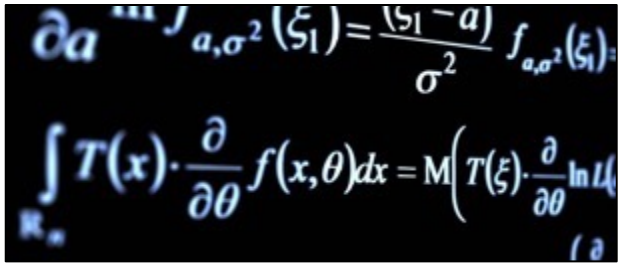
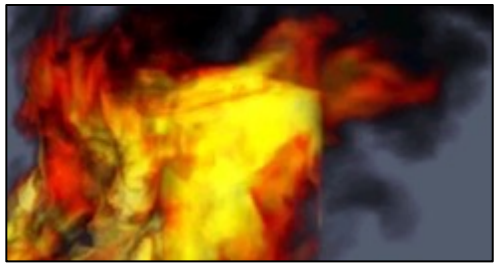
Exceptional service in the national interest

Sandia National Laboratories



$$\int T(x)\cdot\frac{\partial}{\partial\theta}f(x,\theta)dx = \mathrm{M}\left(T(\xi)\cdot\frac{\partial}{\partial\theta}\ln L\right.$$



Kokkos Sustaining Exascale Codes for the Long Haul

Unclassified Unlimited Release

Christian R. Trott, - Center for Computing Research

Sandia National Laboratories/NM

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-NA-0003525.

SAND2020-9180 C

Sandia National Laboratories

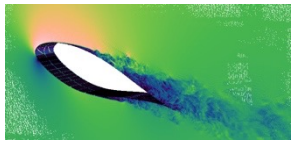
How does Kokkos help applications with sustainability concerns?

