Testing Hypotheses about Climate Prediction at Unprecedented Resolutions on the NSF Blue Waters System

Ross Heikes, David Randall

Dept. of Atmospheric Science
Colorado State University

Blue Waters Symposium – May 21-22, 2013

Outline.

1. Introduction to the model and grid -- boilerplate slides with lots of spheres.
2. Parallel scaling of the MPI portion of the model.
3. Experiences (so far) with the accelerators.

Icosahedral grid. Projecting to the sphere.

- Our models live on an icosahedral grid.
- Starting with an icosahedron (fig. 1)
- We can project the icosahedron onto a unit sphere (fig. 2) forming 20 spherical triangles.

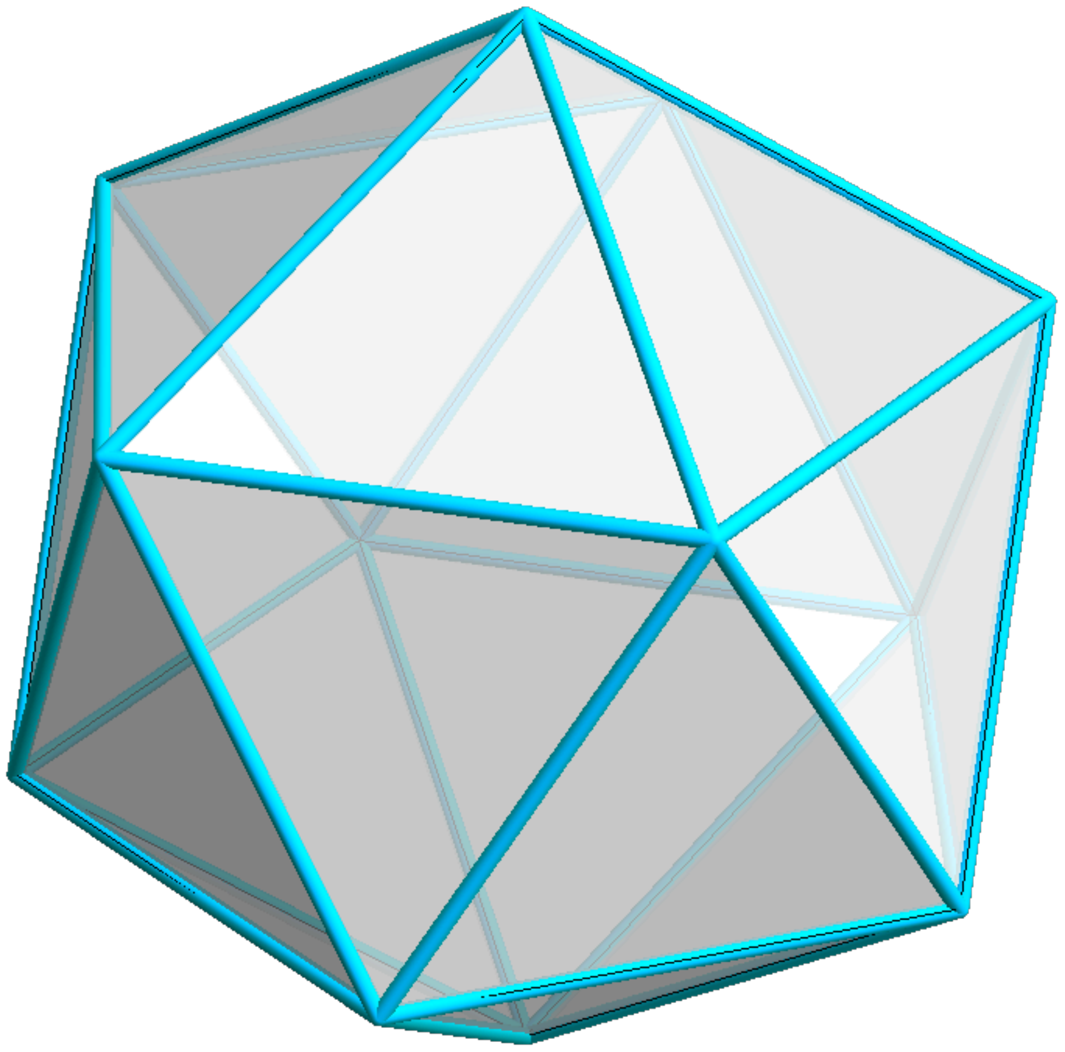


figure 1

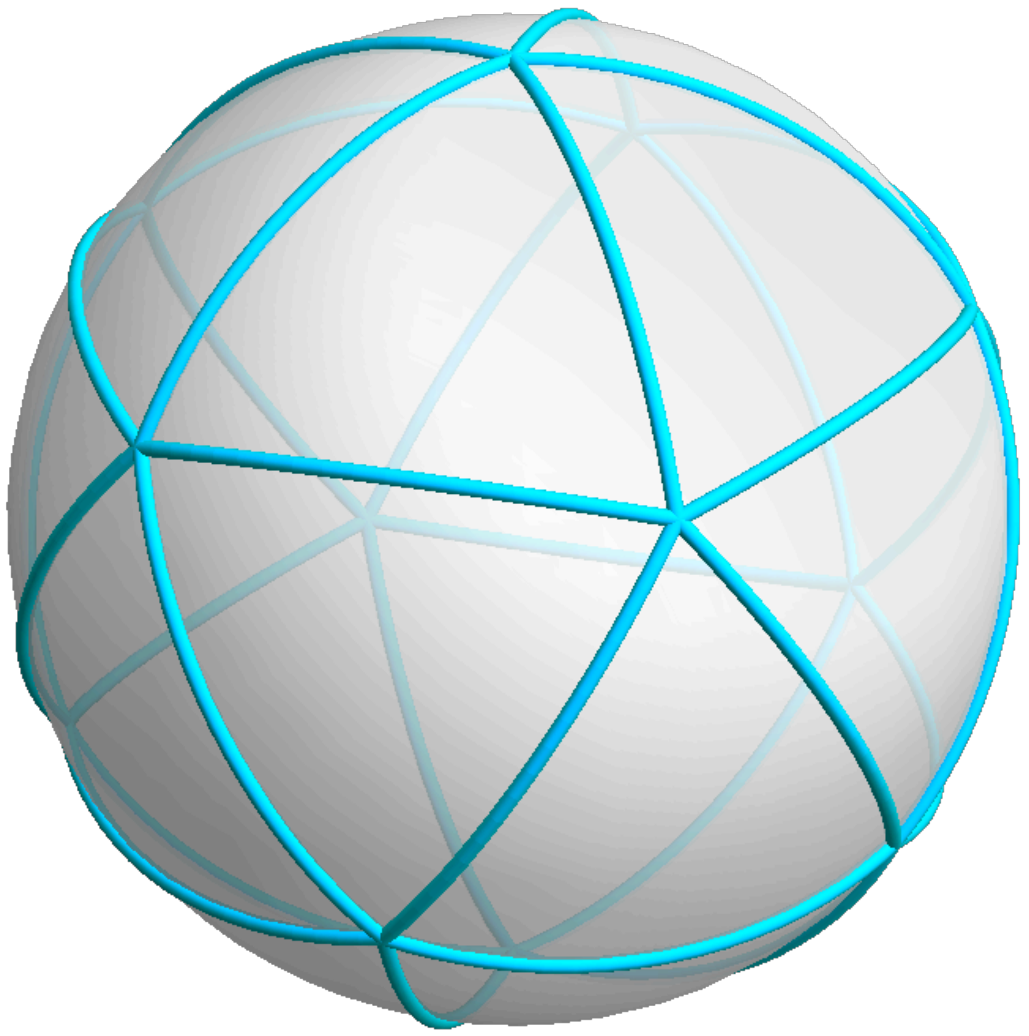
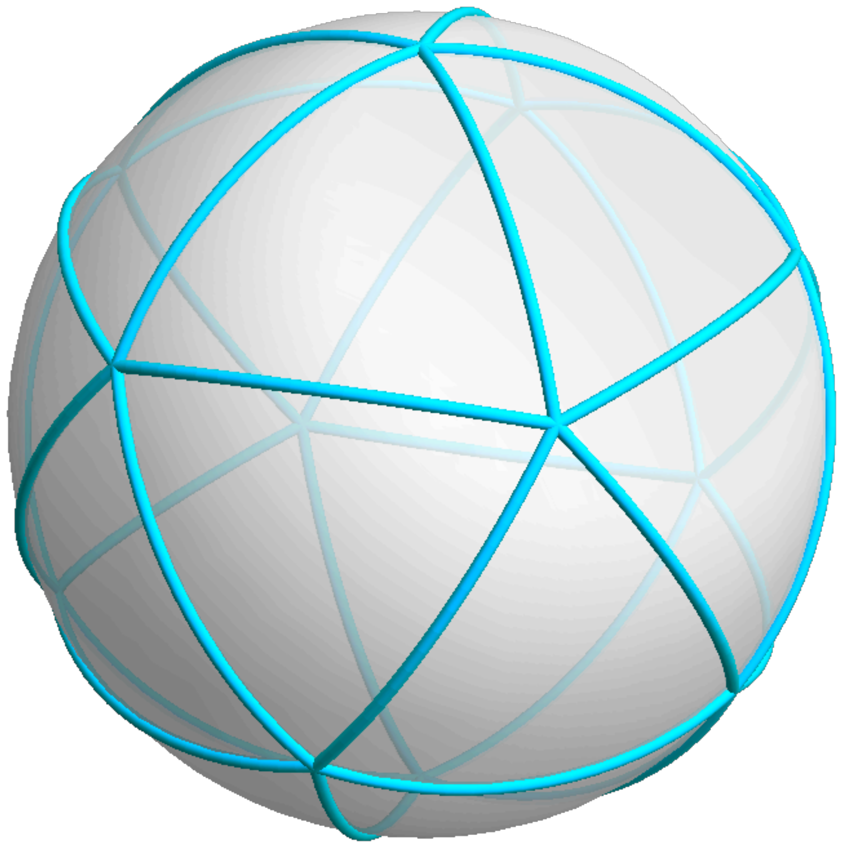
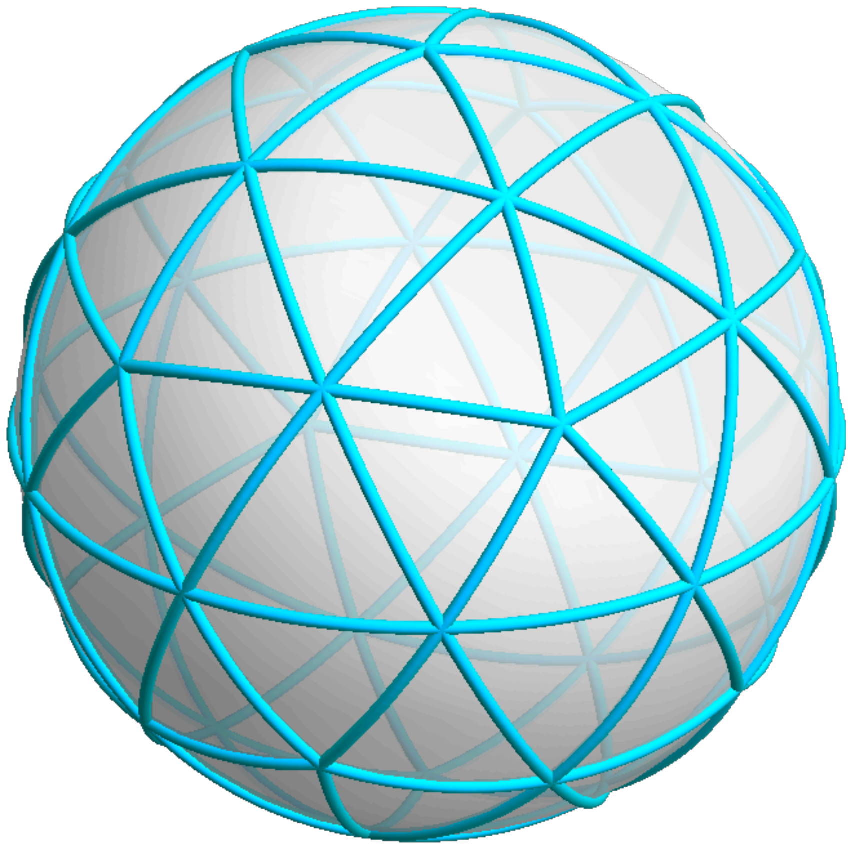


