Kokkos 4.3 Release Briefing

New Capabilities

May 8, 2024

4.3 Release Highlights

- Organizational
- Backend updates
- Build system updates
- `Kokkos::sort_by_key`
- Miscellaneous
- Deprecations and other breaking changes

Find More

Online Resources:

- https://github.com/kokkos:
  - Primary Kokkos GitHub Organization
- https://github.com/kokkos/kokkos-tutorials/wiki/Kokkos-Lecture-Series:
  - Slides, recording and Q&A for the Full Lectures
- https://kokkos.github.io/kokkos-core-wiki:
  - Wiki including API reference
- https://kokkosteam.slack.com:
  - Slack channel 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

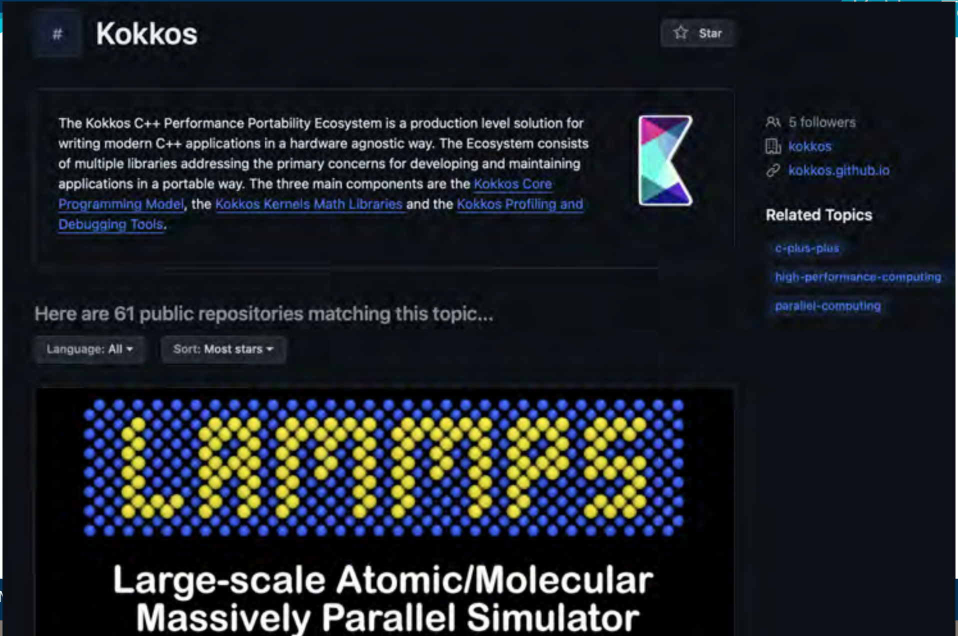
List of Applications and Libraries

- Add your app to https://github.com/kokkos/kokkos/issues/1950
- We are planning to add that to a 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.

opic



Kokkos

Star

The Kokkos C++ Performance Portability Ecosystem is a production level solution for writing modern C++ applications in a hardware agnostic way. The Ecosystem consists of multiple libraries addressing the primary concerns for developing and maintaining applications in a portable way. The three main components are the Kokkos Core Programming Model, the Kokkos Kernels Math Libraries and the Kokkos Profiling and Debugging Tools.

5 followers
kokkos
kokkos.github.io

Related Topics

c-plus-plus
high-performance-computing
parallel-computing

Here are 61 public repositories matching this topic...

Language: All ▾ Sort: Most stars ▾

Large-scale Atomic/Molecular Massively Parallel Simulator

Organizational

Content:

- Linux Foundation
- `master` Branch
- Kokkos Tea Time

Kokkos is now a project of the Linux Foundation!

At ISC HPC in Hamburg (May 2024) we will also join the *High Performance Software Foundation*!

Why did you do that? To grow developer base!

- LF provides neutral ground
- Well defined governance
- Not a "DOE" or even worse "Sandia" project

What changes for users? Mostly nothing!

