

Kokkos Kernels: FY20 update



Approved for public release

Luc Berger-Vergiat
Sivasankaran Rajamanickam (PI), Seher Acer, Vinh Dang, Nathan
Ellingwood, Evan Harvey, Brian Kelley, Jeremiah Wilke, Ichitaro Yamazaki
Sandia National Laboratories

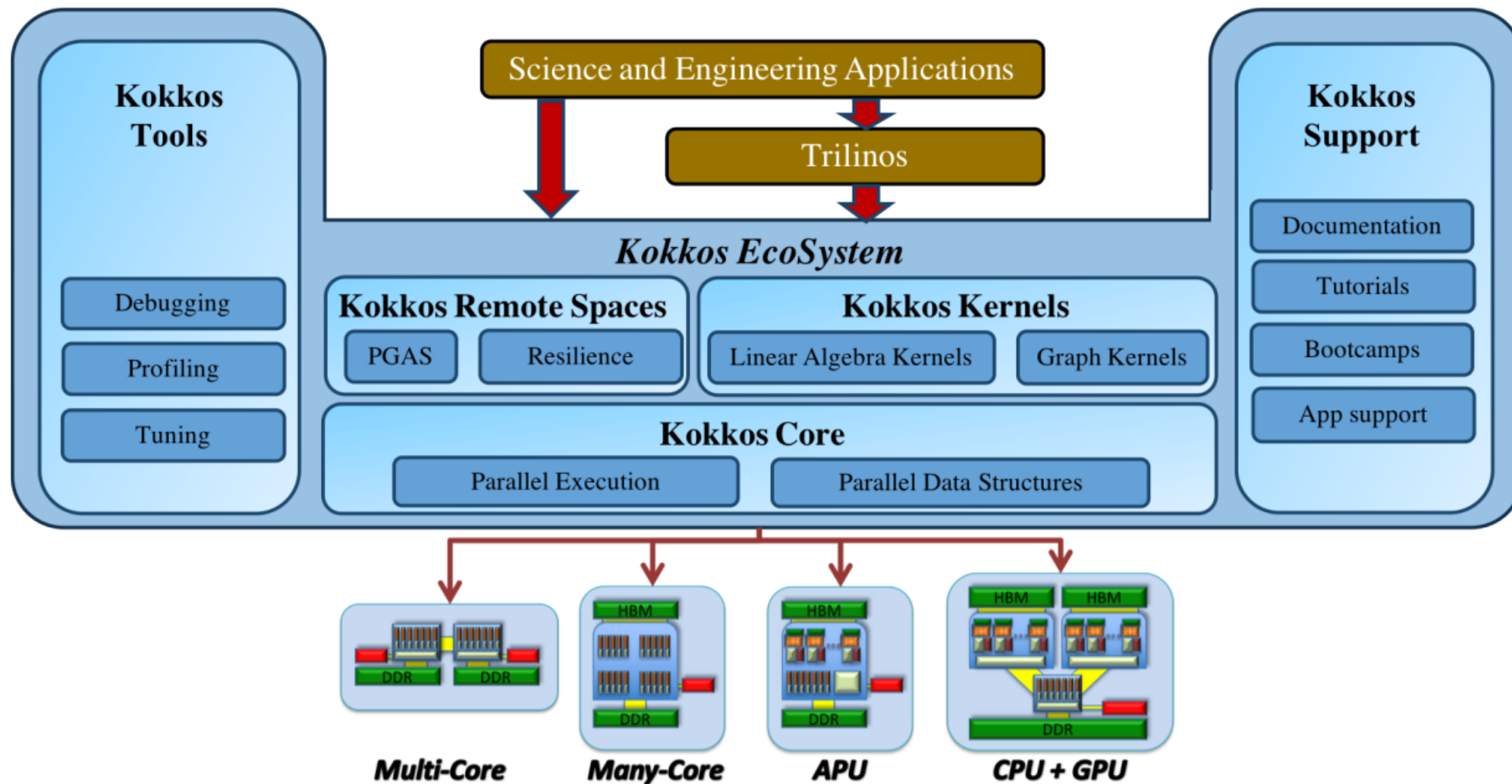
Outline

- Introduction
 - What is Kokkos Kernels
 - Who do we work/partner with
 - What kernels do we provide
- Highlights from FY20
- Planned work
- Questions

