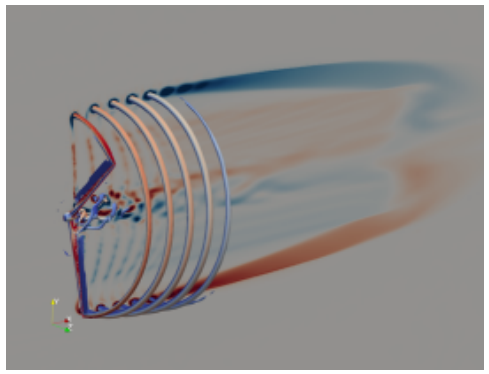
- Performance portable nonhydrostatic atmosphere dycore rewritten in C++ and Kokkos

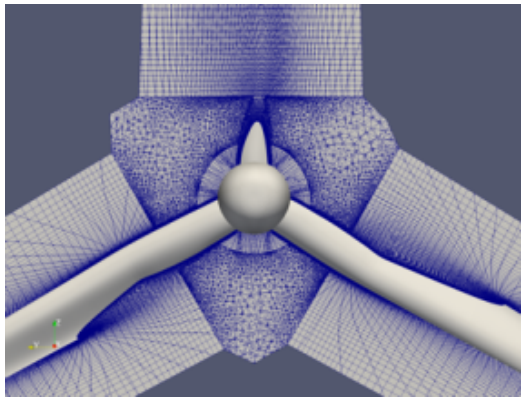
- Energy Exascale Earth System Model's nonhydrostatic atmosphere dynamical core
- Major source of uncertainty in climate models is from parameterized effects of convective cloud systems from 25 km resolutions
- Cloud resolving simulations of 3km and 1 km needed (7.2 billion / 65 billion grid points, 16 unknowns per grid point)
- Solve the the fully compressible Navier-Stokes equations
- Result: 0.97 simulated years per day on fill Summit system, 27600 GPUs
- Unique mesh structure, 2D mesh with extruded for the height dimension
- Unique features of computational frameworks used: Kokkos for performance portability

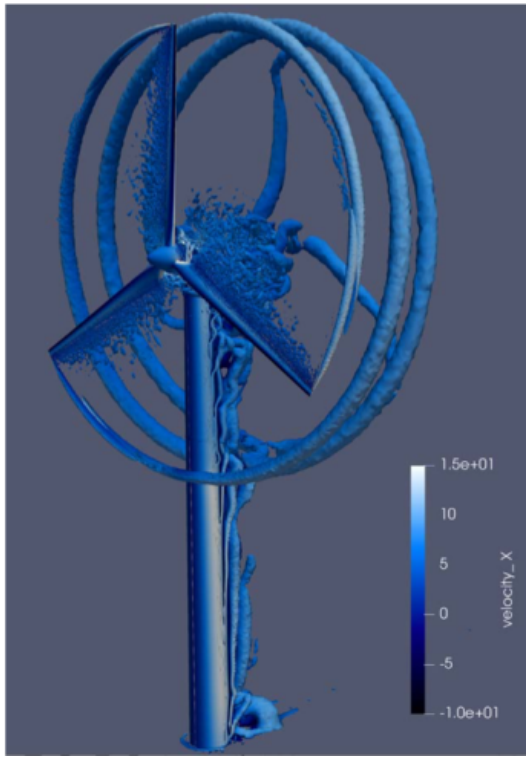
POC: Luca Bertagna, Mark Taylor

Bertagna, Luca, Oksana Guba, Mark A. Taylor, James G. Foucar, Jeff Larkin, Andrew M. Bradley, Sivasankaran Rajamanickam, and Andrew G. Salinger. "A Performance-Portable Nonhydrostatic Atmospheric Dycore for the Energy Exascale Earth System Model Running at Cloud-Resolving Resolutions." In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1-14. IEEE, 2020.

Exawind: Wind plant performance simulation







- Simulate the physics of entire wind power plants.
- Improve wind turbine and plant structures to advance performance.
- Nalu-Wind fluids code uses Trilinos
  - Based of Nalu code used for fire simulations (Lin et al.)
- Evaluate the energy output and the structural loading on wind turbines by modeling the incoming turbulent wind field and the evolution of turbine wakes and their interaction with the downstream turbines.
- Unique features of computational frameworks used: Multigrid/ one level domain decomposition, iterative solvers (distributed), linear algebra data structures and kernels like Gauss-Seidel preconditioner, Kokkos ecosystem for performance portability

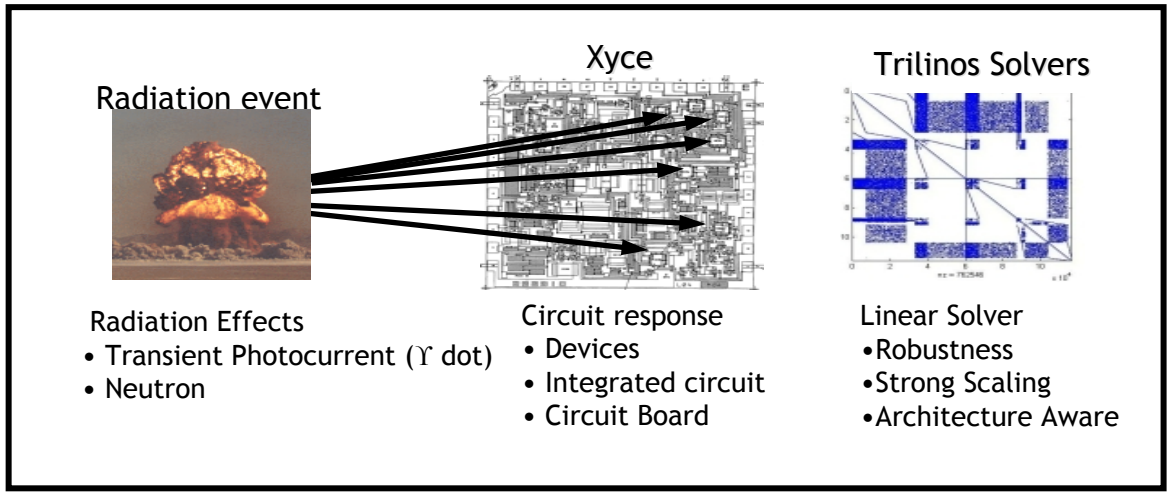
- Performance portable nonhydrostatic atmosphere dycore rewritten in C++ and Kokkos

Trilinos POC: Jonathan Hu

Read more: "**ExaWind: A multifidelity modeling and simulation environment for wind energy**" M Sprague, S Ananthan, G Vijayakumar and M Robinson, 2020 *J. Phys.: Conf. Ser.* **1452** 012071
Lin, Paul, et al. "**Towards extreme-scale simulations for low Mach fluids with second-generation Trilinos.**" *Parallel processing letters* 24.04 (2014): 1442005.

Xyce: Parallel Circuit Simulator

Xyce application drivers: stockpile stewardship, RF design, microwave/satellite apps

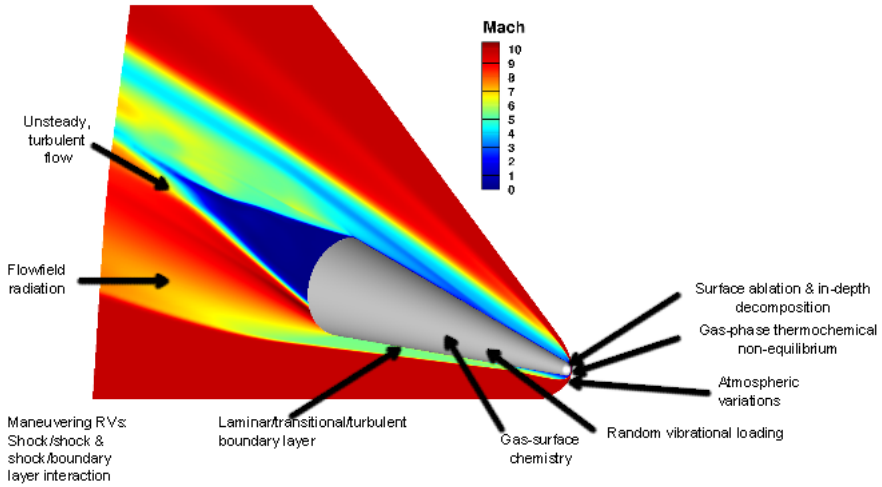


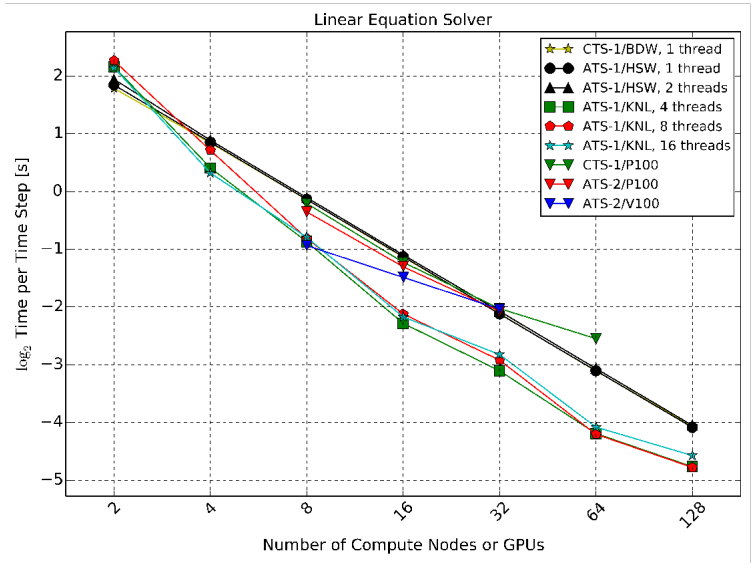
- Unique direct/hybrid solver capabilities in Trilinos developed over the years that support DOE requirements for robustness and scalability

