SAND2019-9827R

EXASCALE COMPUTING PROJECT

ECP-U-2018-XXX

Documented Kokkos API

WBS STPR 04 Milestone 13

Authors

Christian Trott (1

Author Affiliations

(1) Sandia National Laboratories

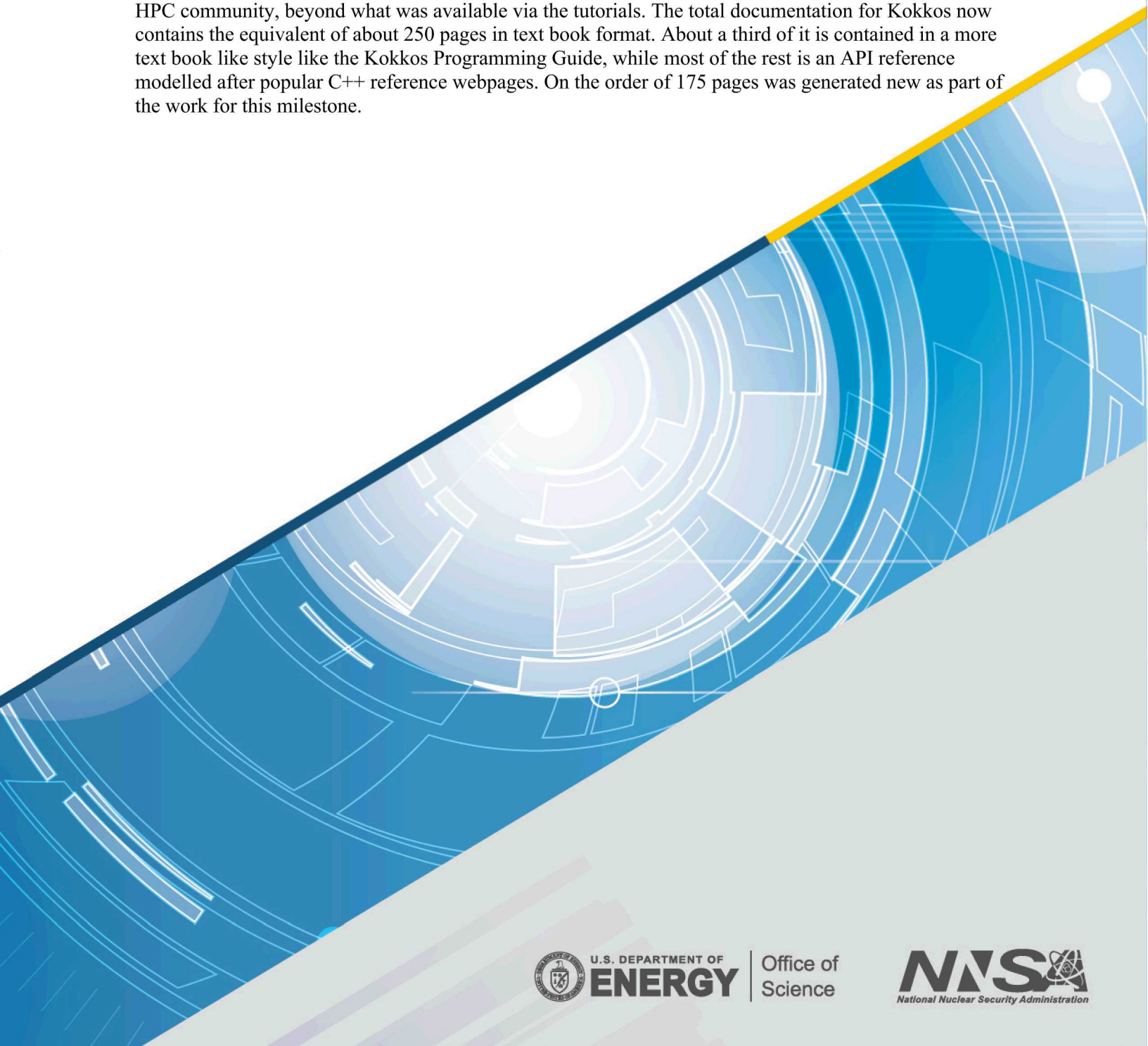
August 12th 2019

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.