Introduction



Kokkos Ecosystem for Performance Portability



Focus of Kokkos Kernels

Provide performance portable dense/sparse linear algebra and graph kernels

- These kernels often account for more than 50% of applications runtime
- Kernels often require different implementations on different architectures for best performance
- High level of expertise is needed to write these performance kernels on top of Kokkos
- Users should not need that high level of expertise to use the kernels
- Focus on performance for platform of interest to DOE

Kokkos Kernels delivers portable, high-performance kernels in a robust software ecosystem to support CSE applications

Focus of Kokkos Kernels

Deliver a robust software ecosystem that can be built open by other software technology and applications

- Production software capabilities that are high performance, portable and turn-key solutions
- A layered testing approach
 - CI testing for pull request with GitHub automation
 - Nightly testing of all backends on various architectures using common type combinations enabled by application users
 - Larger release/integration testing with Trilinos and applications
- Kokkos support through slack, GitHub issues, tutorials, user group meetings and hackathons

Kokkos Kernels delivers portable, high-performance kernels in a robust software ecosystem to support CSE applications

Focus of Kokkos Kernels

Serves as reference implementation of key kernels needed by applications

- Actively work with vendor to introduce high performance implementation of kernels in their libraries
- Provide interface to vendor implementation when they provide better performance or more features
- Actively publish algorithmic results to help the community develop build on top of our work

Kokkos Kernels delivers portable, high-performance kernels in a robust software ecosystem to support CSE applications

Focus of Kokkos Kernels

Actively partner with Applications to identify new opportunities for performance

- Team-level dense, sparse linear algebra
- Team-level data structures (hashmap) and utilities
- (sorting) for better performance
- Fused Kernels
- Symbolic and Numeric separation in interface design

Kokkos Kernels delivers portable, high-performance kernels in a robust software ecosystem to support CSE applications

Collaborations with vendors

NVIDIA

- Summit on Summit meetings
- Bi-weekly work stream meeting to guide NVIDIA's math libraries plans
- Kernel requirements prioritized by applications needs and milestones
- Long history of interaction within CoE
- SpGEMM, GEMM, Solvers are all improved

ARM

- Working with the math libraries team both on algorithms
- SpGEMM, SpMV, Batched linear algebra in ARM PL

Collaborations with vendors

AMD

- Kokkos backend fully implemented enabling Kokkos Kernels backend deployment and testing
- Bi-weekly meetings to improve runtime stability and discuss performance of linear algebra kernels
- Discussion on math library interfaces

Intel (in collaboration with ANL)

- Compact API on KNL
- Kokkos backend experimental and under development
- Kokkos Kernels backend development and compiler tests underway
- Target Kokkos Kernels algorithms will serve as performance benchmarks

Collaborations with ECP Applications

SPARC: state-of-the-art hypersonic unsteady hybrid structured/unstructured finite volume CFD code

- High performance line solvers; batched BLAS on CPUs and GPUs
- Performance-portable programming models

EMPIRE: next-gen unstructured-mesh FEM PIC/multifluid plasma simulation code

- Scalable solvers for electrostatic and electromagnetic systems for Trinity and Sierra architectures
- Thread-scalable, performance-portable, on-node linear algebra kernels to support multigrid methods
- Performance-portable programming models
- Non-linear solvers, discretization, and automatic differentiation approaches

Collaborations with ECP Applications

ExaWind: next-gen wind energy simulation code

- Scalable solvers for Trinity and Sierra architectures
- Thread-scalable, performance-portable, on-node linear algebra kernels to support multigrid methods
- Performance-portable programming models

QMCPACK: Electronic structure code with Quantum Monte Carlo Algorithms

- Team level BLAS and LAPACK within the Kokkos ecosystem

Kokkos Kernels integrated into several applications in an agile manner at all stages from requirements solicitation, designing kernels and integration

Currently available kernels

Dense Linear Algebra (BLAS and Batched BLAS)

- Driven by applications needs
- Include specialization on application use cases (dot based GEMM)

Sparse Linear Algebra

- Sparse Containers (CrsMatrix, StaticCrsGraph)
- Sparse Matrix-Vector Multiplication (SpMV)
- Sparse Matrix-Matrix Addition (SpADD)
- Sparse Matrix-Matrix Multiplication (SpGEMM)

