Software development for performance in the Energy Exascale Earth System Model

Robert Jacob

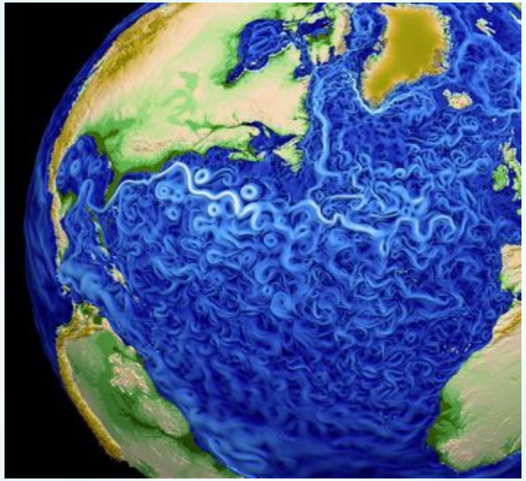
Argonne National Laboratory

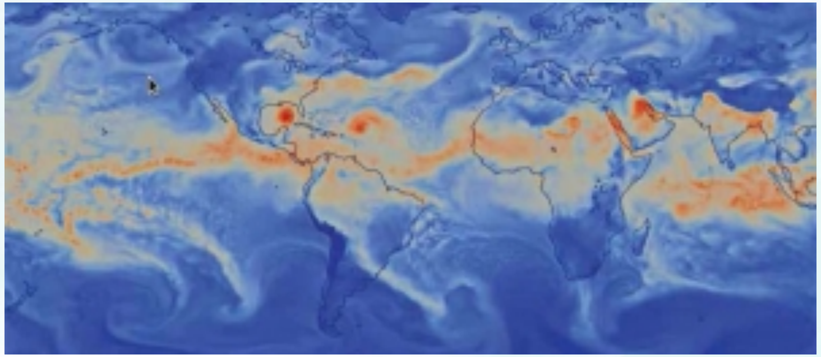
Lemont, IL, USA

(Conveying work from many people in the E3SM project.)

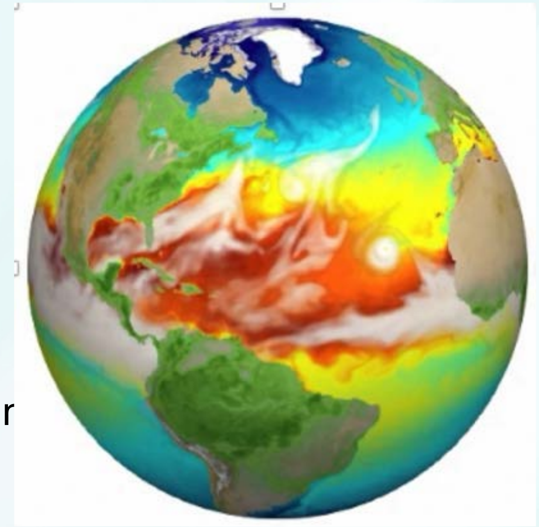
E3SM: Energy Exascale Earth System Model

- 8 U.S. DOE labs + universities. Total ~50 FTEs spread over 80 staff
- Atmosphere, Land, Ocean and Ice (land and sea) component models
- Development and applications driven by DOE-SC mission interests: Energy/water issues looking out 40 years
- **Computational goal: Ensure an Earth system model will run well on upcoming DOE pre-exascale and exascale computers**
- https://github.com/E3SM-Project/E3SM
  - Open source and open development
- http://e3sm.org





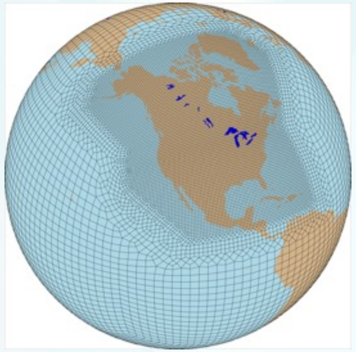
E3SM v1 (released April, 2018)



- New MPAS Ocean/Sea Ice/Land Ice components
- Atmosphere
  - Spectral Element (HOMME) dynamical core
  - Increased vertical resolution: 72L, 40 tracers.
  - MAM4 aerosols, MG2 microphysics, RRTMG radiation, CLUBBv1 boundary layer, Zhang-McFarlane deep convectior
- Land
  - New river routing (MOSART), soil hydrology (VSFM), dynamic rooting distribution, dynamic C:N:P stoichiometry option
- 2 Resolutions:
  - Standard: Ocean/sea-ice is 30 to 60km quasi-uniform. 60L, 100km atm/land
  - High: Ocean/sea-ice is 6 to 18 km, 80L, 25 km atm/land. Integrations ongoing
- Code: Fortran with MPI/OpenMP (including some nested OpenMP threading)

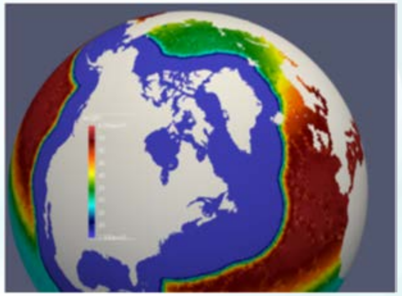
Golaz, J.-C., et al. (2019). "The DOE E3SM coupled model version 1: Overview and evaluation at standard resolution." *J. Adv. Model. Earth Syst.*, 11, 2089-2129. https://doi.org/10.1029/2018MS001603

