Pacific Northwest
NATIONAL LABORATORY

Running climate simulations on Cori

Throughput per job or throughput per year?

January 27, 2022

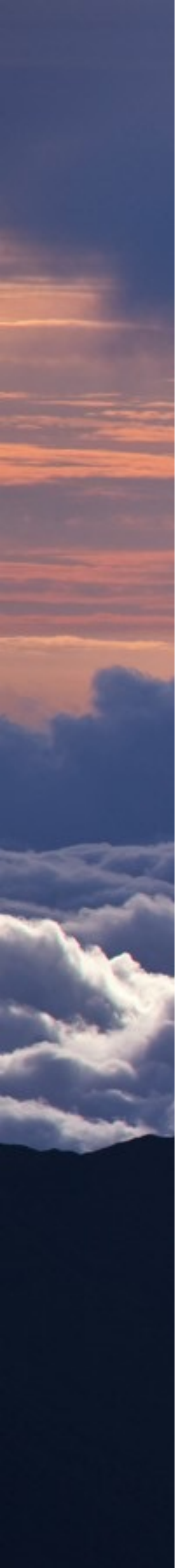
Koichi Sakaguchi

Atmospheric Sciences and Global Change Division

PNNL

U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy



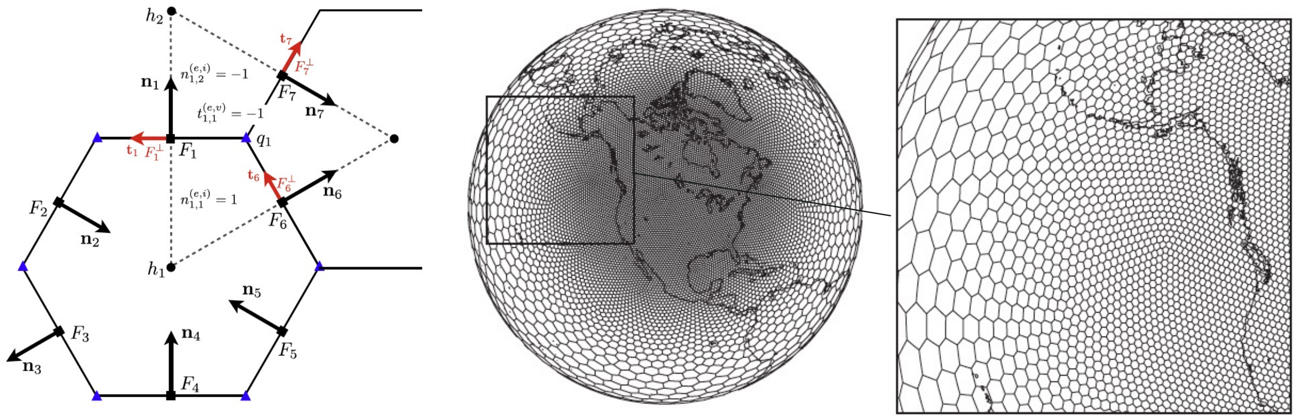
My research topics

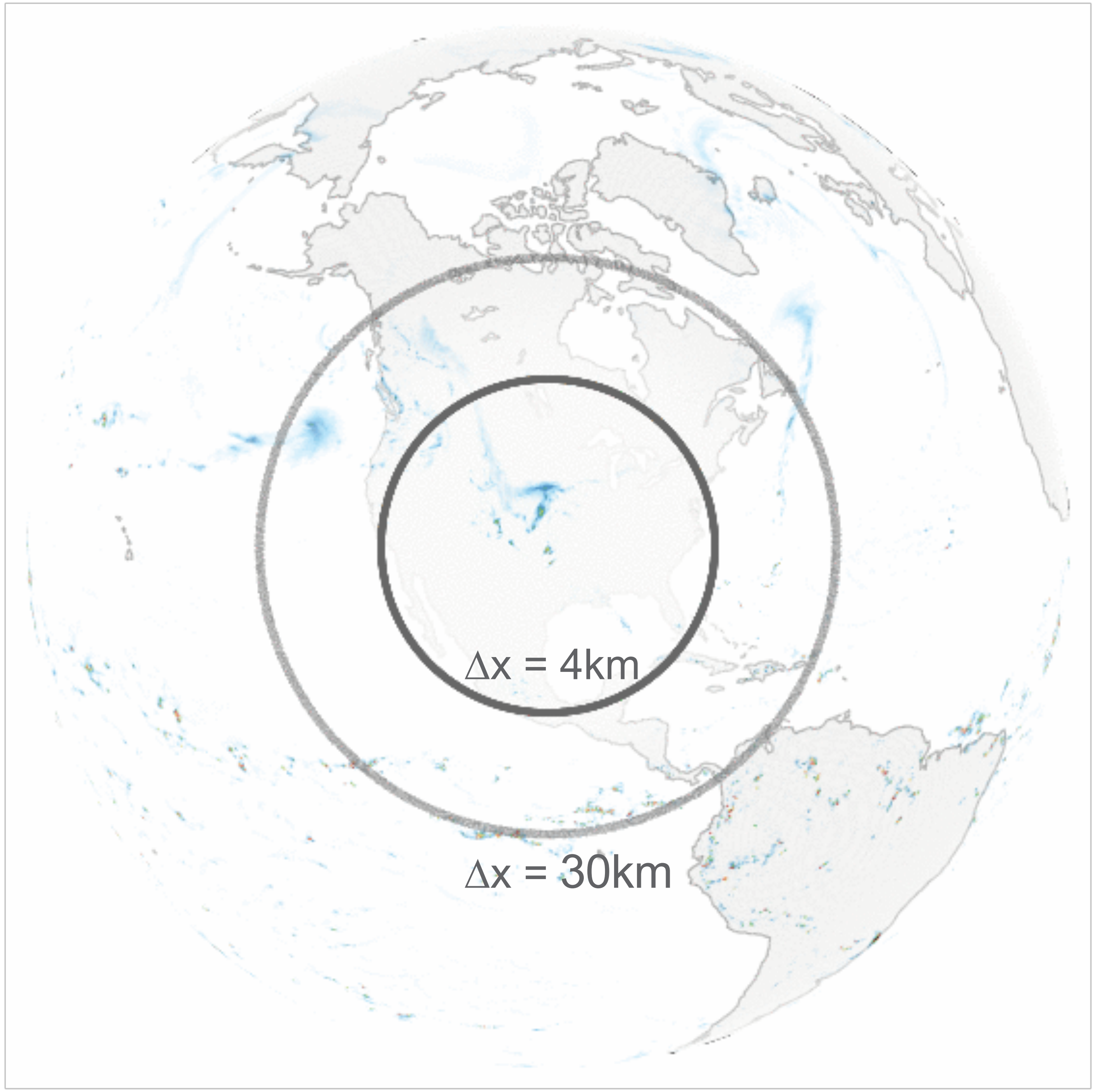
Atmospheric turbulence in the planetary boundary layer and moist convection

