SAND2018-5797C

June 4, 2018

A Performance-Portable C++ Implementation of Atmospheric Dynamics

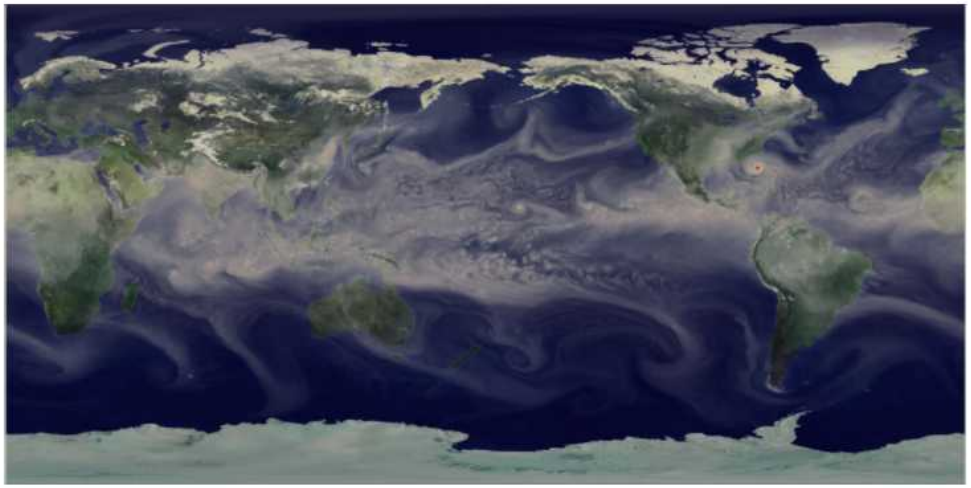
A. Salinger, L. Bertagna, A. Bradley,
M. Deakin, O. Guba, D. Sunderland,
I. Tezaur
Sandia National Laboratories
Albuquerque, NM, USA

ories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of San
well International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under cont

Outline

Energy Exascale Earth System Model (E3SM)

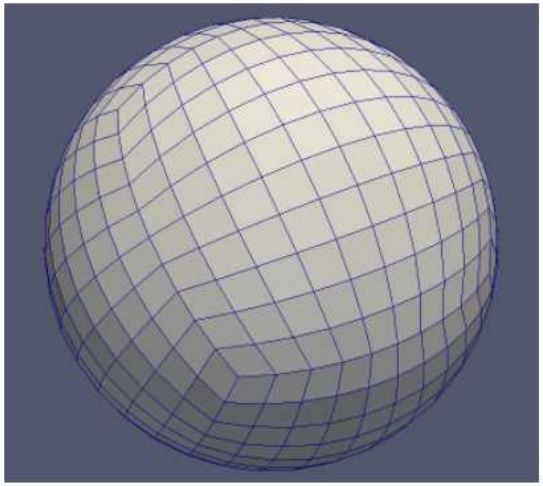
Sandia National Laboratories

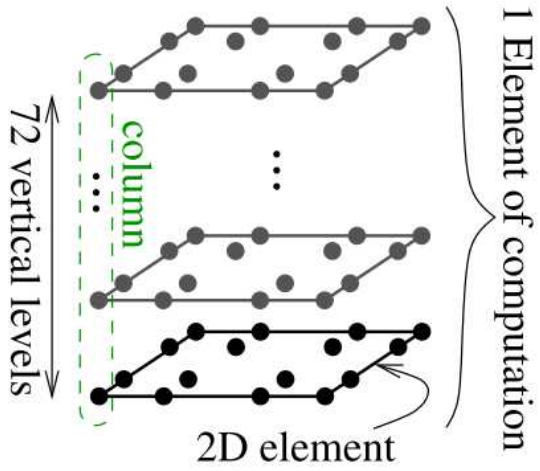


- Open source with open development at `https://github.com/E3SM-Project/E3SM`
- Fork of Community Earth System Model (CESM)[9] from 2014
- Several Components: **Atmosphere Dynamics**, Atmosphere Physics, Ocean, Sea Ice, ...
  - We are only concerned with the Atmosphere Dynamics

