Kokkos 5.1 Release Briefing

4/14/2026

5.1 Release Highlights

- Organizational
- Feature Highlights
- General Enhancements
- Backend updates
- Build system updates
- Deprecations and other breaking changes
- Bug Fixes

Online Resources:

- Visit https://kokkos.org
  - Community>Mailing Lists: Subscribe to kokkos-announcements@lists.hpsf.io
  - Community>Chat: Join us on Slack (https://kokkosteam.slack.com/)
  - Community>Tea Time: Call in to our webinar series
  - Documentation>Get Started
  - Documentation>Programming Guide
  - Documentation>API References
  - Documentation>Tutorials

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:

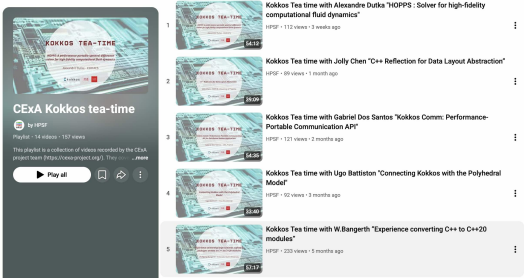
- Kokkos User Group @ HPSF Conference '26 (US)
- Mailing Lists

- Videos are online
  - Browse the agenda with slides uploaded and recordings
  - Playlist with all the presentations available on HPSF's YouTube Channel
- Checkout the trip report on our blog
  `https://kokkos.org/blog/2026-03-KUG-report/`

- Ensure reproducibility
  - Version-specific citations for Core and Kernels since 5.0
  - Format: *Kokkos Core v5.0.2, DOI: 10.5281/zenodo.18408291*
  - See release page to find the specific DOI for the version you used
- In addition to that, please also cite relevant papers describing the component of the Ecosystem you used

Visit `https://kokkos.org/citing-kokkos/`

- On the 3rd Wednesday of the month at 4 PM Paris time (usually 10 AM Eastern, 8 AM Mountain, 11 PM Tokyo time)
- Find the playlist on HPSF YouTube channel
- Reach out if you would like to get invited to speak



Next session scheduled for tomorrow

Parthenon – a performance portable block-structured adaptive mesh refinement framework by Philipp Grete

Visit `https://kokkos.org/community/tea-time/`

Kokkos 5.2: Major Removal

- Most `KOKKOS_ENABLE_DEPRECATED_CODE_4` paths will be **removed**.
- **Action:** Compile with deprecation warnings enabled to identify remaining 4.x legacy code.

Temporary Reprieve (Final Removal in 5.3)

These remain in 5.2 but will be deleted in the following release:

`MemoryManaged` → Use `MemoryTraits<>`

`View::HostMirror` → Use `View::host_mirror_type`

- *Note: Version 5.3 will have zero exceptions for 4.x legacy code.*

Final Transition

The legacy `View` implementation will be removed in 5.3.

- **Status:** `std::mdspan`-based `View` has been the default since 5.0.
- **Contingency Flag Removal:** `-DKokkos_ENABLE_IMPL_VIEW_LEGACY=ON` will no longer be available.

Are you still using the legacy implementation?

Please reach out to the team via GitHub or Slack!

We need to address any blocking bugs or regressions before 5.3.

Feature highlights

Now defining the following concepts

- `Kokkos::ExecutionSpace`
- `Kokkos::MemorySpace`
- `Kokkos::ExecutionPolicy`
- `Kokkos::TeamHandle`
- `Kokkos::Reducer`

```
// Old way: SFINAE-based constraint
template <typename Exec, class Numbers>
std::enable_if_t<Kokkos::is_execution_space_v<Exec>>
    crunch1(Exec const& exec, Numbers numbers) { /* ... */ }

// With concepts introduced in Kokkos 5.1
template <Kokkos::ExecutionSpace Exec, class Numbers>
void crunch2(Exec const& exec, Numbers numbers) { /* ... */ }
```

