Kokkos 5.2 Release Briefing

08/04/2026

5.2 Release Highlights

- Organizational
- Feature Highlights
- General Enhancements
- Backend updates
- Build system updates
- Deprecations and other breaking changes
- Bug Fixes
- Kokkos Tools 5.2

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)
- Save the Date
- 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/`

KUG 2027 @ HPSFCon

- April 12th - 16th in Montreal Canada
- Save the date website: https://events.linuxfoundation.org/hpsf-conference/
- Call for submissions expected mid September

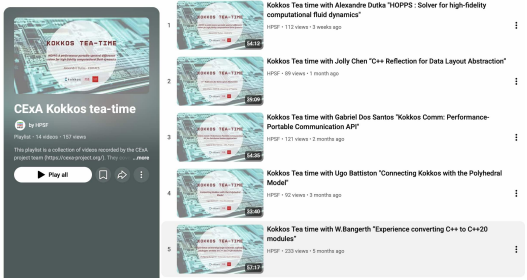
Kokkos Tutorial @ SC26

- November 15th-20th Chicago, IL (exact date tbd)
- Beginner tutorial with hands-on
- Let your colleagues know!

- Ensure reproducibility
  - Version-specific citations for Core and Kernels since 5.0
  - Format: *Kokkos Core v5.2.0, DOI: 10.5281/zenodo.21648054*
  - 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



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

Kokkos 5.2: Major Removal

Most `KOKKOS_ENABLE_DEPRECATED_CODE_4` paths has been **removed**.

Final Removal in 5.3

These remain in 5.2 but will be removed in 5.3:

`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.

Extended support for 4.7 has reached its end.

Feature highlights

Kokkos always allowed reuse of functions on host and device

```
KOKKOS_INLINE_FUNCTION
double awesome(double val, int a) {...}

void bar_host() {
  double val = awesome(1.0, 2);
  parallel_for("bar", N, KOKKOS_LAMBDA(int i) {
    double val2 = awesome(val, 3);
  });
}
```

Kokkos always allowed reuse of functions on host and device

```
KOKKOS_INLINE_FUNCTION
double awesome(double val, int a) {...}

void bar_host() {
  double val = awesome(1.0, 2);
  parallel_for("bar", N, KOKKOS_LAMBDA(int i) {
    double val2 = awesome(val, 3);
  });
}
```

But what if `awesome` contains parallelism itself?

You need to distinguish code paths!

- Different functions.
- Policy factory.
- `if constexpr`

```
template<class ExecOrTeam>
KOKKOS_FUNCTION double awesome(const ExecOrTeam& exec, int M) {
  if constexpr (is_execution_space_v<ExecOrTeam>) {
    parallel_for(RangePolicy(exec, 0, M), KOKKOS_LAMBDA(int i) {
      ...
    });
  } else {
    parallel_for(TeamVectorRange(exec, 0, M), [&](int i) {
      ...
    });
} }
```

You need to distinguish code paths!

- Different functions.
- Policy factory.
- `if constexpr`

```
template<class ExecOrTeam>
KOKKOS_FUNCTION double awesome(const ExecOrTeam& exec, int M) {
  if constexpr (is_execution_space_v<ExecOrTeam>) {
    parallel_for(RangePolicy(exec, 0, M), KOKKOS_LAMBDA(int i) {
      ...
    });
  } else {
    parallel_for(TeamVectorRange(exec, 0, M), [&](int i) {
      ...
    });
} }
```

This still warns about calling host-only `parallel_for` from host-device function!

Generalizes `RangePolicy` to support both execution policies and team-level ranges

Enables writing generic code callable from both host and device:

```
template <class Exec, class X, class Y>
KOKKOS_INLINE_FUNCTION void sum_views(const Exec& exec,
                                      const X& x, const Y& y) {
  Kokkos::parallel_for(Kokkos::RangePolicy(exec, 0, x.extent(0)),
    KOKKOS_LAMBDA(const int& i) { x(i) += y(i); });
}
```

- Can be called with execution space on host: `sum_views(exec_space, x, y);`
- Can be called with team handle in device code: `sum_views(team, x, y);`

- `std::mdspan` uses different style of arguments than classic `View`

```
View<DataType, LayoutType, MemorySpace, MemoryTraits>

mdspan<ElementType, ExtentsType, LayoutType, AccessorType>
```

- `DataType` ⇒ `ElementType` + `ExtentsType`
- `LayoutType` ⇒ `LayoutType`
- `MemorySpace` + `MemoryTraits` ⇒ `AccessorType`

Kokkos Views now support mdspan-compatible template parameters

- Enables specifying index types other than `size_t` for potentially smaller View memory footprint
- Aligns with ISO C++ standard `std::mdspan`

Need to use `mdspan` compatible layout and accessors from Kokkos:

```
layout_left layout_right layout_stride
Experimental::layout_right_padded<Pad> Experimental::layout_left_padded<Pad>
Experimental::Accessor<ElementType, MemSpace, MemTraits>
```

An Example:

```
// before
View<T**, LayoutLeft, HostSpace, MemoryTraits<>>
// equivalent in new style:
View<T, dextents<size_t, 2>, layout_left, Accessor<T, HostSpace, MemoryTraits<>>>
```

Current Limitations:

- None of the template arguments are defaulted.
- Compiler warnings inside Kokkos when using *signed* index types.
- Arbitrarily mixing static/dynamic extents is not well tested.
- Custom layouts and accessors not officially supported yet.