Trilinos POC: Siva Rajamanickam; Heidi Thornquist

- Verify design of electrical components prior to manufacture and deployment.
- Xyce: Analog time domain and frequency domain parallel circuit simulation software.
- Application: Predict the reliability of weapon system components when exposed to hostile radiation and electromagnetic environments.
- Simulate effects of ionizing radiation in transistors and circuits. This radiation could cause the circuit to become unpowered.
- Unique Features of Trilinos used: Sparse hybrid/direct solvers (distributed, multithreaded), graph partitioning/ordering techniques, linear algebra data structures and kernels, iterative solvers, preconditioners

SPARC: Sandia Parallel Aerodynamics Reentry Code





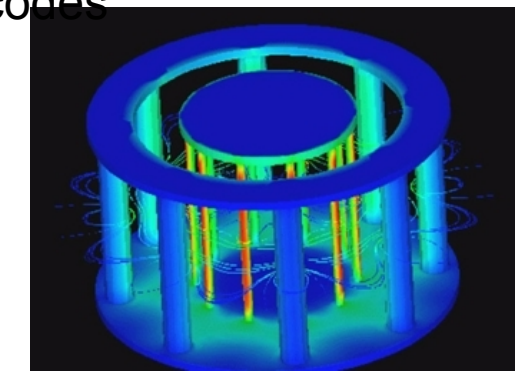
- Hypersonic speeds result in intense heating environment
- Thermal protection needed to protect internal components
- Need to simulate aerodynamics and material thermal response
- Hybrid structured-unstructured finite volume code
- Perfect and thermo-chemical non equilibrium gas models
- Unique Features of Frameworks used: Sparse tridiagonal solvers, linear algebra data structures and kernels, preconditioners, Kokkos ecosystem for portability

Kokkos Kernels + Ifpack2 provides nearly perfect scalability for Intel KNL and GPU platforms

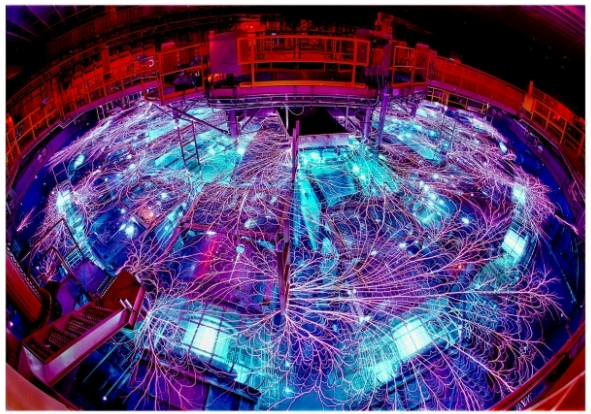
Trilinos POC: Siva Rajamanickam, Jonathan Hu

More applications …

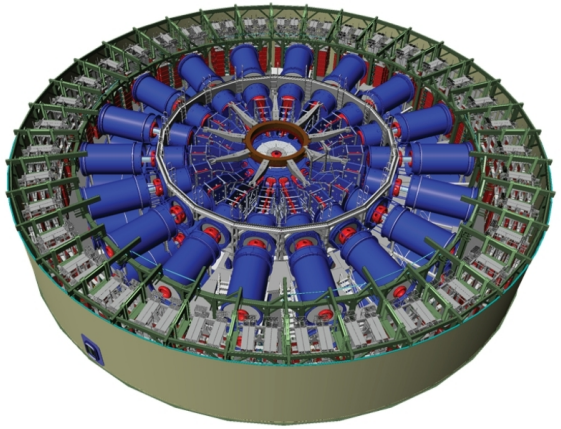
ALEGRA – Shock and multphysics codes



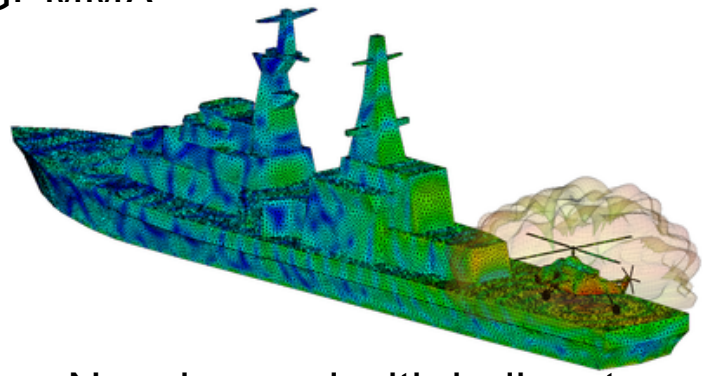
Magnetic field strength in z-pinch array



ElectroMagnetic Plasma In Realistic Environments (or EMPIRE) is a modeling and design tool for plasma environments.



Massively Parallel Frequency Domain Electromagnetic Simulation Codes : Eiger and GEMMA



Naval vessel with helicopter on de
(Source: FEKO)

. The Boundary Element Method (BEM) version of the Method of Moments (MoM) for solving integral equations results in dense linear systems.

Framework for Engineering Mechanics

Computational Frameworks: Trilinos

Trilinos: Open-Source Toolkit of Mathematical Algorithms for HPC

Trilinos Software

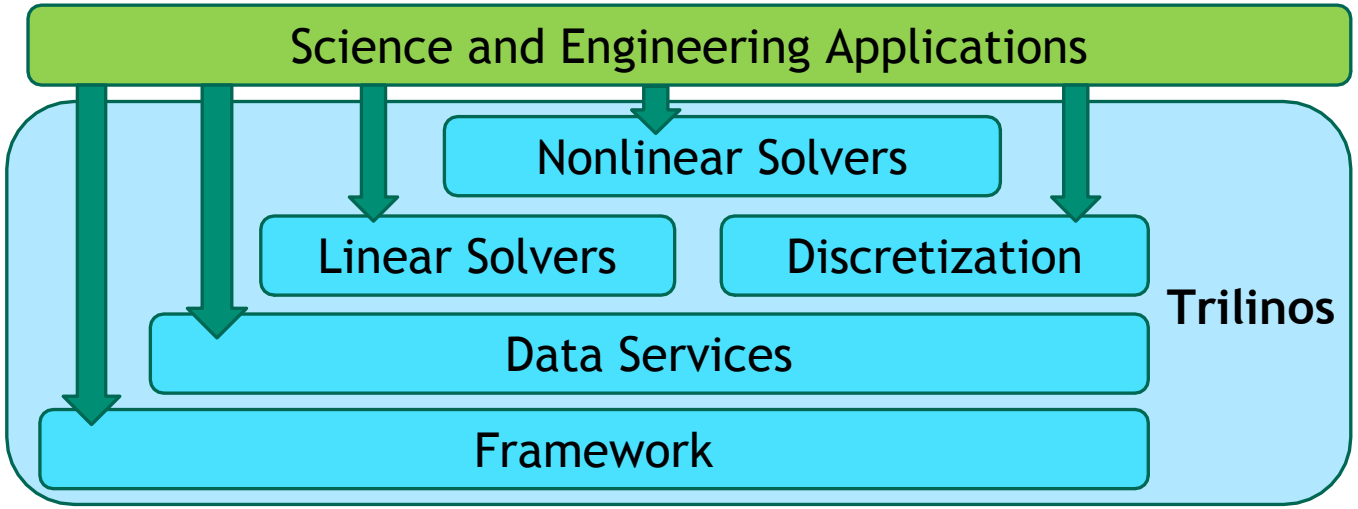
55 packages in five areas

~100 contributors in total

~50+ active contributors

30-140 commits per week

400 forks



Application Impact

- Solid mechanics, fluid dynamics, electrical circuits, etc.
- SIERRA, Empire, SPARC, Xyce, Drekar, Charon, etc.

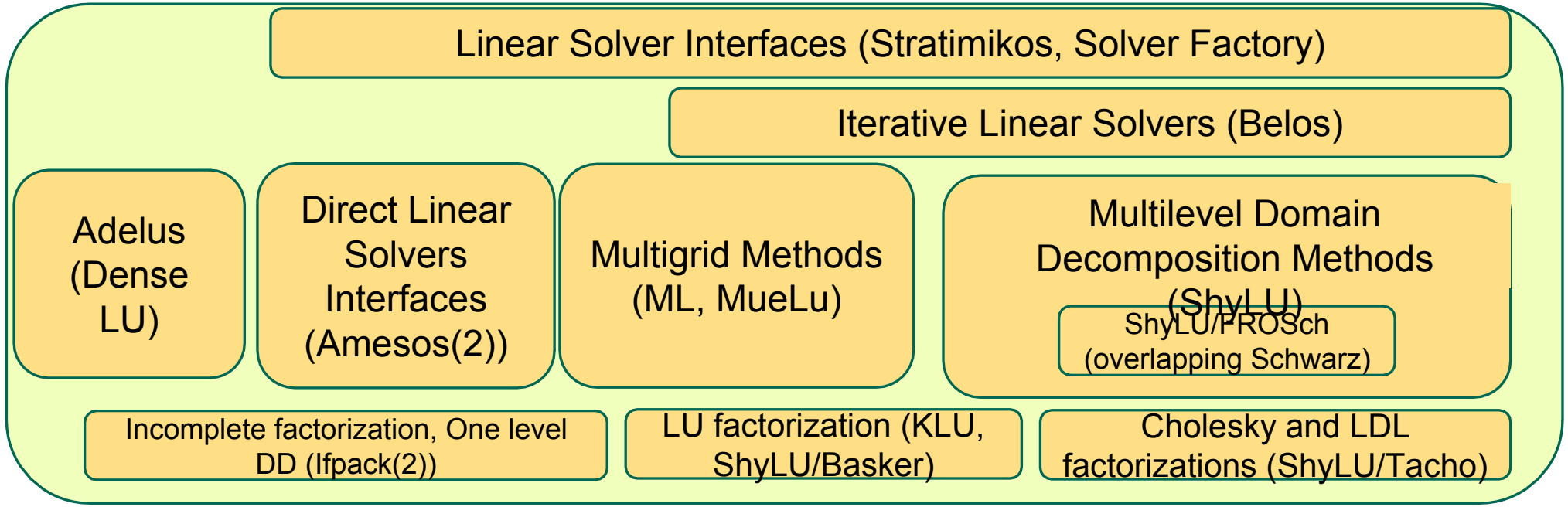
Two main codes paths:
- 32-bit stack (maintenance)
- Templated C++ stack (active)

