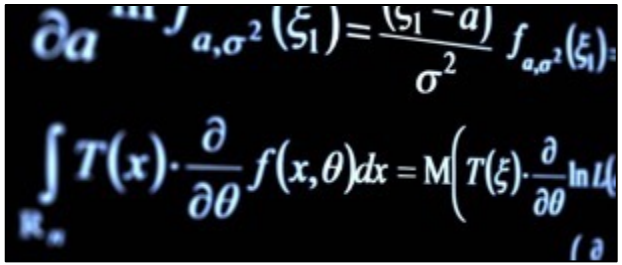
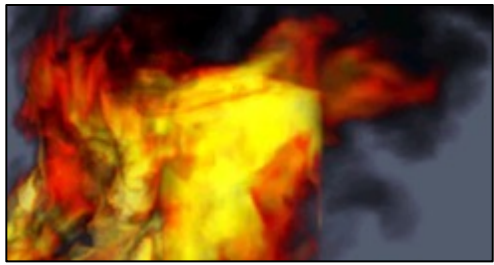
Exceptional service in the national interest

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 Ecosystem 4.0 Update

Unclassified Unlimited Release

Nathan Ellingwood, - 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.

Kokkos 4.0 – what to expect

Sandia National Laboratories

- **Kokkos 4.0 will require C++17**
- Will support C++17, C++20, C++23
  - Support for compilers lacking full C++17 will be dropped
- Allows us to keep testing amount manageable
- Will enable new interfaces and streamlined implementation
- Use of class template argument deduction (CTAD) reduces the need to spell template arguments out
  - Fold expressions help with internal implementation, and improve compile times
  - constexpr if reduces use of clunky substitution failure is not an error (SFINAE) patterns

New compiler minimums

Sandia National Laboratories

Compiler	Version
GCC	8.2
Clang	8
Clang as CUDA compiler	10
Intel	19.0.5
CUDA-NVCC	11
CUDA with Clang as CUDA compiler	10.0.1
ROCM	5.2.0
IntelLLVM (CPU)	2021.1.1
IntelLLVM (SYCL)	2022.2.0
NVC++	22.3
MSVC	19.29
IBM XL	Not Supported
Classic PGI	Not Supported

Discussion at https://github.com/kokkos/kokkos/issues/5285

Kokkos 4.0 – what to expect

Sandia National Laboratories

- HIP backend will be promoted from Experimental
  - Use Kokkos::HIP instead of Kokkos::Experimental::HIP
  - Will support ROCM versions longer
  - For transition time, HIP will be available in both namespaces
- TeamMDRangePolicy!
- Moving SIMD to the core repository – testing and feedback encouraged!
  - Incremental process, all capabilities planned to be moved by Kokkos 4.2

Migrating to 4.0

Sandia National Laboratories

- Build with deprecated code disabled

  -DKokkos_ENABLE_DEPRECATED_CODE_3=OFF

- Kokkos version 3.7 will be maintained for a patch release to ease transition (bug fixes only, no new features)
- Deprecated code has been removed from Kokkos develop branch
  - There are just a handful of exceptions we will leave in for one or two more minor cycles to give more transition time
- Additional details in the release briefing slides: https://github.com/kokkos/kokkos-tutorials/tree/main/Other/ReleaseBriefings

Partial list of code deprecations (3.7.00)

- Do not include private Kokkos headers (use #include “Kokkos_Core.hpp”)
- Reducer join member function taking volatile-qualified arguments are deprecated (remove the volatile qualifier)
- Array reductions with pointer return types (use a Kokkos::View)
- Name your kernels by passing a string as first argument
- InitArguments replaced by InitializationSettings
  - Command line arguments and environment variables updated to increase consistency
- ScopeGuard behavior change with respect to prior initialization
- Kokkos::sort does not accept trailing boolean argument any more
- More details: https://github.com/kokkos/kokkos-tutorials/blob/main/Other/ReleaseBriefings/release-37.pdf

New documentation websites!

