SAND2019-14095R

ECP EXASCALE COMPUTING PROJECT

SAND2019- 9827 R

Engage the ISO C++ Standard Committee

WBS STPR 04 Milestone 26

Authors

Christian Trott (1)

Author Affiliations

(1) Sandia National Laboratories

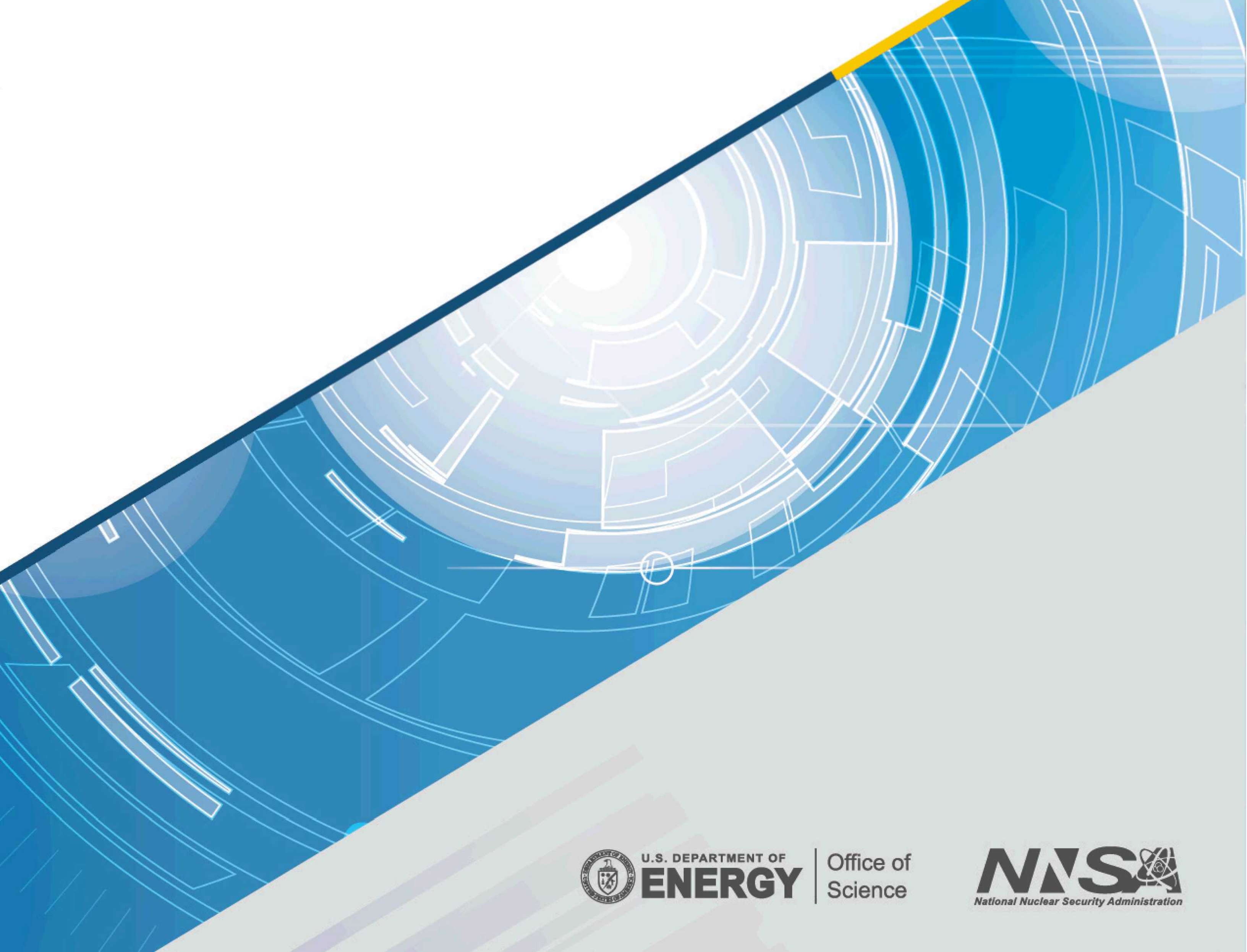
September 29th 2019

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ECP EXASCALE COMPUTING PROJECT

EXECUTIVE SUMMARY

This report documents the completion of milestone STPR04-26 Engaging the C++ Committee. The Kokkos team attended the three C++ Committee meetings in San Diego, Hawaii, and Cologne with multiple members, updated multiple in-flight proposals (e.g. MDSpan, atomic_ref), contributed to numerous proposals central for future capabilities in C++ (e.g. executors, affinity) and organized a new effort to introduce a Basic Linear Algebra library into the C++ standard. We also implemented a production quality version of mdspan as the basis for replacing the vast majority of the implementation of Kokkos::View, and thus start the transitioning of one of the core features in Kokkos to its future replacement.



U.S. DEPARTMENT OF ENERGY | Office of Science

NNSA National Nuclear Security Administration

1. INTRODUCTION

The Milestone is tracking engagement with the C++ Committee.

2. MILESTONE OVERVIEW

2.1 DESCRIPTION

We will participate in all official ISO C++ committee meetings, retaining our voting right.