Interactions between land surface and atmosphere/clouds

Interactions between small- (~km) and large-scale(~10^3 km) atmospheric phenomena

A hierarchy of models/resolutions — from Large Eddy Simulations (Δx~100m) to General Circulation Models (Δx~100km)





Ringler et al., 2010; Ju et al., 2011; Skamarock et al., 2012

How we see the Earth system

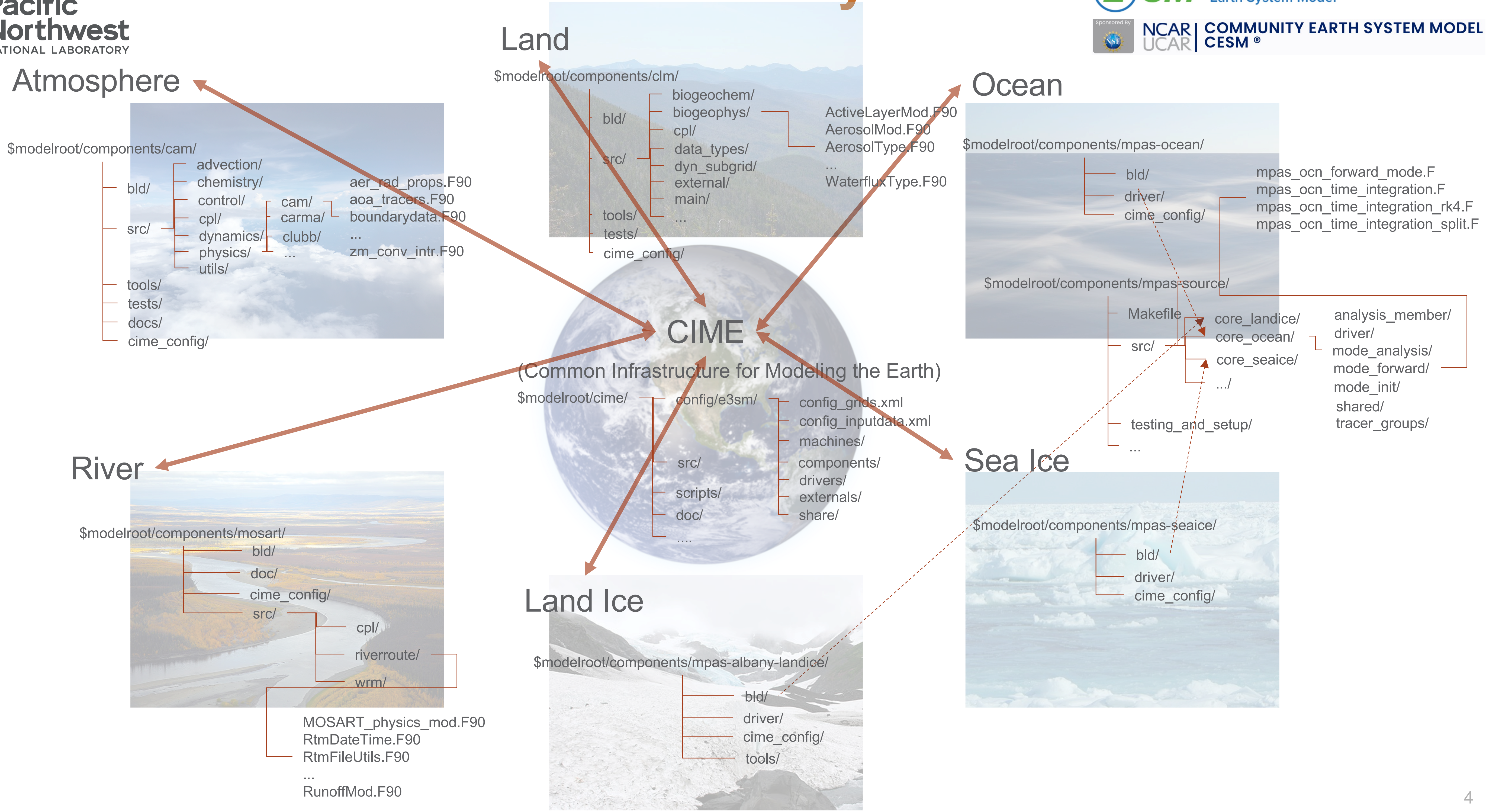
Pacific Northwest
NATIONAL LABORATORY

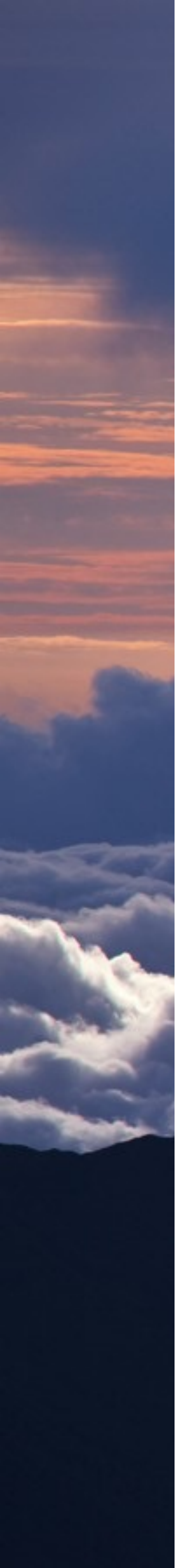
How we model the Earth system

Example from :

E3SM Energy Exascale Earth System Model

NCAR UCAR | COMMUNITY EARTH SYSTEM MODEL CESM ®





How we model (a part of) the atmosphere

1. Simplify the equations governing atmospheric state and motion

$$\frac{\partial \bar{\rho}\mathbf{u}}{\partial t} = -\nabla \cdot (\bar{\rho}\mathbf{u}\mathbf{u}) - \nabla p + \cdots + S_d + S_p$$