Sandia National Laboratories

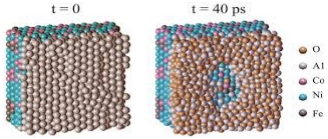
Applications

Libraries

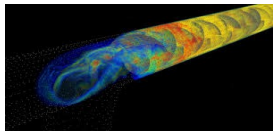
Frameworks



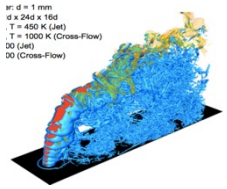
SNL NALU
Wind Turbine CFD



SNL LAMMPS
Molecular Dynamics



UT Uintah
Combustine



ORNL Raptor
Large Eddy Sim

ORNL Frontier
Cray / AMD
GPU

LANL/SNL Trinity
Intel Haswell / Intel
KNL

ANL Aurora
Intel Xeon CPUs +
Intel Xe GPUs

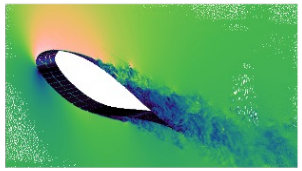
SNL Astra
ARM Architecture

LLNL SIERRA
IBM Power9 / NVIDIA Volta

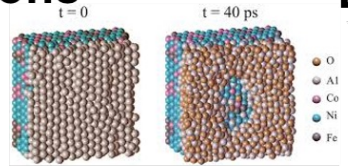
Applications

Libraries

Frameworks

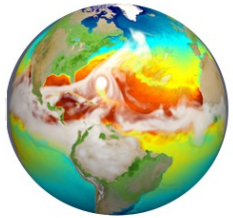


Computational Fluid Dynamics



Molecular Dynamics

Electro Magnetics



Climate Simulation

Kokkos

ORNL Frontier
AMD GPUs

LANL/SNL Trinity
Intel CPUs

ANL Aurora
Intel GPUs

Riken Fugaku
ARM CPUs

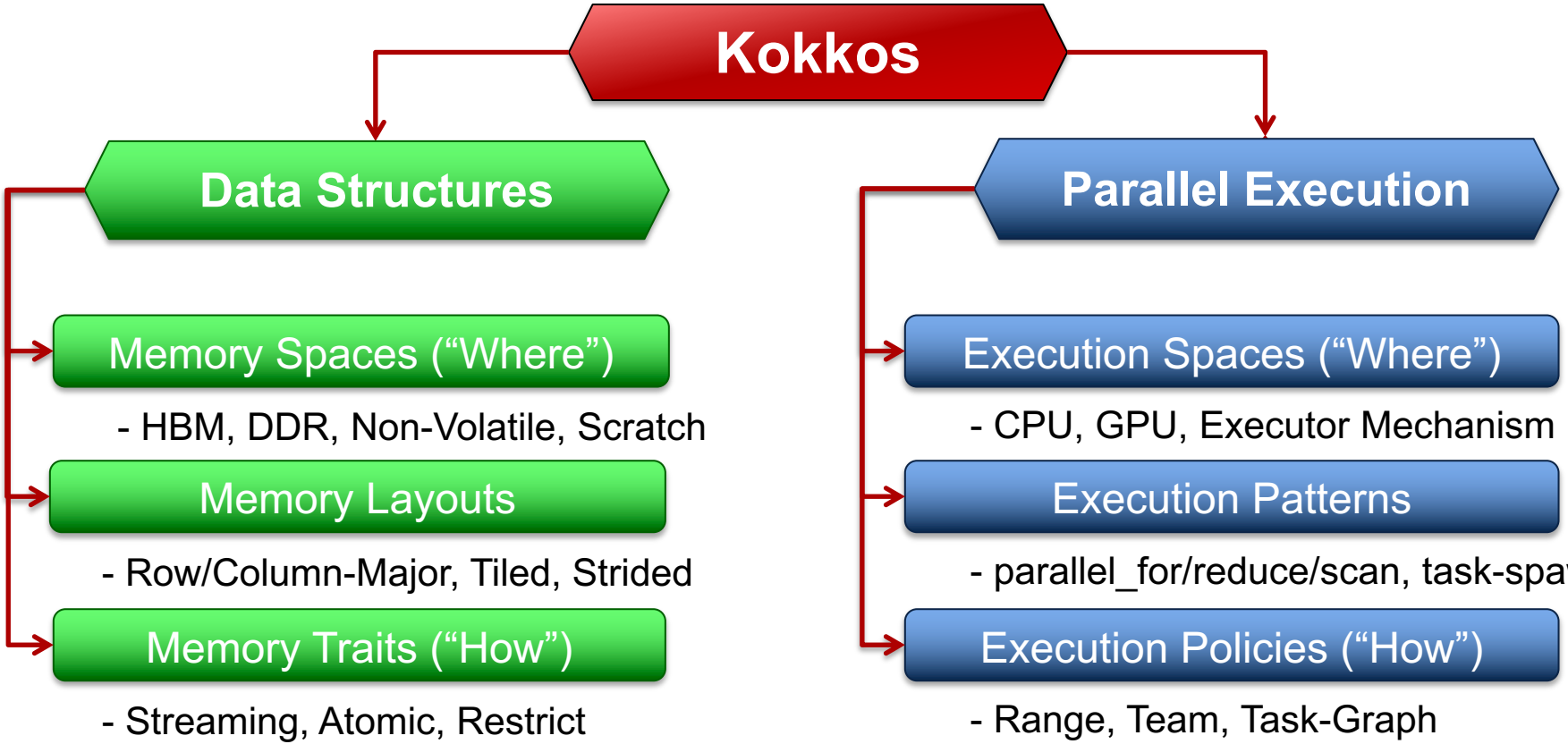
NERSC Permutter
NVIDIA GPUS

What is Kokkos?

Sandia National Laboratories

- A C++ Programming Model for Performance Portability
  - Implemented as a template library on top of CUDA, OpenMP, HPX, …
  - Aims to be descriptive not prescriptive
  - Aligns with developments in the C++ standard
- Expanding solution for common needs of modern science/engineering codes
  - Math libraries based on Kokkos
  - Tools which enable insight into Kokkos
- It is Open Source
  - Maintained and developed at https://github.com/kokkos
- It has many users at wide range of institutions.