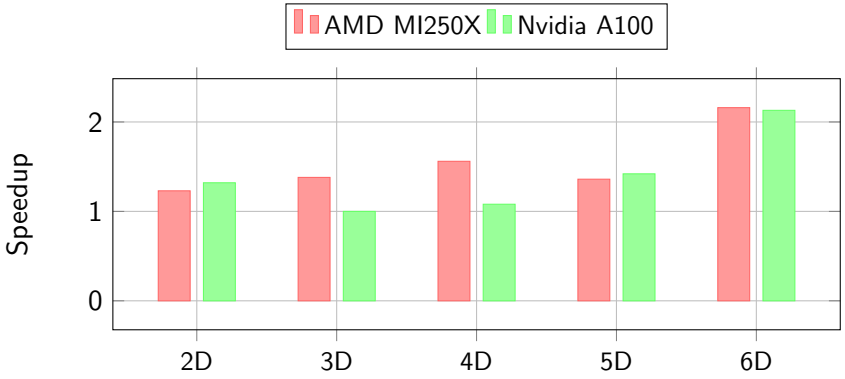
Improved performance for `MDRangePolicy` for all rank (2 to 6) for `CUDA`, `HIP` and `SYCL`

- Rewrite internal logic for computing functor indices
- Fix performance discrepancies between iterate Left and Right

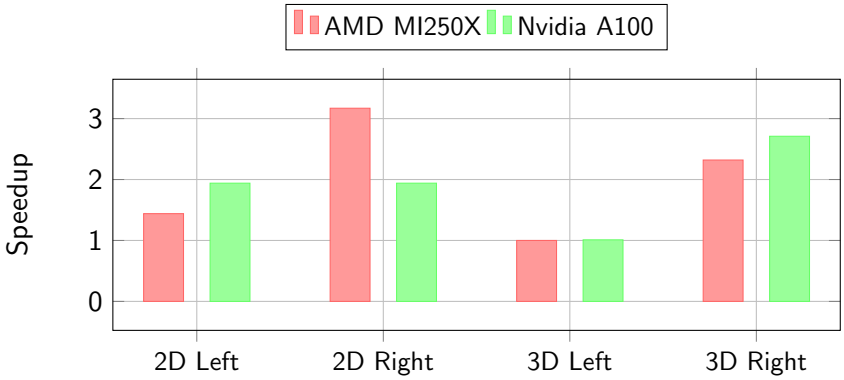
Note: Doesn't concern nested multidimensional policy like `TeamThreadMDRange` or `TeamVectorMDRange`

Change the default tile size for `CUDA`, `HIP`, and `SYCL` backends. Use more threads for the dimension that accesses memory in a coalesced way.

- Lower register usage for the same kernel
- Better memory throughput for memory-intensive kernels.
- Better GPU occupancy for simple kernels.



- Stream Add kernel ($C = A + B$) for different dimensions.



- Stencil kernel with 7 points and 9 points in 2D and 3D respectively.

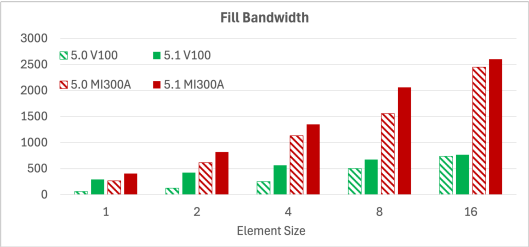
General Enhancements

Now providing all common math functions from the standard library

- Basic operations
  - `remquo`
- Nearest integer floating point operations
  - `round`, `lround`, `llround`, `rint`, `lrint`, `llrint`
- Floating point manipulation functions
  - `frexp`, `ldexp`, `modf`, `scalebn`, `scalebln`, `ilogb`, `nexttoward`
- Classification and comparison
  - `fpclassify`, `isnormal`, `isgreater`, `isgreaterequal`, `isless`, `islessequal`, `islessgreater`, `isunordered`

