Kokkos 5.0 Release Briefing

11/25/2025

Outline

Find More

Online Resources:

- https://github.com/kokkos:
  - Primary Kokkos GitHub Organization
- https://kokkos.org/kokkos-core-wiki/tutorials-and-examples.html:
  - Tutorials, video lectures, and examples
- https://kokkos.org/kokkos-core-wiki:
  - Wiki including API reference
- https://kokkosteam.slack.com:
  - Slack workspace for Kokkos.
  - Please join: fastest way to get your questions answered.
  - Can whitelist domains, or invite individual people.

Would like to strengthen community bonds and discoverability

List of Applications and Libraries

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

GitHub Topics

- Use *kokkos* tag on your repos.
- If you click on the topic you get a list of all projects on github with that topic.

Organizational

Content:

- Kokkos Training & Hackathon @ CEA
- Kokkos User Group @ HPSF Community Summit '26 (Europe)
- Kokkos User Group @ HPSF Conference '26 (US)
- Internship program
- Mailing Lists

Kokkos Training & Hackathon @ CEA, France

- *When:* January 12th-16th 2026 (noon to noon)
- *Where:* CEA, Saclay ("near" Paris), France
- *What:*
  - 1 day of basic training: masterclass + exercises
  - 3 days Hackathon to get your hands dirty & discover Kokkos for real:
    - get a code half-ported to GPU,
    - in small teams, with support from the devs., finish porting & optimize for GPU,
    - compete with the other teams for the best performance!

Registration is free, but mandatory, seats are limited!

https://indico.math.cnrs.fr/e/kokkos-hackathon-26

Kokkos User Group @ HPSF Community Summit, Europe

- *When:* February 25th-27th 2026
- *Where:* TU Braunschweig ("near" Berlin), Germany
- *What:*
  - Few HPSF plenary talks
  - Project community gathering:
    Kokkos, Trilinos and WarpX user & developers groups
  - Some Spack, Module & other HPSF projects

Call for Abstracts open!

https://events.academiccloud.de/e/hpsfcs26

Kokkos User Group @ HPSF Conference, USA

- *When:* March 16th-20th 2026
- *Where:* Chicago, IL, USA
- *What:* 2½-days HPSF plenary + 2½-days Project meetings
- *KUG-Content:* Focused on user experiences
  - How do you leverage Kokkos?
  - What are pain points?
  - Kokkos-based libraries of interest to the community

Call for Papers & Registration open!

https://events.linuxfoundation.org/hpsf-conference/

SRP Internship Program

Kokkos will take part in HPSF internship program in 2026
With Sustainable Research Pathways
More info soon

Mailing Lists

Sign up for the Kokkos mailing list to stay up-to-date
with the latest Kokkos news.
https://kokkos.org/community/mailing-lists/

Feature highlights

Minimum compiler requirements after requiring C++20

	Kokkos 4.x	Kokkos 5.0
GCC	8.2.0	10.4.0
Clang (CPU)	8.0.0	14.0.0
Clang (CUDA)	10.0.0	15.0.0
IntelLLVM (CPU)	2021.1.1	2022.0.0
IntelLLVM (SYCL)	2023.0.0	2024.2.1
NVCC	11.0	12.2.0
NVC++	22.3	22.3
ROCm	5.2.0	6.2.0
MSVC	19.29	19.30

Note: Clang (CUDA) allows for CUDA 11.8 as underlying runtime.

Note: MSVC is only actually tested with the latest version.

https://kokkos.org/kokkos-core-wiki/get-started/requirements.html

Deprecated Code 4

Deprecated Code 4 is turned off by default and started to be removed!

- In Kokkos 5.0 `Kokkos_ENABLE_DEPRECATED_CODE_4=OFF` is the default!
- Some deprecated feature will immediately be removed.
- Features where we expect users still needing time, will stick around for a couple more minor releases.