Kokkos Core Abstractions



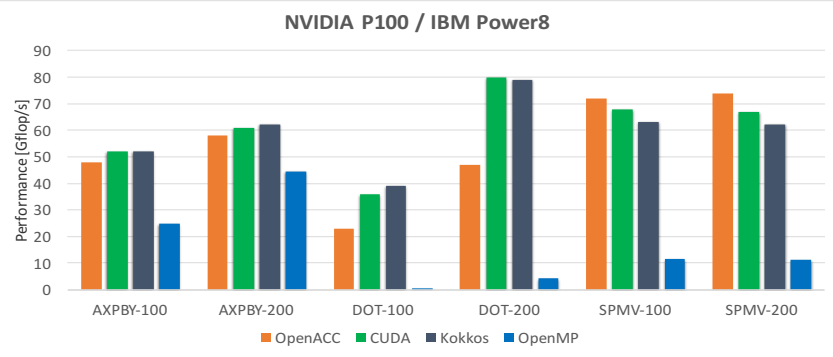
Kokkos Core Capabilities

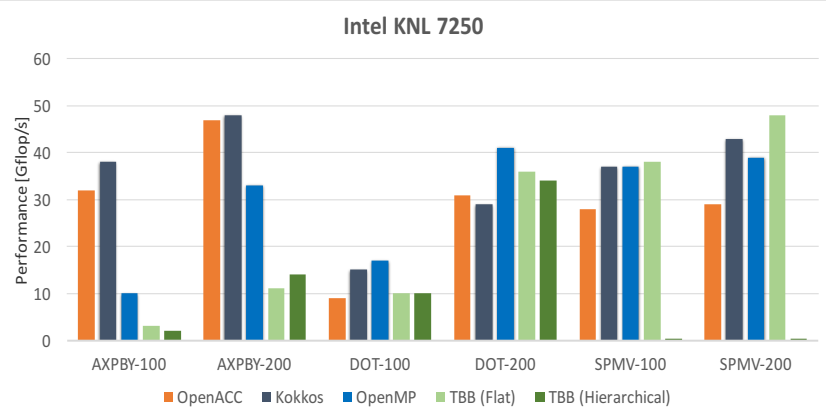
Sandia National Laboratories

Concept	Example
Parallel Loops	**parallel_for**(N, **KOKKOS_LAMBDA** (int i) { ...BODY… });
Parallel Reduction	**parallel_reduce**(**RangePolicy**<ExecSpace>(0,N), **KOKKOS_LAMBDA** (int i, double& upd) { …BODY... upd += ... }, Sum<>(result));
Tightly Nested Loops	**parallel_for**(**MDRangePolicy**<**Rank**<3> > ({0,0,0},{N1,N2,N3},{T1,T2,T3}, **KOKKOS_LAMBDA** (int i, int j, int k) {…BODY...});
Non-Tightly Nested Loops	**parallel_for**(**TeamPolicy**<**Schedule**<**Dynamic**>>(N, TS), **KOKKOS_LAMBDA** (Team team) { … COMMON CODE 1 ... **parallel_for**(**TeamThreadRange**(team, M(N)), [&] (int j) { ... INNER BODY... }); … COMMON CODE 2 ... });
Task Dag	**task_spawn**(**TaskTeam**(scheduler , priority), **KOKKOS_LAMBDA** (Team team) { … BODY });
Data Allocation	**View**<double**, Layout, MemSpace> a(“A”,N,M);
Data Transfer	**deep_copy**(a,b);
Atomics	atomic_add(&a[i],5.0); View<double*,MemoryTraits<AtomicAccess>> a(); a(i)+=5.0;
Exec Spaces	Serial, Threads, OpenMP, Cuda, HPX (experimental), HIP (experimental), OpenMPTarget (experimental)