- Provide more quadruple precision floating-point manipulation functions
- Implemented `norm` free function acting on complex numbers
- Added non-standard `rcp` reciprocal function
- Increased `half` and `bhalf` support
  - Add missing `denorm_min` numerical trait specialization
  - More reduced-precision math functions

Improved `deep_copy` performance for scalar view fill using `StaticBatchSize`



- Enable ScatterView contribute into a View that is a rvalue

```
contribute(subview(2d_view, ALL, 0), scatterview);
```

- Use `Array::size_type` for subscript operators

```
Kokkos::Array arr{0, 1, 2, 3, 5, 7};
enum { zero = 0 };
// now can also index with class object that are convertible to size_t
std::integral_constant<std::size_t, 1> one;
arr[zero] = arr[one] = arr[2] = 42;
```

- Enable MPI detection with PALS

- Added bitwise operators to all simd masks and simd vectors of integral types
  - Bitwise ops: `~`, `&`, `|`, `^`, `&=`, `|=`, `^=`
- Added simd memory permute functions
  - `[unchecked|partial]_gather_from(in, indices, simd_flag)`
  - `[unchecked|partial]_scatter_to(simd, out, indices, simd_flag)`
  - Replace memory permute functions in deprecated `where_expression`

Backend Updates

CUDA

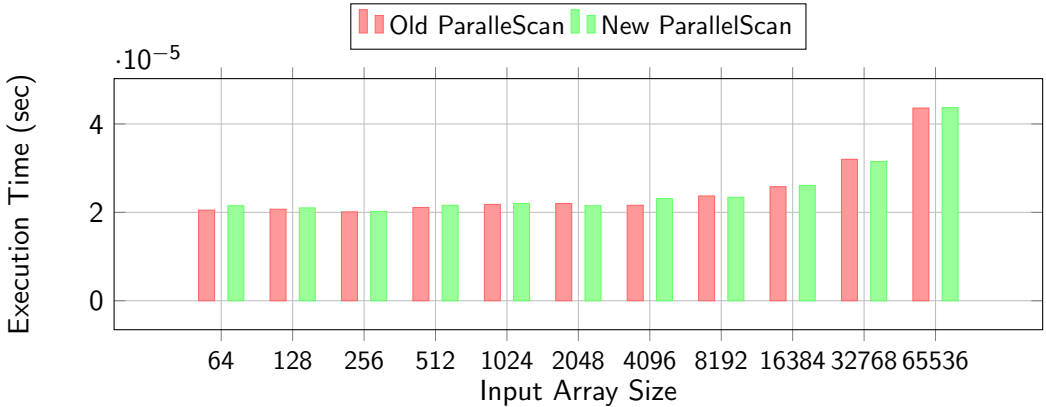
- Added support for `Kokkos_ARCH_BLACKWELL103` for B300 GPUs (requires `CUDA` $\geq$ 12.8).
- Fixed building with `clang` when enabling both `CUDA` and `OpenMP`.
- Added support for the flags `-Ofc` and `--fdevice-time-trace` to `nvcc_wrapper` (Useful for getting insights into compile time).

`nvcc` $>$ 12.8 allows to use `NEON` instructions.

- With `Kokkos_ARCH_NATIVE=ON` we run checks if the compiler supports `NEON` and `SVE`. If the compiler supports them we enable them.
- For all manually specified ARM architectures we enable them if the architecture supports them by specification. We do **not** check if the compiler supports them.

