SANDIA REPORT

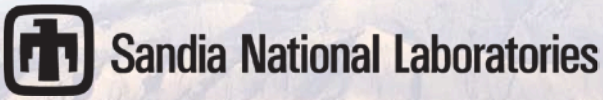
SAND2015-2160
Unlimited Release
Printed March 2015

ASC-ATDM Performance Portability Requirements for 2015-2019

H. Carter Edwards and Christian Trott

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2015-2160
Unlimited Release
Printed March 2015

ASC-ATDM Performance Portability Requirements for 2015-2019

H. Carter Edwards and Christian Trott
Computing Research Center
Sandia National Laboratories
P.O. Box 5800 / MS 1318
Albuquerque, New Mexico 87185

Abstract

This report outlines the research, development, and support requirements for the Advanced Simulation and Computing (ASC) Advanced Technology, Development, and Mitigation (ATDM) Performance Portability (a.k.a., Kokkos) project for 2015-2019.

The research and development (R&D) goal for Kokkos (v2) has been to create and demonstrate a thread-parallel programming model and standard C++ library-based implementation that enables performance portability across diverse manycore architectures such as multicore CPU, Intel Xeon Phi, and NVIDIA Kepler GPU. This R&D goal has been achieved for algorithms that use data parallel patterns including parallel-for, parallel-reduce, and parallel-scan. Current R&D is focusing on hierarchical parallel patterns such as a directed acyclic graph (DAG) of asynchronous tasks where each task contains nested data parallel algorithms. This five year plan includes R&D required to fully and performance portably exploit thread parallelism across current and anticipated next generation platforms (NGP)

The Kokkos library is being evaluated by many projects exploring algorithms and code design for NGP. Some production libraries and applications such as Trilinos and LAMMPS have already committed to Kokkos as their foundation for manycore parallelism and performance portability. These five year requirements includes support required for current and anticipated ASC projects to be effective and productive in their use of Kokkos on NGP.

The greatest risk to the success of Kokkos and ASC projects relying upon Kokkos is a lack of staffing resources to support Kokkos to the degree needed by these ASC projects. This support includes up-to-date tutorials, documentation, multi-platform (hardware and software stack) testing, minor feature enhancements, thread-scalable algorithm consulting, and managing collaborative R&D.

CONTENTS

NOMENCLATURE

ASC	Advanced Simulation and Computing
ATDM	Advanced Technology, Development, and Mitigation
DAG	directed acyclic graph
DOE	Department of Energy
ISO	International Standards Organization
NGP	Next generation platforms
SNL	Sandia National Laboratories

1 INTRODUCTION

Since its initiation in 2011 the research and development (R&D) goal for Kokkos (v2) has been to create and demonstrate a thread-parallel programming model and standard C++ library-based implementation that enables performance portability across diverse manycore architectures such as multicore CPU, Intel Xeon Phi, and NVIDIA Kepler GPU. This R&D goal has been achieved for algorithms that use data parallel patterns including parallel-for, parallel-reduce, and parallel-scan. Recent accomplishments include deployment of multi-level data parallelism based upon the league-of-teams-of-threads concept.

The Kokkos library is being evaluated by ASC-ATDM and other projects that are exploring algorithms and code design for NGP. Some production libraries and applications such as Trilinos and LAMMPS have already committed to Kokkos as their foundation for manycore parallelism and performance portability.

1.1 Greatest Risk: Inadequate Staffing for User Support and R&D

The greatest risk to the success of Kokkos and ASC/ATDM projects relying upon Kokkos is a lack of project staffing sufficient to support Kokkos to the degree required by these projects. This support includes up-to-date tutorials, documentation, multi-platform (hardware and software stack) testing, minor feature enhancements, thread-scalable algorithm consulting, and managing collaborative R&D.

Kokkos has been a highly successful R&D project in creating and demonstrating a performance portable thread-parallel programming model and standard C++ library-based implementation. The unique (versus other library- and compiler-based portable programming models) abstraction contributing to this success is the integration of parallel patterns and polymorphic multidimensional arrays. A consequence of this success is a growing community of domain library and application projects interest and exploration of Kokkos to support their NGP futures.