Trilinos product areas (Lead : Heroux)

- **Framework** – Build, install, and test infrastructure; application integration (Product Lead: Willenbring)
- **Data Services** – Linear algebra, **Kokkos** performance-portability, **load balancing**, mesh services (Product Lead: Devine)
- **Linear Solvers** – Iterative/direct solvers, preconditioners (domain-decomposition, multigrid, block) (Product Lead: Rajamanickam)
- **Nonlinear Solvers** – Time-stepping methods, non-linear solvers (Product Lead: Pawlowski)
- **Discre**

Trilinos provides scalable algorithms to ASC-IC/ATDM applications, enabling high performance on current and next generation HPC platforms

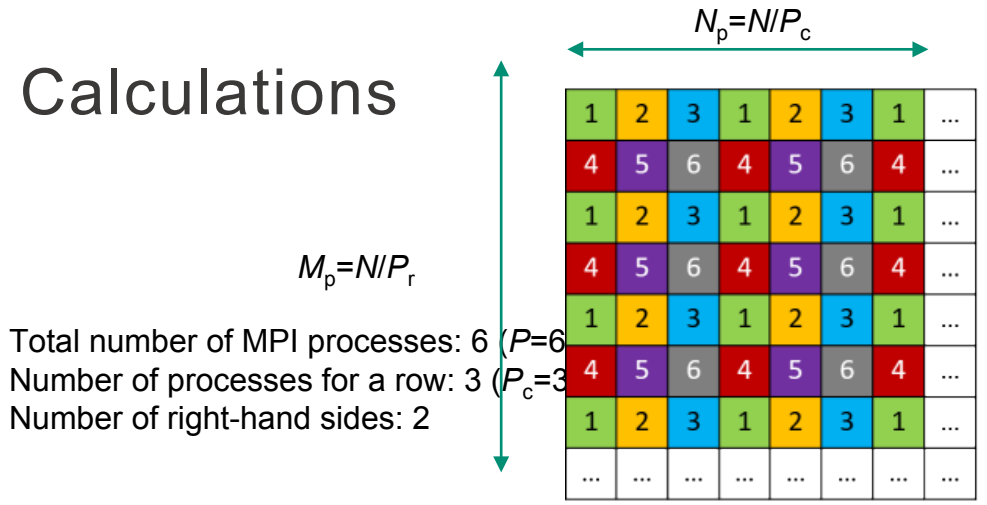
Trilinos Solvers Overview



- Comprehensive suite of solvers that covers entire spectrum of solver needs
  - **Direct linear solvers** for highly ill-conditioned, unsymmetric, problems (Xyce)
  - **Dense LU solver** for distributed memory complex LU (Eiger/GEMMA)
  - Schur complement solvers for distributed-memory direct solvers (Xyce)
  - Iterative linear solvers that cover broad suite of applications (SIERRA, Xyce, Charon, EMPIRE, SPARC)
  - Multigrid methods for broad suite of applications (Fluid dynamics, Charon, EMPIRE, SPARC)
  - Multilevel domain decomposition methods (SIERRA-SD, SIERRA-SM)
  - Incomplete factorization preconditioners (Smoother for multigrid methods)

Adelus : Dense LU Solver for EM Calculations

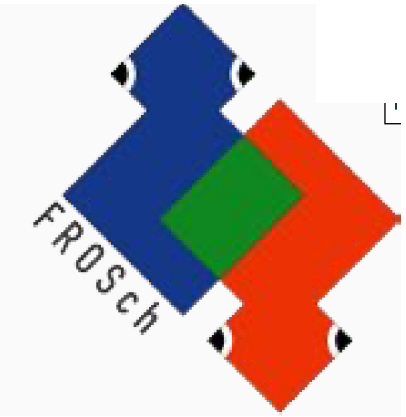
- A distributed-memory, dense LU solver capable of utilizing hardware accelerators available on top supercomputers is in need
- Complex linear systems
- Torus-Wrap Mapping Advantage: each process has nearly the same workload and the process idle time is minimized
- 7.7 Pflops on the Sierra system on full application run



N	Nodes (GPUs)	Solve time (sec.)	TFLOPS	Procs/row
226,647	25 (100)	240.5	1291.0	10
1,065,761	310 (1240)	1905.1	1694.5	31
1,322,920	500 (2,000)	6443.9	958.1	20
1,322,920	500 (2,000)	2300.2	2684.1	50
1,322,920	500 (2,000)	2063.6	2991.9	100
2,002,566	1,200 (4,800)	3544.1	6042.6	100
2,564,487	1,900 (7,600)	5825.2	7720.7	80

7.7 Pflops on 7600 GPUs. Still state of the art.

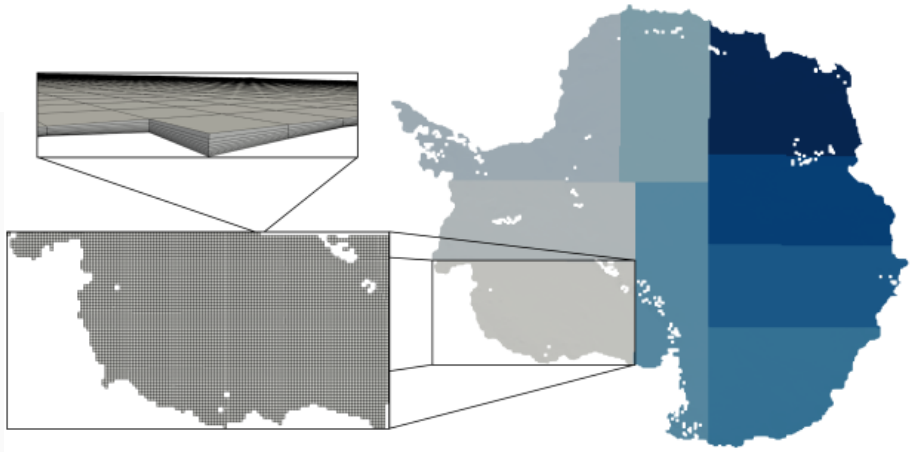
FROSch: Multilevel Domain Decomposition solver



- **Schwarz preconditioners** with **algebraic coarse spaces** based on extension operators, e.g., **GDSW** (Generalized–Dryja–Smith–Widlund) coarse spaces
  - **Algebraic** and **scalable**
- Part of the package ShyLU
- Scalability up to 32,000 cores on Cori system

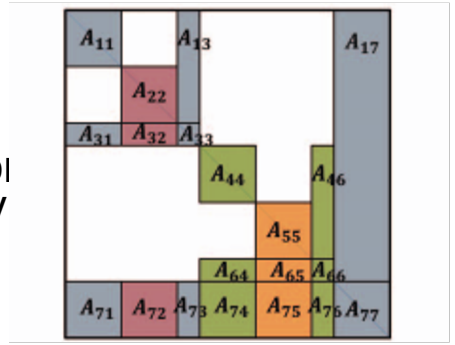
MPI ranks	mesh	# dofs	1 OpenMP thread			4 OpenMP threads		
			avg. its (nl its)	avg. setup	avg. solve	avg. its (nl its)	avg. setup	avg. solve
32	16 km	2.2 m	24.1 (11)	11.97 s	9.47 s	23.5 (11)	4.15 s	3.25 s
128	8 km	8.8 m	32.0 (10)	14.08 s	8.71 s	32.0 (10)	4.97 s	2.85 s
512	4 km	35.3 m	42.6 (11)	14.99 s	12.50 s	42.6 (11)	5.50 s	4.02 s
2048	2 km	141.5 m	61.0 (11)	22.83 s	19.76 s	61.0 (11)	7.36 s	6.55 s
8192	1 km	566.1 m	67.1 (14)	17.36 s	22.91 s	67.1 (14)	6.20 s	7.39 s

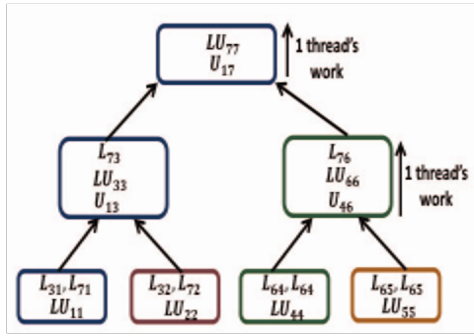
Problem: Velocity **Mesh:** Antarctica, 20 vert. layers **Coarse space:** RGDSW



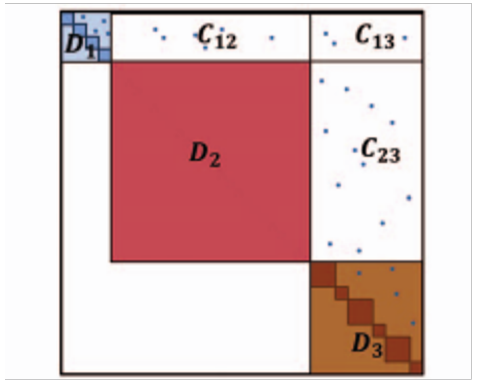
Sparse Linear Solvers: Direct and Iterative

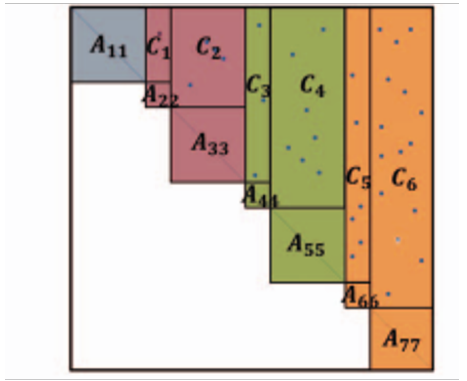
- **Direct Solvers:**
- Factor the sparse matrix to find an exact solution, LDL', LU, Simplicial LU, LL'
- **KLU** (BTF based sequential solver), **ShyLU** (distributed-memory hybrid Schur Complement solver), **Basker** (BTF + Nested-dissection based multithreaded simplicial LU), **Tacho** (multithreaded Cholesky
- **Amesos2**: Interfaces to standard sparse direct solvers libraries: SuperLU, MUMPS, …