New high-quality pseudo-random number generator with reproducible stream partitioning

- 2^{64} streams from one 64-bit seed; period $\geq 2^{64}$ (expected $\sim 2^{255}$)
- Pool can be initialized *in parallel*, faster than XorShift for large pools
- `seed_offset` enables bit-reproducible distributed partitioning
- Drop-in replacement for `Random_XorShift64_Pool`

```
// Pool 0 handles streams [0, 500), pool 1 handles streams [500, 1000)
// Identical results to a single pool of 1000 states with seed_offset 0.
Kokkos::Random_SFC64_Pool<>
  pool0(seed, /*seed_offset=*/0,   /*num_states=*/500),
  pool1(seed, /*seed_offset=*/500, /*num_states=*/500);

// ... use pool0 / pool1 exactly like Random_XorShift64_Pool
```

Extends `MDRangePolicy` to support one-dimensional ranges

- Enables rank-1 construction:

```
MDRangePolicy<ExecSpace, Rank<1>>(0, N[, tile_size])
```

- GPU backends (CUDA/HIP/SYCL) use tile size 256
- Host backends (OpenMP/Threads) use chunk-size logic like `RangePolicy`
- Includes full `parallel_for` and `parallel_reduce` support

New single-argument overload for mirroring and copying Views

```
// Old API (two arguments required)
auto mirror = Kokkos::create_mirror_view_and_copy(space, view);
```

- Problem: If `view` uses `SharedSpace` using `HostSpace` as space argument will allocate a new `View`

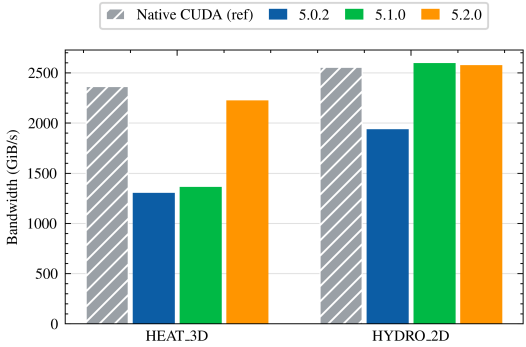
Always get a host accessible copy - avoiding unnecessary allocation:

```
// New convenience overload (one argument)
auto mirror = Kokkos::create_mirror_view_and_copy(view);
```

Consistent with `create_mirror()` and `create_mirror_view()`, simplifying View management

General Enhancements

- Improvements on the MDRange internals on device when the iteration domain is small enough



- Add `std::uint32_t` support for NEON
- Improve the performance of generator constructors on SVE
- Make the flag argument for memory operations optional

```
Experimental::simd<T> vec(ptr, Experimental::simd_flag_default);
Experimental::simd_unchecked_store(vec, ptr,
    Experimental::simd_flag_default);
```

Becomes:

```
Experimental::simd<T> vec(ptr);
Experimental::simd_unchecked_store(vec, ptr);
```

- Rename `native_graph()` and `native_graph_exec()` to backend specific names `{cuda,hip,sycl}_graph()` and `{cuda,hip,sycl}_graph_exec()`
- Add a `GraphNodeKind` enum and allow to retrieve the node type
  - at compile time: `NodeType::node_kind`
  - at runtime: `node.get_node_kind()`
- Allow access to vendor nodes with `node.{cuda,hip,sycl}_node()`

Improve performance of load/store atomic by leveraging compiler built-ins instead of generating them via CAS

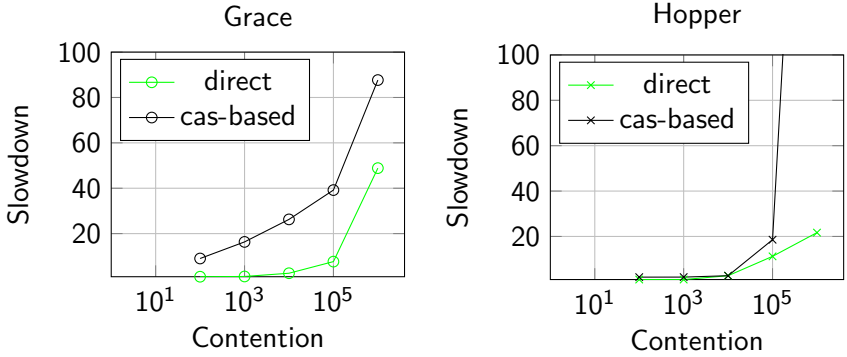


Figure: 1e9 `atomic_load/store` operations on `int` with various contention

- Introduce `KOKKOS_FORCEINLINE_[CLASS_]LAMBDA` macros
- Support `FORCEINLINE` macros with MSVC
- Support building and using Kokkos libraries as C++ modules with MSVC and GCC 16
- Throw a new `Experimental::BadAlloc` exception on allocation failures
- Increase the maximum level 1 scratch size from 20MB to 80MB