- Hopefully even better Kokkos in the long run!
- Will leverage LF to organize community events.
- Only caveat: Trademark rules

Linux Foundation - Trademark

Trademark??? Yes: Kokkos™...

- Don't call your project "KokkosFoo" unless inside the Kokkos organization
- You can say "Foo based on Kokkos" "Foo for Kokkos" and disclaim "Kokkos is a LF project"
- You can organize classes, presentations and what not on Kokkos, and refer to it as much as you like in presentations.
- The goal is to avoid confusion of what is part of the official Kokkos LF project vs efforts which are just leveraging the project.

If you have questions: asks us!
And look at the Linux Foundation Trademark rules:
https://www.linuxfoundation.org/legal/trademark-usage

We are getting rid of the master branch for Kokkos

- `master` branch has been the "latest release"
- Not generally a common practice on github e.g. normally `main` is what we call `develop`
- Often lead to PRs based on wrong thing

What will change?

- `develop` branch now the default branch
- `master` will still exist for a little while

Kokkos Tea Time

Kokkos tea-time is a monthly time online to meet the community and discuss anything related to **Kokkos**, its **ecosystem**, or even **GPU-programming** at large.

- 3rd Wednesday of the month
  - 7AM PT, 8AM MT, 10AM ET, 2PM UTC, 4PM CEST
- a 45min time slot for either
  - a 30min presentation followed by questions,
  - or a more informal discussion on a select topic.
- by zoom, by visio, by phone, or in your nearest kokkos shop
  - Get you link `https://cexa-project.org/news/`

We want you to tell us about your Kokkos use, development, ideas, ...
`contact@cexa-project.org`

Discover Kokkos Resilience with Nic Morales on the 16th of May.

Two new Kokkos subprojects:

- Very Experimental: expect changes of interface etc.
- Brave early experimenters welcome to help give feedback or get involved.

KokkosFFT

- Goal: wrapping existing MPI libraries such as fftw, cufft, mkl and rocfft.
- `https://github.com/kokkos/kokkos-fft`
- POC: Yuuichi Asahi (CEA)

KokkosComm

- Goal: provide communication facilities: for now MPI-like interfaces taking `Views`.
- `https://github.com/kokkos/kokkos-comm`
- Join the `mpi-interop` channel on the Kokkos Slack

Backend Updates

Content:

- Backend Updates CUDA
- Backend Updates HIP
- Backend Updates SYCL
- Backend Updates OpenMPTarget
- Backend Updates OpenACC

Miscellaneous

- Link against CUDA libraries even with KOKKOS_ENABLE_COMPILE_AS_CMAKE_LANGUAGE
- Don't use the compiler launcher script if the compile language is CUDA.
- nvcc(wrapper): adding "long" and "short" versions for all flags

Multi-GPU Support from single process

- Highly Experimental! (But most things should work now)
- For now: via interoperability interfaces - i.e. no native interface to create execution space instances on different GPU
- If you have interest: try it out and provide feedback

CUDA - multi-GPU support

```
cudaStream_t[2] streams;
cudaSetDevice(0);
cudaStreamCreate(&streams[0]));
cudaSetDevice(1);
cudaStreamCreate(&streams[1]));
{
  Kokkos::Cuda exec0(streams[0]), exec1[streams[1]);

  Kokkos::View<int *, TEST_EXECSPACE> view(Kokkos::view_alloc("v0", exec0), n);
  Kokkos::View<int *, TEST_EXECSPACE> view(Kokkos::view_alloc("v1", exec1), m);

  // run concurrently
  Kokkos::parallel_for(Kokkos::RangePolicy<Cuda>(exec0, 0, n), functor);
  Kokkos::parallel_for(Kokkos::RangePolicy<Cuda>(exec1, 0, m), functor);
}
cudaStreamDestroy(streams[0]);
cudaStreamDestroy(streams[1]);
```

Backend Updates HIP

- Fix compilation error with amdclang++ 5.7 and newer when relocatable device code is enabled. We are aware of issues with older versions of amdclang++ when relocatable device code is enabled.
- Added support for rocThrust (used in sort). Note that some installation of ROCm do not have rocThrust installed. rocThrust support can be enable/disabled using kokkos_ENABLE_ROCTHRUST

