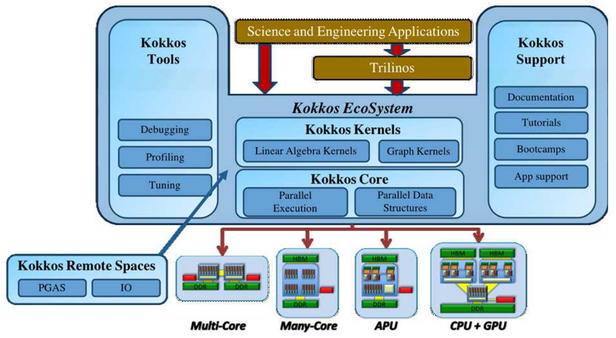
Deep dive technical story:
2.3.6.03 (SNL NNSA Software) Kokkos:

SAND2019-5585PE

- Kokkos is the primary on-node programming model at Sandia
  - Estimate around 40 projects at Sandia are using Kokkos
  - Slightly more than that outside of Sandia
- Vision: Performance Portability through C++
  - Heavy emphasis on work in the C++ standard committee
  - Transition capabilities into the standard
  - Make Kokkos a “sliding window” of advanced capabilities
- Tight collaboration with vendors allows Kokkos to isolate apps from the ever faster changing HPC architecture landscape
- Strong focus on designing a general programming model
  - No ”one-offs”
  - No application specific features
  - No hardware specific API features

Some Kokkos Users

Foundation for the Kokkos EcoSystem



Kokkos Development Team 2.3.6.03 + 2.3.1.10 (soon 2.3.1.18 with RAJA)

- Dedicated team with a number of staff working most of their time on Kokkos
  - Main development team at Sandia in CCR
  - Additional teams for Kokkos Kernels, Tools and Support

Kokkos Core:	***C.R. Trott,*** *D. Sunderland, N. Ellingwood, D. Ibanez, J. Miles, D. Hollman, V. Dang, Mikael Simberg, H. Finkel, N. Liber, D. Lebrun-Grandie, B. Turcksin*
	former: ***H.C. Edwards****, D. Labreche, G. Mackey, S. Bova*
Kokkos Kernels:	***S. Rajamanickam,*** *N. Ellingwood, K. Kim, C.R. Trott, V. Dang, L. Berger, J. Wilke, W. McLendon*
Kokkos Tools:	***S. Hammond,*** *C.R. Trott, D. Ibanez, S. Moore; soon:* ***D. Poliakoff***
Kokkos Support:	***C.R. Trott,*** *G. Shipman, G. Lopez, G. Womeldorff,*
	former: ***H.C. Edwards****, D. Labreche, Fernanda Foertter*

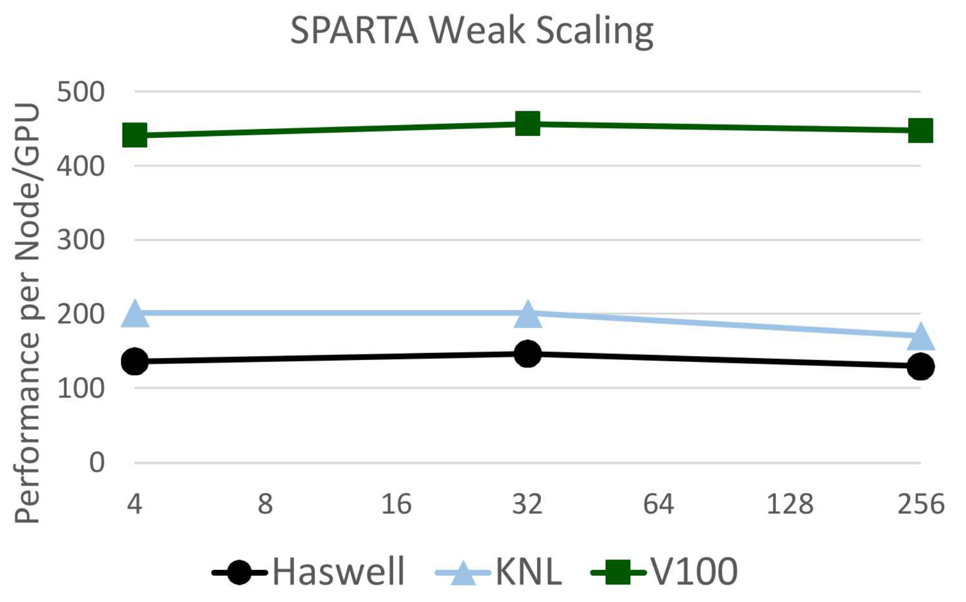
- DOE Exascale Machine Support (also supports RAJA via 2.3.1.18)
  - ORNL Cray with AMD GPUs + AMD CPUs via AMD HIP developed at ORNL
  - ANL Cray with Intel Xe Compute + Intel Xeon via Intel One API compiler developed at ANL