The Kokkos project is currently addressing the transition from a successful pure-R&D project to a project that both supports consumers of deployed capabilities and continues cutting-edge R&D for new functionality and NGP features. These five year project requirements assume that the ASC/ATDM program will adequately staff the Kokkos project for both essential Kokkos R&D and to support ASC/ATDM projects relying upon Kokkos.

1.2 Development of these Requirements

These five year requirements are developed from previously identified support and R&D requirements, and input from ASC/ATDM projects at Sandia.

2 USER AND COLLABORATOR SUPPORT

Many ASC-ATDM projects at SNL have identified Kokkos as a key technology for developing performance portable intra-node parallel capabilities. Other non-ATDM DOE projects have also expressed a similar need and intent to utilize Kokkos for their performance portable intra-node parallelism. For this growing community of DOE users to be productive the Kokkos project must include user and collaborator support activities.

2.1 DOE projects planning or investigating use of Kokkos

- ASC-ATDM
  - Application Components; Roger Pawlowski (PI); Glen Hansen (Mgr)
  - Analysis Components; Eric Phipps (PI); Jim Stewart (Mgr)
  - Thermal-Mechanical; Steve Bova (PI); Ryan Bond (Mgr)
  - Algorithm Components; Eric Cyr (PI); Mike Parks (Mgr)
  - SREMP/SGEMP; Matt Bettencourt (PI); Scott Hutchinson (Mgr)
  - Data Management; Craig Ulmer (PI); Ron Oldfield (Mgr)
  - Task Parallel; Janine Bennett (PI); Robert Clay (Mgr)
- ASC-Algorithms Trilinos suite
- ASC-Integrated Codes Sierra
- ASC-CSSE / Codesign
- PSAAP University of Utah

2.2 Rigorous multi-platform testing

Kokkos is intended to be a performance portable library for intra-node parallelism. Insuring portability requires rigorous multi-platform verification testing. The multidimensional "platform" space includes different hardware and their versions, operating systems and their versions, runtimes and their versions, compilers and their versions, language standard versions, compiler options, and integration with third party libraries (TPLs) and their versions. Build configuration files and scripts for tested platforms will be configuration managed and publically available. The SNL Computing Research Center's SEMS team will be engaged to support automated multi-platform nightly testing.

Rigorous multi-platform testing goals include 100% passing unit tests and examples, coverage of ATDM-prioritized users' platforms, integration testing with ATDM-prioritized users' applications and domain libraries, warning-free builds on selected platforms. Warning-free builds across all platforms can be problematic due to some platforms' competing or spurious warning analysis.

2.3 Collaborative R&D Configuration Management

The ease and acceptance of collaborations with researchers and users outside of SNL's Computing Research Center will be significantly improved by hosting Kokkos on the publically

accessible github site. Moving Kokkos to this site requires deploying new collaborative configuration management processes to insure that Kokkos code maintains sufficient quality and compatibility with ATDM-prioritized users' code.

2.4 Documentation and Tutorials

A productive and growing user and collaborative developer community requires user and developer documentation for Kokkos. Efficient learning by new users requires tutorial classes with associated materials and hands-on exercises. Production quality documentation and tutorials have not been generated to date as part of Kokkos' previous pure R&D effort.

2.5 Examples and Mini-applications

Early "alpha" users of Kokkos provided feedback that working examples utilizing Kokkos capabilities were highly valuable to their learning and effective use of Kokkos. These examples are designed to be both simple and reflect patterns found in application codes and domain libraries. Such examples are not unit tests; they are intended to demonstrate recommended usage patterns for Kokkos capabilities. Mini-applications represent the high-end of examples. Mini-applications, or lesser examples, are also used to measure performance and evaluate usability of Kokkos for "realistic" usage patterns. As such development and maintenance of examples and mini-applications is valued support activity.

