Performance Portability of Shallow Water Model with Kokkos

Zephaniah Connell[1,2] and Leila Ghaffari[1,3]

[1]National Center for Atmospheric Research
[2]Univeristy of Wyoming
[3]University of Colorado Boulder

Motivation

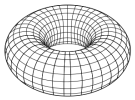
- Portability is a desired capability which enables us to run our code on ever-changing hardware and software platforms.
- It can be difficult and time-consuming to port or develop multiple versions of code that only run on specific architectures.
- Kokkos is a new framework that advertises the ability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

Goal

- Port the Shallow Water Model (SWM) mini-app to Kokkos with limited modifications
- Optimize the performance of the ported code on different hardware platforms

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

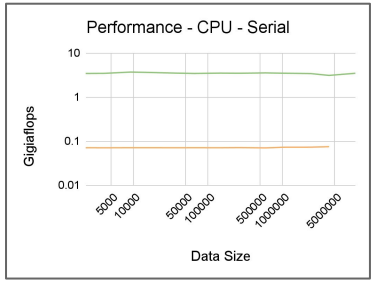
Kokkos is a C++ library that can be used to write a single source code that can execute serially on a CPU, in parallel on a CPU using OpenMP backend, and in parallel on a GPU using CUDA backend. It is performance portable because it is architecture aware.

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

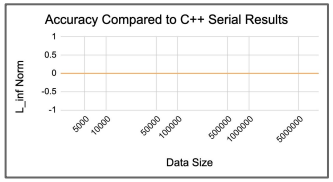
Pattern	Parallel structure
Policy	Index space
Views	Multi-dimensional data class
Kernel	Work performed on each index
Execution / Memory Spaces	Memory location, execution hardware, and execution method

Performance & Accuracy - CPU - Serial

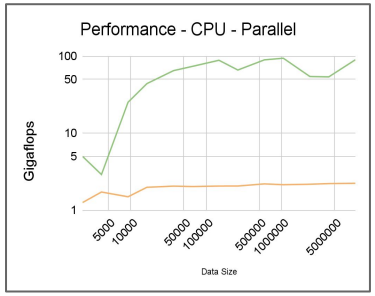


Kokkos performed ~50x slower than C++

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

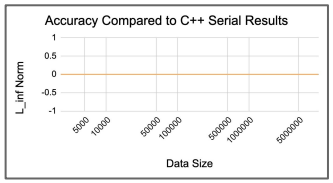


Performance & Accuracy - CPU - Parallel

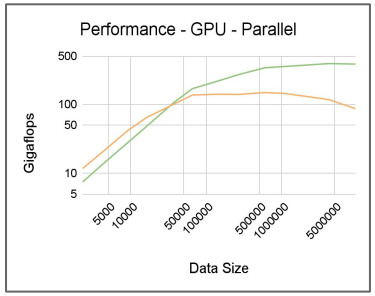


Kokkos performed ~44x slower than OpenMP and ~1.67x slower than C++ Serial

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

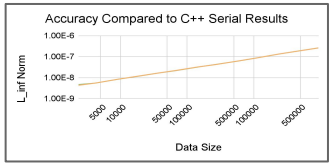


Performance & Accuracy - GPU - Parallel



Best performance: ~1.6x faster than OpenACC and ~43x faster than C++ Serial
Worst performance: ~4.4x slower than OpenACC and ~3.4x faster than C++ Serial

- OpenACC V100 nvhpc/21.3 cuda/10.1 -O2
- Kokkos V100 gnu/8.3.0 cuda/10.1 -O2



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

Carter Edwards, H., Trott, C. R., & Sunderland, D. (2014). Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, *74*(12), 3202–3216. https://doi.org/10.1016/j.jpdc.2014.07.003

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