- Support: https://github.com/kokkos https://kokkosteam.slack.com

Kokkos ATDM Adoption

- All Sandia ATDM Applications based on Kokkos
  - SPARC, Empire, Sparta, Cheetah, Gemma
  - Target Sierra, Trinity, Astra and later machines
  - Kokkos releases are qualified against ATDM applications
    - No special Kokkos versions for ATDM though, everybody uses the same public release
  - Issues are generally fed into the public tracking system and prioritized according to severity
- Los Alamost ATDM projects started working on incorporating Kokkos
  - Regular collaborations with core members from Kokkos going for multi-day visits to LANL, and LANL teams are frequent attendees at Kokkos training events
  - LANL leads Kokkos/Fortran Interop efforts
    - Public release imminent
  - FleCSI now working on exploiting Kokkos for node level parallelism

- **Sparta - Production DSMC**
  - **S**tochastic **PA**rallel **R**arefied-gas **T**ime-accurate **A**nalyzer
  - *Steve Plimpton, Stan Moore, Michael Gallis*
  - Only code to have run on all of Trinity
    - 3 Trillion particle simulation using both HSW and KNL partition in a single MPI run
  - Production runs now at 5k GPUs
    - Benchmarked on 16k GPUs on Sierra
  - Co-Designed Kokkos::ScatterView

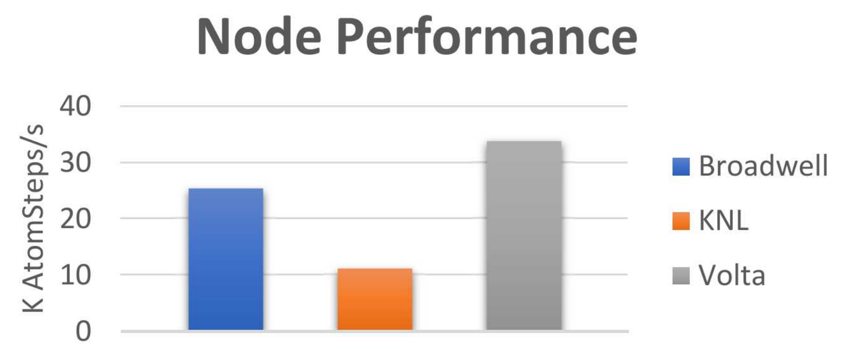


Kokkos Broader ECP Community Adoption

Stakeholder (WBS)	Anticipated new capabilities they will use	How integration will be achieved
2.2.1.02	NWChemEx: Kokkos support for A21/Frontier; maybe Kokkos support for on-node PGAS; C++ Compatibility and Backport features	CMake build against release Kokkos version; Currently experimenting
2.2.1.04	Exaalt: Kokkos support for A21/Frontier; C++ Compatibility and Backport features	LAMMPS snapshots Kokkos regularly into its repository
2.2.1.06	QMCPack: Kokkos support for A21/Frontier; maybe Kokkos support for on-node PGAS; C++ Compatibility and Backport features	CMake build against Kokkos; Attending bootcamps + potential extra meeting
2.2.2.01	ExaWind: Kokkos support for A21/Frontier; C++ Compatibility and Backport features	Using Kokkos via Trilinos
2.2.2.02	Pele: Kokkos support for A21/Frontier; C++ Compatibility and Backport features	
2.2.5.01	LANL ATDM: Frontier Support; C++ Compatibility and Backport features;	CMake against Kokkos releases.
2.2.5.03	SNL ATDM: Frontier Support; C++ Compatibility and Backport features;	Through Trilinos and direct snapshotting of Kokkos
2.2.6.01	Proxy Apps: support for A21/Frontier; maybe Kokkos support for on-node PGAS; C++ Compatibility and Backport features; Kokkos/RAJA interoperability	Various build systems using Kokkos and RAJA releases; RAJA Performance Suite is in Proxy App collection;
2.2.6.04	COPA: Kokkos support for A21/Frontier; maybe Kokkos support for on-node PGAS; C++ Compatibility and Backport features	CMake + Kokkos releases
2.2.6.07	ExaGraph: Kokkos support for A21/Frontier; C++ Compatibility and Backport features	
2.2.6.08	ExaLearn: Kokkos support for A21/Frontier; C++ Compatibility and Backport features	CMake + Kokkos releases
2.3.3.13	Slate: Kokkos support for A21/Frontier; C++ Compatibility and Backport features	
2.3.3.14	ALExa: Kokkos support for A21/Frontier; C++ Compatibility and Backport features	Through Trilinos.