- **Iterative Solvers:**
- Choose an approximate solution out of an appropriate subspace.
- **Belos**: Templated Krylov solvers: Conjugate Gradient (CG), MINRES, GMRES, BiCGStab, GCRO-DR
- Simple interface for matrix-free operators
- Block solvers for multiple right-hand sides



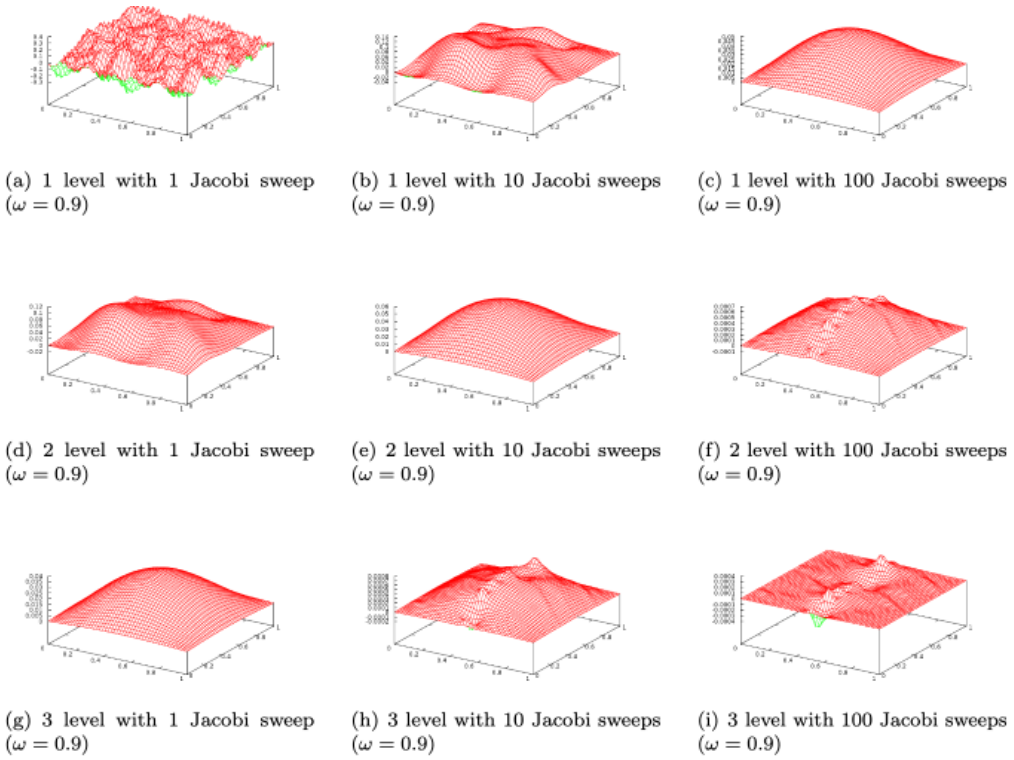


Preconditioners and Smoothers

- **Multigrid methods**
- **MueLU** package offers scalable preconditioning
- Based on second generation solver stack with scalable solves on GPUs

- **Domain Decomposition (1-level):**
- IfPack2 package offers 1-level DD or smoothers
- ILU "Incomplete LU" factorization on the node
- Chebyshev (polynomial) iteration
- Relaxation: Jacobi, Gauss-Seidel

- **Multigrid:** MueLu



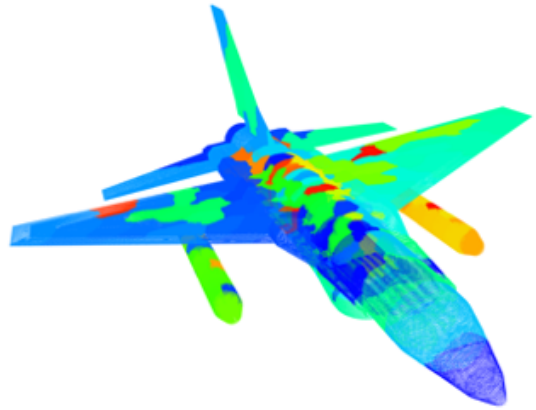
(a) 1 level with 1 Jacobi sweep ($\omega = 0.9$)
(b) 1 level with 10 Jacobi sweeps ($\omega = 0.9$)
(c) 1 level with 100 Jacobi sweeps ($\omega = 0.9$)
(d) 2 level with 1 Jacobi sweep ($\omega = 0.9$)
(e) 2 level with 10 Jacobi sweeps ($\omega = 0.9$)
(f) 2 level with 100 Jacobi sweeps ($\omega = 0.9$)
(g) 3 level with 1 Jacobi sweep ($\omega = 0.9$)
(h) 3 level with 10 Jacobi sweeps ($\omega = 0.9$)
(i) 3 level with 100 Jacobi sweeps ($\omega = 0.9$)

Zoltan2 supports rebalancing in STK mesh database

Mesh database: critical component of Sierra Toolkit (STK)

STK Rebalance capability uses Zoltan2's geometric and graph-based methods for a range of applications

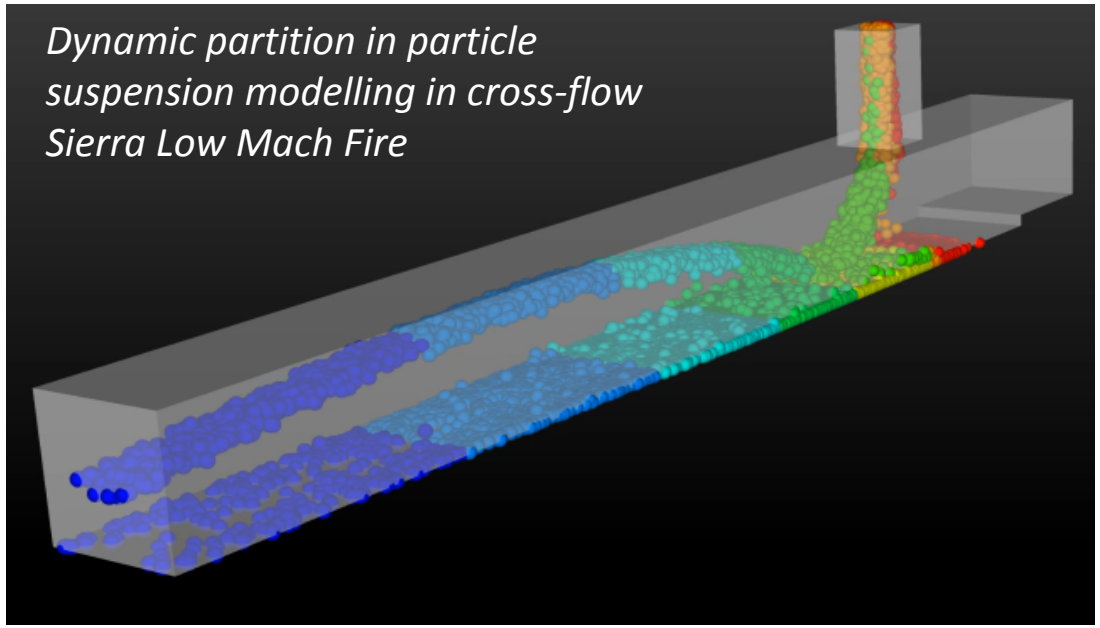
- Static parallel partitioning of huge meshes
- Dynamic load balancing for deforming meshes, particle codes



Parallel static partitioning for 8B elements on 4K cores

Impact example:

- Zoltan2's multicriteria load balancing enabled ~2x faster matrix assembly and ~1.5x faster solution in Aria for mock abnormal thermal use case with 655K elements

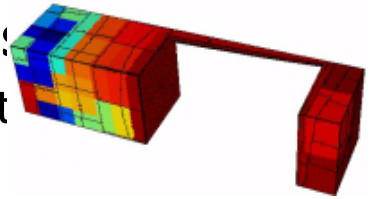


POC: Karen Devine

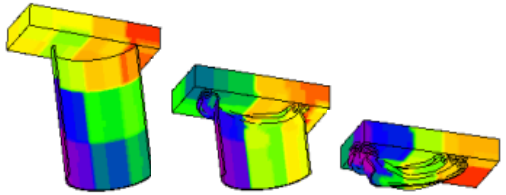
Zoltan/Zoltan2 for Partitioning, Dynamic Load Balancing, and Task Placement

Zoltan/Zoltan2 in Science applications

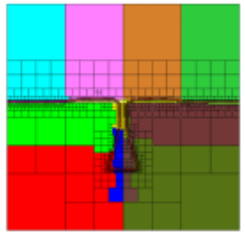
- Maintain load balance and geometric locality for particles (Aleph, SIERRA)



- Increase geometric locality in crash/contact simulations for efficient contact search (SIERRA)



- Partition extremely large meshes, electrical networks & linear systems in parallel (Xyce, SIERRA, Charon)
- Repartition adaptive meshes to maintain load balar (CTH-AMR, SIERRA)



- Redistribute coarse operators in MueLu multigrid to maintain scalability
  (Benefits all applications using MueLu)

Zoltan/Zoltan2 key capabilities:

- **Partitioning & load-balancing**
  - Fast geometric methods maintain spatial locality
  - Graph and hypergraph methods explicitly account for communication costs
  - Single interface to popular partitioning TPLs: XtraPuLP (SNL, RPI); PT-Scotch (U. Bordeaux); ParMETIS (U. Minn.)
- **Architecture-aware MPI task placement**
  - Places interdependent tasks on "nearby" nodes in computing architecture
  - Reduces communication time and network congestion
- **Also: graph coloring, matrix**

Zoltan/Zoltan2 improves parallel efficiency of DOE applications through better load balancing and task placement

Other product areas and packages

- **Tools for the discretization of integral and differential equations.**
  - Local (on-Node) Finite Element Discretizations: **Intrepid2**
  - Global Finite Elements: **Panzer**
    - Provides local-to-global mappings of finite elements from Intrepid
  - Automatic Differentiation: **Sacado**
  - Model building tools: **Phalanx**
    - Decomposes problems into simpler pieces and manages dependencies.
- **Non Linear Solvers Product**
  - Continuation and Bifurcation: **LOCA** (“Library of Continuation Algorithms”)
  - Nonlinear solvers: **NOX**
  - Optimization: **ROL** (“Rapid Optimization Library”)
  - Time Integration: **Tempus**
- **Distribute Linear Algebra**
  - **Tpetra**: (Underneath has Kokkos + MPI)
    - Data structures for Sparse matrices (CRSMatrix), map to distributed ranks, vectors, linear algebera kernels

Computational Frameworks: Kokkos Ecosystem

Changing HPC Landscape and Need for Performance Portability

Several many/multi-core architecture central to DOE/NNSA HPC

Intel Multicore


NVIDIA GPU


IBM Power

Intel Manycore

AMD Multicore/APU

ARM

2012	2016	2018	2021
IBM BGQ (Sequoia, Mira)	Intel KNL (Trinity, Cori)	NVIDIA Volta (Summit, Sierra)	Intel A21
NVIDIA Kepler (Titan)		ARM (Astra)	AMD GPU
			NVIDIA GPU

Decade of DOE HPC will have seen 4-5 "new" paradigms!