HOMME Atmosphere Dynamics

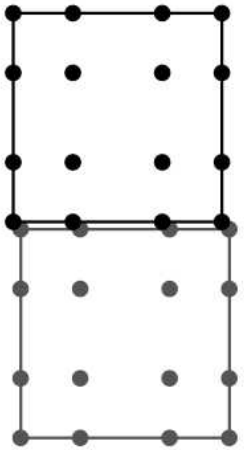
- High-Order Methods Modeling Environment (HOMME)[6]
- Joint project among the DOE, NSF, NCAR, and several laboratories and universities
- Cubed Sphere Mesh with 72 level extruded elements
  - Terrain following vertical coordinate with material surfaces, ie no vertical flux

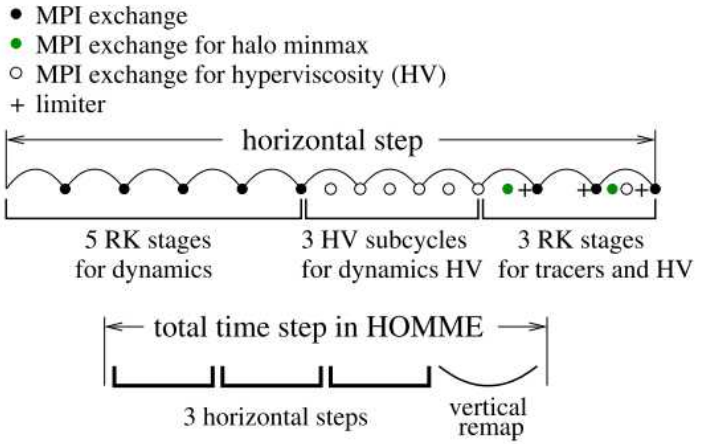




HOMME Atmosphere Dynamics

- Spectral Element Method (SEM) horizontal discretization [4]
  - Continuous Galerkin method with high accuracy and scalability
  - DOFs and quadrature points chosen to coincide
- Solves for 4 prognostic variables: Pressure, Temperature, and 2 Velocity components

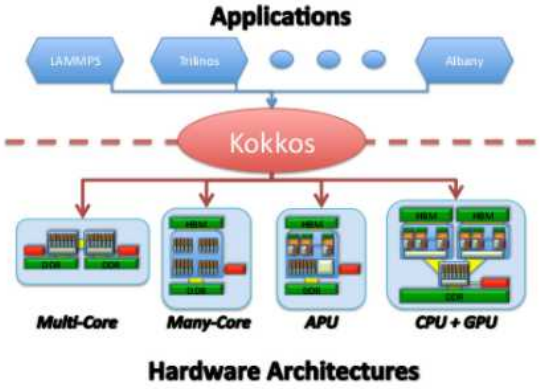




Kokkos [5]

Sandia National Laboratories

- Open source with open development at `https://github.com/kokkos/kokkos`
- Performance Portability Layer, similar to RAJA [8], HEMI [7], OpenACC [3], and future C++ STL features[1]
- Not pragma based, C++11 compliant, uses template metaprogramming
- Provides memory space abstractions in addition to parallel dispatch abstractions
- Performance overhead of at most 10%



[11]

Kokkos

Sandia National Laboratories

Key Definitions:

- **Kernel**: A body of work to be performed
- **Execution Space**: Describes the device kernels should execute on
- **Execution Pattern**: Scan, Reduce, or Map; primitives for parallel algorithms
- **Execution Policy**: How a kernel receives work items.
  - Range: Work is mapped to a range of integers from $[L, U)$
  - Team: Provides a team of warps/threads to each work item. Exposes hierarchical parallelism, avoids duplicating work and extra buffers
- **View**: A multi-dimensional array reference. Abstracts memory (de)allocation and accesses

Kokkos Examples

Sandia National Laboratories

```
Kokkos::parallel_for(
  Kokkos::RangePolicy<ExecSpace>(0, elements.num_elems()*NP*NP*NUM_LEV),
  KOKKOS_LAMBDA(const int it) {
     const int ie = it / (NP*NP*NUM_LEV);
     const int igp = (it / (NP*NUM_LEV)) % NP;
     const int jgp = (it / NUM_LEV) % NP;
     const int ilev = it % NUM_LEV;
     t(ie,nm1,igp,jgp,ilev) = (5.0*t(ie,nm1,igp,jgp,ilev)-t(ie,n0,igp,jgp,ilev))/4.0;
     v(ie,nm1,0,igp,jgp,ilev) = (5.0*v(ie,nm1,0,igp,jgp,ilev)-v(ie,n0,0,igp,jgp,ilev))/4.0;
     v(ie,nm1,1,igp,jgp,ilev) = (5.0*v(ie,nm1,1,igp,jgp,ilev)-v(ie,n0,1,igp,jgp,ilev))/4.0;
     dp3d(ie,nm1,igp,jgp,ilev) = (5.0*dp3d(ie,nm1,igp,jgp,ilev)-dp3d(ie,n0,igp,jgp,ilev))/4.0;
});
```