E3SM v2 Development



E3SMv2 was supposed to incorporate minor changes compared to v1

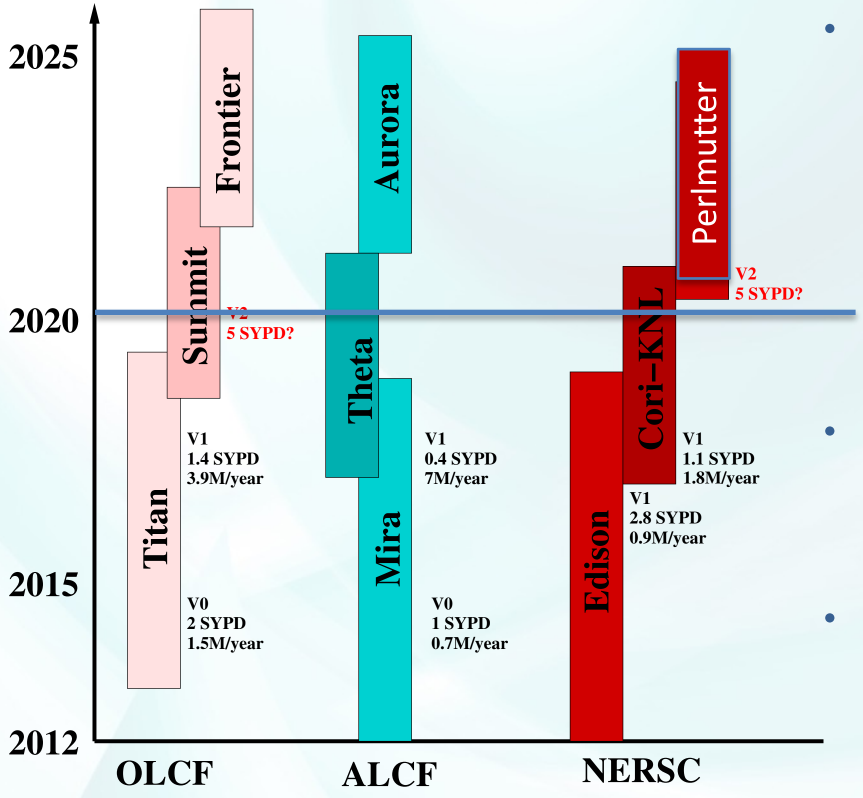
- Not quite… significant structural changes have taken place
- **Tri-grid configuration**
  - Land and river component now on separate common ½ deg grid
  - Increased land resolution, especially over Northern land areas
  - Enables closer coupling between land and river model: water management, irrigation, inundation, …
  - 3 grids: atmosphere, land/river, ocean/sea-ice.
- **Non-hydrostatic atm dynamical core** with **semi Lagrangian** (SL) tracer transport
- **Phys-grid** (reduced Finite-Volume grid for atmosphere physics) (2x speedup)
- **RRM** (regionally refined meshes) capability in Atmosphere, Ocean, and Sea ice.
- Improved MPAS-ocean numerics (3D GM, Redi mixing)
- More….



But computational approach will not change much

- Still Fortran with MPI/OpenMP
- Will do most of simulations on CPU systems including ones dedicated to the project.
  - 420 Skylake node system at PNNL, 520 AMD node system coming to Argonne this Fall.
- Scheduled for release Sept, 2021.