CG Solve: Performance 2016

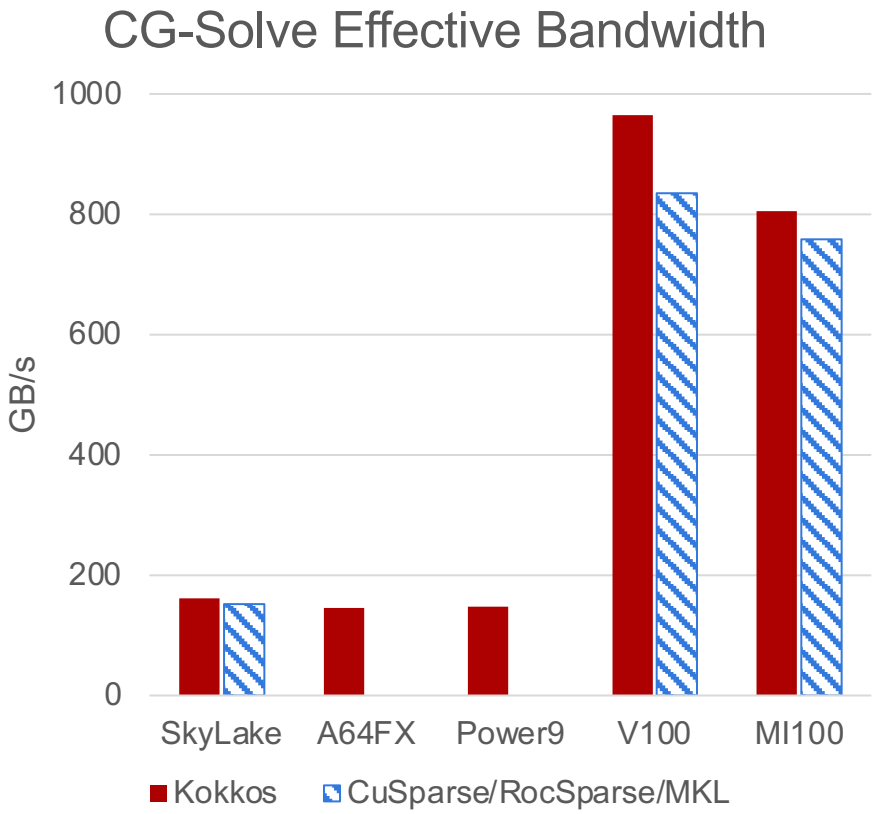
- Comparison with other Programming Models
- Straight forward implementation of kernels
- OpenMP 4.5 was immature at that point
- Two problem sizes: 100x100x100 and 200x200x200 elements





CG Solve Performance 2022

- CG-Solve as discussed above
- Also try replacing SPMV with TPL
- Running 100x100x100 heat conduction problem
  - ”MiniFE” Proxyapp setup
- Measure effective Bandwidth
  - Algorithmical memory ops per time
- Why is this beating vendor libs?
  - Its complicated, but a real effect