EXASCALE COMPUTING PROJECT

EXECUTIVE SUMMARY

This report documents the completion of milestone STPR04-13 “Documented Kokkos API”. The goal of this Milestone was to generate documentation for the Kokkos programming model accessible to the open HPC community, beyond what was available via the tutorials. The total documentation for Kokkos now contains the equivalent of about 250 pages in text book format. About a third of it is contained in a more text book like style like the Kokkos Programming Guide, while most of the rest is an API reference modelled after popular C++ reference webpages. On the order of 175 pages was generated new as part of the work for this milestone.



U.S. DEPARTMENT OF ENERGY | Office of Science

NNSA National Nuclear Security Administration

1. INTRODUCTION

In order to enable a growing user base across a wide array of institutions a quality documentation for Kokkos is required. This will relieve pressure on answering questions via email, GitHub issues and Slack, while making sure that the semantics of Kokkos capabilities are formaly documented.

2. MILESTONE OVERVIEW

2.1 DESCRIPTION

Develop a web based documentation of the Kokkos Core API that will enable users to quickly navigate and review the semantics of Kokkos Core. This documentation will include detailed information on parallel constructs, kernels, and data views of Kokkos Core.

2.2 EXECUTION PLAN

1) Review Kokkos core API and the explicit semantics of each.
2) Categorize the Kokkos core API by functionality and intent.
3) Document intent of each API and semantics.
4) Gather feedback on documentation from Kokkos users.
5) Iterate on 1 - 4 to improve documentation quality.
6) Release documentation via public Kokkos website.

2.3 COMPLETION CRITERIA

A publicly available documentation for Kokkos's API is implemented.

3. TECHNICAL WORK SCOPE, APPROACH, RESULTS

The Kokkos Wiki is integrated into https://github.com/kokkos/kokkos via the built-in Wiki capabilities of GitHub. The content is provided via MarkDown text, which renders well as part of GitHub, can be viewed via standalone plugins in most web browsers, and is still well readable as raw text. The Kokkos Wiki currently consists of a total of 110 MarkDown documents, representing the equivalent of about 250 pages text book.

3.1 STRUCTURE OF THE KOKKOS WIKI

The Kokkos Wiki contains three major sections:

1) The Programming Guide

2) API Reference
3) A Guide to Kokkos Testing and Issue Tracking.

The Programming Guide contains documentation for users written in text book style. That means capabilities are introduced in prose, with motivation for the capability, use case descriptions as well as wording which places the specific capability in the larger context of Kokkos and parallel programming in general. As a compromise, features are often not exhaustively described with all their nuances and various corner use cases. The audience of the programming guide are mainly new Kokkos developers.

The API Reference is meant as a complete description of each public API feature in Kokkos. The inspiration for its style is taken from websites such as http://www.cplusplus.com/reference/ and https://cppreference.com . It targets active Kokkos developers who want to quickly check syntax and semantics of specific Kokkos functions, or want to check whether a certain capability is available. Going forward, every new feature in Kokkos must as part of its pull request document that the appropriate API Reference entry was generated. The API Reference contains the equivalent of about 150 pages in approximately 60 different markdown documents.

A Guide to Testing and Issue Tracking documents the Kokkos Projects processes for testing as well as how users report issues and request new features or enhancements of existing capabilities. It also documents the release procedure in detail. In particular developers on the core Kokkos team need to be aware of these processes. This section is the smallest with about 20 pages content.