U.S. Exascale Architecture Roadmap

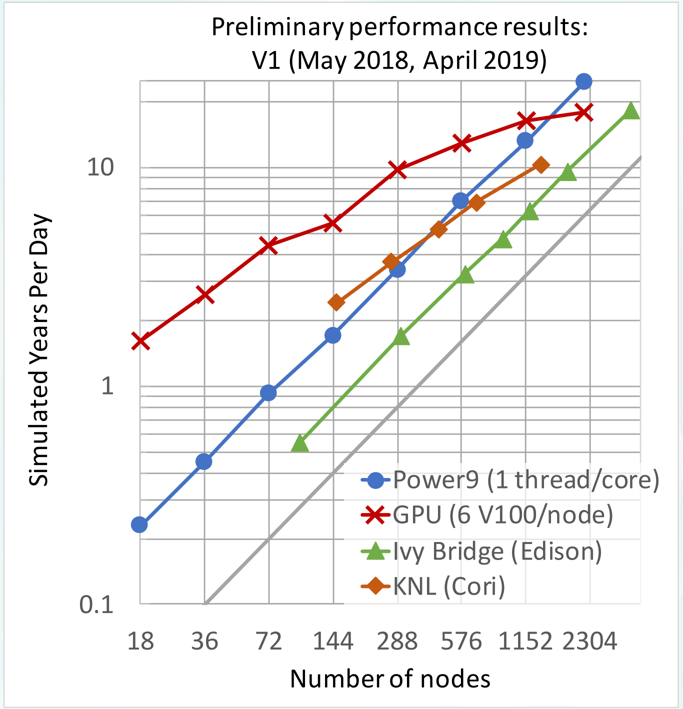


Exascale Strategy -> GPU Strategy

- NERSC:
  - Now: Cori-KNL, Intel KNL, 28 PF peak
  - Perlmutter Phase 1: late 2020
    - 1500 CPU-GPU nodes, 256 GB
    - 1 AMD Milan + 4 **NVIDIA** A100 **GPU**
  - Perlmutter Phase 2: mid 2021
    - 3000+ CPU-only nodes
    - 2 AMD Milan per node, 512 GB
    - 84-118PF total (phase 1 and 2)
- ALCF:
  - Now: Theta, Intel KNL, 11PF peak
  - Aurora21: 1.0 EF, late 2021
    - 2 Intel Xeon "Sapphire Rapids" + 6 **Intel Xe GPU**
- OLCF:
  - Now: Summit, 2 IBM Power9 + 6 NVIDIA v100s GPU 200PF
  - Frontier: 1.5EF, late 2021-early 2022
    - 1 AMD EPYC + 4 **AMD Radeon GPU**

E3SM Computational Performance Strategy

- Focus on simulation regimes where GPU systems can outperform similar size CPU systems
- GPU systems require large amount of work per node in order to outperform CPU systems (per watt)
  - We have a good understanding of GPU performance based on detailed benchmarks of parts of the code (especially the Atmosphere dycore HOMME)
- E3SM has excellent MPI performance allowing us to obtain throughput by running in the strong scaling limit (little work per node).
  - In this regime, GPUs are not competitive with CPUs.
- Will need to run on possibly 3 different GPUs.



Two science regimes that allow a E3SMv3 where GPUs outperform CPUs

- **Ultra-high resolution atmosphere**
  - 3km resolutions and higher: plenty of work per node to keep the GPU happy if entire atmosphere model runs on GPU.
  - At these resolutions, GPU systems could be up to 3x faster than CPUs (per Watt)
  - Large U.S. exascale allocation would allow O(100) simulated years per calendar year.
  - Can start to examine many large uncertainties in climate models but with SDPD
  - E3SM "SCREAM" project will take this on

- **Super-parameterized atmosphere**
  - GPUs will also allow us to increase complexity with proportionally less increase in cost
  - Use **super-parameterization** and put the 2D CRM on the GPU.
  - Could still get multiple SYPD while providing benefits of resolved convection/clouds (to the limit of the 2D CRM)
  - E3SM-MMF project (part of ECP) will take this on

How do we do all this on multiple vendor's GPUs?

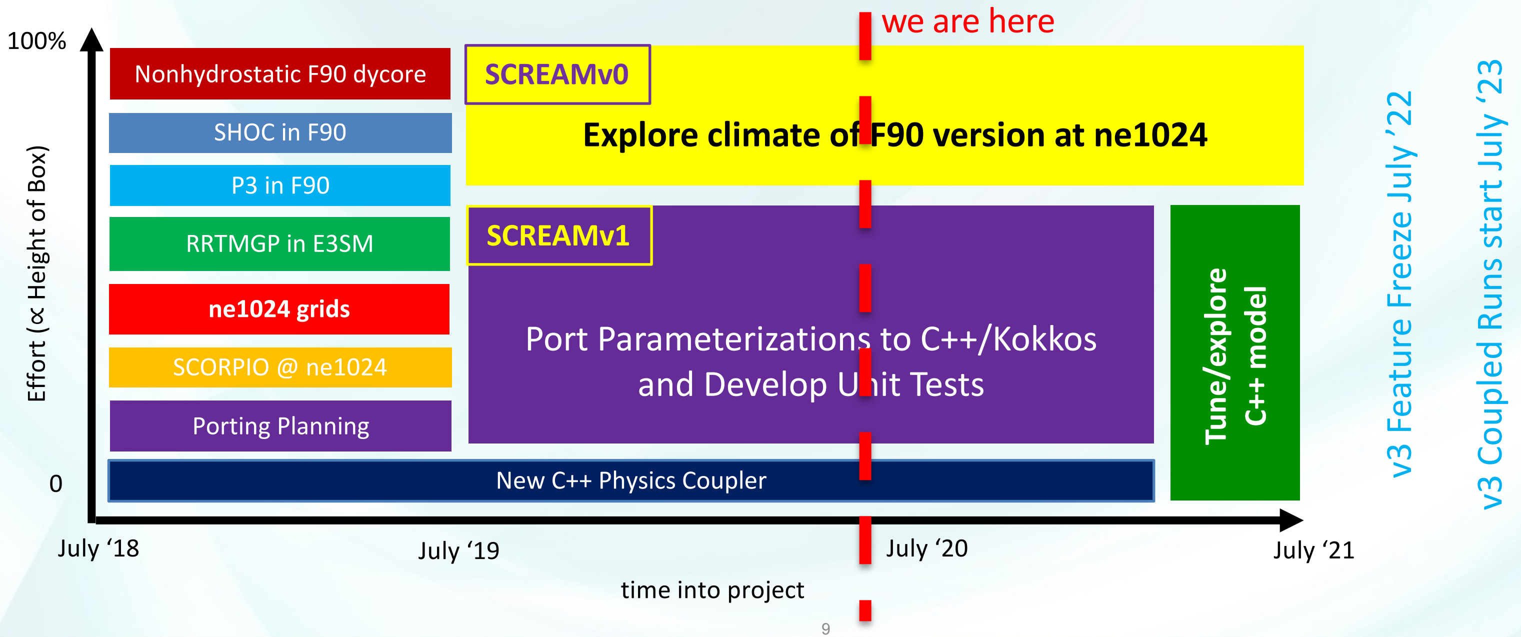
Performance Portability Strategy for E3SMv3