Sandia National Laboratories

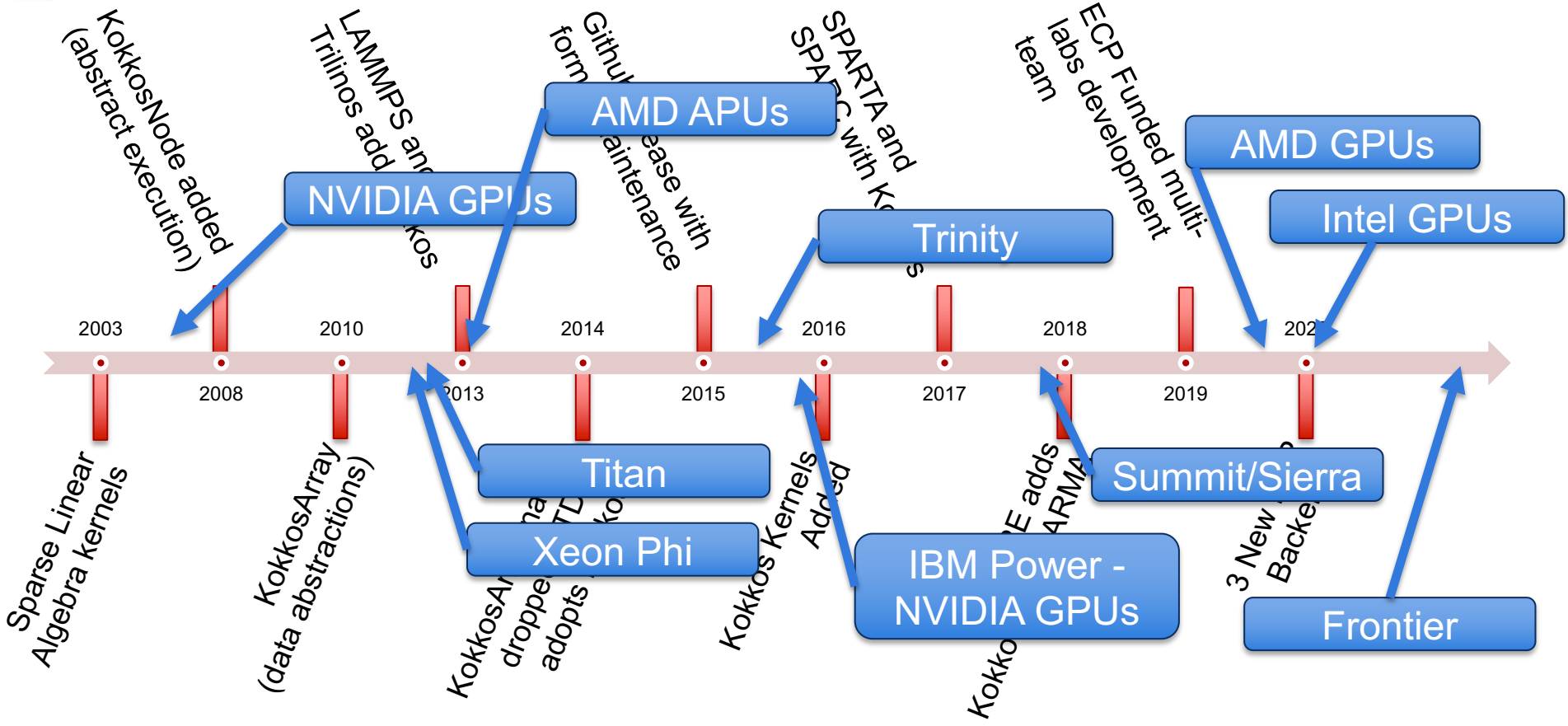
Developing a sustainable project.

