Kokkos 4.6 Release Briefing

New Capabilities

04/10/2025

4.6 Release Highlights

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

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:

- HPSF and Kokkos Meeting 2025
- Targeting C++20 for Kokkos 5.0
- Makefile deprecation

Kokkos User Group Meeting 2025 @ HPSF Conference

- *When:* May 5th-8th 2025
- *Where:* Chicago
- *What:* 2-days HPSF plenary + 2-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

Registration open now!

What to expect from KUG

- Eight 90-minute sessions featuring a dynamic blend of Kokkos developers and community users

- *Day 1 Highlights:*
  - Essential Updates
  - Kokkos in Applications
  - Adopting Kokkos
  - Lightning Talks

- *Day 2 Highlights:*
  - Kokkos Ecosystem
  - Tuning and Performance
  - Algorithms
  - Panel Discussion

Other reasons to go

- General Poster Session
- Updates on the HPSF project
- Introduction to various working groups
- Various Panel Discussions
- Chance to meet all other members of HPSF
- ...

- HPSF will be present at ISC BOF 2025
- Kokkos Tea-Time on 2nd or 3rd Wed of the month
  - April 16th @ 11am EST "Solomon: unified schemes for directive-based GPU offloading"

Kokkos 5 is comming Summer 2025

We will require C++20!

Start preparing now:

- Check availability of compilers on your systems
- Test with C++20 enabled: start with a CPU build
- Minimum Compiler requirements will change (more details later)

Nothing wrong for your project to require C++20 now if you feel ready!

Makefile is officially deprecated and will be removed in the next major release

Start preparing now:

- Check if you can transition to CMake
- Comment on pinned issue 7610

We reached “passed” on the OSSF Best Practices Program

www.bestpractices.dev

This means Kokkos is continuously tracking and openly reporting the conformity with open source software practices.

Feature highlights

- describes asynchronous workloads organised as a direct acyclic graph (DAG)
- executed using `submit()`, possibly many times, observing dependencies

```
auto graph = Kokkos::create_graph([&](auto root) {
    auto node_A = root.then_parallel_for("A", ...policy..., ...functor...);
    auto node_B = node_A.then_parallel_for("B", ...policy..., ...functor...);
    auto node_C = node_A.then_parallel_for("C", ...policy..., ...functor...);

    auto node_D = Kokkos::when_all(node_B, node_C).
                  then_parallel_for("D", ...policy..., ...functor...);
});

graph.instantiate();

graph.submit();
```

- `then` node: executes a callable on device
- Single call of the functor per `submit()`
- Executed in the `ExecutionSpace` the graph is submitted to

```
auto graph = Kokkos::create_graph([&](auto root) {
    auto node_A = root.then_parallel_for("A", ...policy..., ...functor...);
    auto node_B = node_A.then("B", ...functor...);
});
```

- Functor passed to `then` must be callable without arguments and marked with `KOKKOS_FUNCTION`

- Interoperability: create a `Kokkos::Graph` from a native Cuda/HIP/Sycl graph

```
cudaGraph_t native_graph = nullptr;
cudaGraphCreate(&native_graph, 0);

auto graph_from_native =
  Kokkos::Experimental::create_graph_from_native(exec, native_graph);
```

- Experimental, does not yet allow adding nodes created using the native API to a `Kokkos::Graph`

- Launch kernels on multiple devices from a single host process
- Available for ROCm 5.6 and later
- Requires direct use of HIP runtime API for creating and destroying streams
- Experimental, still looking for feedback from new users
- New documentation (for all backends)

  https://kokkos.org/kokkos-core-wiki/API/core/MultiGPUSupport.html

```
// Create streams on different devices
hipStream_t streams[2];
hipSetDevice(0); hipStreamCreate(&streams[0]);
hipSetDevice(1); hipStreamCreate(&streams[1]);
{
  // Creating execution spaces
  Kokkos::HIP exec0(streams[0]), exec1(streams[1]);

  // Allocating views
  Kokkos::View<int*> v0(Kokkos::view_alloc("v0", exec0), N);
  Kokkos::View<int*> v1(Kokkos::view_alloc("v1", exec1), M);

  // Launch kernels (run concurrently)
  Kokkos::parallel_for(Kokkos::RangePolicy(exec0, 0, N), functor0);
  Kokkos::parallel_for(Kokkos::RangePolicy(exec1, 0, M), functor1);
}
// Destroy streams (after execution spaces are deleted)
hipStreamDestroy(streams[0]); hipStreamDestroy(streams[1]);
```