Sandia National Laboratories

- Kokkos Documentation now on https://kokkos.github.io
  - Transition to Sphinx syntax
  - More flexibility in site layout and style
  - Better update processes
    - Source for core documentation at https://github.com/kokkos/kokkos-core-wiki
    - Using pull requests with auto deploy
    - Pull requests to improve documentation are welcome!
- KokkosKernels Read The Docs!
  - Additional source code documentation to the wiki
  - https://kokkos-kernels.readthedocs.io/en/latest/developer/index.html

Kokkos documentation

Sandia National Laboratories

Kokkos: The Programming Model

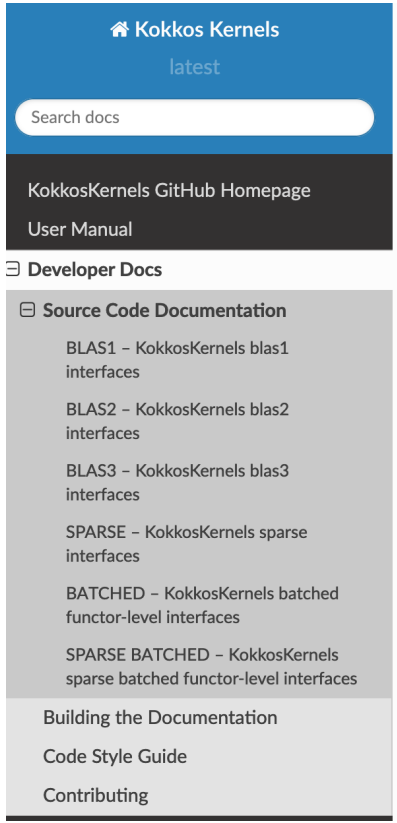
C++ Performance Portability Programming Model

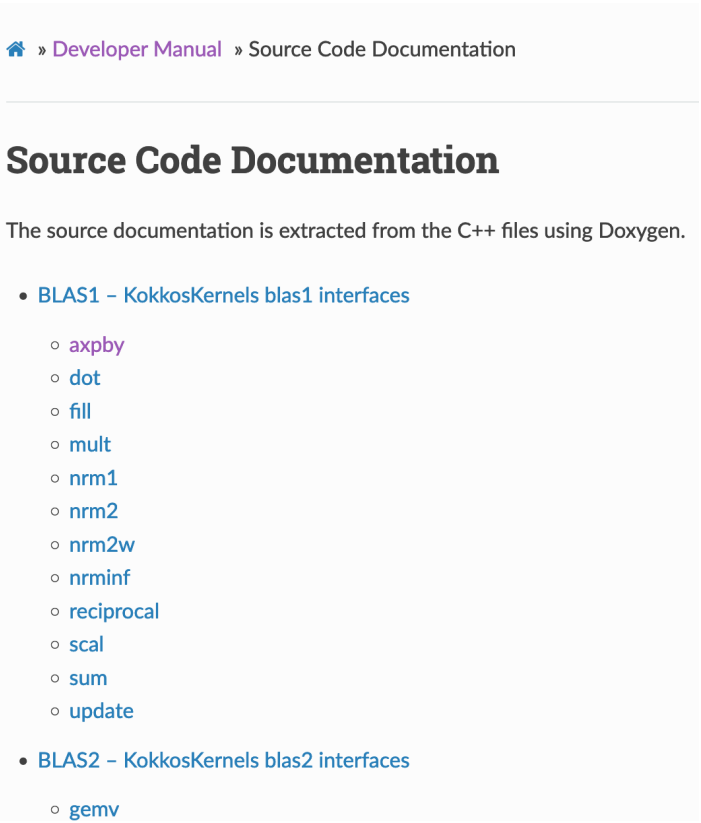
Kokkos Core implements a programming model in C++ for writing performance portable applications targeting all major HPC platforms. For that purpose it provides abstractions for both parallel execution of code and data management. Kokkos is designed to target complex node architectures with N-level memory hierarchies and multiple types of execution resources. It currently can use CUDA, HIP, SYCL, HPX, OpenMP and C++ threads as backend programming models with several other backends development.