3.2 API REFERENCE CONTENT

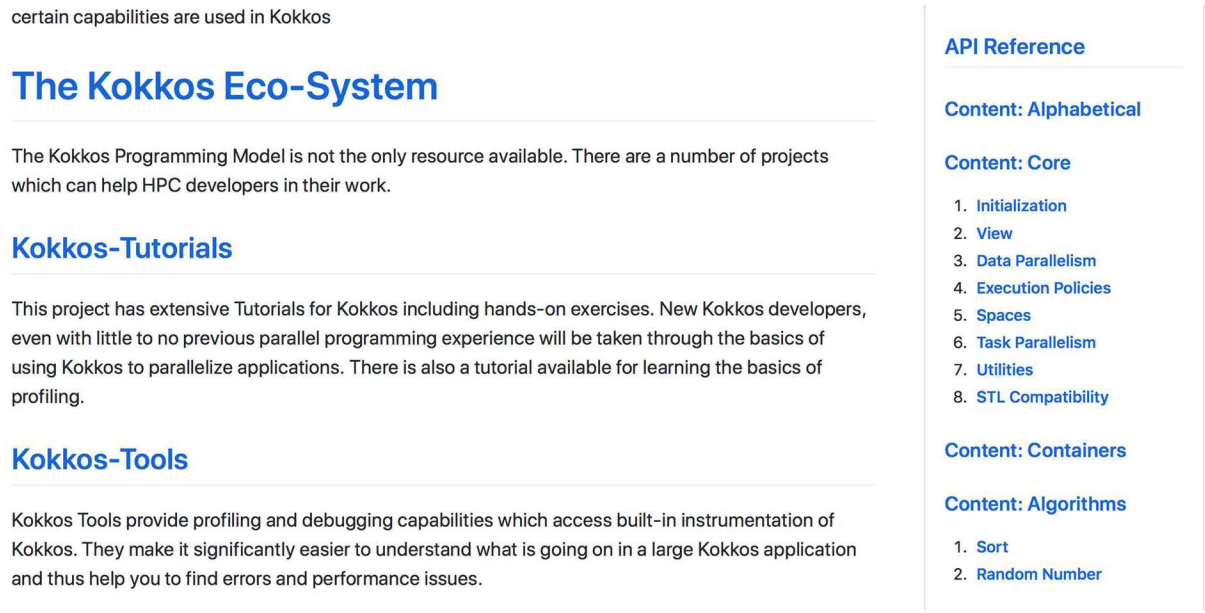
As described in the execution plan Kokkos capabilities had to be categorized into various feature areas. This categorization is reflected in the content structure of the API Reference subsection.

At the highest level the API Reference is split into the Core, Algorithms and Containers sections. Below that only Core is further divided into subcategories.

1) Initialization
2) View
    a. View
    b. subview
    c. realloc
    d. resize
    e. create_mirror
    f. create_mirror_view
3) Data Parallelism
    a. parallel_for
    b. parallel_reduce
    c. parallel_scan
    d. Built-in Reducers
4) Execution Policies
    a. RangePolicy
    b. MDRangePolicy
    c. TeamPolicy
    d. NestedPolicies

5) Spaces
    a. Execution Spaces
    b. Memory Spaces
6) Task Parallelism
7) Utilities
    a. Timer
8) STL Compatibility
    a. Array
    b. complex
    c. pair

This categorization is reflected in a permanent sidebar of the Wiki, allowing easy navigation back to each of the main points.


certain capabilities are used in Kokkos

The Kokkos Eco-System

The Kokkos Programming Model is not the only resource available. There are a number of projects which can help HPC developers in their work.

Kokkos-Tutorials

This project has extensive Tutorials for Kokkos including hands-on exercises. New Kokkos developers, even with little to no previous parallel programming experience will be taken through the basics of using Kokkos to parallelize applications. There is also a tutorial available for learning the basics of profiling.

Kokkos-Tools

Kokkos Tools provide profiling and debugging capabilities which access built-in instrumentation of Kokkos. They make it significantly easier to understand what is going on in a large Kokkos application and thus help you to find errors and performance issues.