- Make execution space's move constructor and assignment operator `noexcept`
- Make `View`'s move constructor `noexcept`
- Add signed `index_type` and unsigned `size_type` aliases to execution and memory spaces
- Promote numeric traits out of the `Experimental::` namespace into `Kokkos::`
- Use the mathematical constants from the standard library

- Extract iterator related functions from `algorithms` to `core`
  - `Kokkos::Experimental::begin`
  - `Kokkos::Experimental::end`
  - `Kokkos::Experimental::cbegin`
  - `Kokkos::Experimental::cend`
  - `Kokkos::Experimental::distance`
- Check preconditions on view extents in the algorithms

Backend Updates

CUDA

- Support more than 48kB of scratch memory request for level 0 (shared memory)
- Enable C++23 support with NVCC (available since CUDA 13.3.0)

HIP

- Add AMD GFX1151 (Strix Halo / Radeon 8060S) architecture support
- Add AMD Radeon 860M / RDNA3.5 / gfx1152 architecture support
- Add AMD GFX1101 (Radeon RX 7800 XT, RX 7700 XT, RX 7700) architecture support
- Improve performance of `deep_copy(v, 0)` on MI300A

SYCL

- Fix Windows build issues

OpenACC

- Fix OpenACC `parallel_scan` chunk boundary race

NextSilicon

- Add initial and preliminary ‘NextSilicon‘ support

HPX

- Fix team `parallel_reduce` buffer indexing
- Fix unregistering execution spaces on finalize

Build System Updates

- Flag checks now warn instead of raising a fatal error when a flag is rejected.
- Failed checks now print the CMake configure log (CMake $\geq$ 3.26) for easier diagnosis.
- `nvcc_wrapper` now handles response files (`@file.rsp`).
- Remove the `Kokkos_CXX_STANDARD` user option.

Deprecations and Breaking Changes

- Deprecate `Experimental::{HIP,SYCL}` aliases and related memory spaces
  - Use `Kokkos::HIP` / `Kokkos::SYCL` (and matching spaces) instead
- Deprecate out-of-rank `View` extent queries to align with `mdspan`
  - `extent(r)`, `extent_int(r)`, and `static_extent(r)` with `r >= rank()`
  - Still returns 1 when `Kokkos_ENABLE_DEPRECATED_CODE_5` is on (default)

```
View<double**> a("a", 10, 20);
a.extent(0);  // OK: 10
a.extent(1);  // OK: 20
a.extent(2);  // deprecated (rank is 2); returns 1 if DEPRECATED_CODE_5 on
```

- `deep_copy(dst, src)` aborts if the `View` arguments are identical

```
deep_copy(v, v);  // precondition violation (was a no-op)
```

- `is_assignable(dst, src)` now returns false if `dst` is `const`

```
View<int*> a("A",5) , b("B", 5);
const View<int*> c_a = a;
assert(is_assignable(c_a, b)); // fails since 5.2
```

Bug Fixes

- Fix simd arithmetic operators calling host-only functions on device
- Make masked version of simd reduce with default arguments unambiguous
- Fix segfaults when doing unaligned simd stores on AVX512
- Fix compile failures on AppleClang 14 and Clang 14 and 15 when using simd

- Only use `__atomic_max/min_fetch` with floating point types with LLVM >22.1
- Fix RangePolicy `HIP` performance regression with static batch size 1
- Fix bug in compiler detection for RISC-V with clang
- Fix thread-safety of `parallel_scan` when resizing shared memory between kernels in `CUDA`
- Fix construction of an unmanaged subview from a `View`
- Fix using `Kokkos::UnorderedMap` as a set

Kokkos Tools 5.2

Content:

- Aligned release cycle with Kokkos Core
- Maturing life-cycle stage and testing status

Aligned release: Kokkos Tools **5.2** ships with the Kokkos Core 5.2 cycle.

Lightweight profiling and debugging utilities that hook into the Kokkos runtime.
Allows for Kokkos-centric analysis (kernel labels, Views) vs. vendor tools alone (NVTX, ROCTx, ...).

- Load at runtime via `KOKKOS_TOOLS_LIBS` or `--kokkos-tools-libs`
- No application rebuild required (needs `Kokkos_ENABLE_LIBDL=ON`)
- CMake build; C++20 required

Docs: https://github.com/kokkos/kokkos-tools/wiki
Repo: https://github.com/kokkos/kokkos-tools

Life-cycle

Kokkos Tools is at the **Maturing** stage in the Kokkos life-cycle. We are ramping up testing and CI coverage.

Currently expanding testing and CI coverage:

- **CI:** PR builds track Kokkos 5.2; nightlies track `develop`.

Backend	Compile	Runtime
OpenMP	yes	yes
CUDA 12.2 / 12.9	yes	—
ROCm HIP 6.2 / 6.3	yes	—
Standalone	yes	—

- **LAMMPS sanity (Hopper, CUDA):** `lmp -k on g 1 -sf kk -in scripts/in.lj.short` Passed all tool checks. (Tracking: kokkos-tools#338)

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)