S_d :unresolved dynamics

S_p :radiative, chemical, thermodynamics

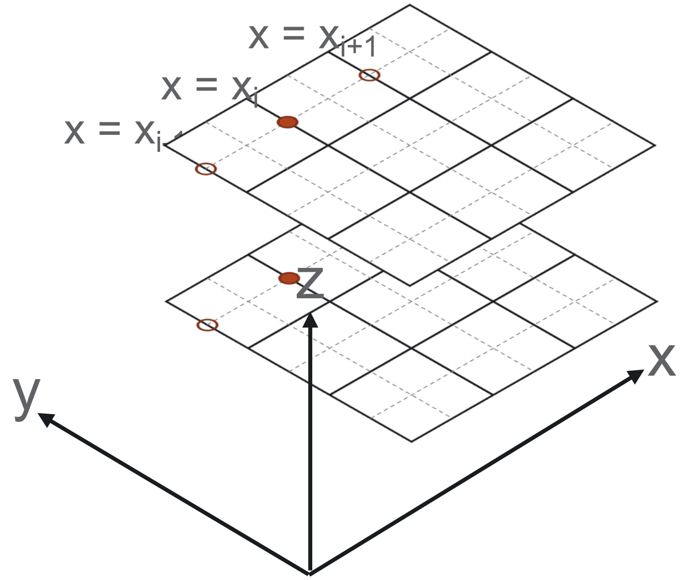
x-component:

$$\frac{\partial u}{\partial t} = -\frac{\partial uu}{\partial x} - \frac{1}{\bar{\rho}}\frac{\partial \bar{\rho}uw}{\partial z} - \frac{1}{\bar{\rho}}\frac{\partial p}{\partial x} + \cdots$$

2. discretize in time and space to solve numerically

$$\frac{\partial u}{\partial t} \approx \frac{\Delta u}{\Delta t} = \frac{u^{t^{n+1}} - u^{t^n}}{t^{n+1} - t^n}$$

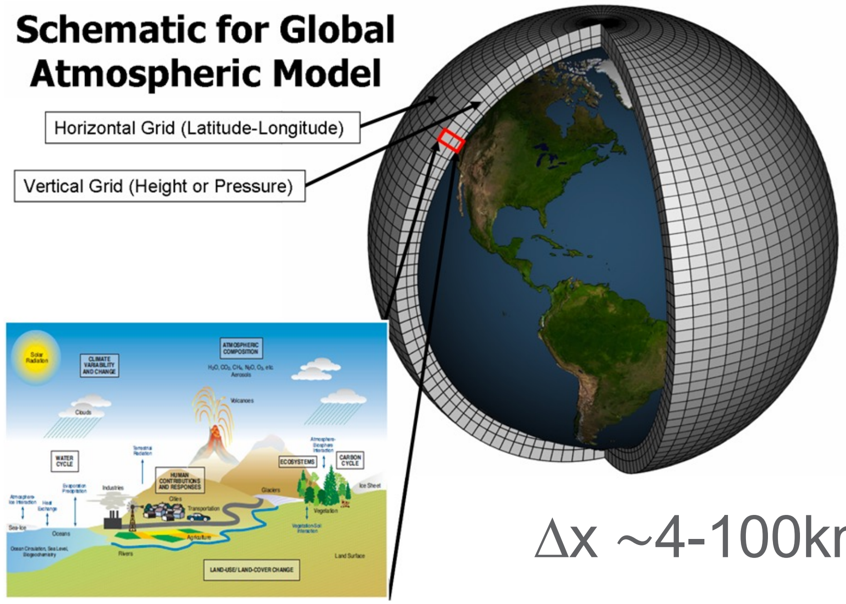
$$\frac{\partial u}{\partial x} \approx \frac{\Delta u}{\Delta x} = \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}}$$



3. numerically integrate in time at each grid cell

$$\frac{u^{n+1}_{i,j,k} - u^n_{i,j,k}}{\Delta t} = -\frac{1}{\Delta x}\left[\left[\frac{u^n_{i+1,j,k} + u^n_{i,j,k}}{2}\right]^2 - \left[\frac{u^n_{i,j,k} + u^n_{i-1,j,k}}{2}\right]^2\right] - \ldots$$

$$u^{n+1}_{i,j,k} = \left[u^n_{i,j,k} - \frac{1}{\Delta x}\left[\left[\frac{u^n_{i+1,j,k} + u^n_{i,j,k}}{2}\right]^2 - \left[\frac{u^n_{i,j,k} + u^n_{i-1,j,k}}{2}\right]^2\right] - \ldots\right]\Delta t$$



Δx ~4-100km

future value = sum of spatially dependent processes operating **now**

Pacific Northwest NATIONAL LABORATORY

How we model the Earth system

Example from :
E3SM Energy Exascale Earth System Model
NCAR | UCAR COMMUNITY EARTH SYSTEM MODEL CESM ®

Atmosphere

Fluid dynamics (AKA dynamical core)
Different schemes available

```
$modelroot/components/cam/
    bld/
    src/
        advection/
        chemistry/
        control/
        cpl/
        dynamics/
            eul/
            fv/
            se/
                trunc.F90
                stepon.F90
                spmd_dyn.F90
                restart_dynamics.F90
                ...
                external/
            mpas/
                mpas_atm_core.f90
                mpas_atm_core_interface.f90
                mpas_atm_time_integration.f90
                ...
                xml_stream_parser.c
        physics/
            cam/
                aer_rad_props.F90
                aoa_tracers.F90
                boundarydata.F90
                ...
                micro_mg2_0.F90
                micro_mg_cam.F90
                ...
                zm_conv_intr.F90
                (122 files)
            carma/
            clubb/
            ...
        utils/
    tools/
    tests/
    docs/
    cime_config/
```

- Unresolved fluid dynamics
- Radiative transfer
- Molecular scale processes (e.g., cloud particle growth)
- Chemistry

Different schemes often available for each process
Each scheme developed by different groups from labs and universities, including students