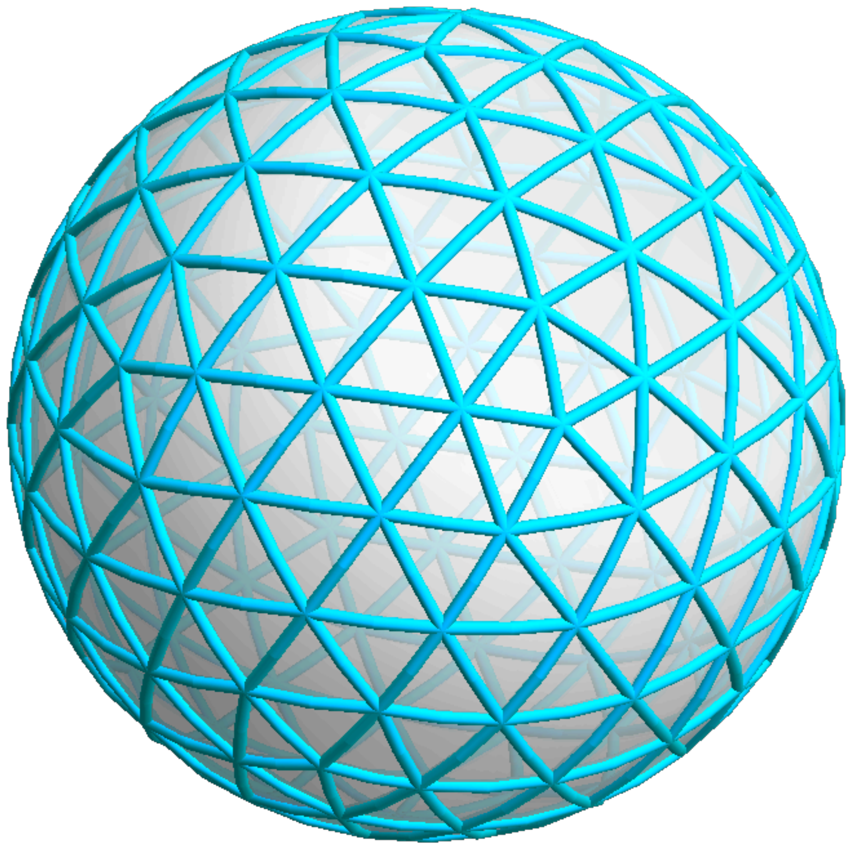
figure 2

Icosahedral grid. Generating polyhedron.