Stepping Stones To Success

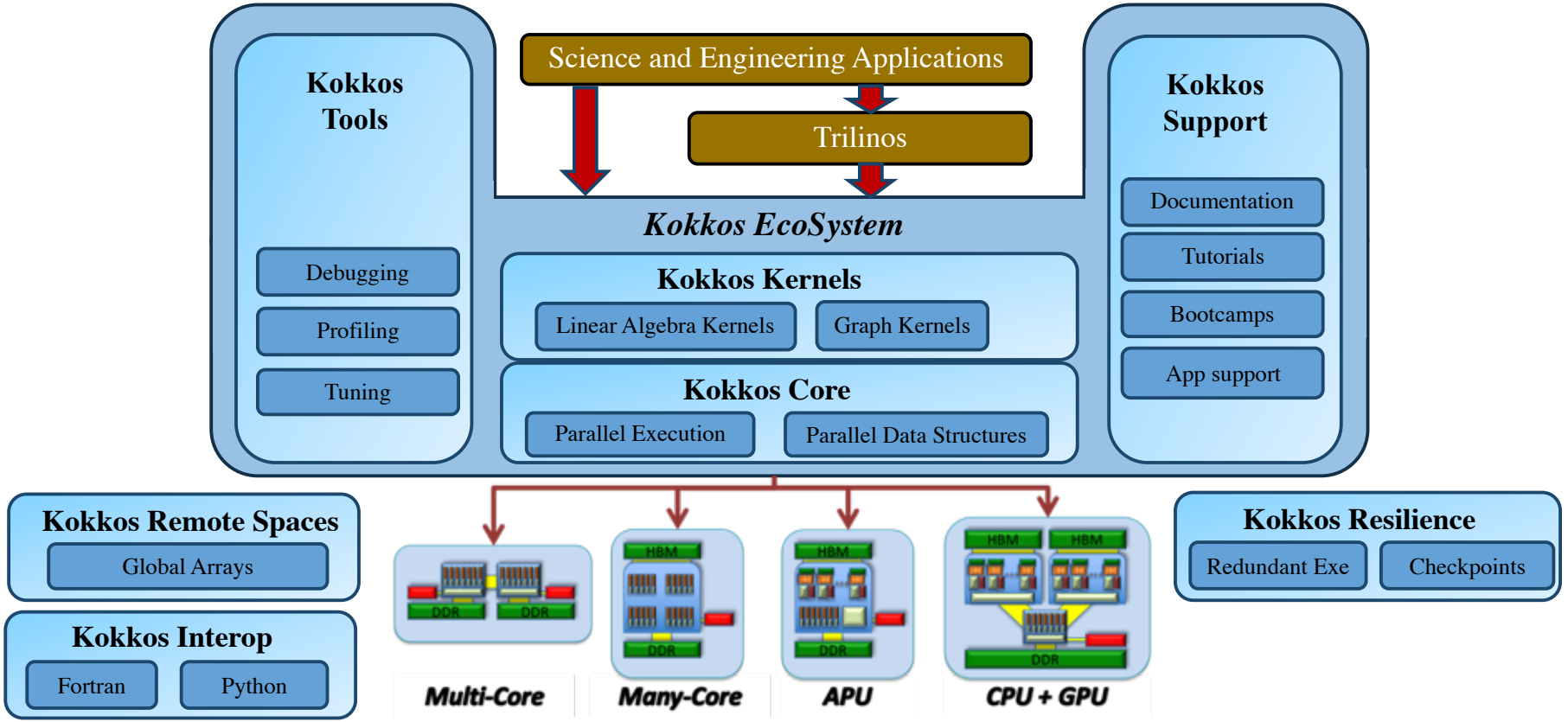
Sandia National Laboratories

- Making the transition from research to actually deployed production software required years with a number of key aspects being critical:
  - ***Address a Critical Need:*** More platforms were coming and apps simply can't rewrite for every single one
  - ***Demonstrate Viability:*** The Kokkos team ported chunks of LAMMPS and Trilinos to demonstrate that it works and performs; We also demonstrated performance parity to native models in miniApps
  - ***User outreach:*** The Kokkos team spend large amounts of its time in outreach, training and support to build the initial community
  - ***Long term stability:*** Having the support from Sandia's HPC management ensured stable long term funding

Kokkos Timeline at Sandia



Kokkos EcoSystem



Testing and Verification

Sandia National Laboratories

- Github based developments with Pull Requests
- Pull Request Testing
  - 30 configurations
  - Includes: CUDA, HIP, OpenMPTarget, SYCL, Windows, Linux, OSX
- “Rule of three”: two additional developers need to approve a pull request
  - Approving means: “I feel I could maintain this if the author left”
- Strong backwards compatibility policy
  - Can’t break backwards compatibility too much, and even minor ones only every couple years or so
- Releases require integration testing with major customers
  - Includes multi-platform test of Trilinos and LAMMPS among others

Sandia National Laboratories

Building a Community

Kokkos Uptake

Community Project

Sandia National Laboratories

OAK RIDGE National Laboratory

Argonne NATIONAL LABORATORY

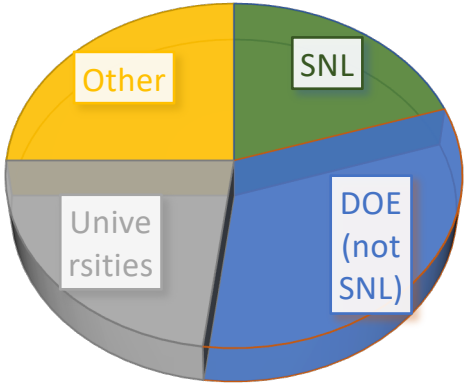
Los Alamos NATIONAL LABORATORY EST. 1943

BERKELEY LAB

- 25 Developers
  - 15 for Core
- Regular contributions from HPC vendors

Kokkos Slack Channel

- >1200 registered users
  - 150 Institutions
  - Every continent
    - (-Antarctica)