Programming models have (somewhat) reduced the barrier for GPU programming.   E3SM will support 2 going forward.

- OpenMP4.5/5.0 directives in Fortran
  - OpenMP 4.5 has GPU target offload and available in some Fortran compilers.
  - OpenMP 5.0 standard includes unified memory. Not fully implemented yet in any Fortran compiler.
  - OpenACC/CUDA not future-proof enough (but working great on Summit!)
- C++ and Kokkos
  - Template programming model that use C++11 language features. From Sandia NL.
  - Backends for CUDA, OpenMP, Pthreads, HIP, and SYCL/DPC++
  - Similar technology:
    - “Raja”/“Umpire” from Lawrence Livermore NL
    - “YAKL” Yet Another Kernel Launcher (https://github.com/mrnorman/YAKL)

Ultra-high res atm implementation strategy (SCREAM)

Get resolution and new physics (SHOC, P3) working in Fortran code base.
Port physics subroutine by subroutine to C++, building tests along the way.
Call C++ versions from Fortran as they are completed
Design/build C++ driver for full C++ atmosphere dynamics and physics.



Status: SCREAMv0 (Fortran, no GPU code)

- Goal: DYAMOND Phase 2 Intercomparison
  - Includes ~10 global storm-system resolving models (GSRMs)
  - 40 day run starting Jan 20, 2020
  - Results due Jan 1, 2021
- Ne1024pg2 (full physics) gets 5.2 simulated ***days*** per wallday on 3072 nodes (32%) of Cori-knl at NERSC
  - without performance optimization
  - ⇒ 40 day run costs 22.7M NERSC hrs
- Simulated world looks like earth
  - Need more in-depth validation

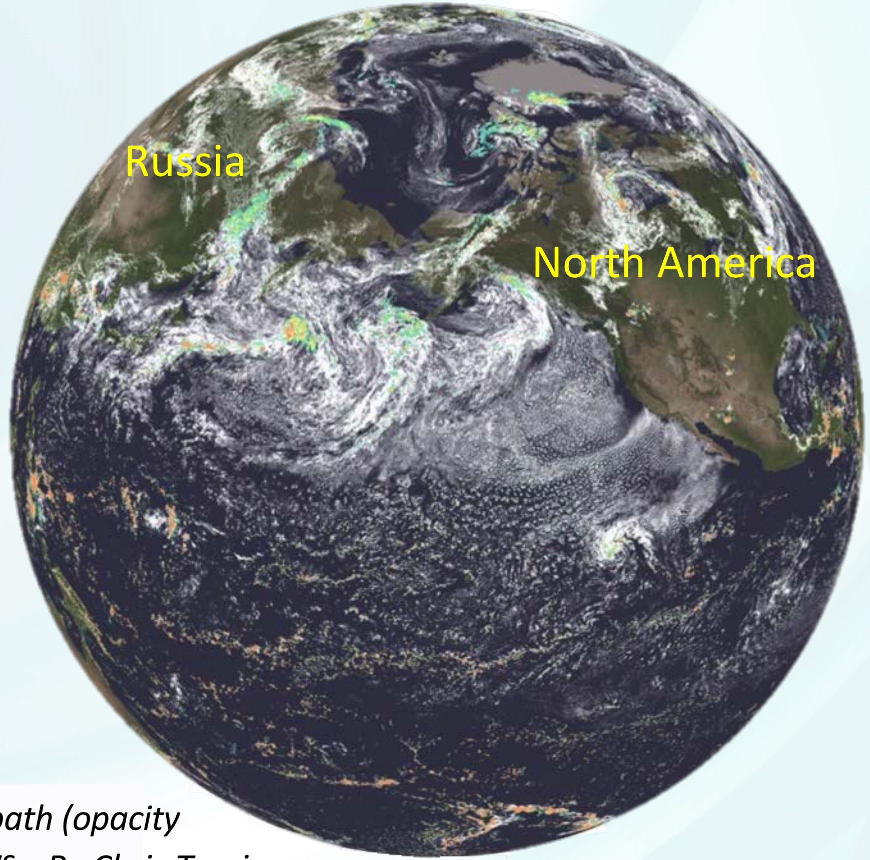
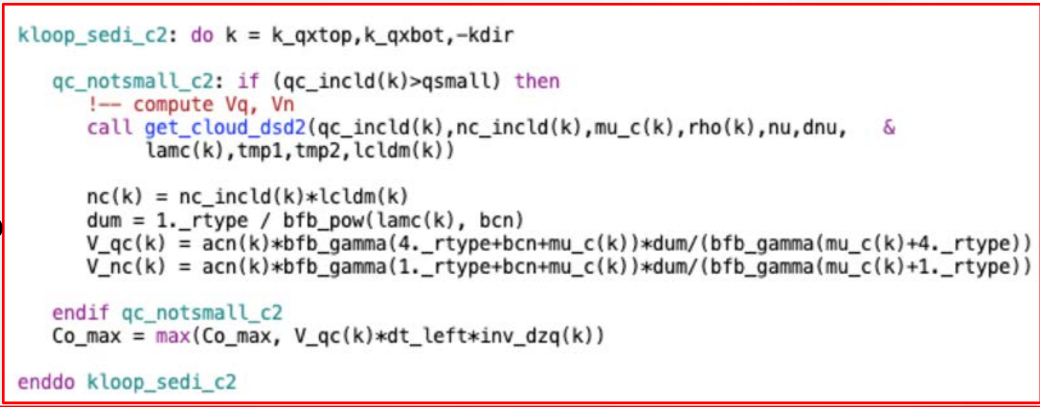