Removed features

- Tasking
- Makefile support

MDSpan-based View Implementation

- This release contains an extensive refactoring of `Kokkos::View`
- View was refactored to use `mdspan`, a C++23 addition to the standard library
  - `mdspan` is backported to C++17/C++20, and our implementation can be found at `github.com/kokkos/mdspan/`
- The goal of this refactor was to provide better library interoperability, more API flexibility, and reduced maintenance burden
- In principle, this update should be fully transparent; i.e. your existing code should work as it did before and we've done extensive testing to ensure this
- What does this mean for applications?
  - We use the same customization points as mdspan now, including accessors and layout mappings
  - In the future, we may provide a mechanism for users to customize these. Would that be useful for people?

Haven't we seen this before??

Yes we did: first attempt to enable in 4.7.0, then disabled in 4.7.1

- Had to disable in Trilinos (and thus Spack) anyway due to needed Trilinos refactor
  - They used lots of `Impl` namespace features from Kokkos
- Got reports in 4.7.0 of performance regressions
  - Addressed with workaround for implicit integer conversion when indexing into `View`
  - Where is a bit more pain to come to clean up 32bit vs 64bit indexing
- More issues when hoped for with older compilers
  - Most are not allowed anymore due to C++20 requirements
  - We are getting initial reports about issues with GCC 10.4 as well as some older CUDA (before 12.6) in corner cases

Recommendation

Consider updating to CUDA 12.8 and GCC 12 if possible – Clang variants are all fine.

C++20 module support

Kokkos 5 added support for C++20 modules

- Tested with clang-19 and expected to work with host backends

Available modules

- `kokkos.core`
- `kokkos.bitset`
- `kokkos.dual_view`
- `kokkos.dyn_rank_view`
- `kokkos.dynamic_view`
- `kokkos.error_reporter`
- `kokkos.functional`
- `kokkos.offset_view`
- `kokkos.scatter_view`
- `kokkos.unordered_map`
- `kokkos.random`
- `kokkos.sort`
- `kokkos.std_algorithms`
- `kokkos.simd`

C++20 module support

See https://github.com/kokkos/kokkos/tree/develop/example/build_cmake_installed_modules

CMake

```
# Enable default source code file scanning
cmake_minimum_required(VERSION 3.28.2)
project(Example CXX)
find_package(Kokkos REQUIRED)
add_executable(example cmake_example.cpp)
# FIXME_MODULES Use Kokkos::kokkos again
target_link_libraries(
  example
  Kokkos::kokkoscore)
```

C++

```
import kokkos.core;
#include <Kokkos_Macros.hpp>

int main(int argc, char* argv[]) {
  Kokkos::initialize(argc, argv);
  [...]
  Kokkos::finalize();
}
```

General Enhancements

View and UnorderedMap Enhancements

- Improved first-touch handling for `View` initialization in Host parallel backends
  - Initialization is now done with `parallel_for` for large allocations
  - `std::memset` is used for small allocations to avoid parallel overhead
  - Threshold is chosen based on benchmarking results
- Extended `UnorderedMap` to support View as `value_type`
  - The `SequentialHostInit` tag can be used during construction for all value types
  - This tag is **mandatory** when `value_type` is a `View`
  - Requires that `UnorderedMap` is allocated in host-accessible memory space

Example:

```
using view_type = Kokkos::View<int*, ExecSpace>;
using map_type  = Kokkos::UnorderedMap<int, view_type, HostSpace>;
map_type umap(Kokkos::view_alloc(Kokkos::SequentialHostInit, ""), N);
```

General Enhancements I

- Add support for type conversions between `simd` types
- Make reduction identity of half-precision types return the correct type
  - Return `[b]half_t` instead of `float` for `reduction_identity<...>` member functions
  - Concerns the functions: `sum()`, `prod()`, `min()`, and `max()`
- Improve `Kokkos::Array` consistency and compatibility with STL `array`
  - Add `[c]begin()` and `[c]end()` methods
  - Add missing `noexcept` specifier to some methods

General Enhancements II

- Print the commit hash of the embedded dependencies at configure time and when calling `print_configuration`
- Desul atomics: Use Clang atomic min/max GCC-style builtins
- Enable running tests on systems with only 2GB of device memory
- Updated `ErrorReporter`: adhere to Kokkos naming conventions and change `get_reports` to return values