Applications and Libraries

- Estimated 200-400 HPC projects using Kokkos
- On the order of two-dozen apps run science and engineering production runs with Kokkos
- Similar distribution as the Slack User

Training and Documentation

Sandia National Laboratories

- The Kokkos Lectures
  - 8 lectures covering most aspects of Kokkos
  - 15 hours of recordings
  - > 500 slides
  - >20 exercises
- Extensive Wiki
  - API Reference
  - Programming Guide
- Slack as primary direct support

https://kokkos.github.io

- Module 1: Introduction
  - Introduction, Basic Parallelism, Build System
- Module 2: Views and Spaces
  - Execution and Memory Spaces, Data Layout
- Module 3: Data Structures and MDRangePolicy
  - Tightly Nested Loops, Subviews, ScatterView,…
- Module 4: Hierarchical Parallelism
  - Nested Parallelism, Scratch Pads, Unique Token
- Module 5: Advanced Optimizations
  - Streams, Tasking and SIMD
- Module 6: Language Interoperability
  - Fortran, Python, MPI and PGAS
- Module 7: Tools
  - Profiling, Tuning , Debugging, Static Analysis
- Module 8: Kokkos Kernels
  - Dense LA, Sparse LA, Solvers, Graph Kernels

Sandia National Laboratories

Recently Added Capabilities

New Capabilities: ISO C++ Standard Algorithms

- Kokkos provides vast majority of all ISO C++ standard algorithms since v. 3.6
- Take iterators or Kokkos::View
- Available at top level and at team level in hierarchical parallelism (new in 4.2)
  - Top level take optional “Label” like other top level Kokkos parallel patterns
  - Always take either Execution Space instance or Team Handle

```
auto iter = find(”Label”, DefaultExecutionSpace(),
                 cbegin(view), cend(view), find_value);
auto iter = find(“Label”, DefaultExecutionSpace(),
                 view, find_value);
auto iter = find(DefaultExecutionSpace(), cbegin(view), cend(view), find_value);
auto iter = find(DefaultExecutionSpace(), view, find_value);
auto iter = find(team_handle, cbegin(view), cend(view), find_value);
auto iter = find(team_handle, view, find_value);
```

New Capabilities: ISO C++ SIMD Types

Sandia National Laboratories

- - Previously available as separate repository (github.com/kokkos/simd-math)
- - Now part of Kokkos Core
- - Implements proposed ISO C++ SIMD types (for C++26)
- - Work across all backends, may just fall back to scalar operations

```
using simd_t = Kokkos::Experimental::simd<double, Kokkos::Experimental::simd_abi::native<double>>;

int N = N_in / simd_t::size();
  Kokkos::View<simd_t **> data("D", N, M);
  Kokkos::View<simd_t *> results("R", N);

  double a = 3.0;
  Kokkos::parallel_for( "Combine", data.extent(0), KOKKOS_LAMBDA(const int i) {
    simd_t tmp = 0.0;
    double b = a;
    for (int j = 0; j < data.extent(1); j++) {
      tmp += b * data(i, j);
      b += a + 1.0 * (j + 1);
    }
    results(i) = tmp;
  });
```

New Capabilities: MDRangePolicy for Teams

- Support MDRangePolicy at nested hierarchical parallelism level
- Parallelizes with both vector and thread level if available
- Three policies in analogy to existing nested policies:
  - TeamThreadMDRange
  - TeamVectorMDRange
  - ThreadVectorMDRange
- Support all typical things: parallel_for, parallel_reduce, Rank argument etc.

```
Kokkos::parallel_for(Kokkos::TeamPolicy<ExecSpace>(leagueSize, Kokkos::AUTO),
    KOKKOS_LAMBDA(const TeamType& team) {
      int leagueRank = team.league_rank();

      auto teamRange =
          Kokkos::TeamThreadMDRange<Kokkos::Rank<2, Direction>, TeamType>(team, n0, n1);

      Kokkos::parallel_for(teamRange, [=](int i, int j) {
        v(leagueRank, i, j) += fillFlattenedIndex(leagueRank, i, j);
      });
    });
```


Sandia
National
Laboratories