Performance Portability of Shallow Water Model with Kokkos
Zephaniah Connell\textsuperscript{1,2} and Leila Ghaffari\textsuperscript{1,3}

\textsuperscript{1}National Center for Atmospheric Research
\textsuperscript{2}University of Wyoming
\textsuperscript{3}University of Colorado Boulder

Motivation

➢ Portability is a desired capability which enables us to run our code on ever-changing hardware and software platforms.
➢ It can be difficult and time-consuming to port or develop multiple versions of code that only run on specific architectures.
➢ Kokkos is a new framework that advertises the ability to execute the same code on CPU or accelerators with limited or no modifications.

Introduction to the Shallow Water Model (SWM) mini-app

SWM is a venerable 2D shallow water model benchmark on staggered finite difference equations on a torus.

Introduction to Kokkos

Kokkos is a C++ library that can be used to write a single source code that can execute serially on a CPU, in parallel on a CPU using OpenMP backend, and in parallel on a GPU using CUDA backend. It is performance portable because it is architecture aware.

Architectures:
GPU: Nvidia, AMD, Intel GPUs
CPU: x86, Power 8, KNL, ARM

Compilers:
GNU 5.3.0 or newer
Intel 17.0.1 or newer
Clang 4.0.0 or newer
PGI 18.7 or newer
CUDA 9.1 or newer

Pattern
Parallel structure

Policy
Index space

Views
Multi-dimensional data class

Kernel
Work performed on each index

Execution / Memory Spaces
Memory location, execution hardware, and execution method

Performance & Accuracy - CPU - Serial

Kokkos performed \textasciitilde 50x slower than C++

Performance - CPU - Serial

Kokkos Skylake 1-Core gnu/8.3.0 -O2
C++ Skylake 1-Core gnu/8.3.0 -O2

Accuracy Compared to C++ Serial Results

Data Size

Data Size

Performance & Accuracy - GPU - Parallel

Kokkos performed \textasciitilde 44x slower than OpenMP and \textasciitilde 1.67x slower than C++ Serial

Performance - GPU - Parallel

OpenMP Skylake 36-Core gnu/8.3.0 -O2
Kokkos Skylake 36-Core gnu/8.3.0 -O2

Accuracy Compared to C++ Serial Results

Data Size

Data Size

Performance & Accuracy - CPU - Parallel

Kokkos performed \textasciitilde 44x slower than OpenMP and \textasciitilde 1.67x slower than C++ Serial

Performance - CPU - Parallel

OpenMP Skylake 36-Core gnu/8.3.0 -O2
Kokkos Skylake 36-Core gnu/8.3.0 -O2

Accuracy Compared to C++ Serial Results

Data Size

Data Size

Performance & Accuracy - GPU - Serial

Kokkos performed \textasciitilde 50x slower than C++

Performance - GPU - Serial

Kokkos Skylake 1-Core gnu/8.3.0 -O2
C++ Skylake 1-Core gnu/8.3.0 -O2

Accuracy Compared to C++ Serial Results

Data Size

Data Size

Conclusions

➢ A Kokkos source code file can execute on many architectures
➢ Most Kokkos concepts are straightforward, so porting to Kokkos generally isn't difficult but time consuming
➢ The CPU performance for Serial and Parallel versions of Kokkos was poor and needs further investigation
➢ The GPU performance of Kokkos was reasonable, but also needs further investigation
➢ The Kokkos GitHub repository Wiki contains relatively comprehensive documentation
➢ The Kokkos developers provide helpful assistance on Slack within minutes

In my opinion, for any project that may benefit from executing code on different GPU architectures, Kokkos is worthwhile.

Future work

➢ Run ported SWM code on Intel and AMD GPUs
➢ Remeasure performance after implementing the following or other optimizations discovered after further research:
  ○ Explicit memory layouts
  ○ Refactoring the SWM data structures
  ○ Enabling vectorization for Views
  ○ Using TeamPolicy w/ lower level optimizations and indexing
➢ Test performance of multi-node and multiple GPU runs w/ MPI
➢ Further explore interoperability with 3rd party profilers

References


Acknowledgements

Mentors: Supreeth Suresh, Cena Miller, Jian Sun, and John Dennis
Research Support: Richard Loft and Thomas Hauser
SIParCS Admins and CODE Assistants: AJ Lauer, Virginia Do, Jerry Cyccone, Max Cordes Galbraith