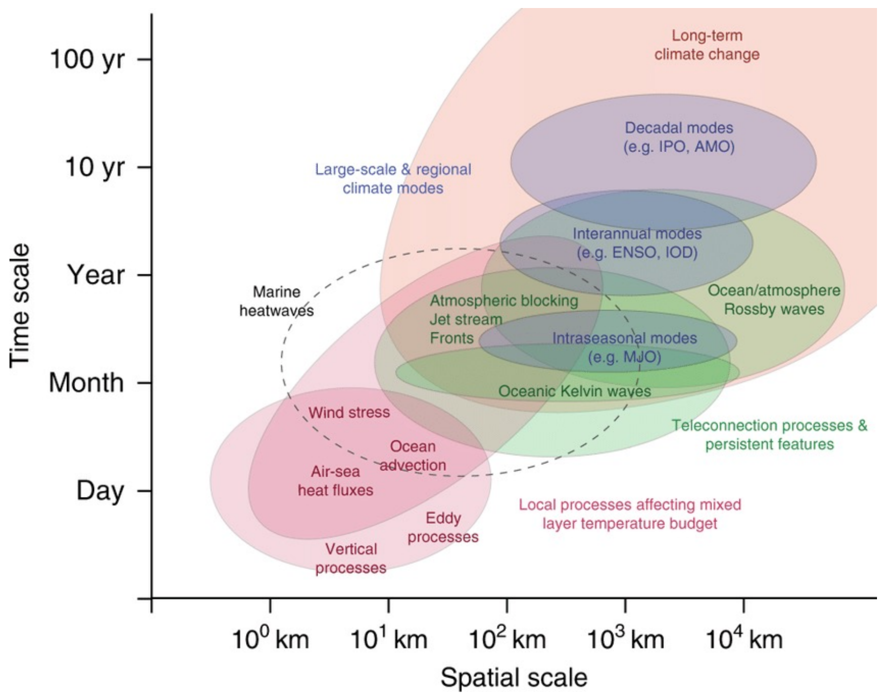
Test cases in the previous NERSC Exascale Science Application Program (NESAP) for optimization for KNL

(Barns et al., 2016)

Large community model code with many options ➡ Not easy to optimize the code for a given (new) architecture

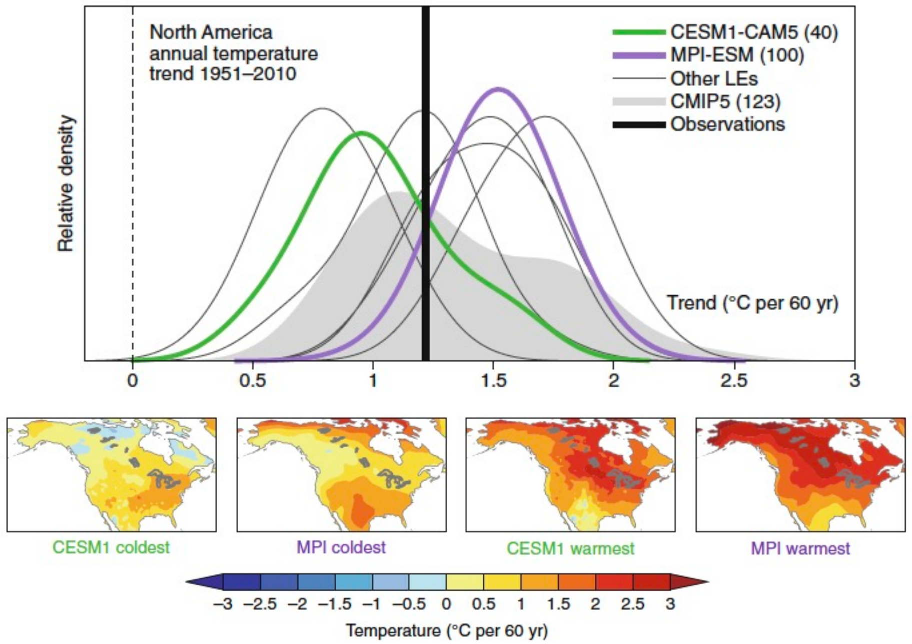
(exceptions emerging: e.g., DOE E3SM/SCREAM model)