2.6 Minor Feature Enhancements

Projects currently using Kokkos occasionally request minor feature enhancements within Kokkos that will improve their productivity. It is anticipated that such requests will grow with the number of projects adopting Kokkos.

2.7 Consulting and Collaboration

Application and domain library developers inexperienced with intra-node parallel algorithms and programming have often requested consulting to address thread safety, thread scalability, and productive use of Kokkos. In some cases the challenges presented by their functional requirements leads to both new algorithms and new requirements for Kokkos functionality to support those algorithms. Inter-project consulting and collaboration is critical for our community's effective transition to manycore technology.

Consulting and collaboration should include coordination of a Kokkos user community through workshops, mailing lists, blogs, or other appropriate activities. A coordinated user community will facilitate communication of usage scenarios and patterns, and aid in identifying common patterns which could be incorporated into Kokkos or library layered upon Kokkos.

3 RESEARCH AND DEVELOPMENT

3.1 Extensibility of Polymorphic Multidimensional Arrays

The Kokkos View polymorphic multidimensional array functionality is extensible with respect to array layouts, aggregate data types such as embedded automatic differentiation and uncertainty quantification stochastic variables, and special access capabilities such as GPU texture cache and atomic operations. The initial design for this extensibility requires excessive code duplication across specialized variants of the View implementation. A new design has been developed to reduce this duplication. The View implementation must be refactored to this new.

Primary customer is ASC-ATDM Analysis Components.

3.2 Atomic Operations on Aggregate Data Types

Kokkos currently maps a portable atomic operation interface to compiler extensions for atomics up to 64bit data types supported in hardware. Atomic operation on larger aggregate data types such as complex<double>, double-double, embedded uncertainty quantification, and automatic differentiation types are also required.

Primary customers are ASC-ATDM Analysis Components and Algorithm Components.

3.3 Array Data Compression

The LLNL floating point array compression library (ZFP) http://computation.llnl.gov/casc/zfp/ has demonstrated improved memory throughput in computational kernels. Such compression will also reduce inter-process message sizes. The design of this capability is compatible with the Kokkos View access annotation design-extension point. After the View implementation is refactored to reduce the effort associated with extensibility the floating point array compression functionality should be integrated as an option for Views.

Primary customers are ASC-ATDM Thermal-Mechanical and ASC-ATDM Data Management.

3.4 Multilevel Thread-Team Parallel Execution Capability

An initial Kokkos capability for multilevel thread-team parallelism has been implemented and exercised in tests and by alpha-users in their mini-applications. This capability features nested data-parallel operations and team-shared scratchpad memory. The current thread-team execution policy performs static scheduling and a dynamic schedule option is also required. A prototype enhancement of this capability includes a third level of parallelism intended to map to vectorised loops.

Primary customers are ASC-ATDM SREMP/SGEMP, Analysis Components, Algorithm Components, Application Components, and Thermal-Mechanical.

3.5 Multilevel Task-Thread-Team Parallel Execution Policy

The Kokkos/Qthreads LDRD is creating a prototype execution policy for a directed acyclic graph (DAG) of tasks that can execute on thread teams. This capability is being evaluated within a collection of mini-applications. This research must be matured for production use across back-end implementations. A significant challenge will be to support this capability on GPU architectures.

Primary customers are ASC-ATDM Algorithm Components and Task Parallel.

3.6 Thread-Scalable Memory Management and Containers

Applications with dynamic structures such as particle-in-cell, mesh adaptation, and dynamic task-DAG require the capability to allocate or reallocate data structures from within thread parallel computations. Develop thread-scalable memory management to support dynamically resizable rank-one arrays and dynamic task-DAGs.

Primary customers are ASC-ATDM SREMP/SGEMP, Algorithm Components, and Task Parallel.

3.7 Index-Set Execution Policy (RAJA compatibility)

The LLNL RAJA functionality can be implemented through Kokkos parallel-foreach pattern composed with a new index-set execution policy. Such a Kokkos execution policy could be composed with Kokkos parallel-reduce and parallel-scan patterns to provide LLNL's RAJA-based applications with an opportunity to use Kokkos to extend their thread-parallel capabilities.