General Enhancements

- `kokkos_check`: Check at configure time that Kokkos was built with the requested backends and target architectures.
- Fix a warning when a user calls the cmake function `kokkos_check` from a `<PackageName>Config.cmake` "Find Module" file

```
CMake Warning (dev) at /usr/share/cmake-3.22/Modules/FindPackageHandleStandardArgs.cmake:438 (message):
  The package name passed to ‘find_package_handle_standard_args‘
  (Kokkos_DEVICES) does not match the name of the calling package (SomePackage).
  This can lead to problems in calling code that expects ‘find_package‘
  result variables (e.g., ‘_FOUND‘) to follow a certain pattern.
Call Stack (most recent call first):
  ... /kokkos/lib/cmake/Kokkos/KokkosConfigCommon.cmake:110 (find_package_handle_standard_args)
  ...
This warning is for project developers.  Use -Wno-dev to suppress it.
```

With the Cuda and HIP backends, `Kokkos::Experimental::inclusive_scan` now calls the vendor versions in Thrust

- The vendor versions are up to 3x faster than the `Kokkos::parallel_scan`-based default implementation
- Thrust requires `Kokkos_ENABLE_ROCTHRUST` to be `ON` (which is the default)
- Approximately 1.5-3x speed up (V100, MI300A)

Reduced the overhead of Kokkos tools related checks

- Store the information whether Kokkos tools are enabled after each modification to the tools' callbacks
- Previously, this value was recomputed for every event (`parallel_for`, `fence`, etc.)
- Most noticeable in small serial kernels (around 100 elements)
- Reduction of launch time of approximately 10ns (About the time to increment 100 elements in a kernel) on CPU

Added reductions and remaining compound assignments to Kokkos SIMD

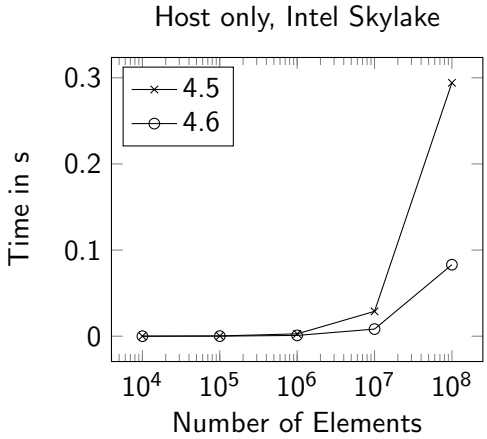
- `basic_simd& operator/=(basic_simd&, U&&)`
- `basic_simd& operator>>=(basic_simd&, U&&)`
- `basic_simd& operator<<=(basic_simd&, U&&)`

- `T reduce_min(const basic_simd& x)`
- `T reduce_max(const basic_simd& x)`
- `T reduce(const basic_simd& x, const mask_type& mask, T identity_element, BinaryOperation binary_op)`
  - Supported binary operations are: std::plus, std::multiplies, std::bit_and, std::bit_or and std::bit_xor
  - std::plus is used if binary op is not specified

- In Kokkos 4.5, we fixed a performance bug in `Kokkos::sort`
- Root cause is in an implementation detail in `RandomAccessIterator`

- In Kokkos 4.6 the root cause is fixed
- Fixes all algorithms that rely on `RandomAccessIterator` (e.g. sort,search,etc.)

Improvement dependent on algorithm and hardware!



`print_configuration` outputs if system allocated memory is accessible on GPU

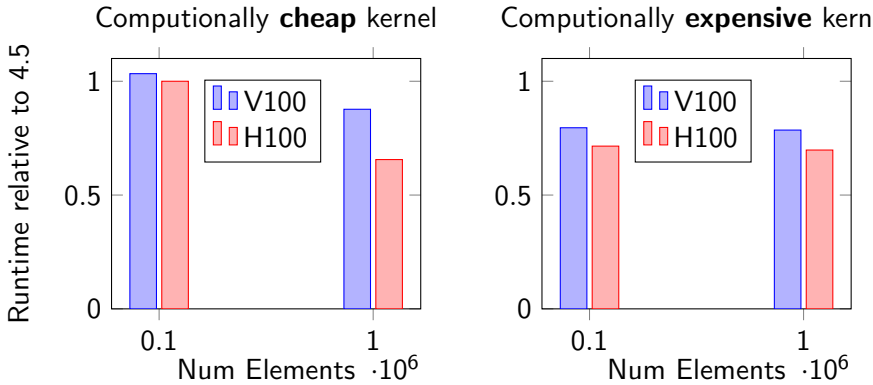
No guarantees about the print format!