Backend Updates SYCL

- Only allow ext_oneapi_*-type devices when targeting GPUs
- Avoid unnecessary zero-memset of the scratch flags in SYCL
- Use host-pinned memory to copy reduction/scan result
- Address deprecations after oneAPI 2023.2.0
- Make sure to call find_dependency for oneDPL if necessary

Backend Updates OpenMPTarget

- Intel GPUs - Allow printing from GPUs.
- NVIDIA and AMD GPUs - LLVM extensions for dynamic shared memory
  - Available since LLVM/18
  - Only available in upstream LLVM
  - Not part of OpenMP API

Backend Updates OpenACC

- Add support for atomic operations
  - OpenACC does not support atomic-compare-exchange operations; implemented using CUDA intrinics (atomicCAS).
  - Can be compiled by NVIDIA NVHPC compiler (nvc++) but not by CLACC compiler.
  - Support only NVIDIA GPUs.
- Change the default execution policy behavior from synchronous to asynchronous executions.

Build System Updates

- Support `CMAKE_CXX_STANDARD` of 26
- New architecture CMake flag: `KOKKOS_ARCH_RISCV_MILKV`
  - Allows building with Milk-V Pioneer
- New CMake Flag: `KOKKOS_ENABLE_ATOMICS_BYPASS`
  - Previously when building with only `Serial` backend atomics were always bypassed
  - Now requires explicit opt-in, but only allowed for `Serial` backend builds
  - Useful for MPI-only builds, but beware of pitfalls!
- Fix generated CMake CUDA targets when using CMake 3.28.4
- Fix Makefile bugs with GNU make version 4.3

Introduced sort_by_key to dispatch to optimized vendor libraries

```
ExecutionSpace exec_space;
Kokkos::View<int*> keys("keys", n);
Kokkos::View<float*> values("values", n);
Kokkos::Experimental::sort_by_key(exec_space, keys, values);
```

- 1D views only
- Sizes of `keys` and `values` must match
- Both `keys` and `values` are modified
- Dispatches to vendor libraries (Thrust, rocThrust, oneDPL) when available

General Enhancements

Querying the Number of Devices

Content:

- Runtime function for querying the number of devices
- Device ID consistency with `KOKKOS_VISIBLE_DEVICES`

A runtime function to query the number of devices

```
[[nodiscard]] int Kokkos::num_devices() noexcept ...
```

- Callable before `Kokkos::initialize()`
- Returns the device count based on visible devices
- Returns -1 if no GPU backend is enabled
- Replaces `Cuda,HIP::detect_device_count()`

Fixed a defect in `Kokkos::device_id()`

`KOKKOS_VISIBLE_DEVICES` were not being considered for `Kokkos::device_id()`.

initialization settings	**Pre-4.3**	**4.3**
<none>	0	0
device_id=1	1	1
KOKKOS_VISIBLE_DEVICES=0	0	0
KOKKOS_VISIBLE_DEVICES=3	3	0
KOKKOS_VISIBLE_DEVICES=1,0	1	0
device_id=1 KOKKOS_VISIBLE_DEVICES=1,0	0	1

Kokkos SIMD

Content:

- `simd_flags`
- `vector_aligned_tag`

Introduced `simd_flags` to match latest ISO C++ proposal on `std::simd`

`template <typename... Flags> struct simd_flags`

- `element_aligned_tag <-> simd_flag_default`
- `vector_aligned_tag <-> simd_flag_aligned`

`simd_flags` are used in:

- `template <class U, class Flags> void copy_from(const U* mem, Flags flags)`
- `template <class U, class Flags> void copy_to(U* mem, Flags flags)`

simd_flags

```
void init_var() {
  constexpr size_t alignment =
    Kokkos::Experimental::native_simd::size() * sizeof(DataType);

  alignas(alignment) DataType const src[N] = { ... };

  simd_type var;
  var.copy_from(src, Kokkos::Experimental::simd_flag_aligned);

  ...
}
```

Bitset Constructor Update

Bitset constructor update

```
Bitset(unsigned arg_size = 0u)
```

- A `Bitset` constructor with a default size of `Bitset` was making a deferred constructor call.
- Caused an unnecessary memory allocation when `Bitset` was constructed with the default size of 0.