- **Several architectures, many with different programming models**
- **Applications struggle to obtain good performance on all of these**

Changing HPC Landscape and Need for Performance Portability

Upcoming Exascale systems use different programming models from different architecture vendors

Approaches to Programming GPUs

Native Programming Models

- CUDA (NVIDIA), HIP (AMD), SYCL (Intel)
- Pros: Customized for each architecture, so low level control
- Cons: Rewrite code every time you buy a hardware from a new vendor

Directive Based Approach

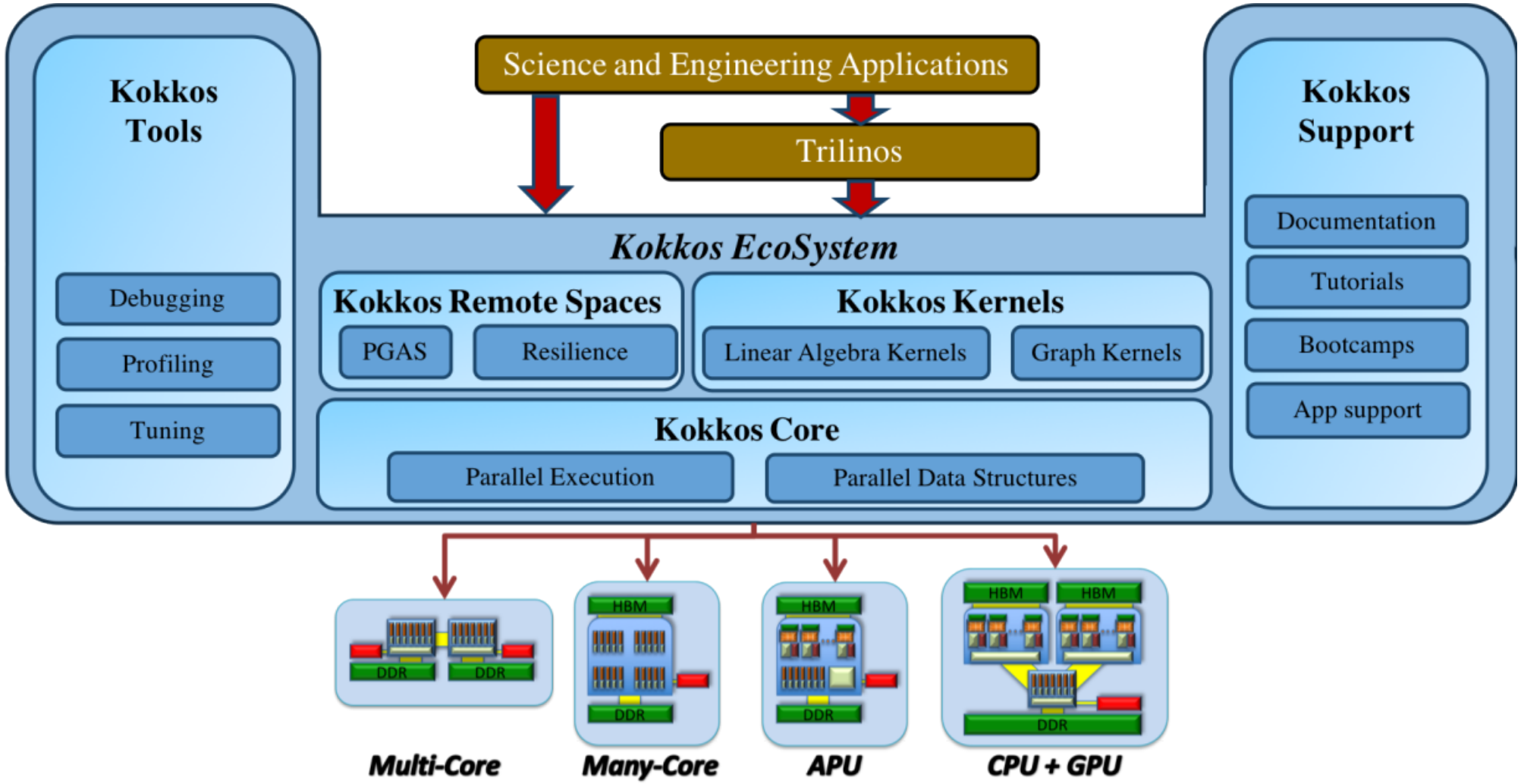
- OpenMP, OpenACC
- Pros: Standards based, General
- Cons: Long lag time between what is needed and when they are needed, Might have to resort to #ifdef after all, Different level of support from vendors

Library Based Approach

- Kokkos, RAJA
- Pros: Portable, Clean abstractions, Quicker turnaround, Reference implementations of standards
- Cons: Dependency on libraries

Library based performance portability allows for writing applications to several architectures with limited dependencies

Kokkos Ecosystem for Performance Portability



Kokkos Ecosystem addresses complexity of supporting numerous many/multi-core architectures that are central to DOE HPC enterprise

Kokkos Overview

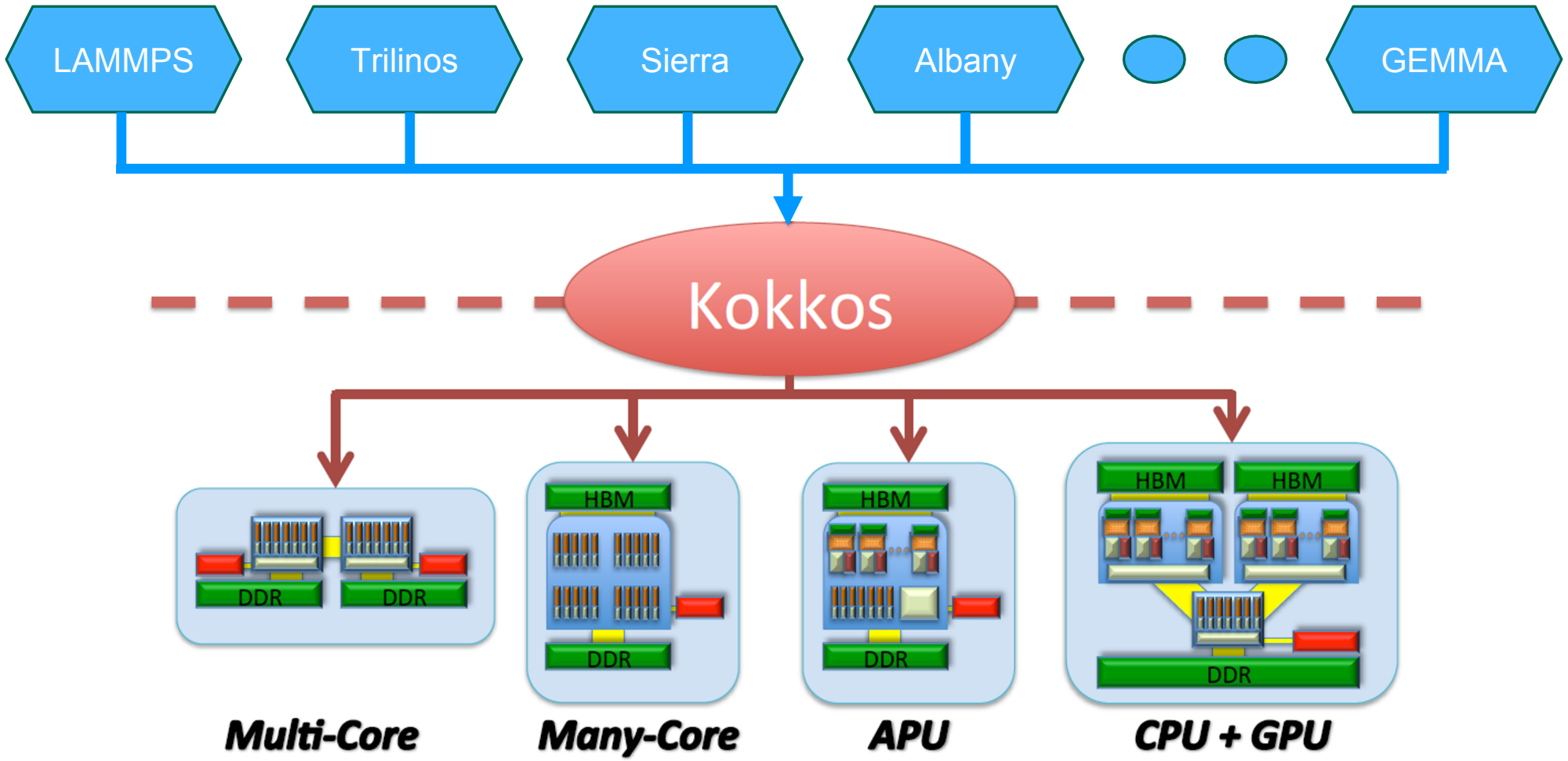
Kokkos is a **productive**, **portable**, **performant**, shared-memory programming model.