Figure : Example of range parallelism in HOMMEXX

Kokkos Examples

Sandia National Laboratories

```
Kokkos::parallel_for(
  Homme::get_default_team_policy<ExecSpace>(m_elements.num_elems()),
  KOKKOS_LAMBDA (const TeamMember& team) {
    KernelVariables kv(team);
    Kokkos::parallel_for (
      Kokkos::TeamThreadRange(kv.team, NP*NP),
      [&] (const int loop_idx) {
        const int i = loop_idx / NP;
        const int j = loop_idx % NP;
        Kokkos::parallel_for(
          Kokkos::ThreadVectorRange(kv.team, NUM_LEV),
          [&] (const int& k) {
            buf(kv.ie,i,j,k) =
              dp(kv.ie,i,j,k) - rhsmdt * divdp_proj(kv.ie,i,j,k);
          });
      });
  });
```

Figure : Example of hierarchical parallelism in HOMMEXX

HOMMEXX Design Overview

Sandia National Laboratories

- Incremental conversion of algorithms from FORTRAN to C++
- Maintained bit-for-bit compatibility (even on GPU!)
- Heavily unit tested, $\sim 85\%$ of kernels individually verified
- Abstracted most architecture dependent code
- First converted RK stages and spherical operators; used to evaluate design choices such as data structures
- Primary performance goals: Expose Parallelism, Minimize Memory Movement, Expose Vectorization
- In addition, minimize code specific to different architectures, and use C++11 in the design

HOMMEXX Design - Expose Parallelism

Sandia National Laboratories

4 levels of parallelism in HOMME: Elements, Variables/Tracers, Vertical Levels, GLL Points.
Expose parallelism by mapping these to hierarchical parallelism

- Kokkos' hierarchical parallelism only supports 3 levels... Solve by treating elements and variables as one level
- Naturally maps to spherical operators and other interior parallel regions

HOMMEXX Design - Vectorization

Sandia National Laboratories

Explicit Vectorization used on CPU

- Abstracted data type `double` as a packed `vector` type
- GPUs use SIMT rather than SIMD, so packs are size 1
- CPU vector size varies, KNL, Skylake size 8, Haswell size 4
- Alternative: pack data and surround loops with `#pragma simd`
  - C++ compilers don't generally vectorize well [10]
  - Explicit vectorization provides a $\sim 15\%$ boost over this method
  - C++ has a proposal for explicit vectorization support [2]

HOMMEXX Design - Data Layout

Sandia National Laboratories

Three priorities

- Enable vectorization
- Reduce required buffers
- Match fast indices with loop order

Two simple non-tiling solutions: make GLL point indices fast, or vertical level indices fast. Each choice has trade-offs:

- Vectorization in (common) spherical operators vs Vectorization in vertical integrals (rare)
- Kokkos' hierarchical parallelism has a parallel scan implementation for fast indices, not for slower ones
- Matching the number of vertical levels with vector sizes is feasible, matching the number of points is more difficult

HOMMEXX Design - Modern C++11

Sandia National Laboratories

- Template metaprogramming enabled static polymorphism.
  Can reduce low level `if` statements
  Can reduce register pressure on GPUs, improving functionality without performance costs
- Pointer to Implementation to reduce compilation times, esp for static polymorphism.
- Memory ownership semantics prevent memory leaks.

