Kokkos Policies

December 14, 2023

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.
SANDXXXX-XXXX PE

Currently, we support

- CMake via `find_package()`
  - Standalone VS as a subpackage of Trilinos
  - CUDA as "CMake language"
- CMake via `add_subdirectory()`
- Generated makefiles via `Makefile.kokkos`

Currently, we support

- CMake via `find_package()`
  - Standalone VS as a subpackage of Trilinos
  - CUDA as "CMake language"
- CMake via `add_subdirectory()`
- Generated makefiles via `Makefile.kokkos`

Supporting multiple ways to build Kokkos has a real cost in increased testing and maintenance work.

Currently, we support

- CMake via `find_package()`
  - Standalone VS as a subpackage of Trilinos
  - CUDA as "CMake language"
- CMake via `add_subdirectory()`
- Generated makefiles via `Makefile.kokkos`

Supporting multiple ways to build Kokkos has a real cost in increased testing and maintenance work.
We want to support the fewest variants necessary to accommodate the most important use cases.

- Maintaining support for any particular C++ standard forever is impractical
  - Difficult to accommodate newer language features that users will increasingly come to expect
  - Eventually, Kokkos' APIs would appear outdated
  - Maintenance and testing burden grows unbounded
- Since C++ language standards are never formally deprecated or EOL'd, we need to come up with our own criteria
  - Support all modern C++ standards, from some oldest standard to the newest
  - Drop support for our oldest supported standard when 5 years has passed since the standard's release
  - support the newest C++ language standard soon after the standard is published and compiler support is available

Minimum required version for compilers and toolchains derives from

- Exclude compilers that do not have workable support for the minimum required C++ standard
- Check versions installed on leadership computing facilities
- Stakeholders survey
- Consider default toolchains on current operating systems
- No compiler that is EOL as defined by the vendor

We aim to support all architectures that are relevant to computing at large

- We guarantee support for all architectures deployed by our sponsors
- We welcome contributions adding support for other architectures
  - We cannot test if we don't have hardware access
- We do not support EOLd architectures (e.g. Fermi or Kepler)

- All breaking changes require a major version bump
  - Dropping support for a language standard
  - Bumping minimum requirements
  - Removing deprecated code
- The highest version number is always supported
- The newest release for non-highest major versions receives 12-months of support
  - Only backporting bug fixes

- All breaking changes require a major version bump
  - Dropping support for a language standard
  - Bumping minimum requirements
  - Removing deprecated code
- The highest version number is always supported
- The newest release for non-highest major versions receives 12-months of support
  - Only backporting bug fixes

With enough users, every change is potentially a breaking change for someone.

- All breaking changes require a major version bump
  - Dropping support for a language standard
  - Bumping minimum requirements
  - Removing deprecated code
- The highest version number is always supported
- The newest release for non-highest major versions receives 12-months of support
  - Only backporting bug fixes

With enough users, every change is potentially a breaking change for someone.

$\Longrightarrow$ Document what constitutes approved and supported usage of Kokkos.

Documented in our Programming Guide

- Unless documented otherwise
  - Don't define things in namespace `Kokkos::`
  - Don't mess with macros starting in `KOKKOS_`
  - Don't create or modify files starting in `Kokkos_`
- Include what you use
- Don't depend upon internal details
  - If the name contains "impl", don't use it
  - Don't include private headers (those not explicitly advertised as public)
- We reserve the right to do certain things such as
  - Add new names to namespace Kokkos
  - Add new default arguments to functions and templates
  - Change return-types of functions in compatible ways (void to anything, etc.)

- Controlled by user at configuration time
  - `-DKokkos_ENABLE_DEPRECATED_CODE_4=ON`
  - `-DKokkos_ENABLE_DEPRECATION_WARNINGS=ON`
- We recommend you regularly build with deprecated code disabled to make sure you don't accidentally rely on deprecated features
- At the beginning of a new major release cycle, deprecated code is `ON` by default, warnings are `ON` unless you explicitly disable them
- When we get closer to the next major release, we change the default and deprecated code becomes `OFF` to make it harder to ignore that the functionality will be removed in the near future

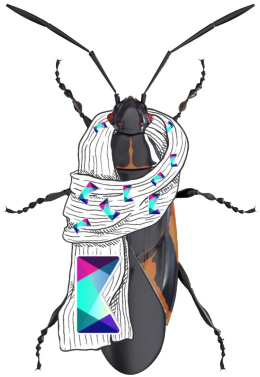
- Anything in namespace `Kokkos::Experimental`
- Users are free to try out those experimemtal features
- The goal is to allow getting feedback on them
- However, we do not provide the same guarantees about those features as we do for the rest of the library
- We reserve the right to change the API
- It is not as widely used so we expect it is more likely there are bugs

Consider asking on Slack first if you are unsure
Open an issue on GH

https://github.com/kokkos/kokkos/issues/new/choose

- Describe the issue (what happened and what did you expect to happen?)
- Steps to reproduce the problem
- What version of Kokkos are you using?
- What compiler and version are you using?
- How did you configure?
- Additional context (e.g. CPU/GPU architecture)

It's not a bug, it's a feature*



*undocumented feature

We welcome patches and other contributions to the project.
There are a few guidelines you need to follow.

- Make sure your pull request are against the **DEVELOP** branch
- Follow our coding style (Apply `clang-format`)
- Make sure tests pass (Configure with `-DKokkos_ENABLE_TESTS=ON`, build, run with `ctest`)
- Avoid mixing in unrelated changes
- Describe the changes and provide a rational


ORDER
ORDER
Out of order