API: StaticBatchSize

New execution policy trait: `Kokkos::Experimental::StaticBatchSize<N>`

- Allows users to specify a loop unrolling factor at compile time
- Only effective for `RangePolicy` in the CUDA backend for now

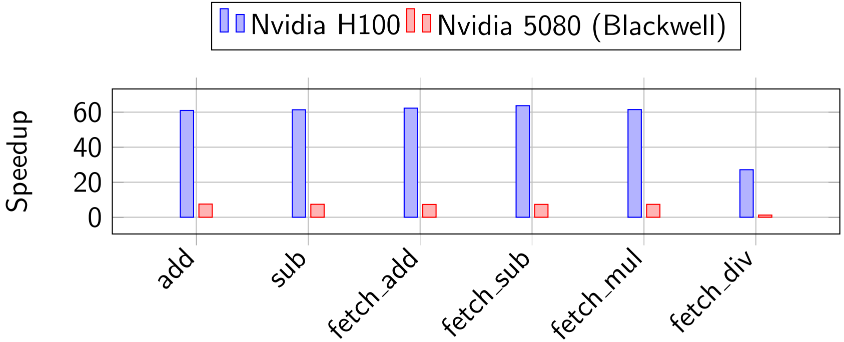
Example usage:

```
// Define a batch size of 4
using unroll_4 = Kokkos::Experimental::StaticBatchSize<4>;
// Use in RangePolicy
Kokkos::RangePolicy<ExecSpace, unroll_4> policy(0, N);
// Execute parallel_for with static batch size
Kokkos::parallel_for(policy, KOKKOS_LAMBDA(const int i) {
  ...
});
```

Backend Updates

CUDA

128 Bit CAS-based device atomics on >= Hopper and CUDA >= 12.8



- Atomics on 10^8 `Kokkos::complex<double>` **without** contention
  - Speedup $\approx$ 60x on H100 and $\approx$ 7x on RTX5080.
  - Same performance for `int128` and `Kokkos::complex<double>`.
  - Division more costly, thus less effect of atomic CAS.

Effect of contention on `atomic_add` on Hopper

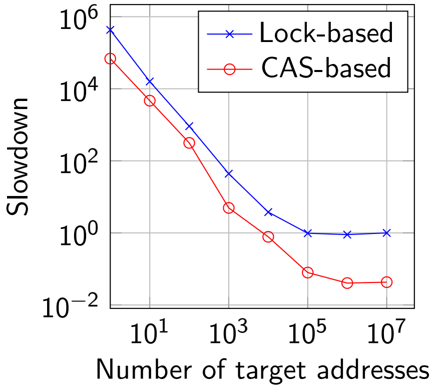


Figure: `atomic_add` of `Kokkos::complex<double>` with 10^8 workers

- Effectiveness of CAS-based atomics reduces similar to Lock-based atomics at high contention.

Leverage larger Kernel Argument

- Allows to launch kernels with up to 32kB of arguments for kernels. Previously it was 4kB.
- Enables us to side-step the "Constant Cache" launch mechanism in Kokkos.
- Effects functors in the 4kB to 32kB size range. No effect on smaller or larger functors.
- This changes the synchronization behavior for functors in this range, due to elimination of an implicit necessary synchronization on constant cache buffer use.
- Does not apply to using Clang as CUDA compiler, nor for GPUs older than Volta (i.e. Compute Capabilities lower than 7).

SYCL

Use unsigned integer type as `size_type` in SYCL

- `SYCL` now uses an unsigned integer type as `size_type`.
- Now unsigned integer type across all backends.

OpenMPTarget

Allow parallel_scan to start anywere in OpenMPTarget

- Previously `parallel_scan` with a `RangePolicy` needed to start at index 0
- Now any starting index smaller than the end index is supported.

Upcoming removal of OpenMPTarget

We decided to remove OpenMPTarget in an upcoming release!

- Never reached feature parity.
- Lower performance than native backends (CUDA, HIP, SYCL)
- Practically no users.
- Little interest in support by any institution.