Fig: Snapshot of precipitation (color) and liquid water path (opacity with opaque white = 200 g m^{-2}) after 2.5 simulated days. *By Chris Terai*

SCREAM approach to converting Fortran to C++/Kokkos

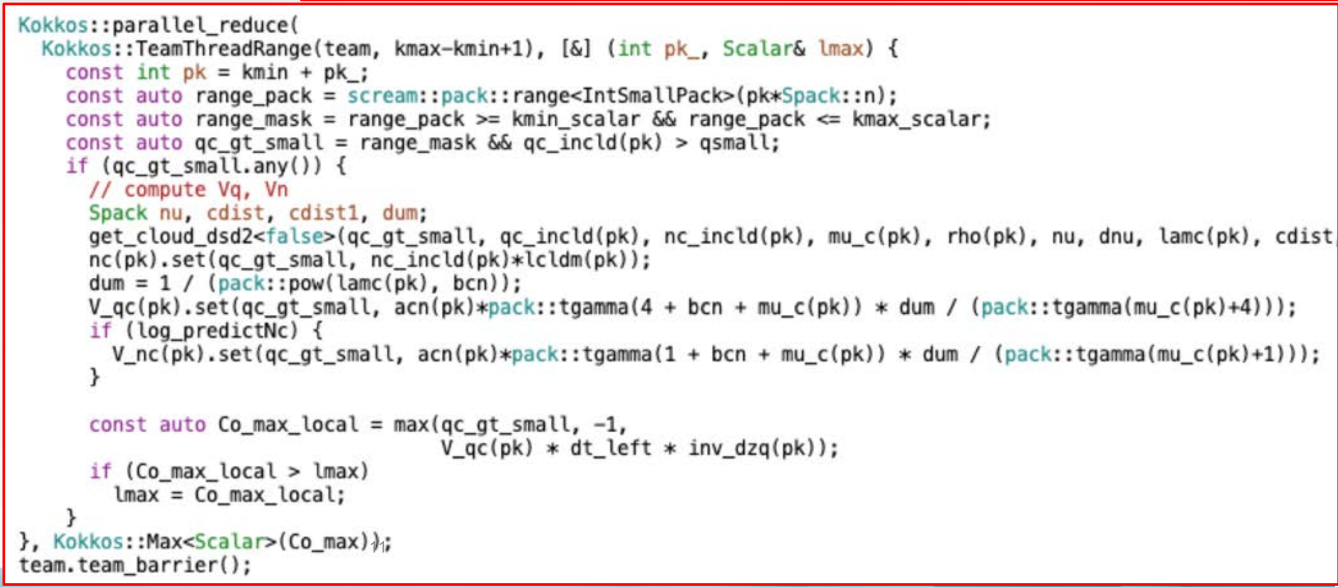
- Start with a single Fortran function or subroutine within a physics package (e.g. P3)
- Convert to C++ with Kokkos.
- Write a C interface too because Fortran can not directly call C++
- From the working Fortran, call the C++ version of the single subroutine via the C interface.
- For testing, call both the original Fortran (on the CPU single threaded) and the C++ (**on the GPU** threaded) and compare answers. **Can be bit-for-bit** (with some effort)
- Repeat
- While doing this, write the physics package driver in C++ and call all the pieces you've ported.
- Also write the C++ atmosphere main driver.