Timescale & chaos of global atmosphere/climate system

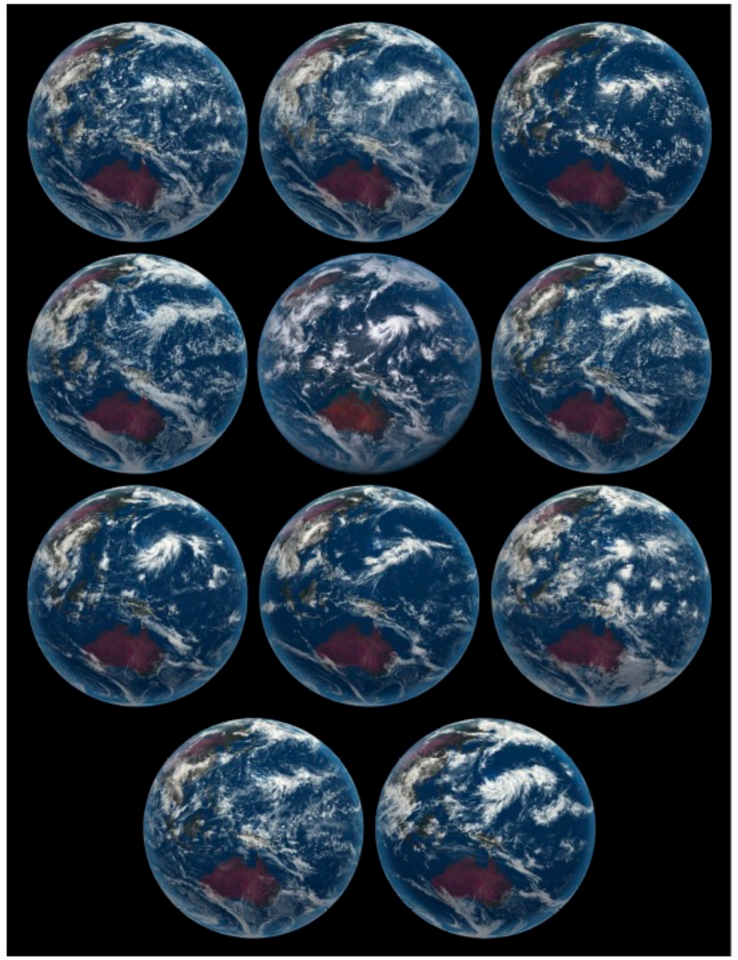


Holbrook et al., 2019

Δx~ 100km
T ~ 100 years
ensemble # = 40-100

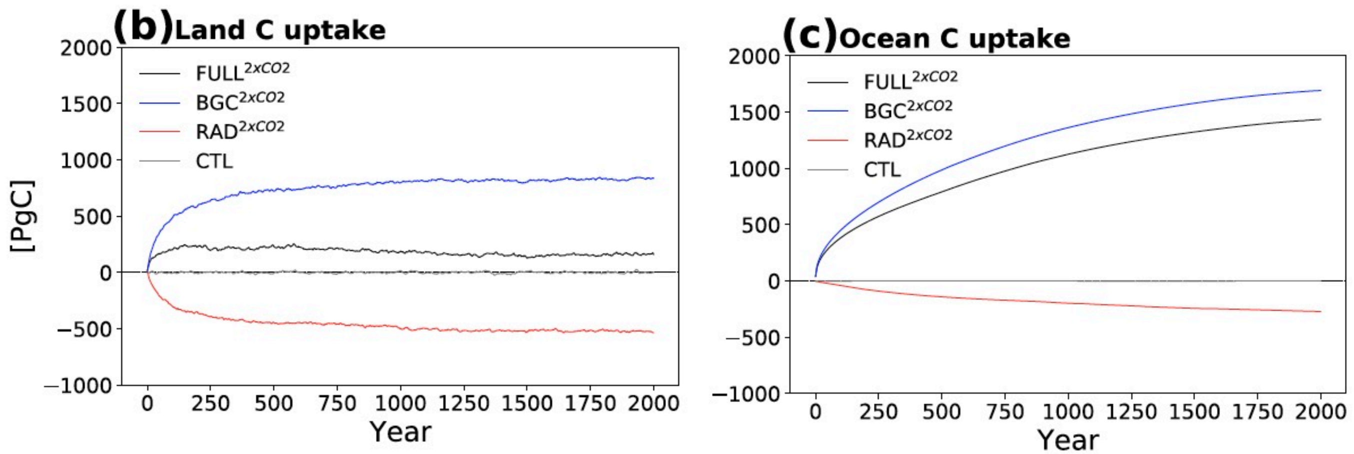


Δx~ 4km
T ~ 3 months
ensemble # = 1



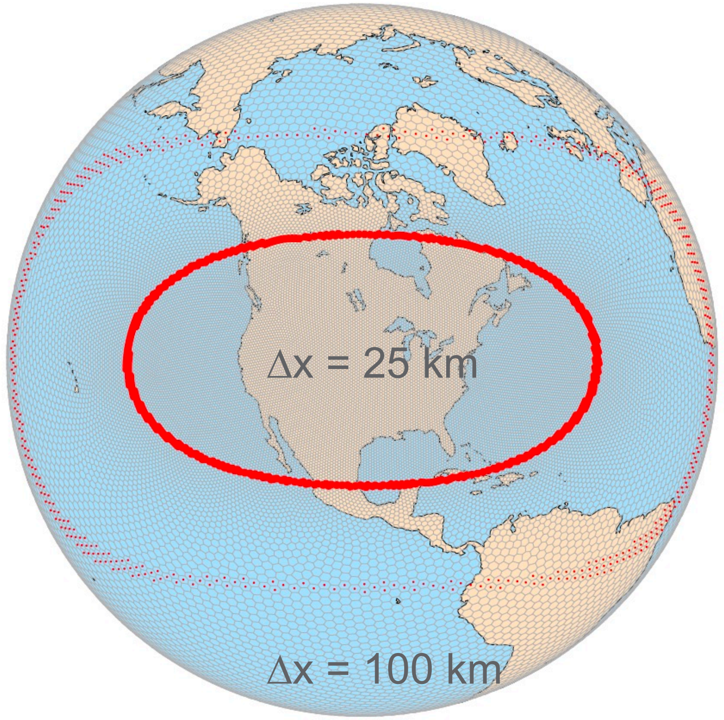
Stevens et al. 2019

Δx~ 200km
T ~ 2000 years
ensemble # = 1-5



My experience on NERSC to run climate simulations: How long it takes?

Example: **experimental** climate model code (CESM2 beta05)



No support for openMP; use only MPI
# of horizontal grid columns = 137,218
# of vertical levels = 32
# of grid boxes = 4,390,976 (**moderate** grid resolution)

Simulation periods: 1989-2010 and 2079-2100 -> 44 years

1 simulation month / ~3 hr realtime using 40 nodes (2560 MPI ranks) on KNL
(1 simulation month / 1.5 hr realtime using 40 nodes (960 MPI ranks) on Edison)

Typically submit a job for ~ 8 hours on KNL for two months simulation

Each job depends on the previous job, which wrote a “restart” file at the end

-> 44 years = 528 months -> 264 jobs

If no queue waiting -> 264 jobs * 6 core hours = 1584 hrs = **66 days**

What is the expected queue time?

NERSC Best Practices are our best friends

I adopted the following best practices (https://docs.nersc.gov/jobs/best-practices/)

- **Set an appropriate Lustre file striping** https://docs.nersc.gov/performance/io/lustre/

  For a high-resolution simulation (230,000,000 grid points), reduced the time spent on writing a restart-file (230GB) from ~ 2 hours (default striping) to ~ 15 minutes

- For large jobs, use --bcast option (copy model executable to the compute-node local path before starting srun

- For small jobs, use the same switch (--switches option)

- Prepare environment when submitting a job (CESM's job management code does this automatically)