Primary customer is ASC-CSSE / Codesign.

3.8 Scalable and Remote Execution and Memory Spaces

Kokkos abstractions currently support multiple types of execution and memory spaces; e.g., a single CPU space and single GPU space. Execution and memory space abstractions are defined to support multiple instances of execution and memory spaces, and map those instances to partitions of execution and memory resources; e.g., mapping instances to memory coherence domains of a Xeon Phi or multiple GPUs within a compute node.

Inter-process task-based programming models such as DHARMA, Uintah, and Legion move and execute coarse grain tasks among processes. These coarse grain tasks may be implemented with fine grain parallelism via Kokkos. For this integration coarse grain tasks will be associated with instances of Kokkos execution spaces and memory spaces, and concurrent memory management and execution dispatch must be supported among memory space and execution space instances.

Execution and memory space abstractions could be extended to access remote execution or memory resources; e.g., active messages or remote memory access. Such an extension has the possibility of allowing tightly integrated racks of compute nodes to be programmed as if it were

a single compute node with many space-instances. Depending upon the intra-rack networking capabilities such a programming model has the potential for improved performance over an MPI+X programming model.

Primary customers are ASC-ATDM Data Management, ASC-ATDM Task Parallel, ASC-CSSE / Codesign, and PSAAP Uintah.

3.9 Checkpointing to Persistent Memory Spaces

Kokkos' memory management abstractions support annotation of arrays with usage-attributes. This feature could be leveraged to denote arrays that must be included in checkpointing. Given this annotation mechanisms could be introduced to automatically copy all marked arrays to a local non-volatile memory space or remote memory space.

Primary customers for R&D collaboration are ASC-ATDM Data Management and ASC-CSSE.

3.10 Embedded Performance and Power Measurement

The Kokkos execution and memory management application programmer interface requires applications and domain libraries to execute thread parallel functions and allocate memory through specific Kokkos interfaces. These interfaces provide an opportunity to automatically introduce performance profiling instrumentation without modifying user code, the compiler, or the underlying runtime. R&D for capability is in collaboration with ASC-CSSE.

Primary customer for R&D collaboration is ASC-CSSE.

3.11 Embedded Data Race Error Detection

Kokkos' parallel functions and memory access interfaces provide an opportunity to automatically introduce data race detection mechanisms such as tracking Write-After-Write dependencies between threads where sync or fence operations have not been called. Such errors are extremely difficult to detect due to their non-deterministic behavior. Integrating with Kokkos' interface and semantics of could enable debugging with smaller executables than a fully debug-built application.

Primary customer for R&D collaboration is ASC-CSSE.

3.12 Research Automated Performance Tuning and Power Management

Given embedded performance and power measurement mechanisms parameterized variants of algorithms (e.g., parallel nesting levels or data tiling) can be automatically evaluated for their performance and optimal parameters automatically determined for particular architectures.

Primary customer for R&D collaboration is ASC-CSSE.

3.13 Support AMD Integrated CPU/GPU

Develop a Kokkos back-end for AMD integrated CPU/GPU architectures.

4 DOMAIN-COMMUNITY ENGAGEMENT

4.1 ISO / C++ Standard Committee

The ISO/C++ standard committee is addressing intra-node parallelism and multi-dimensional array functionality similar to what is implemented in Kokkos. Sustained participation in this committee is essential to realize the long-term vision that Kokkos functionality, in whole or in part, is replaced by features required by the ISO/C++ standard and implemented by vendors or communities implementing this standard.

4.2 OpenMP and OpenACC

OpenMP and OpenACC implementation projects are addressing challenges with the interaction between the semantics of these evolving standards, C++ semantics, and increasingly complex computational node architectures. Due to the success of the Kokkos programming model and library implementation; the Kokkos team has requested to consult with some of these projects to improve how implementation of these standards integrates with C++.