Original F90



```
kloop_sedi_c2: do k = k_qxtop,k_qxbot,-kdir

   qc_notsmall_c2: if (qc_incld(k)>qsmall) then
      !-- compute Vq, Vn
      call get_cloud_dsd2(qc_incld(k),nc_incld(k),mu_c(k),rho(k),nu,dnu,   &
           lamc(k),tmp1,tmp2,lcldm(k))

      nc(k) = nc_incld(k)*lcldm(k)
      dum = 1._rtype / bfb_pow(lamc(k), bcn)
      V_qc(k) = acn(k)*bfb_gamma(4._rtype+bcn+mu_c(k))*dum/(bfb_gamma(mu_c(k)+4._rtype))
      V_nc(k) = acn(k)*bfb_gamma(1._rtype+bcn+mu_c(k))*dum/(bfb_gamma(mu_c(k)+1._rtype))

   endif qc_notsmall_c2
   Co_max = max(Co_max, V_qc(k)*dt_left*inv_dzq(k))

enddo kloop_sedi_c2
```

Ported to C++/Kokkos



```
Kokkos::parallel_reduce(
  Kokkos::TeamThreadRange(team, kmax-kmin+1), [&] (int pk_, Scalar& lmax) {
    const int pk = kmin + pk_;
    const auto range_pack = scream::pack::range<IntSmallPack>(pk*Spack::n);
    const auto range_mask = range_pack >= kmin_scalar && range_pack <= kmax_scalar;
    const auto qc_gt_small = range_mask && qc_incld(pk) > qsmall;
    if (qc_gt_small.any()) {
      // compute Vq, Vn
      Spack nu, cdist, cdist1, dum;
      get_cloud_dsd2<false>(qc_gt_small, qc_incld(pk), nc_incld(pk), mu_c(pk), rho(pk), nu, dnu, lamc(pk), cdist
      nc(pk).set(qc_gt_small, nc_incld(pk)*lcldm(pk));
      dum = 1 / (pack::pow(lamc(pk), bcn));
      V_qc(pk).set(qc_gt_small, acn(pk)*pack::tgamma(4 + bcn + mu_c(pk)) * dum / (pack::tgamma(mu_c(pk)+4)));
      if (log_predictNc) {
        V_nc(pk).set(qc_gt_small, acn(pk)*pack::tgamma(1 + bcn + mu_c(pk)) * dum / (pack::tgamma(mu_c(pk)+1)));
      }

      const auto Co_max_local = max(qc_gt_small, -1,
                                    V_qc(pk) * dt_left * inv_dzq(pk));
      if (Co_max_local > lmax)
        lmax = Co_max_local;
    }
}, Kokkos::Max<Scalar>(Co_max));
team.team_barrier();
```

SCREAMv1 so far

- C++ version of nonhydrostatic (NH) dycore **done and working**
  - Used for recent Gordon-Bell submission (figure on right)
  - Gets 0.97 SYPD using **all** of Summit (4068 nodes) (354 SDPD)
  - Does not include semi-Lagrangian advection ⇒ further speed up!
- RRTMGP C++ done (with YAKL, not Kokkos) and starting to interface.
- P3 C++ port almost done
- SHOC C++ port starting now.

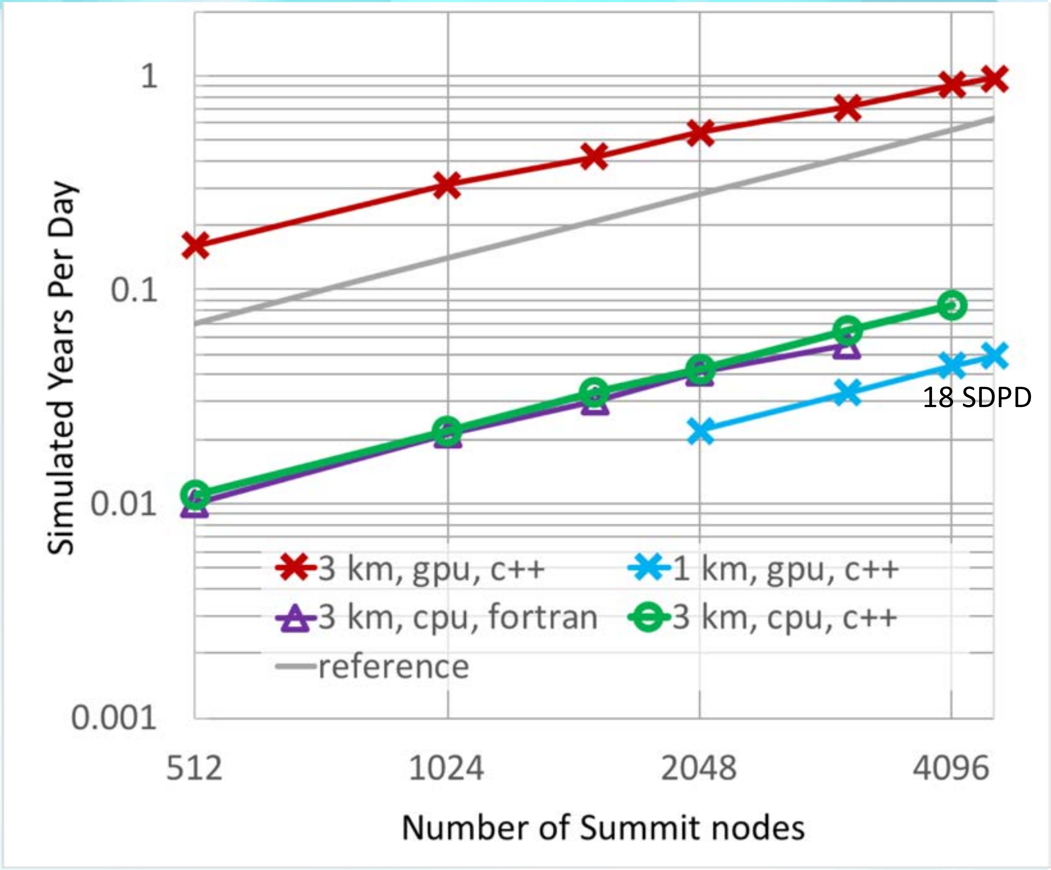
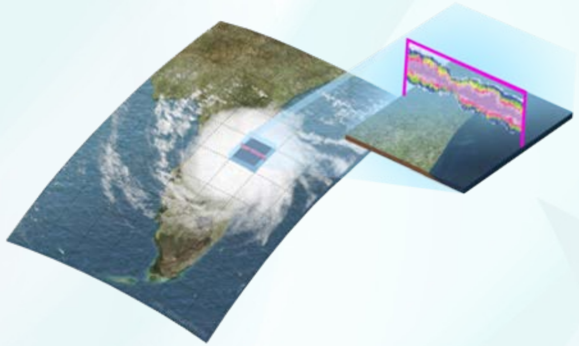


