Experiences with Kokkos in E3SM

L.Bertagna

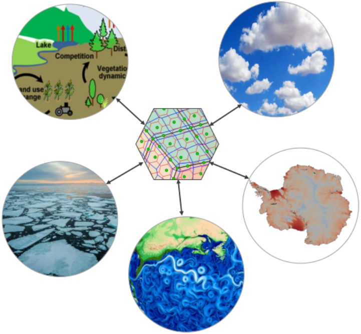
Sandia National Laboratories, Albuquerque, NM

October 7th, 2022

SAND SAND2022-13626 C

Energy Exascale Earth System Model (E^{3}SM)

- DOE's state-of-the-science Earth system model
- Several components: atmosphere, land, ocean, land-ice, sea-ice, etc.
- Components can run at different resolutions.
- Broad variety of time/space scales.
- Mostly written in Fortran



E3SM push to exascale

- Decision makers need reliable climate prediction, with reduced (and quantifiable) uncertainty.
- Increased resolution can help reduce uncertainty, by eliminating the need for certain subgrid approximations.
- In the last decade, improvements in HPC architectures and platforms have taken us to the edge of exascale computing.

This led to big efforts in E3SM, to fully utilize (almost) exascale clusters to answer important science questions. In particular, E3SM needs to run efficiently on all the large DOE supercomputers, most importantly on GPU-based clusters (Summit, Frontier).

Performance portability strategies

Performance Portability: capability of a code base to run "efficiently" on a variety of computer architectures. Three main approaches:

- **compiler directives**: hint/force compiler on how to optimize (OpenMP, OpenACC).
- **general purpose lib**: delegate architecture-specific choices to a TPL (Kokkos, Raja, etc.)
- **domain-specific lang/lib**: add intermediate compilation step, to generate optimal source for the given architecture.

In general, for an efficient use of HPC architectures, three important aspects: **expose parallelism**, **maximize vectorization**, **minimize memory movement**.

Kokkos

- C++ library for on-node parallelism, developed at SNL
- Provides constructs for expressing parallelism: execution space, execution policy, parallel operation.
- Provides constructs for multi-dimensional arrays: data type, memory space, layout, memory access/handling.
- Supports several back ends: OpenMP, Pthreads, Cuda, HIP, SYCL, etc.

EAMxx Kokkos port

Guiding principles for C++/Kokkos port of E3SM Atmosphere Model (EAM)

- Expose parallelism: use teams of threads to share common work and minimize index book-keeping.
- Maximize vectorization: use “pack” data structures and “masked” packed operations, to enhance vectorization.
- Minimize memory movement: keep data on device, except for I/O or coupling with host-only components.
- Minimize memory movement: use (and share) minimally sized scratch memory within team parallel loops.

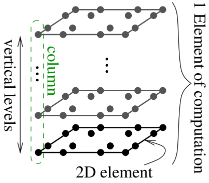
Timeline:

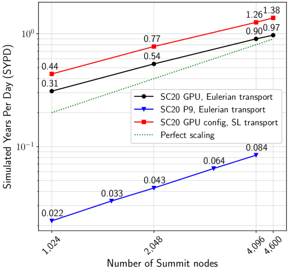
- 2018: port of hydrostatic dynamical core
- 2020: port of non-hydrostatic dynamical core
- 2022: port of full atmosphere component (ultra-high res only, no deep-convection)

EAMxx: Kokkos Performance

Dycore-only performance

- NGPPS benchmark (10 tracers) at 3km and 1km resolution.
- 1.38 SYPD on full Summit system (SL transport)
- At scale, we have roughly 225 horizontal elements/GPU

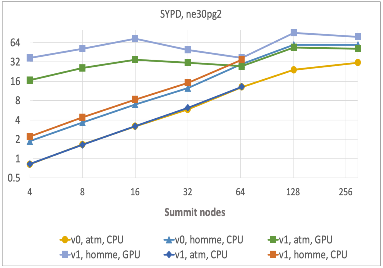




EAMxx: Kokkos Performance

Full atmosphere performance

- 10 tracers, 1 degree horizontal resolution (110km)
- 225 elements/GPU using 4 nodes, 14 elements/GPU at 64 nodes
- Roughly 20x speedup of GPU vs CPU at high workload, roughly 2x at low workload
- 3km resolution simulation campaign is ongoing



Ups and downs

- + Kokkos efficiently maps to current (and upcoming) architectures.
- + Kokkos can reduce disruption from new architectures.
- + Good speedup of GPU vs CPU, as well as CPU performance.
- + Hierarchical parallelism crucial to expose all algorithmic parallelism.
- - C++ syntax is *much* heavier than Fortran, which can be a barrier for field scientists.
- - With a GP performance portability library, simple algorithmic steps can require several lines of code to be expressed (in a performance portable way).
- - C++ does not vectorize as well as F90, so pack-like structures needed for CPU performance. All areas of the code need to be aware of packs (e.g., consider possible array padding). As a result, code becomes longer and more involved.