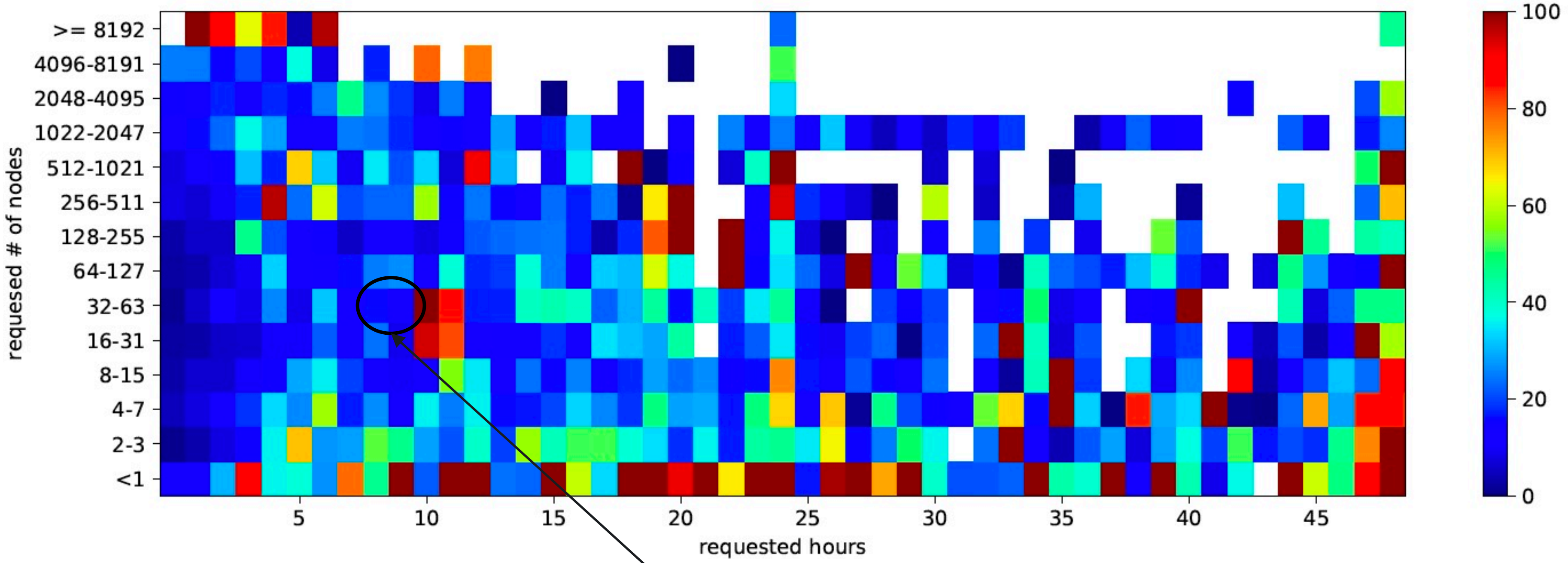
- Core specialization (#SBATCH -S 4; leave 4 cores on a KNL node for system overhead)

- Most of the input/output files on the scratch (fastest I/O with compute nodes)

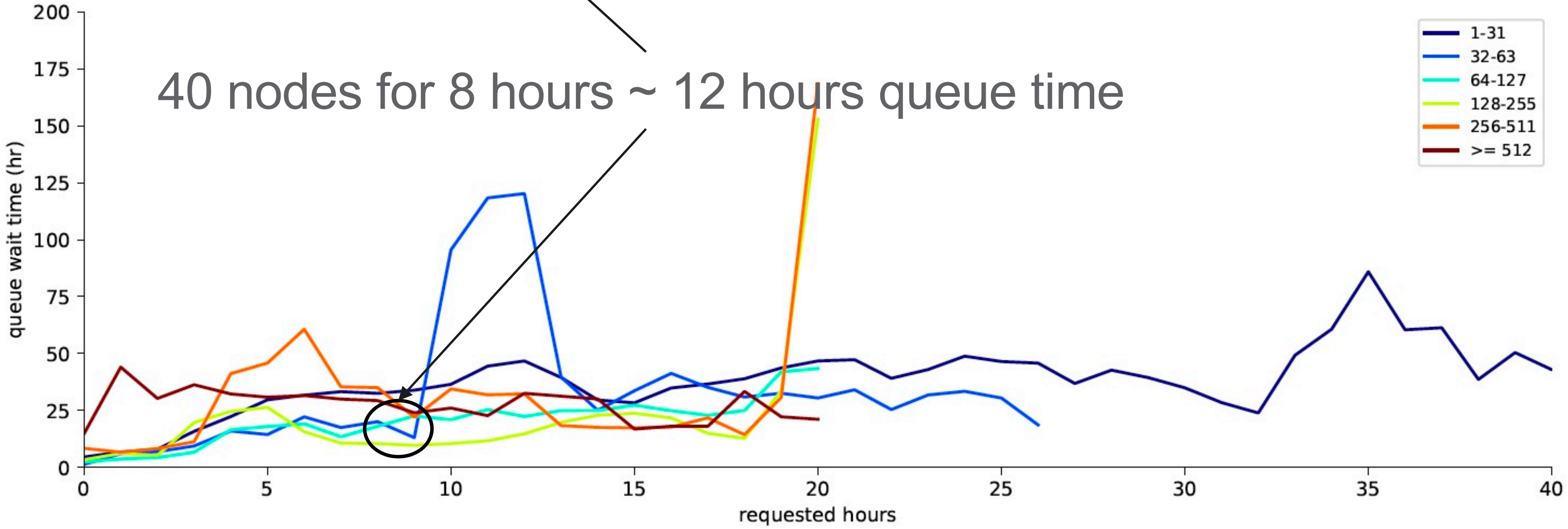
- Burst Buffer -> yet to try (already got advice from Steve Leak on where to put relevant commands in the (complex) CESM job management code)

2020 average regular queue wait time on KNL

Many thanks to Steve Leak for helping me to retrieve the necessary data from MyNERSC!



40 nodes for 8 hours ~ 12 hours queue time



with queue waiting:
264 jobs*(**6** core hrs + **12** queue hours) = 4752 hrs = **198 days**