- Each spherical triangles can be further partitioned into four spherical triangles.
- The algorithm can be applied recursively.
- These polyhedrons are used to generate the icosahedral grid.

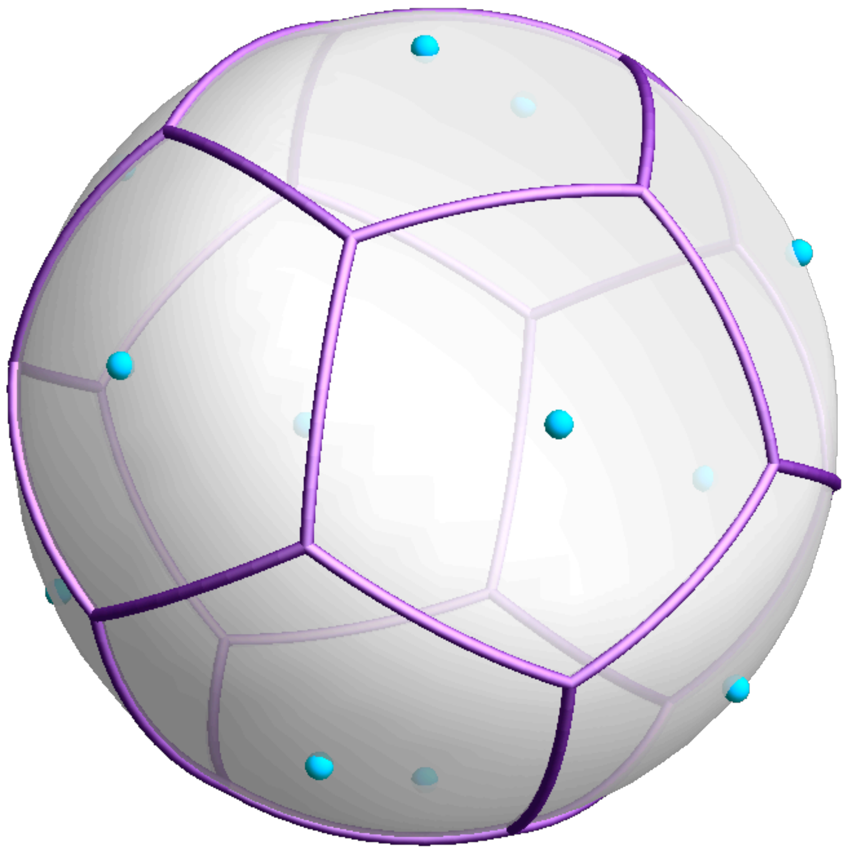

20
triangles

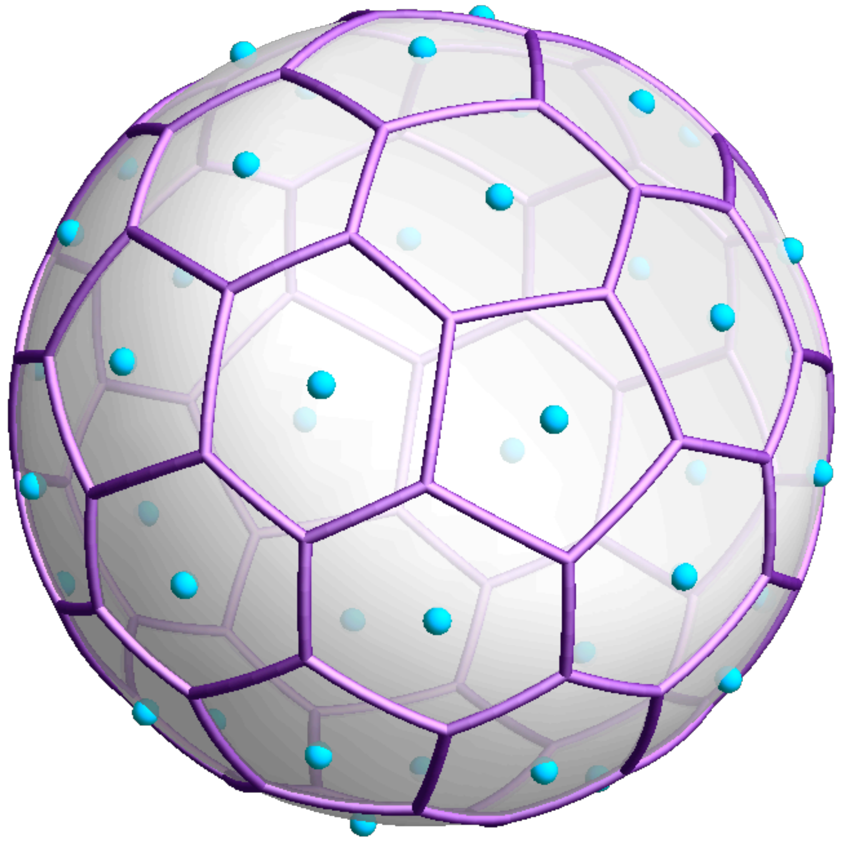

80
triangles

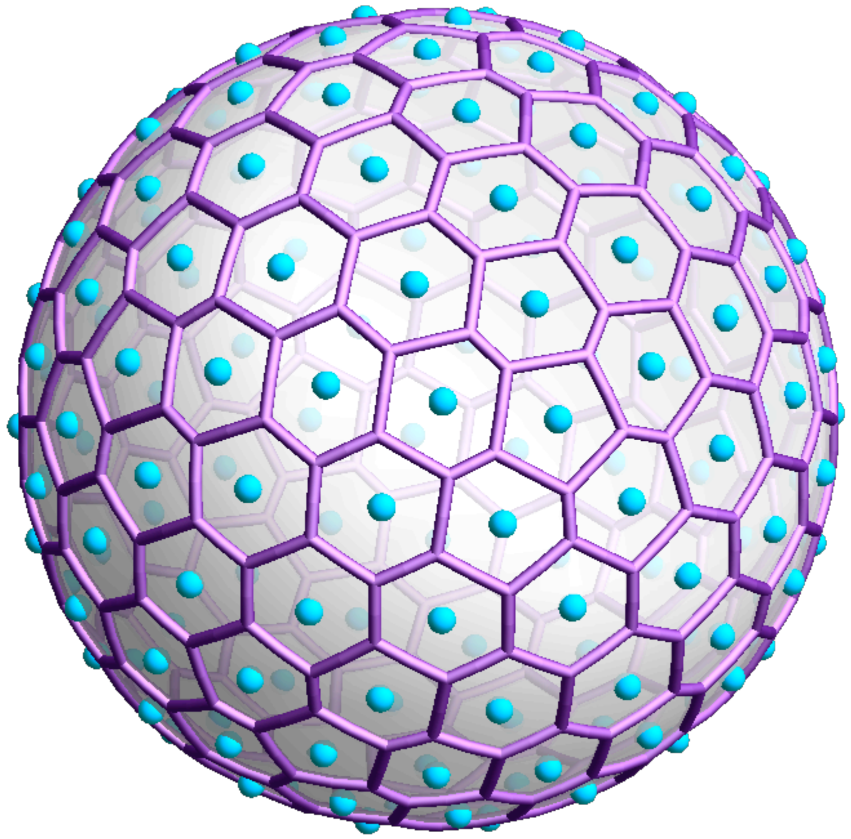

360
triangles

Icosahedral grid.

- The vertices of the previous polyhedrons (shown here as blue points) are used to generate the icosahedral grids. The vertices are called generating points.
- An area (Voronoi cell) on the sphere is associated with each generating point.
- This algorithm allows for an isotropic and homogeneous tiling of the sphere to arbitrarily high resolution.


12 cells


42 cells


162 cells

Unprecedented resolution. Counting the cells.

- Let r denote the number of applications of the subdivision algorithm, that is partitioning one triangle into four triangles.
- Our target resolutions are:

resolution (r)	number of cells	global grid point spacing (km)
9	2,621,442	14.99
10	10,485,762	7.495
11	41,943,042	3.747
12	167,772,162	1.874

- The vertical resolution depends on the horizontal resolution. The vertical resolution is typically 32 to 256 layers.

Model equations of the dynamical core

- **Vorticity**

$$\frac{\partial \zeta}{\partial t} + \nabla_H \bullet (\zeta_a \mathbf{v}) + \mathbf{k} \bullet \nabla_H \times \left(w \frac{\partial \mathbf{v}}{\partial z} \right) + J\left(c_p \theta, \pi_{qs}\right) + J\left(c_p \theta, \delta\pi\right) = F_\zeta$$

- **Divergence**

$$\frac{\partial D}{\partial t} - J(\zeta_a, \chi) - \nabla_H \bullet (\zeta_a \nabla_H \psi) + \nabla_H \bullet \left(w \frac{\partial \mathbf{v}}{\partial z} \right) + \nabla^2 K + \nabla_H \bullet \left(c_p \theta \nabla_H \pi_{qs}\right) + \nabla_H \bullet \left(c_p \theta \nabla_H \delta\pi\right) = F_D$$