The Kokkos EcoSystem includes:

Name	Info	
`kokkos`	(this library) Programming Model - Parallel Execution and Memory Abstraction	Github link
`kokkos-kernels`	Sparse, dense, batched math kernels	Github link
`kokkos-tools`	Profiling and debugging tools	Github link
`pykokkos`	Provides Python bindings to the Kokkos performance portable parallel programming.	Github link
`kokkos-remote-spaces`	Shared memory semantics across multiple processes	Github link
`kokkos-resilience`	Resilience and Checkpointing Extensions for Kokkos	Github link

KokkosKernels documentation





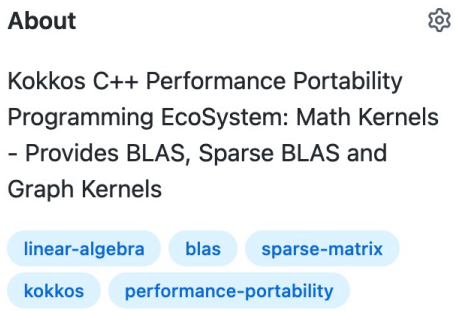
Strengthening community bonds

List of Applications and Libraries

- Add your app to https://github.com/kokkos/kokkos/issues/1950
  - We are planning to add that to a Kokkos website
  - Helps people discover each other when working on similar things

GitHub Topics

- Add kokkos topic to your repo's "About" list of topics
- Click on the topic to get a list of all projects on github with that topic



Additional Support and Resources

Sandia National Laboratories

- Kokkos slack channel: kokkosteam.slack.com
- Github repos
  - https://github.com/kokkos/kokkos
  - https://github.com/kokkos/kokkos-kernels
  - https://github.com/kokkos/kokkos-tools
- Documentation:
  - https://kokkos.github.io/
- The Kokkos Lectures
  - https://kokkos.link/the-lectures

A couple features to advertise…

Sandia National Laboratories

Kokkos std::algorithms

- Provide well known interfaces for C++ standard algorithms
- Limited to 1D Kokkos Views for now
  - Can use iterator interface, or directly passing views which is safer
- >60 algorithms available

```
// C++17 standard
void foo(std::vector<double>& a, std::vector<double>& b) {
  assert(a.size() == b.size());
  std::transform(std::execution::par,
    std::begin(a), std::end(a), std::begin(b),
    [=](const double& in, double& out) { /* do something */ });
}

// Kokkos using View interface (size check inside transform)
void foo(Kokkos::View<double*> a, Kokkos::View<double*> b) {
  Kokkos::transform(Kokkos::DefaultExecutionSpace(), a, b,
    [=](const double& in, double& out) { /* do something */ });
}
```

Multiple Instances

Sandia National Laboratories

- Construct independent instances
  - Allows for overlapping kernels: best for large work per iteration, low count
  - Largely equivalent to CUDA streams or SYCL queues
  - Mini-tutorial at https://github.com/kokkos/kokkos-tutorials/blob/main/Other/ECP-Annualmeeting/2022/KokkosTutorial_ECP_Instances.pdf

```
// Create two instances from streams
auto instances = Kokkos::Experimental::partition_space(
  Kokkos::DefaultExecutionSpace(),1,1);


// Run two kernels which can overlap
parallel_for("F1",RangePolicy<>(instances[0],N),F1);
parallel_for("F2",RangePolicy<>(instances[1],N),F2);
fence();
```

Kernels new features and improvements

Sandia National Laboratories

- Batched linear solvers
  - LU with static pivoting
  - PCG
  - GMRES
- Batched GEMM
  - New heuristics and improved interface
  - Unified interface for all parallelism levels
- Mixed precision linear algebra kernels
- (More in later talks)

- BsrMatrix format
  - Support constant block size sparse matrix
  - Mat-Vec via SpMV interface
  - Mat-Mat via SpGEMM interface
  - Gauss-Seidel smoother


Sandia
National
Laboratories