API Reference

Content: Alphabetical

Content: Core

1. Initialization
2. View
3. Data Parallelism
4. Execution Policies
5. Spaces
6. Task Parallelism
7. Utilities
8. STL Compatibility

Content: Containers

Content: Algorithms

1. Sort
2. Random Number

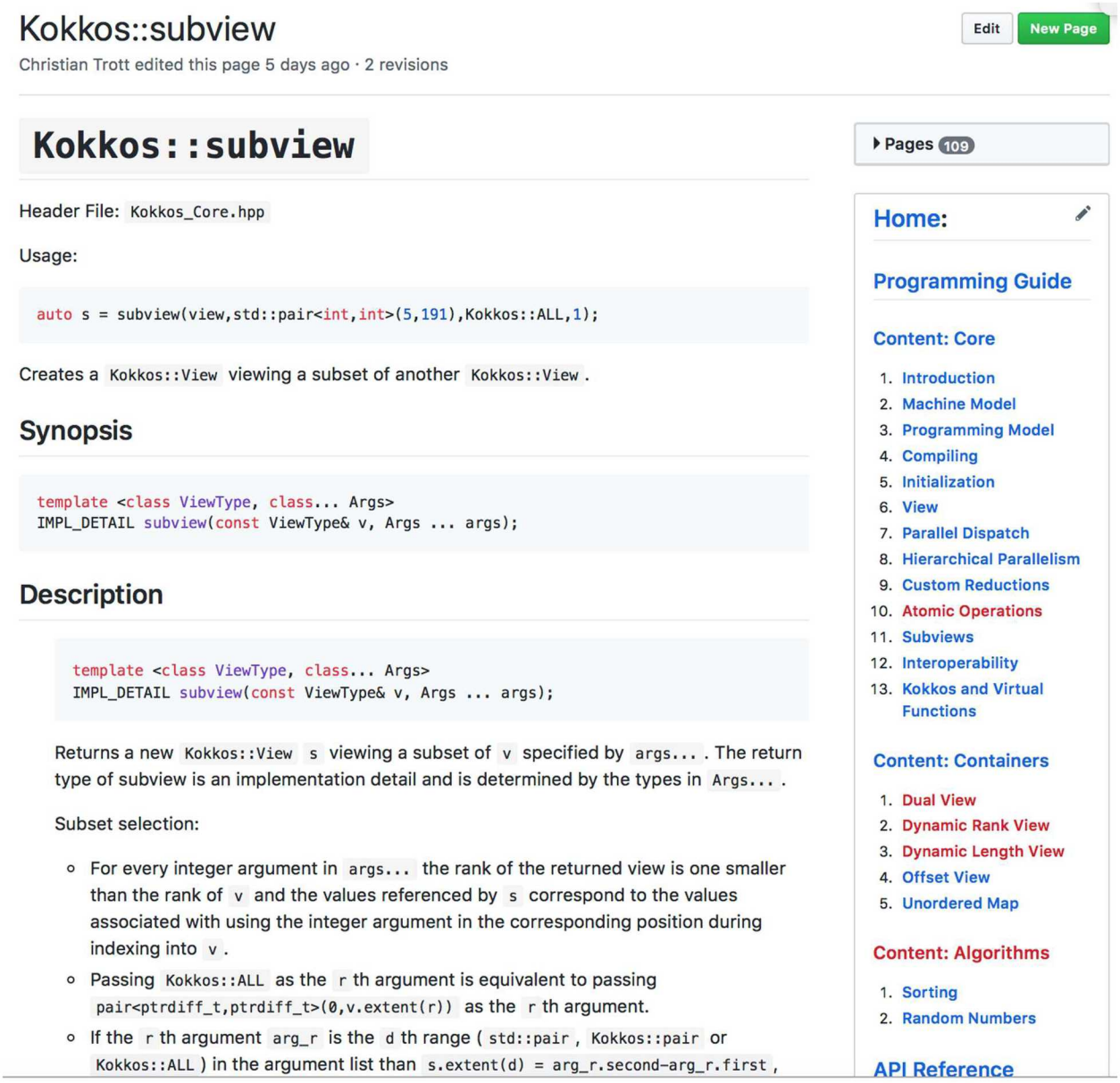
3.3 EXAMPLE: KOKKOS::SUBVIEW

A typical example for an API Reference page is subview. Generally the page is split into four sections:

1) A short description with a simple usage example.
2) A synopsis of the API.
3) A full description of the API.
4) A more detailed example.

