SAND2013-8143P

Extended Abstract Online Submission for the 2014 GPU Technology Conference

Session Title:

Kokkos, a Manycore Device Performance Portability Library for C++ HPC Applications

Session Description (100-150 words):

Prevalent manycore device portable programming models (e.g., OpenMP, OpenACC, OpenCL, Thrust) focus on parallel execution of computational kernels but fail to address memory access patterns critical for achieving best performance. Thus a "portable" code must still be extensively re-written to meet disparate and device specific memory access pattern requirements; e.g., data structures and loops transformed from array-of-structures (AoS) patterns to structure-of-arrays (SoA) patterns. The Kokkos library integrates abstractions for both parallel execution and memory access patterns to enable performance portability across disparate manycore devices such as NVidia Kepler and Intel Xeon Phi. At GTC13 we presented results demonstrating performance portability of several mini-application codes using a research version of Kokkos. At GTC14 we present manycore performance portability of the LAMMPS molecular dynamics code and the Trilinos/Tpetra linear solver implemented with MPI and release version of Kokkos and run on a clusters with Intel Xeon Phi and NVIDIA Kepler devices.

Intended Audience (50 words or less):

Developers of HPC computational science and engineering applications and libraries who are concerned with performance portability to disparate manycore devices.

Extended Abstract: (approx. 500 words):

Kokkos is a C++ library that enables portable execution of kernel in the spirit of Thrust; i.e., we provide a thin portability layer on top of Cuda, OpenMP, or Pthreads for parallel execution. In contrast to Thrust's simple vector abstraction Kokkos provides a multidimensional array abstraction in the spirit of Boost's MultiArray library. Kokkos multidimensional arrays are allocated in a specified device's memory space (similar to Thrust) and have polymorphic array layout (similar to MultiArray). By integrating these abstractions we can transparently choose the array layout for which a computational kernel will have the optimal memory access pattern for the target device. This choice is made at compile-time and implemented through C++ template meta-programming to maximize the C++ compiler's opportunity to optimize array access operators within a computational kernel. Enabled optimizations include vectorization on CPUs and Intel Xeon Phi and loop unrolling.

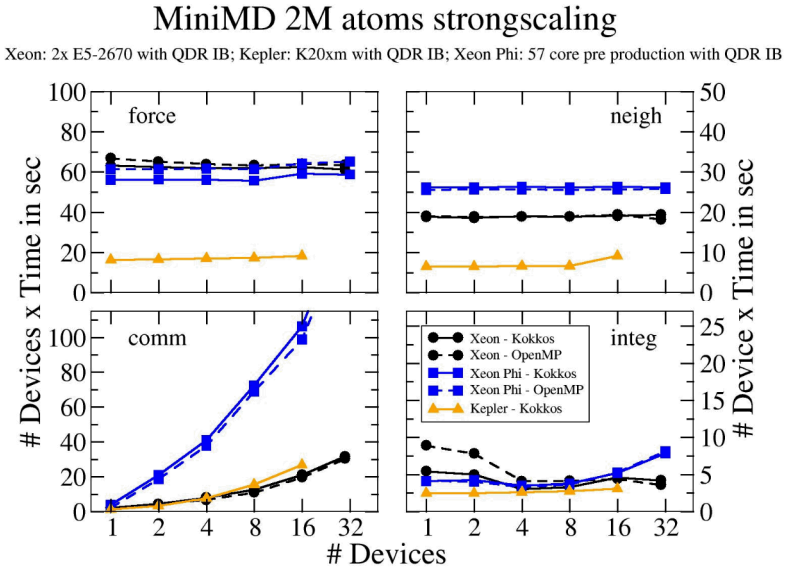
Kokkos research and development has focused on portability, performance, and usability. Our performance goal has been to achieve portable kernels with performance equal (or nearly equal to) corresponding kernels that are hand-coded according to each manycore device's performance capabilities. Our usability goal has been to provide an application programmer interface (API) that is simple and intuitive for developers of scientific and engineering kernels. Our greatest software development challenge has been to provide such an API while refusing to sacrifice performance and portability.