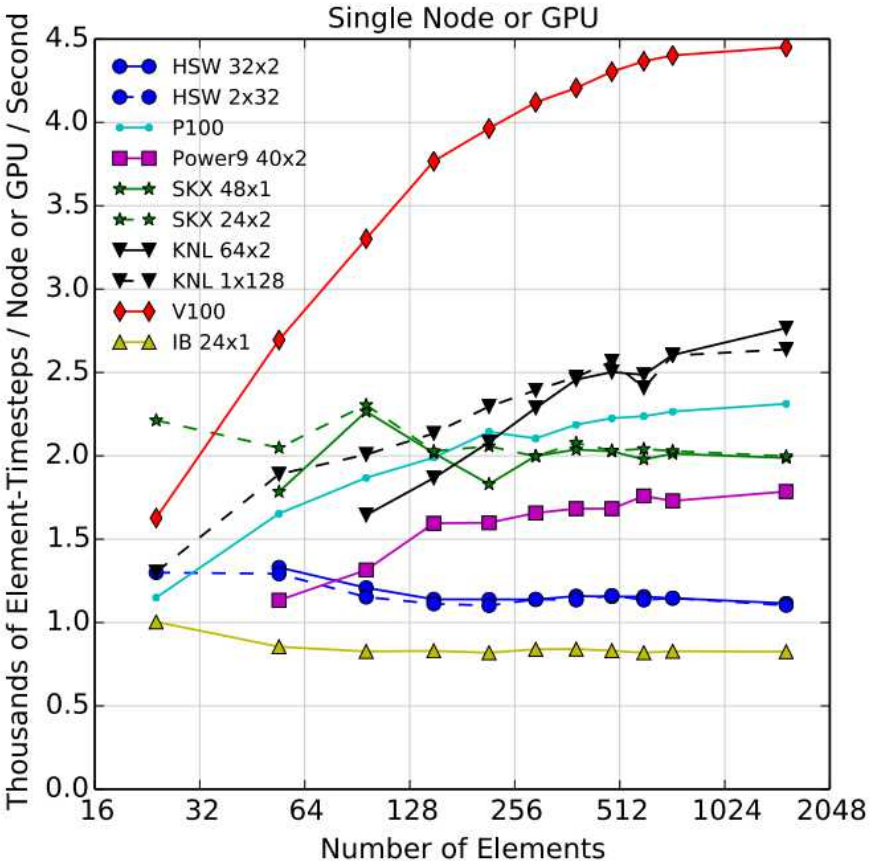
Architectures

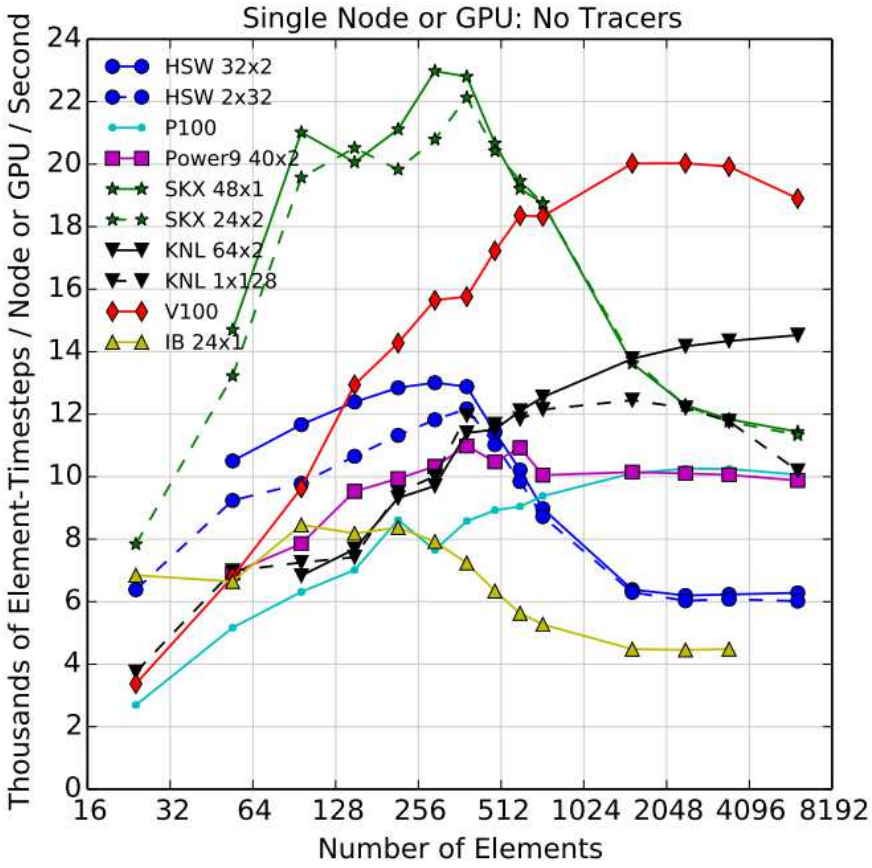
Sandia National Laboratories

Compared performance of CPU to HOMME on several machines, and examined GPU performance.
Architectures tested:

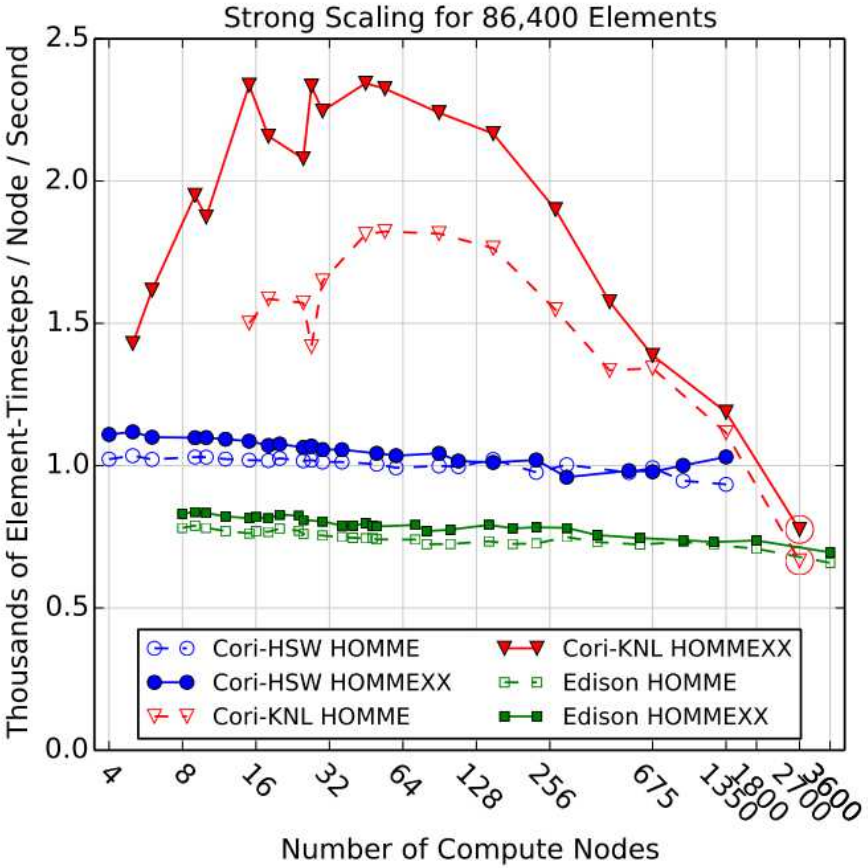
- Edison (IB): 2 sockets, 12 cores, 2 threads, Ivy Bridge, DDR3
- Cori Haswell (HSW): 2 sockets, 16 cores, 32 threads, Haswell, DDR4
- Cori KNL (KNL): 68 cores, 272 threads, Xeon Phi Knights Landing, MCDRAM
- Skylake (SKX): 2 sockets, 24 cores, 2 threads, DDR4
- GPU Testbed (V100, Power 9): 2 POWER9 CPUs, 10 cores, 2 threads, 4 Nvidia V100 GPUs