EXAALT Molecular Dynamics

- Loosely coupled ensemble simulations using LAMMPS
- Employing SNAP potential
  - Most accurate classical potential we have for W and W/He
  - Offline Machine Learning used to get parameters
  - ~500x more expensive than Lennard Jones
  - Large cache footprint good on CPUs, but makes it HBM memory bandwidth bound on GPUs and KNL
- Collaboration with COPA
- Targets both Aurora and Frontier



Kokkos Deep Dive
C++ Standard Connection

- Kokkos Team currently involved in more than a dozen proposals
  - Leading atomic_ref, mdspan and linear algebra related proposals
  - Proposals take years to make it through the committee
  - Kokkos Team represents about a quarter of HPC centric folks at comittee

atomic_ref<T> C++20

- Atomic capabilities like in Kokkos
  - Atomic operations on non-atomic types
  - Almost arbitrary types
- Much better alignment with DOE requirements than std::atomic
- Kokkos and RAJA can drop their own implementations of atomics when this is available

basic_mdspan<T,Extents,Layout,Accessor> C++23

- Based on Kokkos::View
  - Enables all the things Kokkos::View can do
  - Very extensible: could allow PGAS, IO, compression
- Kokkos::View will become thing wrapper around mdspan

Executors C++23

- Getting heterogeneous parallel execution into the standard
- Low level interface, build nicer things on top
  - Parallel STL, Kokkos etc.
- Kokkos Team is instrumental in the design process
  - Helped forge compromise between other participants
  - Thought for direct support of data parallel patterns

Linear Algebra C++23

- BLAS for C++ with nicer interface
- Using mdspan as data handles
  - Scalar type agnostic
  - Data layout aware
  - Support for memory spaces via accessors possible

Kokkos: KPP-3 Impact Goals and Metrics (include all the projects)

Integration Goal 1

- Kokkos is used by ECP applications and software technology projects to run on multiple platforms. This is a shared goal with the Kokkos/RAJA ECP Support project.
- Target all ECP Applications and Software Products
- Metric: Sum over the number of backends used by ECP applications software technology projects to run on different applications in a given year from FY20 up to and including FY23.
- Objective: 200, Threshold: 95, Actual: 35

This goal is on track. Expect passing threshold inFY21 if no technical problems are encountered

- By FY21 four backends in ”production” use for DOE machines (CPUs, Aurora, Frontier, Sierra)
- If all projects at least do regular tests on all four architectures we should get an annual increase of ~40

Tasks for FY20-23

- FY20-21 Support the Kokkos/RAJA ECP development effort for A21 and ATS4
- FY20 Port Kokkos to ATS3 (Crossroads).
- FY20-21 Develop coarse grained tasking capabilities to provide more scheduling flexibility.
- FY21-23 Optimize Kokkos on ATS3 and ATS4.
- FY21-23 Retire implementations of features in Kokkos which can be replaced by ISO C++ standard capabilities.
- FY20-22 Evaluate software stacks for new platforms as they become available and integrate them into Kokkos’ testing suite.
- FY22-23 Start implementing proper C++ executor interfaces to align with future parallelism in the C++ standard.

Primary Risks

- Immature Compilers on untested platform designs
- Time available for new backend development is small