OpenACC

Allow parallel_scan to start anywere in OpenACC

- Previously `parallel_scan` with a `RangePolicy` needed to start at index 0
- Now any starting index smaller than the end index is supported.

Support `Kokkos_Random` algorithms API in OpenACC

- OpenACC now supports the `Kokkos_Random` algorithms API.
- Can be inefficient if the actual team size is different from the default team size.

Support `partition_space` API in OpenACC

- OpenACC now supports the `partition_space` API.
- Execution space instances created by `partition_space` will use OpenACC async IDs in a reserved range (from 64 to 191), which are assigned in a round-robin manner.

Support custom scalar reduction in `parallel_reduce` with a `RangePolicy` in OpenACC

- OpenACC now supports custom scalar reduction with `parallel_reduce` and `RangePolicy`.
- Supports both built-in reducers with custom scalar types, and custom reducers with custom scalar types.

HIP

Improved performance and new architecture support

- Fix a performance regression introduced in 4.6 when using lightweight kernel (`Experimental::WorkItemProperty::HintLightWeight`) in `parallel_reduce`
- Prefer smaller block sizes for `parallel_for` when the requested parallelism is less than the available concurrency
- Use atomic builtins for `atomic_fetch_min/max` with floating point types instead of our own implementation
- Add support for `Navi4` architecture (Radeon AI PRO R9700, Radeon RX 9070 XT)

ROCm 7.1

- Avoid using ROCm 7.1 if possible: **you may get incorrect results**
- On MI100 and MI200 series, use `-DKokkos_ENABLE_IMPL_HIP_MALLOC_ASYNC=OFF`
- On MI300 series, we cannot compile the testsuite yet. Very likely that you will also need to use `-DKokkos_ENABLE_IMPL_HIP_MALLOC_ASYNC=OFF`

Build System Updates

Defaults and Requirements

- New (unsurprising) defaults
  - `Kokkos_ENABLE_DEPRECATED_CODE_4=OFF`
  - `Kokkos_ENABLE_DEPRECATED_CODE_5=ON`
- CMake Requirements
  - CMake 3.22 is the new miminum
  - CMake 3.25.2 is the minimum for `CUDA` CMake language support

Support for multiple CMake Languages

- Kokkos now offers support for multiple CMake languages in one build
  - Allow a single Kokkos installation to be used with the CMake languages `CXX` and `CUDA/HIP`.
  - Kokkos sets the flags for all CMake languages that work with the selected backend.
  - Rules for `kokkos_launch_compiler` stay unchanged: Used for `CUDA` backend **without** `nvcc_wrapper` as `CXX_COMPILER`
  - Enable via `Kokkos_ENABLE_MULTIPLE_CMAKE_LANGUAGES=ON`

Support for multiple CMake Languages

CMake

```
find_package(Kokkos)

# use Kokkos in a CUDA language project
add_library(language_lib library.cu)
target_link_libraries(language_lib PUBLIC Kokkos::kokkos)

#use Kokkos in a CXX project
add_library(cxx_lib library.cpp)
target_link_libraries(cxx_lib PUBLIC Kokkos::kokkos)

#use both in an executable
add_executable(example cmake_example.cpp)
target_link_libraries(example PUBLIC language_lib cxx_lib)
```

- Look into examples/build_cmake_installed_multilanguage

Windows experimental shared library support

- Experimental support for shared library builds in Windows
  - Kokkos now supports `BUILD_SHARED_LIBS=ON` in Windows builds
  - Includes `WINDOWS_EXPORT_ALL_SYMBOLS=ON` for all Kokkos-core libraries

Deprecations

Deprecations

New `Kokkos_ENABLE_DEPRECATED_CODE_5` (= `ON` by default)

- Deprecate `KOKKOS_ATTRIBUTE_NODISCARD` macro
- Deprecate `Owning,ObservingRawPtr` aliases
- Deprecate `Random_XorShift{64,1024}_Pool::init`
- Deprecate `basic_simd::{const_}where_expression`
- Deprecate support for using nested OpenMP parallel regions without nested OpenMP enabled
- Deprecate creating Kokkos::OpenMP instances inside OpenMP parallel regions