- is a C++ **library**, not a new language or language extension.
- supports **clear**, **concise**, **thread-scalable** parallel patterns.
- lets you write algorithms once and run on **many architectures**

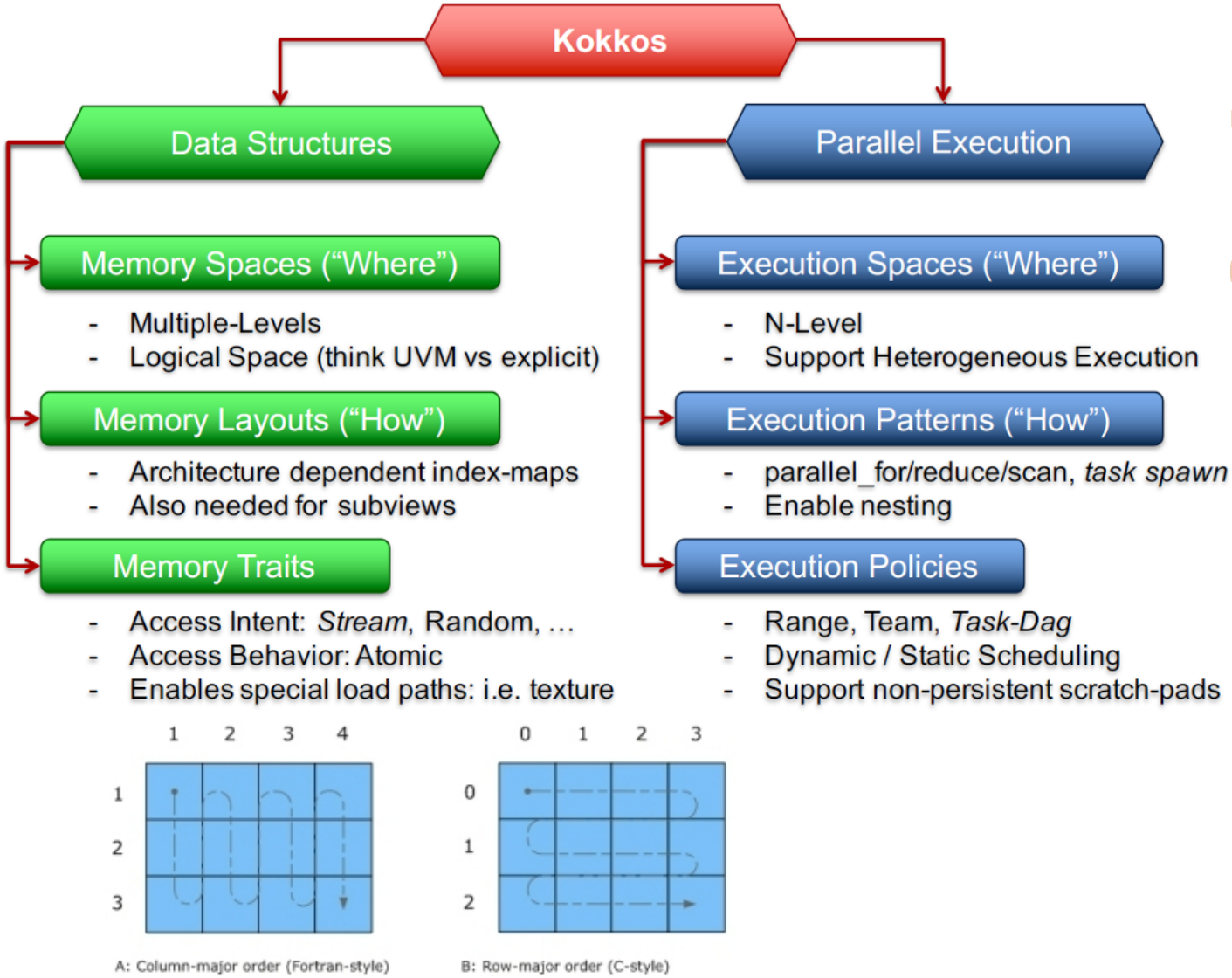
  e.g. multi-core CPU, NVidia GPU, Xeon Phi, ...
- **minimizes** the amount of **architecture-specific implementation details** users must know.
- **solves the data layout problem** by using multi-dimensional arrays with architecture-dependent **layouts**

https://github.com/kokkos/kokkos-tutorials/blob/master/Intro-Short/Slides/KokkosTutorial_ATPESC18.pdf

An Abstraction Layer to Avoid Rewriting an Entire Code



Kokkos Data Management and Execution



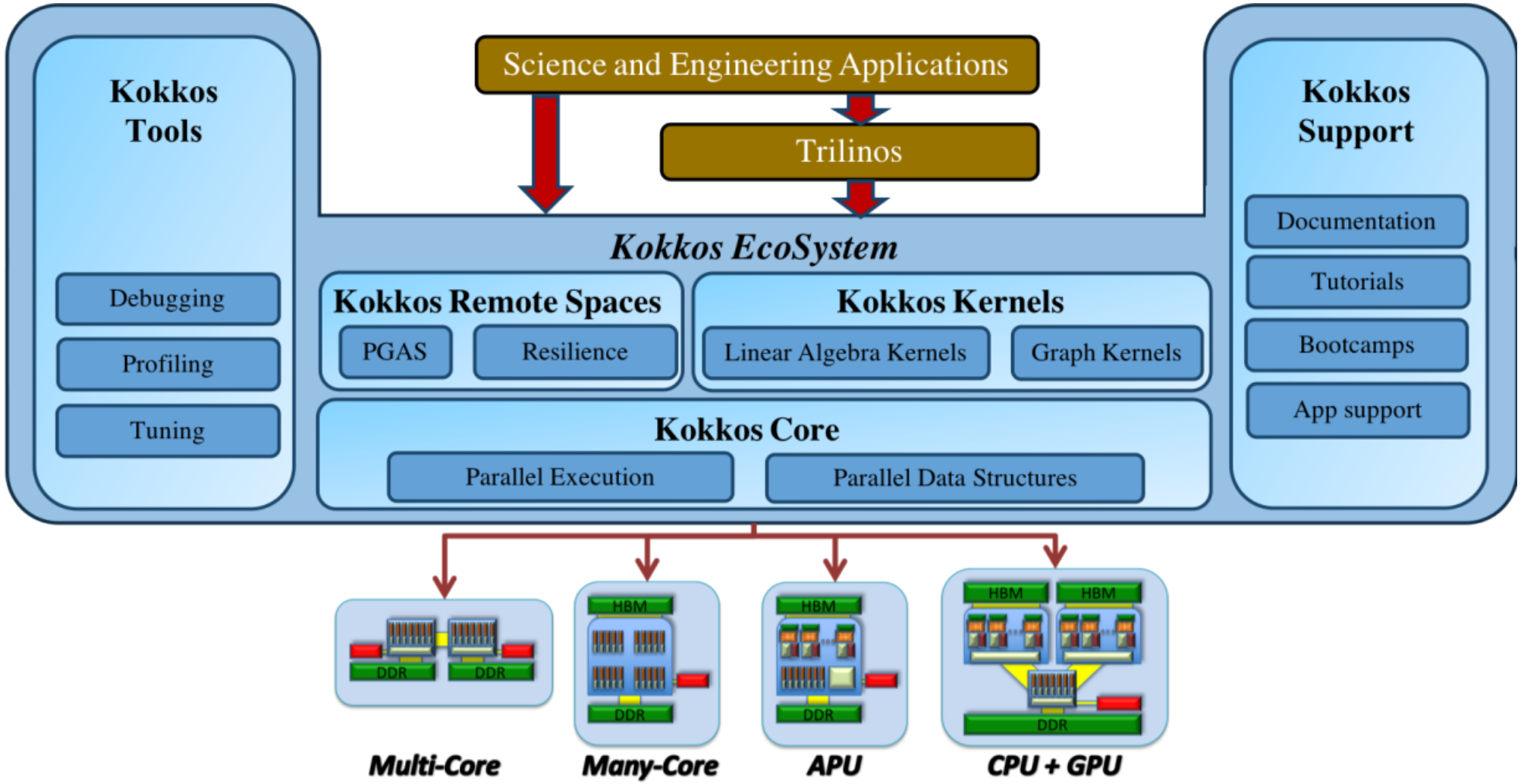
Execution Policies Patterns

```
parallel_for(N, [=] (const size_t i) {
  /* loop body */
});

double totalIntegral = 0;
parallel_reduce(numberOfIntervals,
  [=] (const size_t i, double & valueToUpdate) {
    valueToUpdate += function(...);
  },
  totalIntegral);

parallel_outer(
  TeamPolicy<>(numberOfTeams, teamSize, vectorLength),
  KOKKOS_LAMBDA (const member_type & teamMember[, ...]) {
    /* beginning of outer body */
    parallel_middle(
      TeamThreadRange(teamMember, thisTeamsRangeSize),
      [=] (const int indexWithinBatch[, ...]) {
        /* begin middle body */
        parallel_inner(
          ThreadVectorRange(teamMember, thisVectorRangeSize),
          [=] (const int indexVectorRange[, ...]) {
            /* inner body */
          }[, ...]);
        /* end middle body */
      }[, ...]);
    /* end of outer body */
  }[, ...]);
```

Kokkos Ecosystem for Performance Portability



Kokkos Ecosystem addresses complexity of supporting numerous many/multi-core architectures that are central to DOE HPC enterprise

New Features in Kokkos Kernels 3.X

Sparse Linear Algebra

- ✓Cluster Gauss-Seidel
- ✓ Sparse ILU factorization
- ✓ Sparse triangular solves for sparse L and U
- ✓ Sparse triangular solves for supernodal L and U
- ✓ Structured sparse matrix vector multiply
- ✓ Cluster Gauss Seidel

Dense Linear Algebra

- ✓ Faster kernels for orthogonalization
- ✓ Complex support for dense LU factorization
- ✓Interfaces to vendor libraries
- ✓More BLAS and LAPACK support with Kokkos views