+

manual work: configuration, tests, post-process, etc

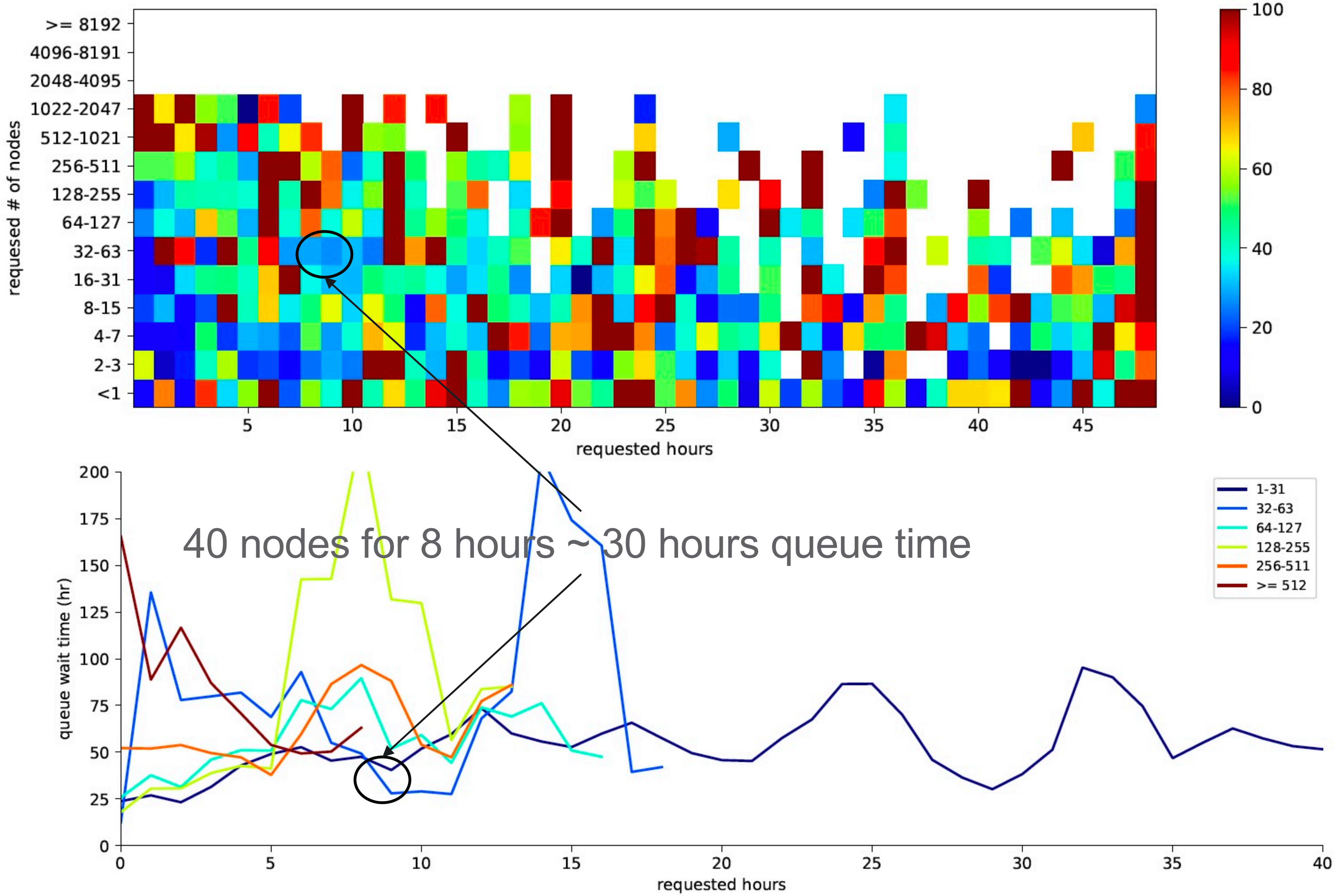
+

system down time

-> ~ 1 year

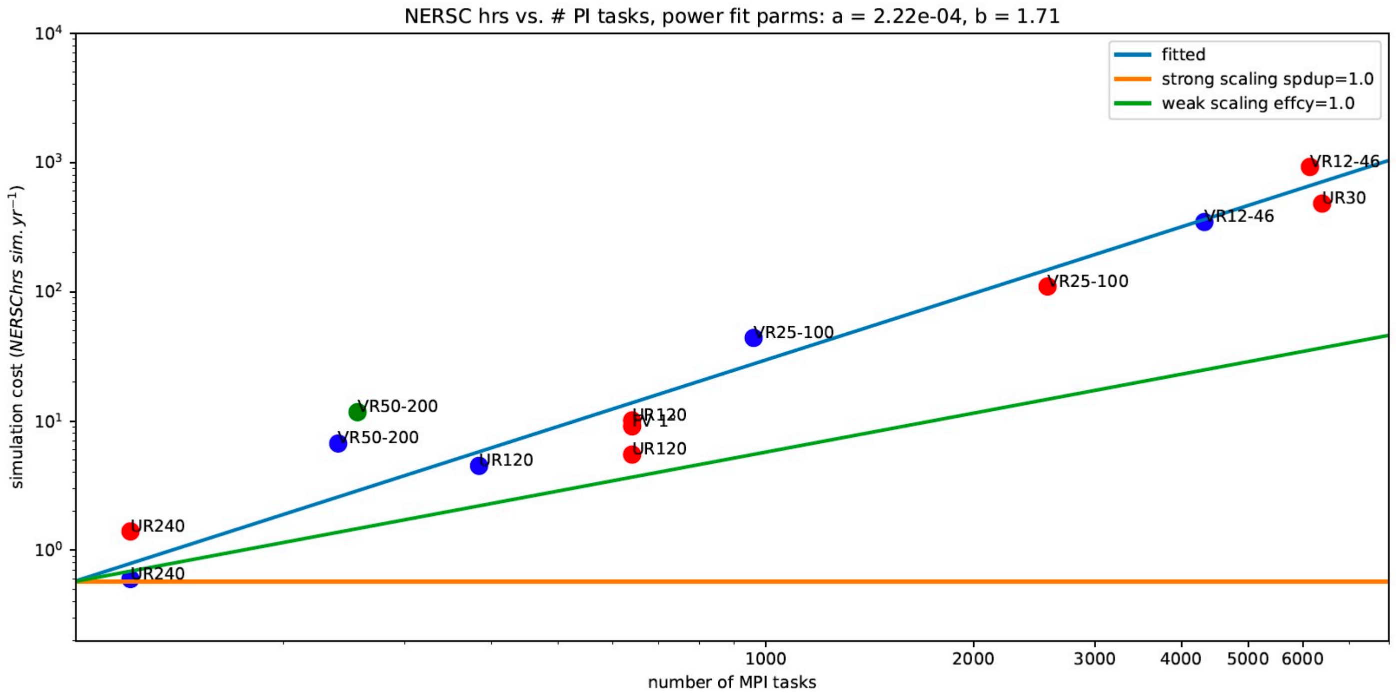
Cannot simply take the best throughput # MPI ranks; need to consider queue wait time for a given number of nodes