Refactored to no longer have the default argument and use a defaulted default `Bitset` constructor instead.

Random Number Generator

`Normal distribution improvements`

- Replace Marsaglia polar method with Box-Muller method
- Box-Muller contains no branching/looping, single code path, ideal for GPU
- Kokkos performance on GPU, Nvidia:
  - 20% faster for 64 bit version
  - 60% faster for 1024 bit version

New Public Headers

Content:

- `Kokkos_Clamp.hpp`
- `Kokkos_MinMax.hpp`

Kokkos_Clamp.hpp

```
// bounded value

template<class T>
constexpr const T& clamp(const T& value, const T& low, const T& high);

template<class T, class ComparatorType>
constexpr T const& clamp(const T& value, const T& low, const T& high,
                         ComparatorType comp);
```

Kokkos_MinMax.hpp Max

```
// max

template <class T>
constexpr const T& max(const T& a, const T& b);

template <class T, class ComparatorType>
constexpr const T& max(const T& a, const T& b, ComparatorType comp);

template <class T>
constexpr T max(std::initializer_list<T> ilist);

template <class T, class Compare>
constexpr T max(std::initializer_list<T> ilist, Compare comp);
```

Kokkos_MinMax.hpp Min

```
// min

template <class T>
constexpr const T& min(const T& a, const T& b);

template <class T, class ComparatorType>
constexpr const T& min(const T& a, const T& b,ComparatorType comp);

template <class T>
constexpr T min(std::initializer_list<T> ilist);

template <class T, class Compare>
constexpr T min(std::initializer_list<T> ilist, Compare comp);
```

Kokkos_MinMax.hpp MinMax

```
// minmax

// minmax
template <class T>
constexpr Kokkos::pair<const T&, const T&> minmax(const T& a, const T& b);

template <class T, class ComparatorType>
constexpr Kokkos::pair<const T&, const T&> minmax(const T& a, const T& b,
                                                  ComparatorType comp);

template <class T>
constexpr Kokkos::pair<T, T> minmax(std::initializer_list<T> ilist);

template <class T, class Compare>
constexpr Kokkos::pair<T, T> minmax(std::initializer_list<T> ilist, Compare comp);
```

Compile-Time Argument Deduction (CTAD / Deduction Guides)

Content:

- What are deduction guides?
- `Kokkos::Array` deduction guide
- `Kokkos::RangePolicy` deduction guides
- `Kokkos::MDRangePolicy` deduction guides

- C++17
- Usability Improvement
- Deduces class template parameters from types and/or number of parameters passed to constructors
- Eliminates need to specify template parameters when declaring automatic variables

Array Deduction Guide

```
// Kokkos::Array<double, 3>
Kokkos::Array a4{3.0, 1.0, 4.0};
```

- matches `std::array` deduction guide

RangePolicy Deduction Guides

```
int64_t work_begin   = /* ... */;  // conversions as well
int64_t work_end     = /* ... */;  // conversions as well
Kokkos::ChunkSize cs = /* ... */;  // conversions as well
Kokkos::DefaultExecutionSpace des; // conversions as well
SomeExecutionSpace ses;            // different from Kokkos::DefaultExecutionSpace

// Kokkos::RangePolicy<>
Kokkos::RangePolicy rp0;
Kokkos::RangePolicy rp1(work_begin, work_end);
Kokkos::RangePolicy rp2(work_begin, work_end, cs);
Kokkos::RangePolicy rp3(des, work_begin, work_end);
Kokkos::RangePolicy rp4(des, work_begin, work_end, cs);

// Kokkos::RangePolicy<SomeExecutionSpace>
Kokkos::RangePolicy rp5(ses, work_begin, work_end);
Kokkos::RangePolicy rp6(ses, work_begin, work_end, cs);
```