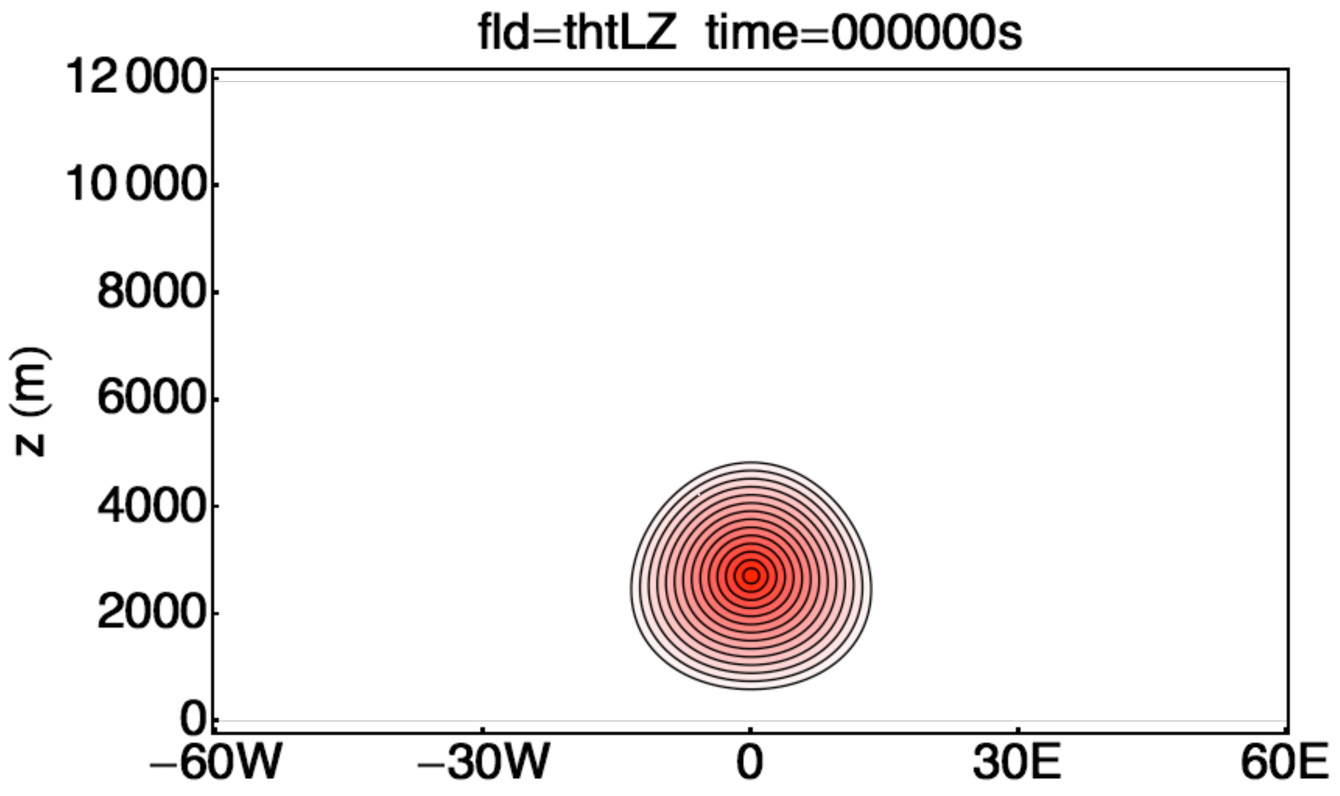
- **Potential Temperature**

$$\frac{\partial \theta}{\partial t} + \frac{1}{\rho_{qs}} \left[\nabla_H \bullet \left(\rho_{qs} \theta \mathbf{v}\right) - \theta \nabla_H \bullet \left(\rho_{qs} \mathbf{v}\right) \right] + \frac{1}{\rho_{qs}} \left[\frac{\partial}{\partial z}\left(\rho_{qs} \theta w\right) - \theta \frac{\partial}{\partial z}\left(\rho_{qs} w\right) \right] = \frac{Q}{\pi_{qs}}$$