Fig: Nonhydrostatic C++ dycore-only NGGPS timings at ne3072 (blue) and ne1024 (other colors).

Super-parameterized E3SM (SP-E3SM)

E3SM with a super-parameterized atmosphere.
An embedded 2D cloud resolving model (CRM) is used to represent sub-grid processes (Convection, Microphysics, Turbulent mixing)



To accelerate SP-E3SM use both **hardware** and **algorithmic** speedup:

- Refactoring the code to enhance parallelism for GPU (also helps the CPU).
- CRM mean-state acceleration reduces the number of CRM time steps needed per GCM time step
- The amount of radiative calculations are reduced by using 4 groups of 16 CRM columns rather than calculating radiation in all columns individually

EXASCALE COMPUTING PROJECT

SP-Atmosphere GPU Implementation Strategy

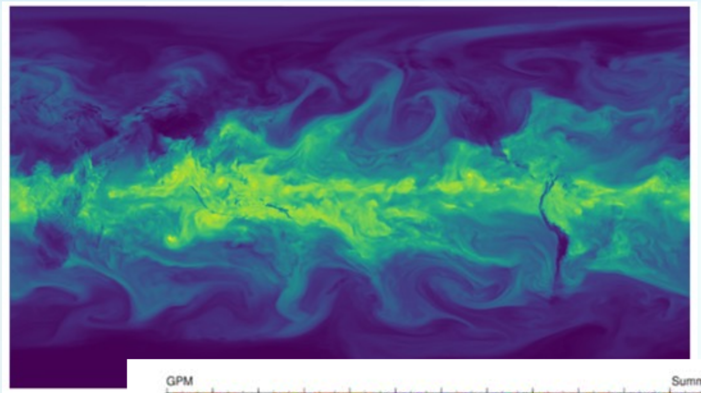
1. Clean up the original CRMFortran code
   - Put all routines in modules, fix bugs through valgrind and bounds checking
2. Expose more parallelism
   - Extra dimension added to all variables to allow multiple CRM instances to run in parallel
   - Move if-statements to expose all loops in a tightly nested manner
   - Push all looping down the call stack to avoid function calls on the GPU
3. Decorate with GPU parallel loop directives (OpenACC)
   - Required fixing / working around a number of PGI compiler bugs
4. Handle race conditions
   - Identify race conditions and handle through reduction clauses and atomics
5. Optimize data movement with CUDA managed memory and prefetching

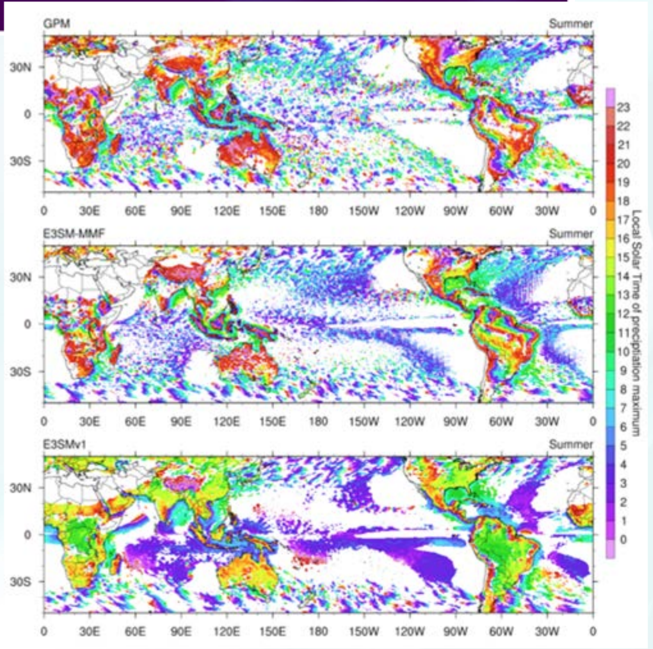
From Matt Norman

SP-E3SM progress so far

- Completed full port of the CRM with OpenACC
  - 20,000 lines of code, using Fortran/openACC
  - 98% of the Atmosphere cost (reduced radiation configuration)
  - Was working on Titan, now Summit