Graph Algorithms

- ✓Distance-2 graph coloring
- ✓Faster distance-1 graph coloring
- ✓Balanced distance-1 coloring
- ✓Balanced “well shaped” graph clustering
- ✓RCM ordering for preconditioners
- ✓ MIS-2 and Coarsening

Portable Vectorization

- ✓ Support ARM platforms
- ✓ Improved application performance on CPU, KNL, GPU and ARM
- ✓ Portable SIMD primitive

Team Level Kernels

- ✓ Team level sorting utilities
- ✓ Team level DFS
- ✓ More team level BLAS and LAPACK support

Software

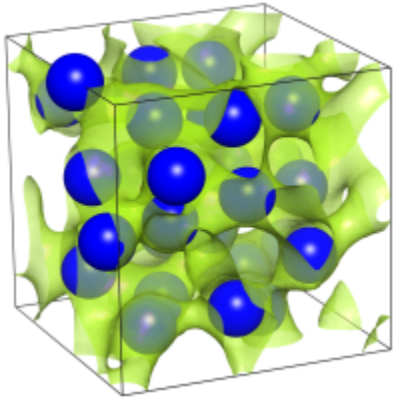
- ✓CMake support
- ✓ETI changes to allow ETI file generation at compile time

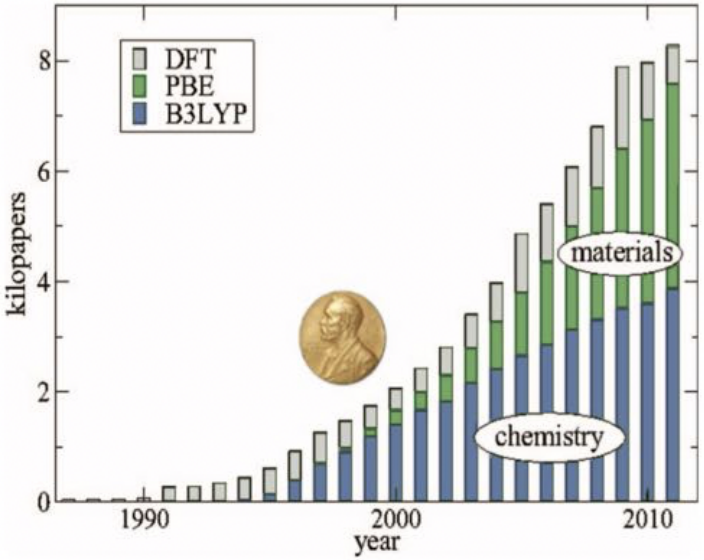
Kokkos Kernels is rapidly growing to support the needs of computational science applications.

Emerging Frameworks: MALA

Motivation

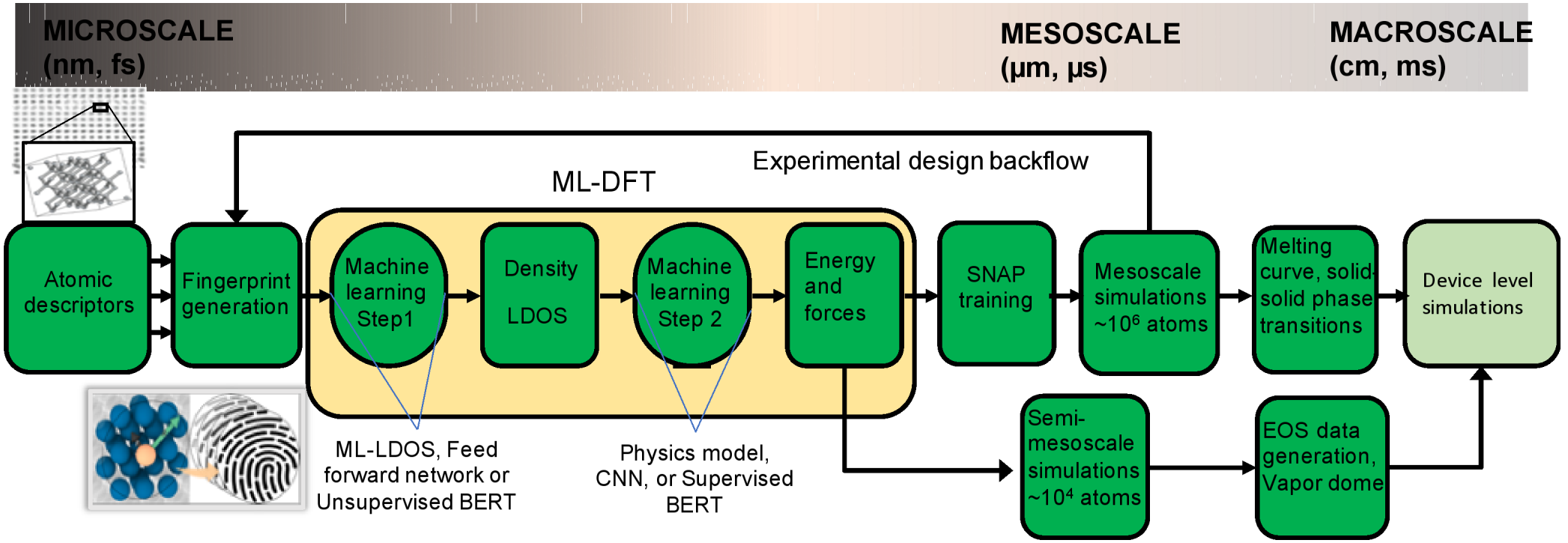
- Multiscale materials modeling (MMM) provide fundamental insight into microscopic mechanisms using:
  - **LAMMPS** molecular dynamics (MD) code.
  - **Quantumespresso/VASP** Density Functional Theory (DFT) codes
- Applications:
  - Rad-hard semiconductors, Advanced manufacturing, Energetic materials
- DFT calculations can be prohibitively expensive, **O (N_{atoms}^3)**
- **Objective:** Develop an **O(N_{atom})** physics-informed machine learning (ML) model to accelerate first-principle DFT calculations





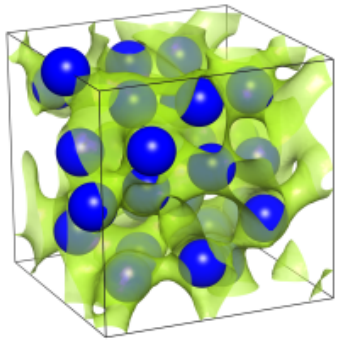
"This new level of technological complexity, combined with the need to search undiscovered areas of the chemical and materials landscape without clear theories or synthesis directions, requires new paradigms that utilize artificial intelligence (AI)." – DOE AI For Science Report

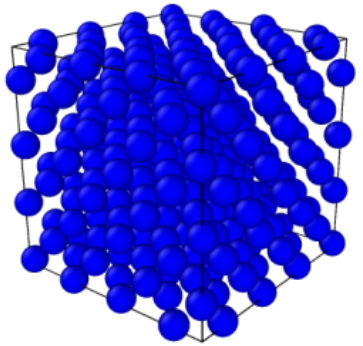
ML-DFT: Accelerating Multiscale Materials Modeling with Machine Learning



- **Physics-informed Machine Learning (ML)** model based Multiscale Materials Modeling (MMM) toolchain.
- Accelerate first-principles data generation, and increase fidelity and robustness of predictive atomistic material simulations.
- ML to accelerate *interpolation* of microscale data (10^2 atoms) and enable *extrapolation* to mesoscale (10^4 atoms)

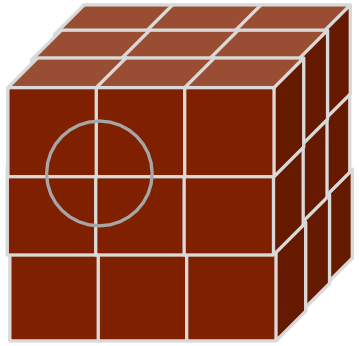
Feature and Data Generation





Fingerprint Inputs:

- **Atom-centered SNAP** descriptor developed at Sandia
  - Rich set of 4-body features, effective for interatomic potentials
  - Heavily optimized LAMMPS implementation $O(N_{neigh} \times N_{atom})$
- We extended to **grid-centered** form $O(N_{neigh} \times N_{grid})$



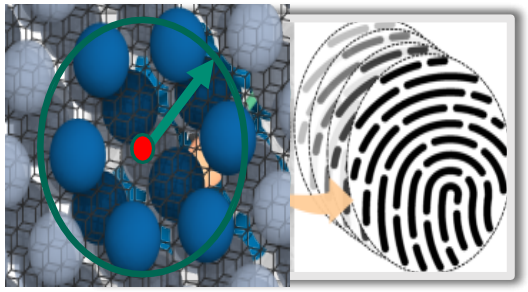
Local Density of States (LDOS) Outputs:

- 10 Aluminum snapshots from an MD-trajectory (previously used by 1600 to calculate the equation-of-state and conductivity of aluminum for 1300)
  - 256 atoms, 298K, 2.699g/cc
  - 8x8x8 k-points
  - LDOS Gaussian smearing
- Post-processed LDOS defined on same **grid-centered** locations

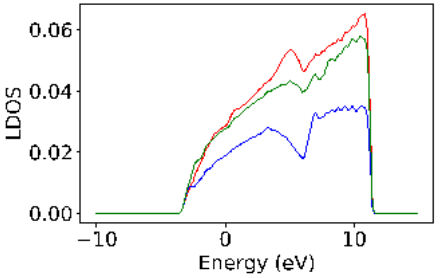
ML Mapping:

- **Input:** Fingerprint features are $N_{fp} \times N_{grid}$ real values
- **Output:** LDOS data are $N_{ldos} \times N_{grid}$ real values