4.3 DOE Program Elements

The Kokkos team engages with other DOE program elements such as co-design, fast-forward, and ASCR to support research in effective intra-node parallel algorithms, programming models, and use of advanced computational architectures.

5 TIMELINE WITH ASSUMED STAFFING

5.1 Staffing Assumptions

- Increase and sustain (FY15-FY19) at 0.5 FTE focus on software quality engineering.
- Increase and sustain (FY15-FY19) at 0.5 FTE focus on research, development, minor feature enhancement, and consultation.
- Summer'15 contract to develop baseline in-depth tutorial

5.2 FY15

Support Objectives

- Deploy rigorous multi-platform testing
- Establish collaborative R&D configuration management repository at github
- Develop in-depth tutorial
- Minor feature enhancements
- Application and domain library consulting

Research & Development Objectives

- Refactor multidimensional array polymorphic layout implementation to reduce effort associated with developing and maintaining extensions.
- Develop thread-scalable memory management and thread-scalable dynamically resizable one dimensional array.
- Develop dynamic scheduling execution policy for thread-team parallelism.
- Collaborate with DHARMA and Uintah projects to R&D the domain boundary between coarse grain inter-process task programming models and Kokkos memory and execution spaces.
- Prototype league-team-vector parallel execution policy to support explicit expression of three-level of parallelism expected from ASC-ATS NGP: league/task, intra-team, and vectorization.

5.3 FY16

Support Objectives

- Demonstrate feature set within well-documented examples and mini-applications.
- Minor feature enhancements
- Application and domain library consulting

Research & Development Objectives

- Mature and deploy multilevel task-dag / thread-team parallel execution policy on CPU-type architectures.
- Develop atomic operations for aggregate data types.
- Develop implementations of execution and memory space **instance** abstractions to support partitioning of execution and memory resources and concurrent utilization by coarse grain inter-process task programming models.

- Collaborate with ASC-CSSE to research automatic, optional embedding of instrumentation for correctness and performance evaluation.
- Evaluate league-team-vector parallel execution policy on ASC-ATS testbeds.
- Develop index set execution policy for interoperability with RAJA-compatible applications.

5.4 FY17

Support Objectives

- Demonstrate new features within well-documented examples and mini-applications.
- Update tutorial to address new task-dag and space instance abstractions.
- Minor feature enhancements
- Application and domain library consulting

Research & Development Objectives

- Mature and deploy multilevel task-dag / thread-team parallel execution policy on GPU-type architectures.
- Develop Kokkos back-end suitable for AMD integrated CPU/GPU architectures.
- Develop Kokkos index-set execution policy that is compatible with LLNL's RAJA functionality.
- Research specification and interaction with remote memory spaces.
- Collaborate with ASC-CSSE to prototype within mini-applications automatic, optional embedded correctness and performance instrumentation.

5.5 FY18

Support Objectives

- Minor feature enhancements
- Update tutorial to address new embedded instrumentation.
- Application and domain library consulting

Research & Development Objectives

- Integrate an optional array data compression capability to enable improved memory-movement performance.
- Collaborate with ASC-ATDM Task Parallel to prototype within mini-applications integrated, hierarchical (inter-node + intra-node) task and task-data parallel functionality.
- Collaborate with ASC-ATDM Data Management to prototype within mini-applications using remote memory space abstractions for persistent data store.
- Collaborate with ASC-CSSE to evaluate within mini-applications automatic, optional embedded correctness and performance instrumentation.

5.6 FY19

Support Objectives

- Minor feature enhancements
- Application and domain library consulting

Research & Development Objectives

- Collaborate with ASC-ATDM Task Parallel to evaluate within mini-applications integrated, hierarchical (inter-node + intra-node) task and task-data parallel functionality.
- Collaborate with ASC-ATDM Data Management to evaluate within mini-applications using remote memory space abstractions for persistent data store.
- Research in collaboration with ASC-CSSE utilizing embedded instrumentation for performance auto-tuning.

6 DISTRIBUTION

1	MS0899	Technical Library	9536 (electronic copy)

Sandia National Laboratories