*Exceptional service in the national interest*

# Performance-portability progress for an ultra-high resolution non-hydrostatic atmosphere model in E3SM

L.Bertagna, O.Guba, J.Foucar, A.Bradley, M.Taylor, A.Salinger

Sandia National Laboratories, Albuquerque, NM

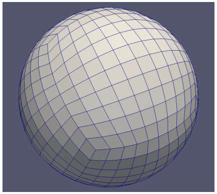
ESMD/E3SM All-Hands Meeting

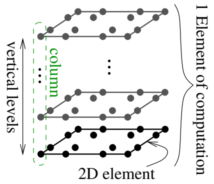
Oct 29, 2020

SAND 2020-10220 C

# High-Order Methods Modeling Environment

- Component of E3SM (and CESM) for dynamics and transport in the atmosphere.
- Accounts for 20-25% of total run time of typical fully-coupled simulation.
- Highly optimized for MPI and OpenMP parallelism.
- Horizontal (2D) and vertical (1D) differential operators are decoupled.
- Spectral Element Method (SEM) in the horizontal direction.
- Eulerian or Lagrangian schemes for vertical operators.





## Simple Cloud-Resolving E3SM Atmosphere Model

Cloud-Resolving Models (CRMs):

- Have horizontal resolution of $\lesssim$3km.
- Can do away with parametrizations of some physical process (deep convection), but needs more complex models for state equations.

SCREAM:

- Atm model for E3SM at ultra-high resolution.
- Targets 3km horizontal resolution, and 128 vertical levels (7.2 billion grid points).
- Uses Homme as dynamics dycore.

## The Kokkos library

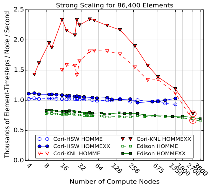
**What is Kokkos?**

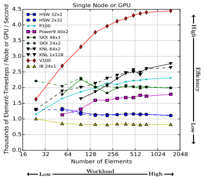
- Developed at Sandia National Labs, written in C++ (with C++11 required).
- Provides templated constructs for on-node parallel execution: execution space (host vs device), execution policy (range vs team), parallel operation (for, scan, reduce).
- Provides template abstraction for a multidimensional array: data type, memory space (host, device, UVM), layout (left, right, ...), memory access/handling (atomic, unmanaged, ...).
- Supports several back-ends: Serial, OpenMP, Cuda, HIP, ....
- Available at `http://github.com/kokkos/kokkos`.

## HOMMEXX: hydrostatic dycore (preqx)

Task completed in 2018, with paper accepted in Geoscientific Model Development[1].

- Incremental F90 $\rightarrow$ C++ conversion.
- Heavily tested.
- Bit-for-bit with F90 implementation.
- Minimization of architecture-specific code.
- Primary design goals:
  - expose parallelism,
  - maximize vectorization,
  - minimize memory movement.





L. Bertagna, M. Deakin, O. Guba, D. Sunderland, A.M. Bradley, I.K. Tezaur., M.A. Taylor, A.G. Salinger, HOMMEXX 1.0: a performance-portable atmospheric dynamical core for the Energy Exascale Earth System Model, Geo. Model Dev., 2019

## HOMMEXX: non-hydrostatic dycore (theta-l)

Task completed in 2020, with paper accepted at SC20[1].

- Same core design goals: expose parallelism and vectorization, minimize memory movement.
- Bit-for-bit with F90 implementation.
- Minimization of architecture-specific code.

With respect to preqx:

- Model differences: adds two state variables, uses potential temperature, can run in both hydrostatic and non-hydrostatic mode.
- Main additional challenges: nonlinear solver, larger memory footprint.

[1] L. Bertagna, O. Guba, M.A. Taylor, J.G.Foucar, J.Larkin, A.M. Bradley, S. Rajamanickam, A.G. Salinger, A Performance-Portable Nonhydrostatic Atmospheric Dycore for the Energy Exascale Earth System Model Running at Cloud-Resolving Resolutions, 2020

## Results: tested architectures

Single node architectures specs:

| | |
|---|---|
| (KNL) | Intel Xeon Phi: 68 cores/node, 4 threads/core, HBM+DDR4 |
| (P9) | IBM Power9: 2 sockets/node, 22 cores/socket, 4 threads/core, DDR4 |
| (V100) | NVidia Volta: 2 sockets/node, 3 GPUs/socket, 2560 DP cores/GPU |

Full cluster specs:

| | |
|---|---|
| (KNL) | Cori: 9688 compute nodes (max used: 9216), located at NERSC. |
| (P9) | Summit: 4608 compute nodes (max used: 4600), located at ORNL. |
| (V100) | Summit: 27648 GPUs (max used: 27600), located at ORNL. |

## Results: strong scaling

- Solve for state and 10 tracers (NGPPS benchmark).
- Run at 3km and 1km horizontal resolution.
- Achieves 0.97 SYPD on GPU on full Summit system.
- Excellent scaling at 3km, perfect scaling at 1km.
- CPU performance comparable (or slightly better) to original F90.
- Approximately 10x speedup when using GPUs vs P9 on Summit.
- 1km resolution features approximately 1 trillion DOFs.