MDRangePolicy initializer_list Deduction Guides

```
Kokkos::DefaultExecutionSpace des;
SomeExecutionSpace ses;             // different from Kokkos::DefaultExecutionSpace
int64_t i;

// Kokkos::MDRangePolicy<Kokkos::Rank<5>>
Kokkos::MDRangePolicy pl0({1, 2, 3, 4, 5}, {1, 2, 3, 4, 5});
Kokkos::MDRangePolicy pl1({1, 2, 3, 4, 5}, {1, 2, 3, 4, 5}, { i });
Kokkos::MDRangePolicy pl2(des, {1, 2, 3, 4, 5}, {1, 2, 3, 4, 5});
Kokkos::MDRangePolicy pl3(des, {1, 2, 3, 4, 5}, {1, 2, 3, 4, 5}, { i });

// Kokkos::MDRangePolicy<SomeExecutionSpace, Kokkos::Rank<5>>
Kokkos::MDRangePolicy pl4(ses, {1, 2, 3, 4, 5}, {1, 2, 3, 4, 5});
Kokkos::MDRangePolicy pl5(ses, {1, 2, 3, 4, 5}, {1, 2, 3, 4, 5}, { i });
```

MDRangePolicy C Array Deduction Guides

```
Kokkos::DefaultExecutionSpace des;
SomeExecutionSpace ses;             // different from Kokkos::DefaultExecutionSpace
int cbegin[3];
int cend[3];
int64_t ctiling[2];

// Kokkos::MDRangePolicy<Kokkos::Rank<3>>
Kokkos::MDRangePolicy pc0(cbegin, cend);
Kokkos::MDRangePolicy pc1(cbegin, cend, ctiling);
Kokkos::MDRangePolicy pc2(des, cbegin, cend);
Kokkos::MDRangePolicy pc3(des, cbegin, cend, ctiling)

// Kokkos::MDRangePolicy<SomeExecutionSpace, Kokkos::Rank<3>>
Kokkos::MDRangePolicy pc4(ses, cbegin, cend);
Kokkos::MDRangePolicy pc5(ses, cbegin, cend, ctiling);
```

MDRangePolicy Array Deduction Guides

```cpp
Kokkos::DefaultExecutionSpace des;
SomeExecutionSpace ses;          // different from Kokkos::DefaultExecutionSpace
Kokkos::Array<int, 2> abegin;
Kokkos::Array<int, 2> aend;
Kokkos::Array<int64_t, 2> atiling;

// Kokkos::MDRangePolicy<Kokkos::Rank<2>>
Kokkos::MDRangePolicy pa0(abegin, aend);
Kokkos::MDRangePolicy pa1(abegin, aend, atiling);
Kokkos::MDRangePolicy pa2(des, abegin, aend);
Kokkos::MDRangePolicy pa3(des, abegin, aend, atiling)

// Kokkos::MDRangePolicy<SomeExecutionSpace, Kokkos::Rank<2>>
Kokkos::MDRangePolicy pa4(ses, abegin, aend);
Kokkos::MDRangePolicy pa5(ses, abegin, aend, atiling);
```

Misc. Algorithmic Improvements/Fixes

Misc. Algorithmic Improvements/Fixes

- Kokkos_Unique.hpp
  - Allocate temporary view with provided execution space
  - Remove unnecessary init for temporary view during construction
- Kokkos_RemoveIf.hpp
  - Allocate temporary view with provided execution space
  - Remove unnecessary init for temporary view during construction
  - Remove unnecessary predicate evaluation

    Important since predicate can be arbitrarily expensive

Range/MDRangePolicy Updates

Content:

- Begin and end bounds check
- Unsafe implicit conversion check
- RangePolicy variadic constructor removal

Asserts that the upper bound is greater than the lower bound

```
Kokkos::RangePolicy<> policy(100, 90);
Kokkos::MDRangePolicy<Kokkos::Rank<2>> policy({100, 100}, {100, 90});
```

Aborts with: *Kokkos::MDRangePolicy bounds error: The lower bound (100) is greater than its upper bound (90) in dimension ...*

- If `KOKKOS_ENABLE_DEPRECATED_CODE_4` is not defined, aborts.
- Else if `KOKKOS_ENABLE_DEPRECATION_WARNINGS` is defined, outputs to `std::cerr`.