HIP

- Added support for `Kokkos_ARCH_AMD_GFX950` for MI350 and MI355 (requires `ROCm` $\geq$ 7.0 recommended $\geq$ 7.1)
- Added support for `bhalf_t` (in `Kokkos::Experimental`).
- Added math functions for `half_t` and `bhalf_t` instead of casting to `float`.
- Workaround race condition reported in `parallel_scan` when running on MI300A
  - Added an extra synchronization at the end of `parallel_scan` implementation.
  - Seems to be a compiler bug on MI300A.
  - Extra overhead is negligible.
- Implemented support for `StaticBatchSize` policy trait



- parallel_scan Performance on MI300A

Build System Updates

Kokkos now uses `find_package(HIP)` to find `ROCm`. The search includes the path in the environment variable `ROCM_PATH`.

- Allows us to get the `ROCm` version independently of the compiler version.
- **Problem**: `find_package(HIP)` runs autodetection of GPU architectures if `GPU_TARGETS` is not defined as `CMake` variable.
- We warn if the architecture in `GPU_TARGETS` does not match the device architecture enabled in Kokkos. This can happen e.g. when cross-compiling.

Breaking Changes

- `TeamPolicy` now validates the `vector_length` argument at construction time and aborts with a descriptive error if it is out of range, making misconfiguration bugs easier to catch.
  - `TeamPolicy` now requires `vector_length` to be a power of two.
- The value returned by `scatter_view.access()` is no longer copyable or movable.
- Kokkos now detects and reports an error if an execution space is outside its valid lifetime scope, i.e., before `initialize()` or after `finalize()`.
  - An example has been added to show users how to migrate from a static execution space variable to comply with the new lifetime scope enforcement.

```
Kokkos::DefaultExecutionSpace replace_static_execution_space() {
  // store the object in a std::optional so it can be released
  static std::optional<Kokkos::DefaultExecutionSpace> exec;
  // acquire resources on 1st invocation in a threadsafe manner
  [[maybe_unused]] static bool once = [] {
    std::tie(exec) = Kokkos::Experimental::partition_space(
        Kokkos::DefaultExecutionSpace(), 1);
    // ensure the underlying object get destroyed before finalize
    Kokkos::push_finalize_hook([] { exec.reset(); });
    return true;
  }();
  KOKKOS_ASSERT(exec.has_value());
  return *exec;
}
```

- Kokkos now warns when an OpenMP execution space instance is created inside an active OpenMP parallel region.
- When enabling nested OpenMP: create execution space outside of the outermost OpenMP parallel region

```
void foo() {
  #pragma omp parallel
  {
    bar(OpenMP()); // not allowed
  }
}

void foobar() { // below is OK
  auto instances = partition_space(OpenMP(), 1, 1);
  #pragma omp parallel num_threads(2)
  {
    bar(instances[omp_get_thread_num()]);
  }
}
```

The OpenMPTarget backend was removed!

Original motivation for backend during ECP:

- Hedge against SYCL failing for Intel GPUs
- Interoperability with other OpenMPTarget based codes
- Common GPU backend for multiple compilers

But it never worked out, and rationale got lost:

- Performance never got good enough
- Backend wasn't feature complete
- Multi compiler support was flaky (only upstream LLVM really worked)
- No one used it
- SYCL was ok on Intel

Bug Fixes

- Fix `reduction_identity` with the `BAnd` reducer
- Update `team_fan_{in|out}` member functions of `ThreadsExecTeamMember` not to call host-only functions on the device
- HPX: Ensure that execution space instances fence when they are destructed
- Correctly identify GCC and LLVM Clang on macOS

The atomics for integers were not restricted on the sizes for which they are `lock-free` (except with `MSVC`).

- If you used 128bit atomics on integers the linker needed to link `latomic`.
- **Problem**: Linking `latomic` by anyone outside of Kokkos' own build system results in **unsafe** atomic operations.
- Kokkos' atomics on integers with 128 bit or more use our internal lock-tables. Linking `libatomic` is **neither** necessary **nor** recommended.

- Add missing constexpr specifiers on `conj()`, and for the `real()` and `imag()` non-member functions taking complex numbers
- Make overloads of `isnormal` compliant with std

- Use intrinsics when calling `min` and `max` on simd vectors of integral types

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)