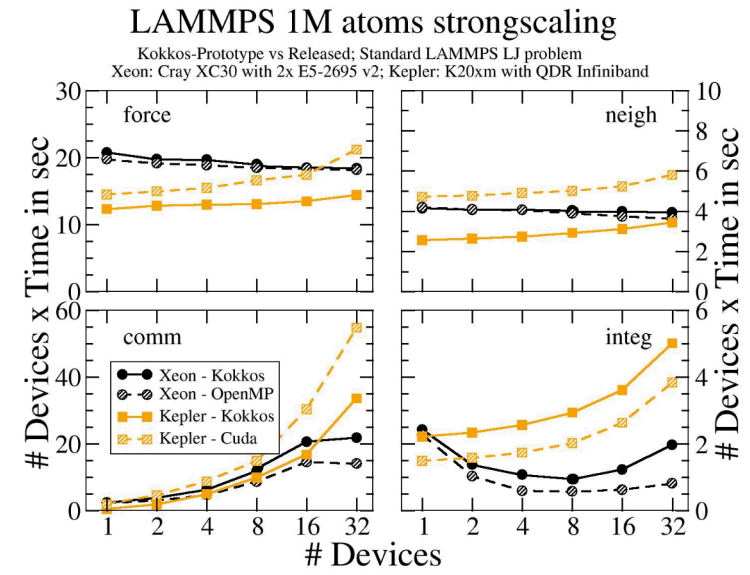
We will demonstrate how Kokkos enables developers to portably and transparently use special hardware features such as texture fetches on GPUs, small and large page sizes for different data allocations on Xeon Phi, and streaming load and store. We will present new capabilities developed since Kokkos was presented at GTC13 of portable multi-level parallelism, thread scalable unordered map, and sparse linear algebra. Multi-level parallelism abstracts OpenMP's league of thread-teams and CUDA grid of thread blocks to a portable interface with thread-team primitives such as barrier, reduction, prefix, and team-shared memory. A significant capability of the Kokkos unordered map is thread scalable insert and delete operations.

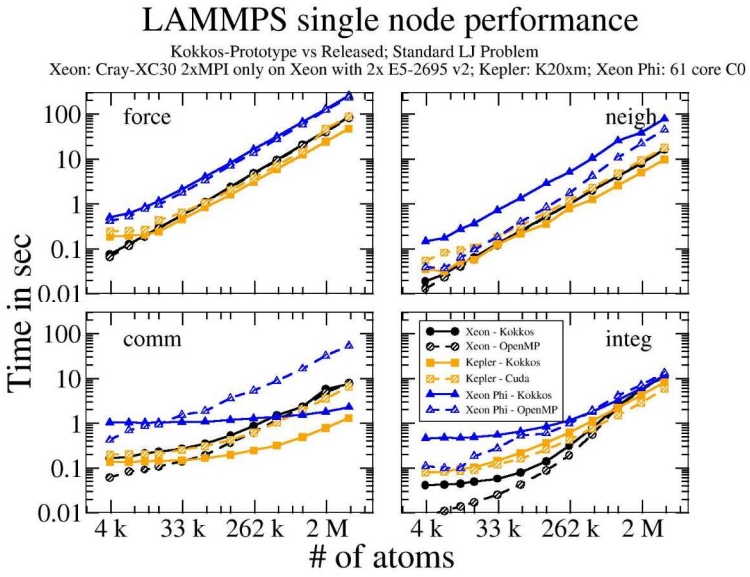
Also new this year, we will describe initial Kokkos prototypes of the LAMMPS molecular dynamics application and Trilinos/Tpetra templated hybrid parallel (using MPI+Kokkos) sparse linear algebra kernel library. We will show that LAMMPS' portable prototype provides as good or better performance than its release version – which uses separate code packages with replicated functionality to support threading via OpenMP and CUDA. We compare performance of Tpetra sparse matrix-vector multiply kernels with NVIDIA cusparse and Intel MKL libraries. Performance results are generated on clusters with NVIDIA Kepler and Intel Xeon Phi manycore devices.

The attached plots demonstrate performance on different platforms achieved with Kokkos versions of miniMD and LAMMPS compared to native programming models. The miniMD comparison uses the same algorithms; however, the LAMMPS prototype of Kokkos uses different algorithms than the released version of LAMMPS resulting in significantly different performance on Xeon Phi. Our experience from miniMD translates to LAMMPS, Kokkos versions achieve comparable (and on GPUs even better) performance than the current native implementations.

Kokkos has been released as part of the Trilinos suite of HPC libraries and is available through http://trilinos.sandia.gov.




LAMMPS 1M atoms strongscaling
Kokkos-Prototype vs Released; Standard LAMMPS LJ problem
Xeon: Cray XC30 with 2x E5-2695 v2; Kepler: K20xm with QDR Infiniband
force
neigh
comm
integ
Xeon - Kokkos
Xeon - OpenMP
Kepler - Kokkos
Kepler - Cuda
Devices x Time in sec
Devices
1
2
4
8
16
32


LAMMPS single node performance
Kokkos-Prototype vs Released; Standard LJ Problem
Xeon: Cray-XC30 2xMPI only on Xeon with 2x E5-2695 v2; Kepler: K20xm; Xeon Phi: 61 core C0
force
neigh
comm
integ
Xeon - Kokkos
Xeon - OpenMP
Kepler - Kokkos
Kepler - Cuda
Xeon Phi - Kokkos
Xeon Phi - OpenMP
Time in sec
# of atoms
4 k
33 k
262 k
2 M