Example output for MI300A with HIP backend

```
XNACK environment variable set: yes
Kernel reports HMM module via ‘CONFIG_HMM_MIRROR=y‘ in ‘/boot/config‘: yes
Architecture capable of accessing system allocated memory: 1,
System allows accessing system allocated memory on GPU: 1,
```

Backend Updates

- SYCL: Improved sorting performance for non-contiguous views
- Serial: Reduce fences overhead when using `Kokkos_ENABLE_ATOMICS_BYPASS`
- CUDA: Improved performance for `Kokkos::parallel_reduce` on H100 and newer by removing limitations on the runtime thread configuration


Computationally **cheap** kernel
Runtime relative to 4.5
1
0.5
0
V100
H100
0.1
1
Num Elements ·10^6
Computationally **expensive** kernel
1
0.5
0
V100
H100
0.1
1
Num Elements ·10^6

- Change block size deduction to prefer smaller blocks/teams if possible
- Allocate memory with stream ordered semantics (*i.e.* use `hipMallocAsync`)
- Fix a segfault when a virtual function called inside a kernel requires too many registers

Build Systems Updates

- Add support for Zen 4 AMD microarchitecture (`Kokkos_ARCH_ZEN4`)
- Enable NVIDIA Grace architecture with NVHPC (`Kokkos_ARCH_ARMV9_GRACE`)
- Support static library builds via `CMAKE_CUDA_RUNTIME_LIBRARY=static` when using CUDA as CMake language

- Spack *develop* branch now supports MI300A with a new variant `apu` (spack/spack#48609)
- To compile Kokkos for MI300A, forcing the APU mode, use the following command: `spack install kokkos +rocm amdgpu_target=gfx942 +apu`

Deprecations and other breaking changes

- Intel has deprecated Intel Classic in 2022, and removed it from oneAPI 2024
- In order to focus on newer compilers, and reduce maintenance burden, we have **removed** support for Intel Classic (oneAPI Intel/icpx still supported of course!)

Deprecate direct access to `d_view` and `h_view`

- Modifying the allocations in d_view and h_view directly is dangerous, especially if `modify` and `sync` are skipped
- Use `view_host()` and `view_device()` instead
- These two functions return by value with deprecated code enabled and by const reference otherwise. This might have perfomance implications if used extensively, e.g., in loop bounds.

- `native_simd`, `native_simd_mask` **deprecated** to align with the C++26 standard
- **Removed** Obtaining a reference from SIMD `operator[]` to align with the C++26 Standard
- **Changed** the return type of SIMD `operator==` and `operator!=` to return SIMD masks instead of `bool`
  - If you want old behavior, use `all_of(a == b)`

- Already discussed deprecating the Makefile
- StaticCrsGraph is **moved** to Kokkos Kernels and **deprecated** in Core
  - See https://github.com/kokkos/kokkos-kernels/pull/2419
  - Symbol is in Kernels under `KokkosSparse::StaticCrsGraph`

Bug Fixes

- Fix execution of ranges with more than 2 billion elements
- Graph:
  - Fix graph node lifetime issues
  - Fix lock-based atomics failure when launching CUDA and HIP graphs
- CUDA backend: Fix incorrect iteration in MDRangePolicy of rank > 4 for high iteration counts
- SIMD:
  - fix a bug in scalar min/max
  - fix a bug in non-masked reductions
- View: fix MSVC compilation

- Fix `clean` target when embedding Kokkos in another project
- Stop generation if ARMv9 Grace arch is not explicitly supported by the compiler when `KOKKOS_ARCH_ARMV9_GRACE` is specified
  - Can still try and configure with `ARCH_NATIVE`
- Fix Zen3 flag for NVHPC
- Use right arch for MI300A in makefiles
- (CUDA) ignore gcc assembler options in `nvcc_wrapper`

- Fix performance bug affecting atomic_fetch_{add,sub,min,max,and,or,xor} on integral types long and unsigned long with HIP
- Fix performance of `RangePolicy` where an error message is generated even if precondition not violated

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)