The short description gives a overview of what this function is supposed to do, as well as a very brief one liner example. The synopsis lists the API in form of a C++ declaration. The full description provides the detailed description of each function, class member, typedef etc. in the particular capability. It will also

specify what each argument is, and what restrictions arguments have. Finally a more detailed example provides more code to demonstrate how to use a Kokkos feature.



Kokkos::subview

Edit New Page

Christian Trott edited this page 5 days ago · 2 revisions

Kokkos::subview

Header File: `Kokkos_Core.hpp`

Usage:

```
auto s = subview(view,std::pair<int,int>(5,191),Kokkos::ALL,1);
```

Creates a `Kokkos::View` viewing a subset of another `Kokkos::View`.

Synopsis

```
template <class ViewType, class... Args>
IMPL_DETAIL subview(const ViewType& v, Args ... args);
```

Description

```
template <class ViewType, class... Args>
IMPL_DETAIL subview(const ViewType& v, Args ... args);
```

Returns a new `Kokkos::View` `s` viewing a subset of `v` specified by `args...`. The return type of subview is an implementation detail and is determined by the types in `Args...`.

Subset selection:

- For every integer argument in `args...` the rank of the returned view is one smaller than the rank of `v` and the values referenced by `s` correspond to the values associated with using the integer argument in the corresponding position during indexing into `v`.
- Passing `Kokkos::ALL` as the `r` th argument is equivalent to passing `pair<ptrdiff_t,ptrdiff_t>(0,v.extent(r))` as the `r` th argument.
- If the `r` th argument `arg_r` is the `d` th range (`std::pair`, `Kokkos::pair` or `Kokkos::ALL`) in the argument list than `s.extent(d) = arg_r.second-arg_r.first`,

Pages 109

Home:

Programming Guide

Content: Core

1. Introduction
2. Machine Model
3. Programming Model
4. Compiling
5. Initialization
6. View
7. Parallel Dispatch
8. Hierarchical Parallelism
9. Custom Reductions
10. Atomic Operations
11. Subviews
12. Interoperability
13. Kokkos and Virtual Functions

Content: Containers

1. Dual View
2. Dynamic Rank View
3. Dynamic Length View
4. Offset View
5. Unordered Map

Content: Algorithms

1. Sorting
2. Random Numbers

API Reference

3.4 RESOURCE ACCESS

All of the resources are available on the public Kokkos wiki.

Feature	Location
General Wiki	https://github.com/kokkos/kokkos/wiki

Programming Guide	https://github.com/kokkos/kokkos/wiki/The-Kokkos-Programming-Guide
API Reference	https://github.com/kokkos/kokkos/wiki/API-Reference

4. RESOURCE REQUIREMENTS

The work performed here required 0.35 FTE.

5. CONCLUSIONS AND FUTURE WORK

One of the obversation made during this work was a difficulty of determining what needs to be documented. It is non-trivial to determine every public API feature of a project like Kokkos. We are exploring a way of making that determination through tools, possibly via plugins in the LLVM/Clang family of tools. Potentially this would allow us to add automatic testing of pull requests, which put in a warning if a new public API capability is added, so that the pull request approver can check whether the appropriate documentation is written. Furthermore, such a test could ensure that nobody accidently adds a new thing in the public namespace of Kokkos.

6. ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration—responsible for the planning and preparation of a capable exascale ecosystem—including software, applications, hardware, advanced system engineering, and early testbed platforms—to support the nation's exascale computing imperative.

Sandia National Laboratories is a multimission 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-NA-0003525.

This work was performed under US Government contract DE-AC52-06NA25396 for Los Alamos National Laboratory, which is operated by Los Alamos National Security, LLC for the U.S. Department of Energy.