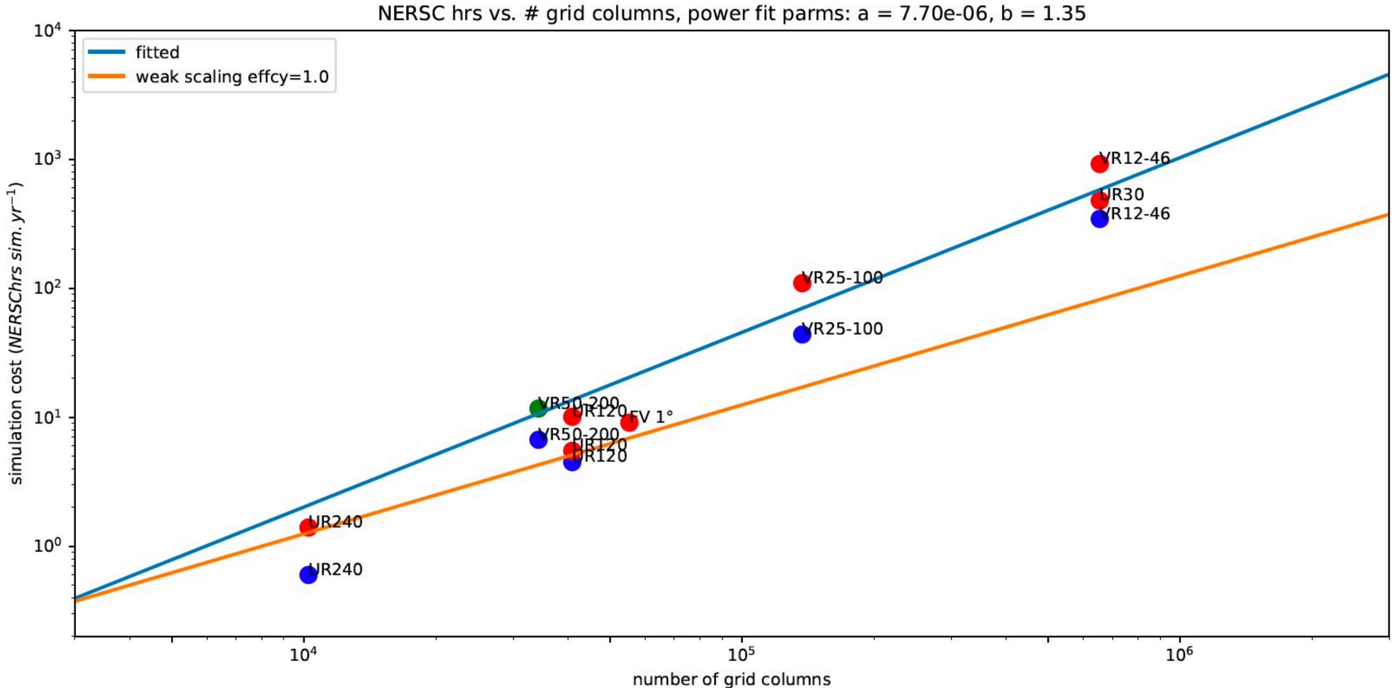
2020 average regular queue wait time on Haswell



40 nodes for 8 hours ~ 30 hours queue time

Science pushes toward higher resolutions: how does simulation costs increase





Simulation cost (NERSC hrs) vs. # of MPI ranks (top) and # of grid columns (bottom)

Higher-resolutions -> more grid boxes (problem size increases)
-> Weak scaling problem

For the current NERSC Charge policy, only the code with a perfect strong scaling (x2 MPI ranks, 1/2 run time for the same problem size) will keep the cost same regardless of nodes used (orange line in the top panel)

My simulations using a range of grid resolutions are not perfect weak scaling as expected (top: green line, bottom: orange line), and they become less and less optimal with higher resolutions (also as expected)

Challenges and my thoughts

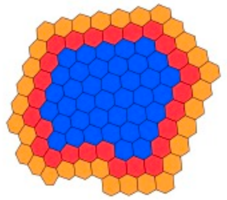
In general, queues for <= 3 hours are less crowded regardless of requested # of nodes (on KNL)

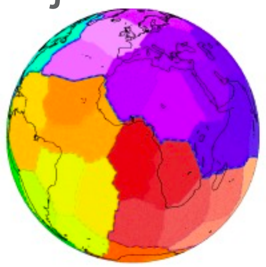
↔ For climate simulations, one month is a convenient time scale (for statistics); less than a month typically requires saving an additional (large) file to keep statistics of many variables at the end of each job

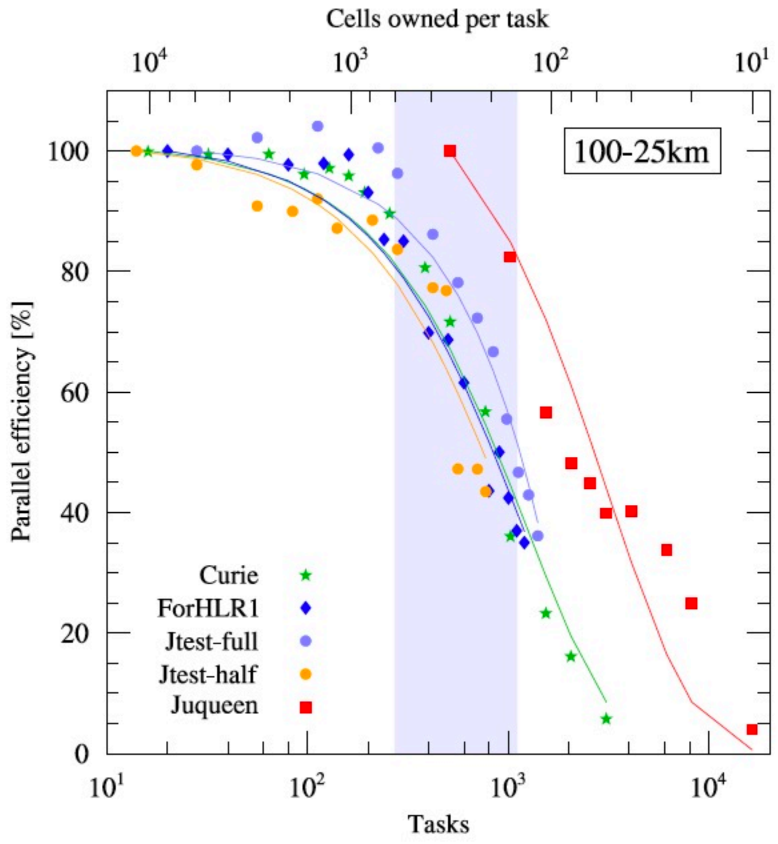
→ On-line calculation of summary statistics/dimension reduction by off-loading to GPU?

↔ Current model code has optimal numbers of MPI ranks (I/O and communication/halo cells); increasing just MPI # to fit the job to be within 3 hours is difficult

→ MPI + threading necessary?







Heinzeller et al., 2016, Geosci. Mod. Dev.

Large community model code cannot be optimized for a given system by an individual science project

→ Experimental/cutting edge models start to support GPU offloading (MPAS -> OpenACC, SCREAM-> Kokkos)

↔ Many model processes are memory-bound

Users need more HPC knowledge to run simulations