Introduction

Problem:
Modern HPC applications, Finite Element Assembly codes in particular, need to be run on many different platforms and performance portability has become a critical issue: parallel code needs to be executed correctly and performant despite variation in the architecture, operating system and software libraries.

Approach:
Kokkos programming model from Trilinos: C++ library, which provide performance portability across diverse devices with different memory models.

Funding:
ASC Exascale Co-design Project

Kokkos programming model


- Provides portability across manycore devices [Multicore CPU, NVidia GPU, Intel Xeon Phi (potential: AMD Fusion)]
- Abstract data layout for non-trivial data structures
- Uses library approach:
  - Maximize amount of user/application/library code that can be compiled without modification and run on these architectures
  - Minimize amount of architecture-specific knowledge that a user is required to have
- Performant: Portable user code performs as well as architecture-specific code

Main abstractions:
- Kokkos executes computational kernels in fine-grained data parallel within an execution space.
- Computational kernels operate on multidimensional arrays [Kokkos::View] residing in Memory spaces.
- Kokkos provides these multidimensional arrays with polymorphic data layout, similar to the Boost.MultiArray flexible storage ordering.

Ice Sheet Simulation Code (FELIX)

We are developing a performance portable implementation of an Ice Sheet simulation code*, build with the Albany [3] application development environment

Project objectives:
- Develop a robust, scalable, unstructured grid finite element code for land ice dynamics.
- Provide sea level rise prediction
- Run on new architecture machines (hybrid systems).

Implementation algorithm:

1) Replace array allocations with Kokkos::Views (in Host space)

2) Replace array access with Kokkos::Views

3) Replace functions with functors, run in parallel on Host

4) Call Kokkos::parallel_for<> [...J] over the number of elements in workset:

5) Set Device to ‘Cuda’, ‘OpenMP’ or ‘Threads’ and run on a specified Device

Performance Results*

Albany MiniDriver test code

Evaluation Environment:
Compton (Intel NUC cluster): 64 nodes:
- Two 8-core Sandy Bridge Xeon E5-2670 @ 2.60GHz (32 threads per node)
- Two Intel Xeon Phi (KNC) 2 per node (20 threads per node)
Shannon (NVidia GPU cluster): 64 nodes:
- Two 8-core Sandy Bridge Xeon E5-2670 @ 2.60GHz (32 threads per node)
- 12GB of DDR4 memory per node
- 12GB of DDR5 memory per node

Albany FELIX code

Graph of Finite Element Assembly Kernels

References:
2) Salinger, Andrea G. Component-based scientific application development. December 01, 2012

*Funding Source: PSC/CMC SciDAC Application (BER & ASCR)

Authors: A. Salinger, I. Kalashnikova, M. Perigo, R. Tuminaro [SNL], S. Price [LANL]