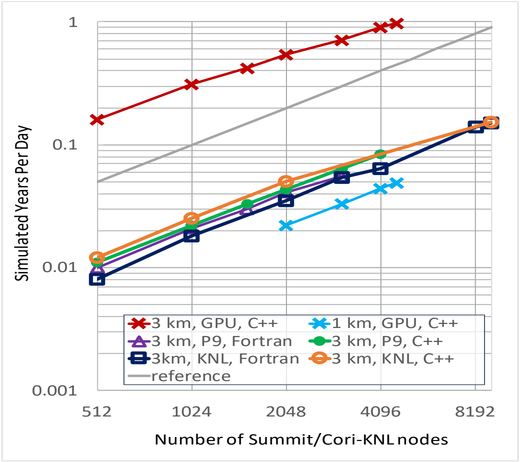


Figure: Achieved SYPD for different implementations and resolutions.

## Results: nonlinear solver

- Homme Timestepping: Horizontally Explicit, Vertically Implicit (HEVI).
- Vertical Implicit: Diagonally Implicit Runge Kutta (DIRK).
- Solves one nonlinear eqn per GLL point (coupling vertical levels).
- Newton step requires tridiag solve.
- Strictly diagonally dominant matrix.
- No pivoting allows packing equations.
- Uses Thomas on CPU, Cyclic Reduction on GPU.
- Independent residual check for each column.
- Only two kernels per solve (initial guess + Newton).

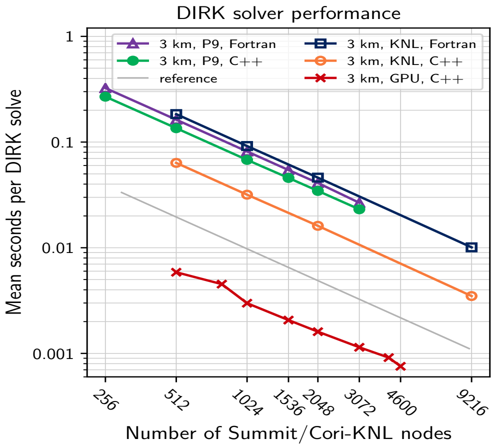


Figure: Performance of the DIRK solver on Cori KNL, Summit Power9, and Summit V100 GPU, for the 3 km benchmark.

# Results: workspace manager

- Allows temporary buffers management within single kernel.
- Simplifies code development in temporary-heavy kernels.
- Only active on GPU builds, at high workload/node.
- Limited overhead when WSM is active, negligible if inactive.
- Necessary to fit 1km problem in memory for less than 4096 nodes on Summit.

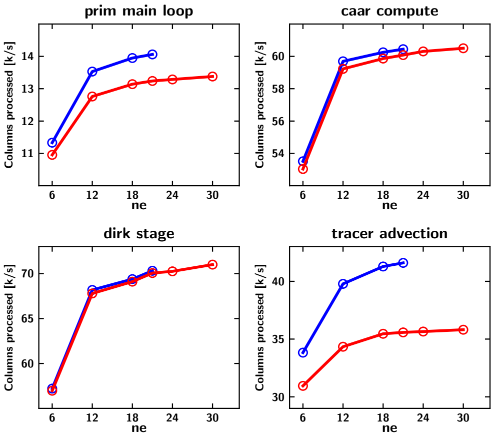


Figure: Processing rate of key functors as well as main time loop timer for the workspace manager (WSM) enabled (red) and disabled (blue).

## E3SM Kokkos Application Toolkit (EKAT)

Motivation:

- Several E3SM subprojects are gravitating toward Kokkos.
- Projects are independent of each other (in the git sense).
- Several common pattern needed across these projects.

EKAT contains utilities needed/useful across different E3SM-related projects:

- Kokkos-friendly structures for vectorization: packs and masks.
- Kokkos implementation of tridiagonal solver.
- Workspace manager for temporary-heavy kokkos kernels.
- Interface to YAML and Catch2 libraries (for inputs and testing).
- CMake utilities, such as handling/configuring Kokkos, or unit testing.

**Note**: EKAT is NOT a generic package. Development is driven by (and tailored for) E3SM needs.

## EKAT: Packs and Masks.

```
using PackType = ekat::Pack<double,8>;
int NP = PHYSICAL_DIM / PackType::n;
some_view_type<PackType*> v1("",NP),v2("",NP),v3("",NP);
{ /* fill views */ }
for(int i=0;i<NP;++i) {
  auto& p1 = v1(i); auto& p2 = v2(i); auto& p3=v3(i);
  p2 += pow(p1,5.0/3.0); // entry-wise
  auto m = p2>=1.0;
  p3.set(m,p2); // Only updates p3[i] if m[i] true
}
```

Packs/Masks serve two main purposes:

- enhance vectorization, b/c of guaranteed alignment, compile-time dimension, and compiler vectorization directives;
- hide implementation details for performance, leaving code easier for scientists.

## EKAT: bit-for-bit (BFB) team reductions.

```
constexpr bool BFB = MY_BFB_MACRO;
using ESU = ekat::ExecSpaceUtils<MyExecSpace>;
using PackType = ekat::Pack<double,8>;
int NP = DIM / PackType::n;
some_view_type<PackType*> v("",NP);
{ /* fill views, create team policy, launch || for */ }
  auto f = [&](int k){ ...};
  // v or f both work, as long as op()(int) works
  ESU::parallel_reduce<BFB>(team,1,DIM-1,v,sum1);
  ESU::parallel_reduce<BFB>(team,1,DIM-1,f,sum2);
}
```

- Template constructs allow to select performant or bit-for-bit implementation based on config options.
- Hide implementation details for bfb-vs-performance, leaving code easier for scientists.
- Crucial to get C++ vs F90 BFB agreement on CPU/GPU, even if PackSize>1.