Currently available kernels

Graph Kernels

- Distance 1 and 2 Graph Coloring
- Bipartite Graph Partial Coloring
- Maximum independent set (MIS) based on distance 2 coloring
- Graph coarsening

Sparse Solvers

- Multicolor and Cluster Gauss Seidel
- Two-Stage Gauss Seidel
- Sparse Incomplete LU Factorization (SpILUK)
- Sparse Triangular Solver (SpTRSV)

Highlights from FY20



Graph coloring and coarsening

- Maximum Independent Set for distance 2 neighbors implemented
 - Provides a deterministic implementation
 - Generates high quality seeds for algebraic multigrid coarsening
 - Speed up compared to CUSP on 3D Laplacian and 3D Elasticity problems is 6x on average
- Graph coarsening
 - Generates graph coarsening based on the MIS-2 algorithm described above
 - Improves performance portability and provides simple interface for multigrid algorithm

Gauss-Seidel improvements

Multiple algorithmic improvements in Gauss-Seidel

- Improved parallelization of classic GS using balloon clustering
 - Balloon clusters expose more balanced parallelism scheme
 - Improves GS setup time and runtime per iteration
 - Can have an impact on number of iterations
- New algorithmic approach using two-stage SpMV based implementation
 - Naïve implementation based on matrix operation
 - Substitutes triangular solve with Richardson iterations (typically only 1 or 2 iterations performed)
 - Exposes good parallelism and scaling due to SpMV performance

Interfaces and instantiations

- Simplified sparse algorithm interfaces

- Pass CrsMatrix objects directly instead of views

```
KokkosSparse::spadd_symbolic(handle, a, b, c);  
KokkosSparse::spadd_numeric (handle, alpha, a, beta, b, c);
```

- View interface still available for fine grained memory management

- Always instantiate all types for TPL allowed types

- Types accepted by c interface of vendors libraries are always instantiated
- Improves availability of TPL and simplifies instantiation logic (can reduce compile time)

GEMM improvements

- Half precision
 - Half precision support is added in Kokkos Core and enabled through Kokkos Kernels
 - Allows kernels to target tensor core more easily
- ARM PL integration
 - ARM math library is available using `-D KOKKOSKERNELS_ENABLE_TPL_ARMPL`
 - Provides improved math performance on ARM architecture
- GEMM performance
 - Additional testing and tuning leads to improved heuristic implementation

HIP backend

HIP backend is enabled for AMD accelerators (in collaboration with ORNL, AMD and Cray)

- Will be included in release 3.4.0 (currently under being tested before public release)
- Link against a Kokkos installation with HIP enabled (-D Kokkos_ENABLE_HIP:BOOL=ON)
- Ongoing work to enabled rocBLAS and rocSPARSE (will not be in 3.4.0)
- Comparisons of native Kokkos Kernels vs rocSPARSE shows that improved heuristic are needed in Kokkos Kernels

Planned work



New backends (ongoing work)

- SYCL backend
- OpenMP Target backend

More multiphysics support

- BlockCrsMatrix already exists
- New kernels support for that format
 - SpMV
 - SpGEMM
 - Jacobi style preconditioner
- Provide TPL interface when possible
 - cuSPARSE BSR
 - rocSPARSE BSR
 - MKL BRS
 - Maybe more?

Improvement to SpGEMM

- Revisit symbolic/numeric interface design
 - Better support for triangle counting
 - Symbolic phase should be identical for CrsMatrix and BlockCrsMatrix
 - Can result in speed-up for numeric phase when reusing the symbolic phase
- Improve internal heuristic implementation
 - Further performance can be gained with more fine grained heuristics
 - Potential gains can also be achieved with different memory management schemes

Advanced math capabilities

- Advanced math functions
 - Support electromagnetic analytical solution evaluations
 - Bessel functions of 1st and 2nd kinds
 - Hankel functions of 1st and 2nd kinds
- Matrix visiting pattern
 - Implements generic ways to visit the rows of a CrsMatrix
 - Will accept a functor/lambda and a matrix as input
 - Should allow users to implement diagonal extraction in 5 lines of code!

Thank you for your attention!
Any questions?