We will collaborate in preparing proposals to transfer successful concepts from Kokkos into the C++ standard, or improving existing proposals.

Furthermore we will explore proposed C++ additions for their suitability for supporting or replacing Kokkos capabilities.

2.2 EXECUTION PLAN

The work for this milestone consists of four main parts:

- Prepare proposals for the C++ standard committee
- Attend virtual meetings with other committee meetings to collaborate on proposals
- Attend the three C++ committee meetings
- Implement production usable reference implementation of future C++ capabilities

2.3 COMPLETION CRITERIA

Completion of this milestone is achieved through submission of updated or new proposals to the C++ standard and release of implementations of such features.

3. TECHNICAL WORK SCOPE, APPROACH, RESULTS

3.1 UPDATED PROPOSALS

We updated in particular atomic_ref and mdspan based on feedback from the C++ committee and its evolving guidelines for how to write proposals. The vast majority of those updates are highly technical and only affect wording, not the actual content of the proposal. Having one of the Kokkos members in the Library Working Group was critical to this effort. The Library Working Group is the final step for proposals for the C++ standard library. The group is responsible to fix the wording of proposals so it conforms with the C++ standard wording requirements. Subtle changes can in fact change the semantic meaning of wording. Even such details as the font of a comma may change the binding meaning of a statement. Learning those subtleties requires commitment and time. Thus Dan Sunderland was assigned to attend all the Library Working Group meeting. The skills he gained through that are helping us now streamline our proposals so they have an increased chance to pass muster.

3.2 NEW PROPOSALS

The team wrote and submitted a number of new proposals in order to further our efforts of getting critical capabilities for HPC applications into the C++ standard. Our next core effort after mdspan is a proposal for including BLAS into the standard. A group originating in the graphics community had proposed linear algebra capabilities which would not meet our needs very well. Largely they only proposed a number data structures, but with few algorithms available. For HPC customers the availability of algorithms is more important than convenience data structures. We organized a group of members of the committee including the major hardware vendors (NVIDIA, Intel, AMD) to develop a proposal for BLAS algorithms based on the BLAS standard. That proposal (P1673) was submitted to the Cologne meeting in July and has passed the first subgroup with approval. The key aspects of the proposal are:

- Support for all algorithms in BLAS 1,2 and 3.
- Data Layout agnostic via the use of mdspan with layouts as arguments.
- Scalar agnostic via templating of input parameters
- Support for mixed precision due to arguments being independently templated
- Straight forward extension to badget BLAS possible.
- Use of execution policies in direct analogy to standard parallel algortihms for interoability with executors on heterogeneous machines

The design is inspired by Kokkos Kernels and can provide the same richness of capabilities with respect to supporting advanced architectures. A reference implementation is currently underway at https://github.com/kokkos/stdblas.

In order to support small linear algebra operations including matrix operations in registers we also developed a proposal for a container version of mdspan called mdarray. The goal is that the storage backing an mdarray can be a std::array and thus reside in registers. This would allow the efficient implementation of the operations such 4x4 matrix multiplies, building blocks of AI algorithms and block sparse operations in DOEs engineering and science codes.

3.3 CODE RELEASES

We three new capabilities in support of the C++ standard work.

MDSPAN

A new implementation of mdspan was written with the goal of serving as a portable, robust implementation for the community with a potential adoption into the main C++ standard library distributions. Furthermore, it should be able to replace the implementation details of Kokkos's View class. The developed implementation can be used portably (i.e. it also works on GPUs), it is usable from C++11 through C++20 (draft), and is well performance tested. A workshop paper was written for the Performance Portability and Productivity workshop at SC19. The paper was accepted and will be presented in November in Denver. The implementation is for now available at https://github.com/kokkos/mdspan. In the future it will be moved into Kokkos core, and then exported to the DESUL library, which we are maintaining jointly with Livermore as part of the RAJA/Kokkos ECP project.

MDARRAY

An implementation of mdarray was implemented, based on the code developed for the mdspan implementation. This implementation is available under https://github.com/kokkos/mdarray.

stdBLAS

The stdBLAS implementation for now serves primarily as a reference for the C++ committee to ponder over. The initial implementation is serial order, so that we have a working interface. The next step will be to add parallel implementations interfacing to KokkosKernels, MKL and cuBLAS. The repository can be found for now under: https://github.com/kokkos/stdBLAS. Currently it is a private repository, but access can be granted to ECP collaborators.

4. RESOURCE REQUIREMENTS

The work performed here required 1.0 FTE, distributed over 8 developers: Steven Bova, Nathan Ellingwood, David Hollman, Dan Ibanez, Duane Labreche, Jeff Miles, Dan Sunderland and Christian Trott.

5. CONCLUSIONS AND FUTURE WORK

Mdspan is getting close to making it into the standard. We will continue pushing for its adoption into the C++23 draft standard this FY. The new BLAS proposal has already support from the vendors, but it is early days in its adoption. We will react to feedback from the committee in the next couple meetings and tweak the proposal. Continued work on executors will help the proposals take shape in a way that they are usefull to the HPC community. We will also continue the work of replacing the internal implementations of Kokkos View and the atomics in Kokkos with implementations of the proposed C++ features.

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.