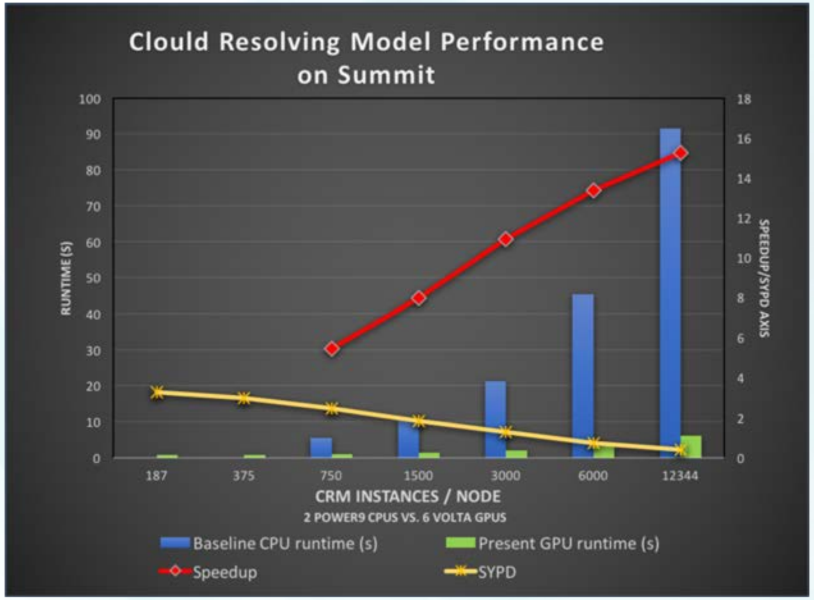
- Early Science Award on Summit
  - Used 350K NODE hours (1024 nodes)
  - High-res ne120 (0.25 degree) exterior (GCM) resolution – highest resolution explored with superparameterization.
  - 1km 2D CRM. 256x1x58
  - Completed 7 year simulation (results on right). 0.5 SYPD





The diurnal cycle of precipitation is improved in SP-E3SM similar to what Mike Pritchard has shown in SP-CAM

SP-E3SM Summit Performance; next steps



- Given enough work, GPU speed-up is > 15x over the Power9 CPUs on Summit
- However, with typical workloads, speed-up is lower due to lack of parallelism per node from strong scaling.
- We found 100% weak scaling efficiency if using a "3D" CRM (64x64x58) and **fix** 169 CRMs on each node (simulation speed is flat with node count)

next steps

- Port OpenACC version to OpenMP
  - Can leverage OpenACC-2-OpenMP Source-to-Source translator: https://github.com/naromero77/ACC2OMP
  - Keep both OpenACC and OpenMP in same source
- Explore more C++/YAKL options
  - Recently completed a C++/YAKL port of CRM

GPU porting for MPAS components (ocean, sea-ice)

- Keep the Fortran and add directives
  - Extended OpenACC (for Summit) coverage across model
  - ~2x overall performance improvement
- Change MPAS framework/data model
  - Currently uses memory pool approach with linked lists pointing to sub-domain structures
  - Converting to contiguous memory blocks and eliminating sub-domain blocking
  - Keeping static mesh arrays on device
- Making use of asynchronous execution within MPAS-ocean
  - Overlapping data transfers with computation
- Sea-ice is an extreme case
  - Sea-ice could not provide enough work to make acceleration feasible
  - Use asynchronous execution within E3SM to run sea-ice on unused CPU cores while ocean and other components run on GPU device

SCREAM developers

Peter Caldwell (PI), Andy Salinger, Luca Bertagna, Hassan Beydoun, Peter Bogenschutz, Andrew Bradley, Aaron Donahue, Chris Eldred, Jim Foucar, Chris Golaz, Oksana Guba, Ben Hillman, Rob Jacob, Jeff Johnson, Noel Keen, Jayesh Krishna, Wuyin Lin, Weiran Liu, Kyle Pressel, Balwinder Singh, Andrew Steyer, Mark Taylor, Chris Terai, Paul Ullrich, Danqing Wu, Xingqiu Yuan

SP-E3SM developers

Mark Taylor (PI, SNL) **ANL:** Jayesh Krishna, Danqing Wu, Nichols Romero, Xingqiu Yuan, **CSU:** David Randall, Don Dazlich, Mark Branson, **LANL:** Philip Jones, Rob Aulwes, Henry Moncada, Matt Turner, **LLNL:** David Bader, Walter Hannah, Jungmin Lee, **ORNL:** Matthew Norman, Sarat Sreepathi, Marcia Branstetter, **PNNL:** Ruby Leung, Mikhail Ovchinnikov, Chris Jones, Guangxing Lin, **SNL:** Andrew Bradley, Chris Eldred, Ben Hillman, **UCI:** Mike Pritchard, Hossein Parishani

Acknowledgments

This research was supported by the E3SM Project (DOE-SC BER) and the Exascale Computing Project (DOE-SC ASCR and DOE NNSA,17-SC-20-SC).

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725 **AND** *resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231* **AND** *resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357*