Deprecations in `Kokkos::View`

Various type alias deprecations and replacements in `View`

- `HostMirror` → `host_mirror_type`
- `scalar_array_type` → `data_type`
- `const_scalar_array_type` → `const_data_type`
- `non_const_scalar_array_type` → `non_const_data_type`
- `array_type` → `type`
- `DynamicView::array_type` → `DynamicView::uniform_type`

Various deprecations and replacements in ErrorReporter

- `getCapacity()` → `capacity()`
- `getNumReports()` → `num_reports()`
- `getNumReportAttempts()` → `num_report_attempts()`
- `getReports()` → `get_reports()`

```
// before
std::vector<int> reporters;
std::vector<report_type> reports;
error_reporter.getReports(reporters, reports);

// now uses structured bindings
auto [reporters, reports] = error_reporter.get_reports();
```

Note

- `Kokkos_ENABLE_DEPRECATED_CODE_4 = ON` → `OFF`

Breaking Changes

Removed options and Version bumps

Makefiles

- Deprecated in 4.6, dropped in 5.0.
- Removes the burden of maintaining two build systems

Kepler architecture (Nvidia GPU)

- `sm_3x` not supported since CUDA 12.
- Dropping support for `Kokkos_ARCH_KEPLER30/32/35/37`.

Google Benchmark

- v1.8.3 is the new minimum.

Task-based parallelism

- Rarely used. Deprecated in 4.5. Now, dropped.
- No more: `TaskSingle`, `task_spawn`, `host_spawn`, `BasicFuture`, etc.

Parallel dispatch: `parallel_for/scan/reduce`

- Dispatch only within the Kokkos execution environment.
- Abort if used before `initialize()` or after `finalize()`.

Execution space chosen by default in `Kokkos::Graph::submit()`

- Used to be the execution space specified during `Graph` construction.
- Now, defaults to `Kokkos::DefaultExecutionSpace`.
- Matches semantics of other parallel constructs in Kokkos.

Default SIMD vector width

- From now, choose the largest SIMD vector width available for type `T`.
- Example (with AVX512): `Kokkos::Experimental::simd<float>` maps to:

```
before:
    Kokkos::Experimental::basic_simd<float, avx512_fixed_size<8>>
after:
   Kokkos::Experimental::basic_simd<float, avx512_fixed_size<16>>
```

- Specifying desired vector width still available:
  `Kokkos::Experimental::simd<float, 8>`

Memory trait alias `Kokkos::MemoryRandomAccess`

- Also meant `Kokkos::Unmanaged` memory trait. Now, fixed.
- Older meaning supported via `Kokkos_ENABLE_DEPRECATED_CODE_4`.

Bug Fixes

Bug Fixes: General

- Track modification for resize only if `DualView` is not using a single device
- Fix MSVC floating-point value does not fit in required floating-point type warning from `reduction_identity`
- Properly delete `Timer` copy constructor and copy assignment operators
- Fix RISC-V support (compiler check at configuration time and missing semicolons at compile time)
- Corrected `bit_width` return type to be `int` instead of `T` to align with the standard library
- Work around a performance regression related to index computation in the mdspan-based `View`

Bug Fixes: HIP & CUDA

- Launch work graph on the instance in the execution policy
- Fix incorrect results from `parallel_reduce` with `LaunchBounds` values smaller than 32

Bug Fixes: OpenMP

- Fix `partition_space` abort when requesting more partitions than available threads

Bug Fixes: SIMD

- Fix a configure-time failure when SVE is enabled and unit tests are disabled

How to Get Your Fixes and Features into Kokkos

- Fork the Kokkos repo (`https://github.com/kokkos/kokkos`)
- Make topic branch from *develop* for your code
- Add tests for your code
- Create a pull request (PR) on the main project *develop*
- Update the documentation (`https://github.com/kokkos/kokkos-core-wiki`) if your code changes the API
- Get in touch if you have any question (`https://kokkosteam.slack.com`)