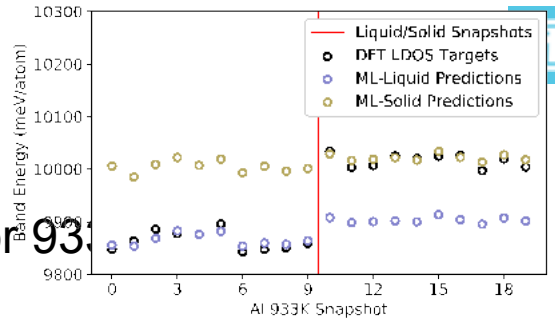
Input fingerprints



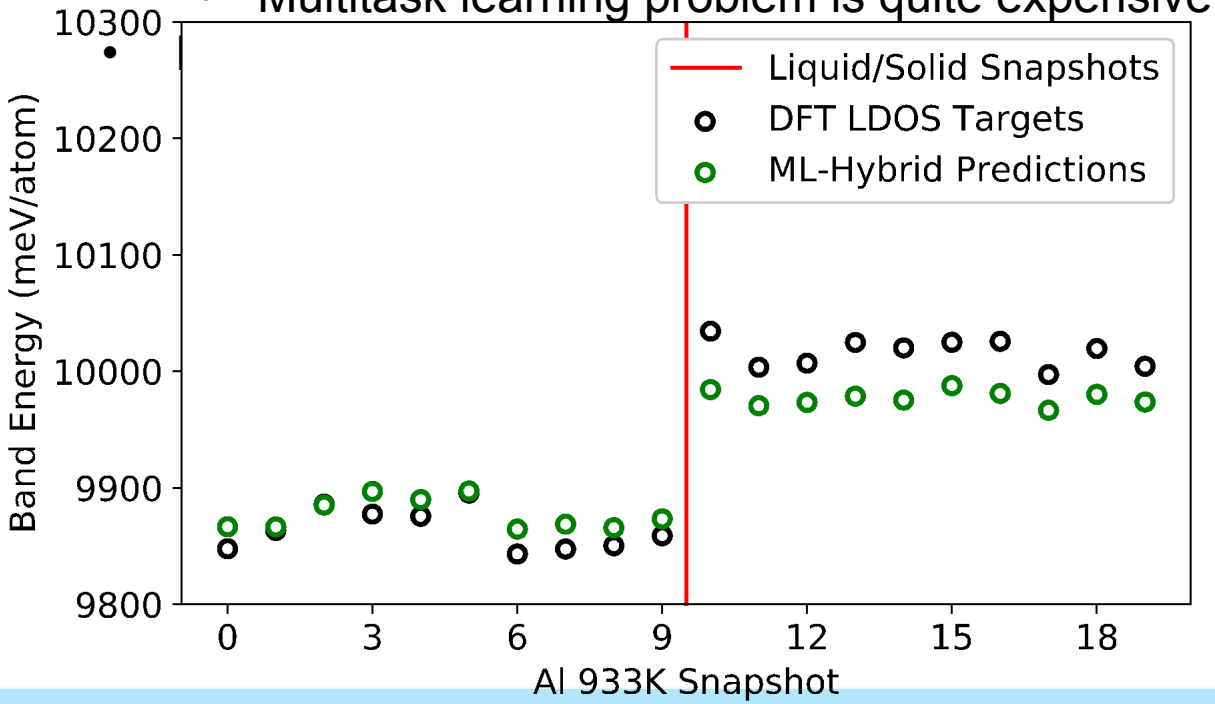
Output LDOS

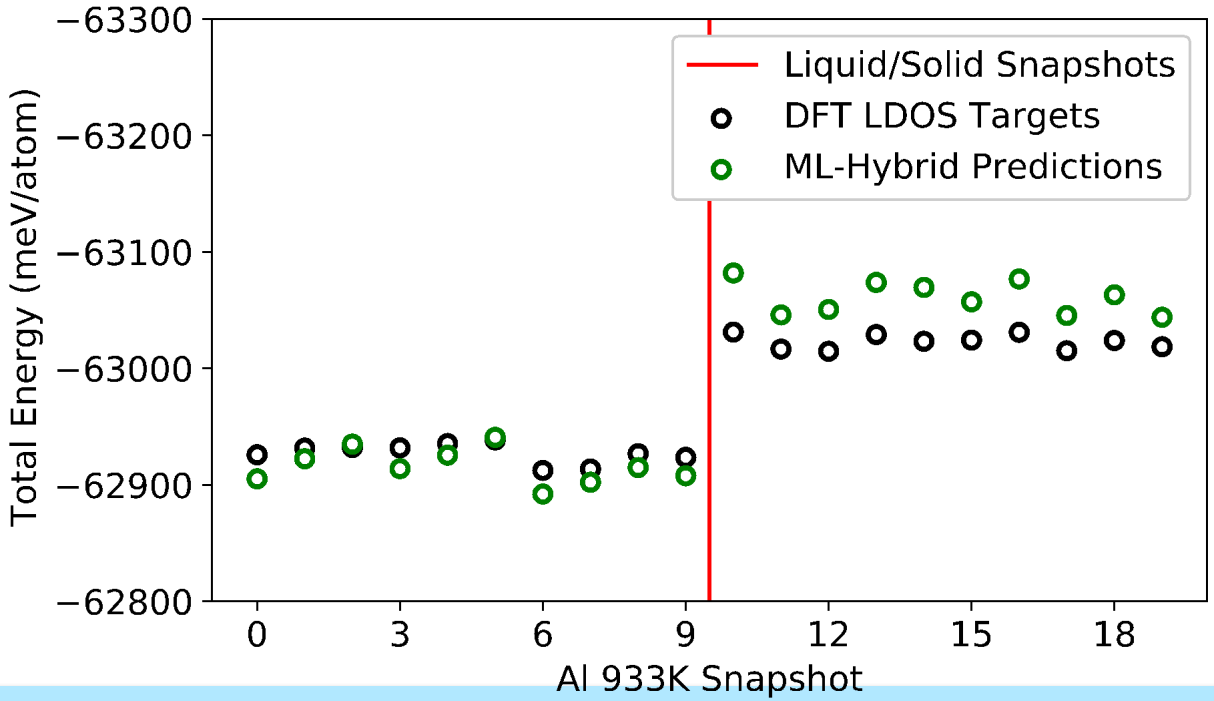


Machine Learning Predictions and Results



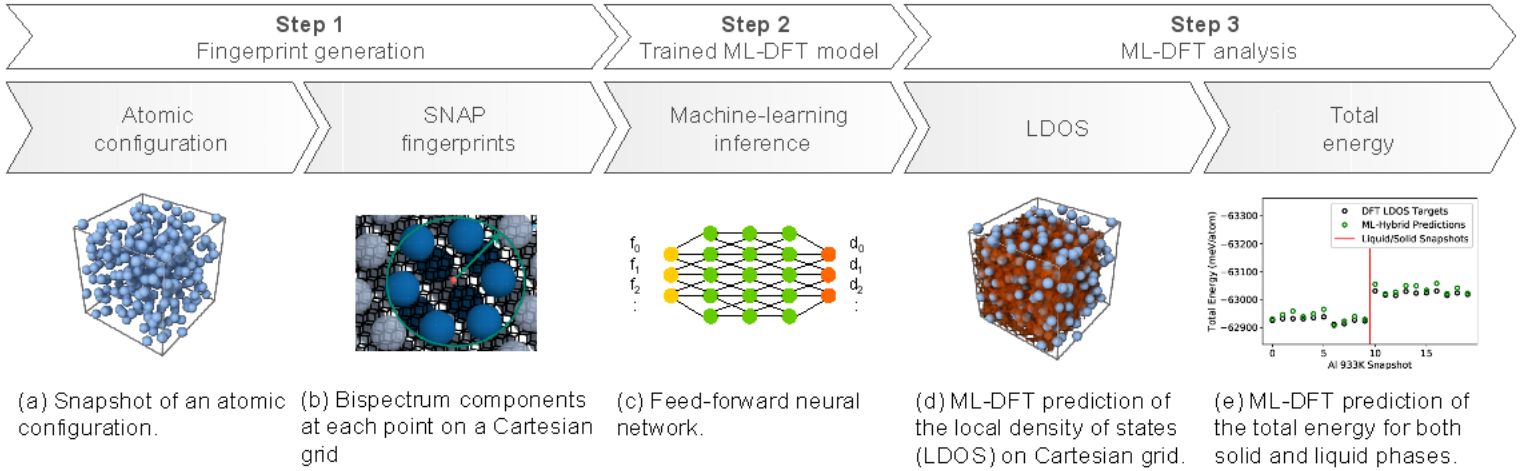
- ML model achieves 5.6 meV/atom max absolute error for 298K
- ML model achieves 15.76 (liquid) 39.23 (solid) meV/atom max absolute error for 93
  - **Test errors less than or equal to intrinsic error in DFT**
- Training time: 151 secs/epoch (298K), 76 minutes/epoch on 2 GPUs (39-100 epochs, 48 and 23 iterations for hyperparameter tuning)
  - Data parallel training has brought down training time
  - Multitask learning problem is quite expensive





ML-DFT reproduces DFT to within chemical accuracy. ML inference takes ~53 seconds on 1 GPU. The training cost can be reduced and amortized by using the model many times on many snapshots.

Methods, Software and Data



(a) Snapshot of an atomic configuration. (b) Bispectrum components at each point on a Cartesian grid (c) Feed-forward neural network. (d) ML-DFT prediction of the local density of states (LDOS) on Cartesian grid. (e) ML-DFT prediction of the total energy for both solid and liquid phases.

Open Source Software:

- New software framework for ML-DFT Calculations
  - **MALA: Materials Learning Algorithms**
  - **https://github.com/mala-project/mala**

Open Training Data, Models, input files:

- **https://rodare.hzdr.de/record/1107**
- **450 GB! - Be careful where you download**

Reference:

- **https://journals.aps.org/prb/abstract/10.1103/PhysRevB.104.035120**
- Accelerating finite-temperature Kohn-Sham density functional theory with deep neural networks

Summary

- Decades long investment in computational frameworks
- State of the art algorithms in the frameworks
- Portable implementation of the algorithms with focus on large petascale/exascale systems at DOE to desktop computers (in some cases)
- Modeling and simulations built on top of the foundational frameworks

Extra Slides