Checks for unsafe implicit index type conversions during RangePolicy construction

- Narrowing conversions
- Sign conversions

Aborts with: *Kokkos::RangePolicy bound type error: an unsafe implicit conversion is performed on a bound (...) which may not preserve its original value.*

- If `KOKKOS_ENABLE_DEPRECATED_CODE_4` is not defined, aborts.
- Else if `KOKKOS_ENABLE_DEPRECATION_WARNINGS` is defined, outputs to `std::cerr`.

Removed RangePolicy variadic constructors

```
template<class ...InitArgs>
RangePolicy(const IndexType&, const IndexType&, const InitArgs...)
template<class ...InitArgs>
RangePolicy(const ExecutionSpace&, const IndexType&, const IndexType&,
            const InitArgs...)

RangePolicy(const IndexType&, const IndexType&, const ChunkSize)
RangePolicy(const ExecutionSpace&, const IndexType&, const IndexType&,
            const ChunkSize)
```

`template <class... Args> inline void set(Args...)` is deprecated in favor of
`inline RangePolicy& set_chunk_size(int chunk_size)`.

Bug Fixes

Bug Fixes

- Fix reductions with types smaller than ints (for Cuda and HIP)
- Fix TeamThreadMDRange, ThreadVectorMDRange, TeamVectorMDRange parallel_reduce to combine results across threads
- Fixed missing broadcast in TeamThreadRange parallel_scan for Threads backend
- Enable {`transform_`}`exclusive_scan` in place (to match `std::`{`transform_`}`exclusive_scan`)
- Fence `fill_random` without execution space
- `cuda_func_set_attributes_wrapper` → `cuda_func_set_attribute_wrapper`
- `cudaFuncSetAttributes` → `cudaFuncSetAttribute`

Potentially Breaking Changes

Breaking Changes DEPRECATED_CODE_3

Remove `Kokkos_ENABLE_DEPRECATED_CODE_3`, including

- `InitArguments`
- `KOKKOS_ACTIVE_EXECUTION_MEMORY_SPACE_*` macros
- `Experimental::clamp, min, max, minmax`
- using declarations in the `Experimental::` namespace for math functions / constants
- `{OpenMP,HPX}::partition_master`
- `MasterLock`

Breaking Changes II

- No longer support users defining `KOKKOS_ASSERT`
- `[[nodiscard]] explicit Kokkos::Profiling::ProfilingSection(std::string)`
- Remove `Kokkos::[b]half_t volatile` overloads
- `Kokkos_Tools_OptimzationGoal` → `Kokkos_Tools_OptimizationGoal`
- Always call `abort` (instead of throwing when on host) for `View` bounds errors
- Check matching static extents in `View` constructor (more like `mdspan`)

Breaking Changes III

- Remove `KOKKOS_ENABLE_INTEL_MM_ALLOC` macro
- Remove `Kokkos::Experimental::LogicalMemorySpace`
- Remove `Kokkos::Experimental::HBWSpace` and `memkind` linking support
- Drop `librt` and `KOKKOS_ENABLE_LIBRT`
- Drop old CPU architectures `ARCH_BGQ`, `ARCH_POWER7`, `ARCH_WSM`, `ARCH_SSE42`
- Drop command line / environment variable support for `num_devices` and `skip_device`

Deprecations

Deprecations

- Add `Kokkos::kokkos_swap` to core and deprecate `Kokkos::Experimental::swap`
- Deprecate `{Cuda,HIP}::detect_device_count()` and `Cuda::[detect_]device_arch()`
- Deprecate `Kokkos::ExecutionSpace::in_parallel()`

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 questions (https://kokkosteam.slack.com)