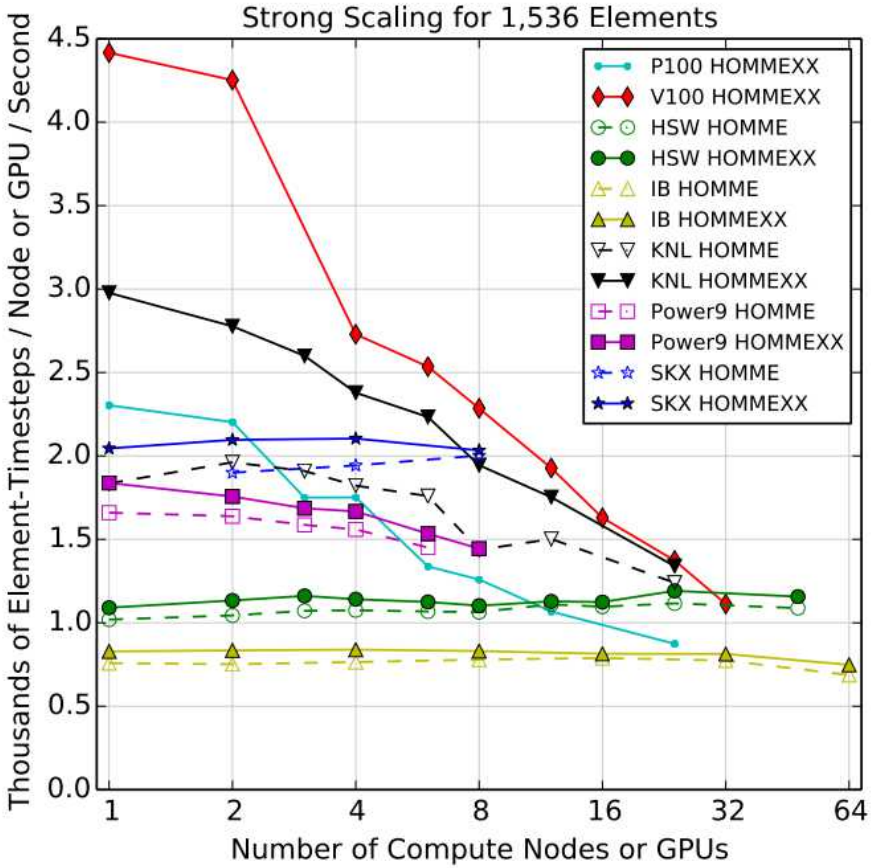
HOMMEXX Performance





HOMMEXX Performance





Conclusions

Sandia National Laboratories

- HOMMEXX is performant across multiple architectures
- C++ and Kokkos are a viable combination to achieve performance portability
- Modern C++ designs with template metaprogramming can be performance portable

Acknowledgements

The authors are grateful to Mark Taylor, Si Hammond, Kyungjoo Kim, Eric Phipps, Steven Plimpton, and the Kokkos team for their suggestions.
This work was part of the CMDV program, funded by the U.S. Department of Energy (DOE) Office of Biological and Environmental Research.
Sandia National Laboratories is a multimission laboratory managed and operated by the National Technology and Engineering Solutions of Sandia, L.L.C., a wholly owned subsidiary of Honeywell International, Inc., for the DOE's National Nuclear Security Administration under contract DE-NA-0003525.
This research used resources of the National Energy Research Scientific Computing Center, a User Facility supported by the Office of Science of DOE under Contract No. DE-AC02-05CH11231.

References


Technical specification for c++ extensions for parallelism.


Working draft, technical specification for c++ extensions for parallelism version 2.


The openacc application programming interface, 2017.


C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang.
Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics.
Scientific Computation. Springer Berlin Heidelberg, 2007.


H. Carter Edwards, C. R. Trott, and D. Sunderland.
Kokkos: Enabling manycore performance portability through polymorphic memory access patterns.
J. Parallel Distr. Com., 74(12):3202–3216, 2014.


J. Dennis, J. Edwards, K. J. Evans, O. Guba, P. H. Lauritzen, A. A. Mirin, A. St-Cyr, M. A. Taylor, and P. Worley.
CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model.
Journal of High Performance Computing Applications, 26(1):74–89, 2012.


M. Harris.
Developing portable cuda c/c++ code with hemi, 2015.


R. D. Hornung and J. A. Keasler.
The raja portability layer: Overview and status.
Technical report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2014.


J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, F. Lamarque, J, W. G. Large, D. Lawrence, K. Lindsay, W. H. Lipscomb, M. C. Long, N. Mahowald, D. R. Marsh, R. B. Neale, P. Rasch, S. Vavrus, M. Vertenstein, D. Bader, W. D. Collins, J. J. Hack, J. Kiehl, and S. Marshall.
The community earth system model: A framework for collaborative research.
Bulletin of the American Meteorological Society, 94(9):13391360, 2013.


M. Rajan, D. Doerfler, M. Tupek, and S. Hammond.
An Investigation of Compiler Vectorization on Current and Next-generation Intel Processors using Benchmarks and Sandias SIERRA Applications.
2015.


C. Trott and H. C. Edwards.
Kokkos: Enabling manycore performance portability for c++ applications and libraries.