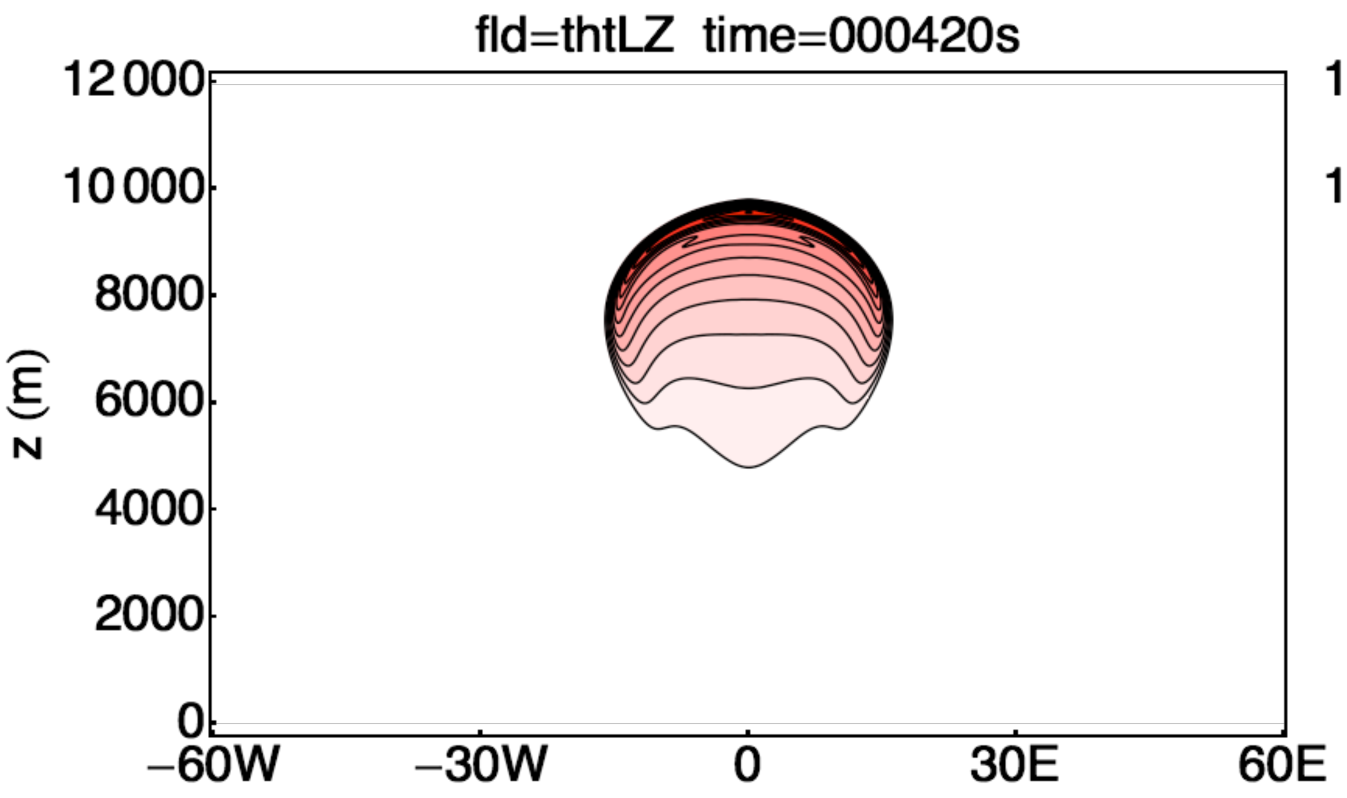
- **Several species of water**

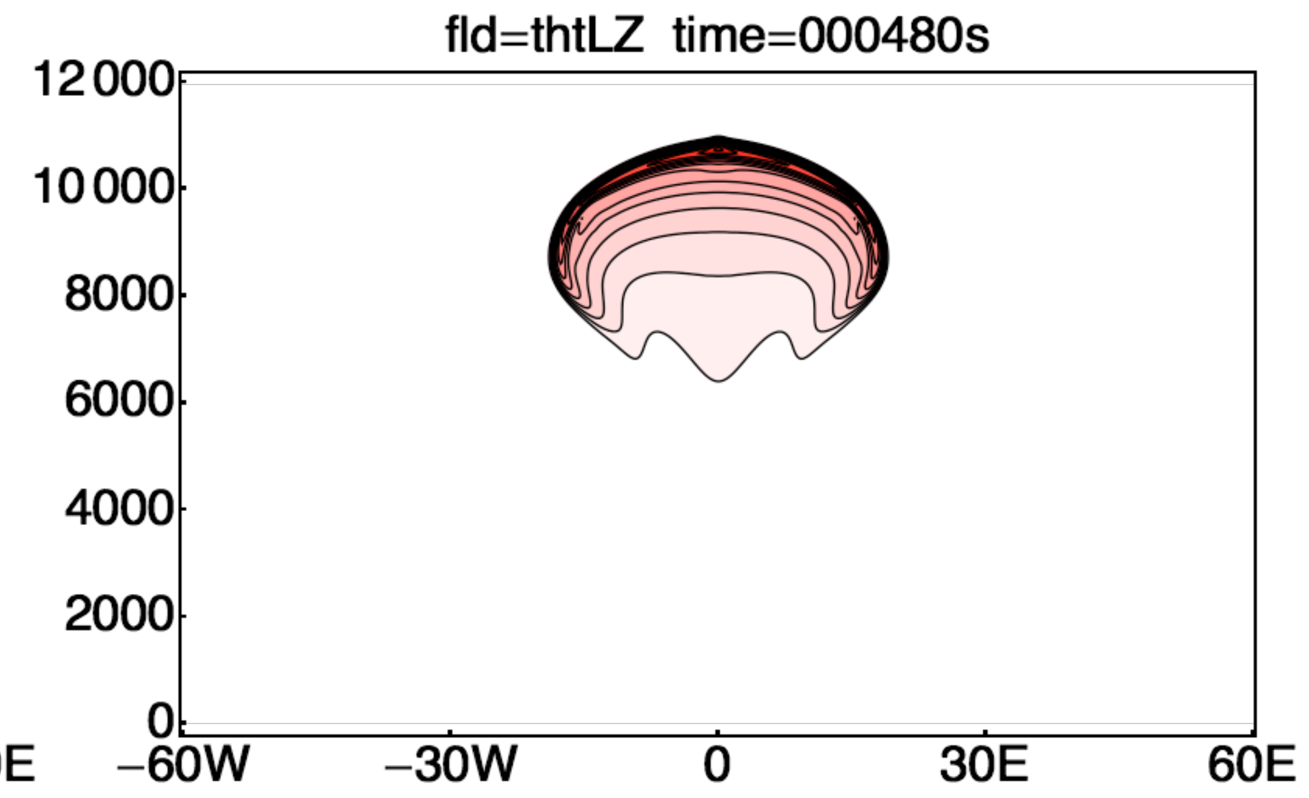
Example 1. Warm Bubble Test

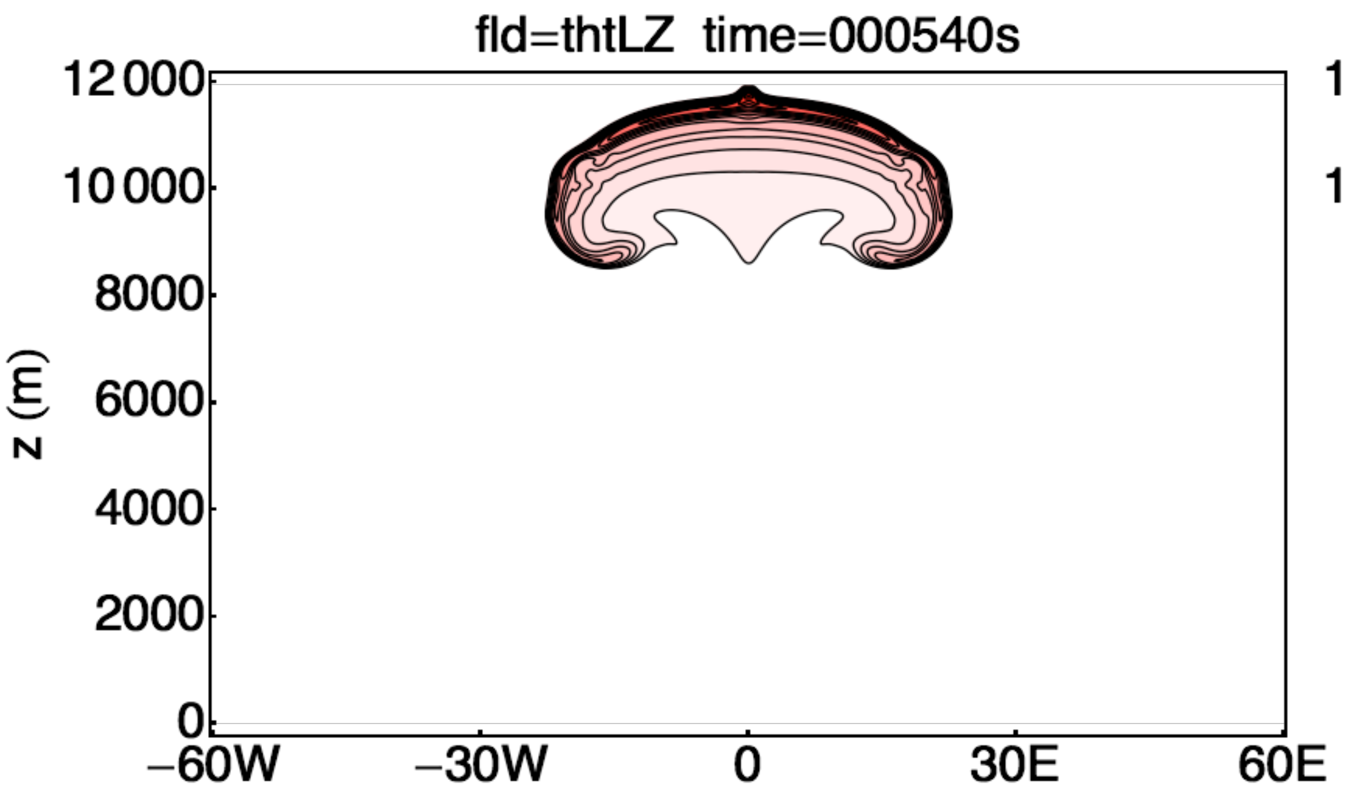
- Initial condition is the 3D version of Mendez-Nunez and Carroll (1994)
- The initial bubble is 6.6K warmer than the environment.
- The globe is 6.37km in radius (1000×smaller)
- The model's resolution is
  - 163842 cells resulting in **63 m horizontally**
  - 160 levels resulting in **75 m vertically**

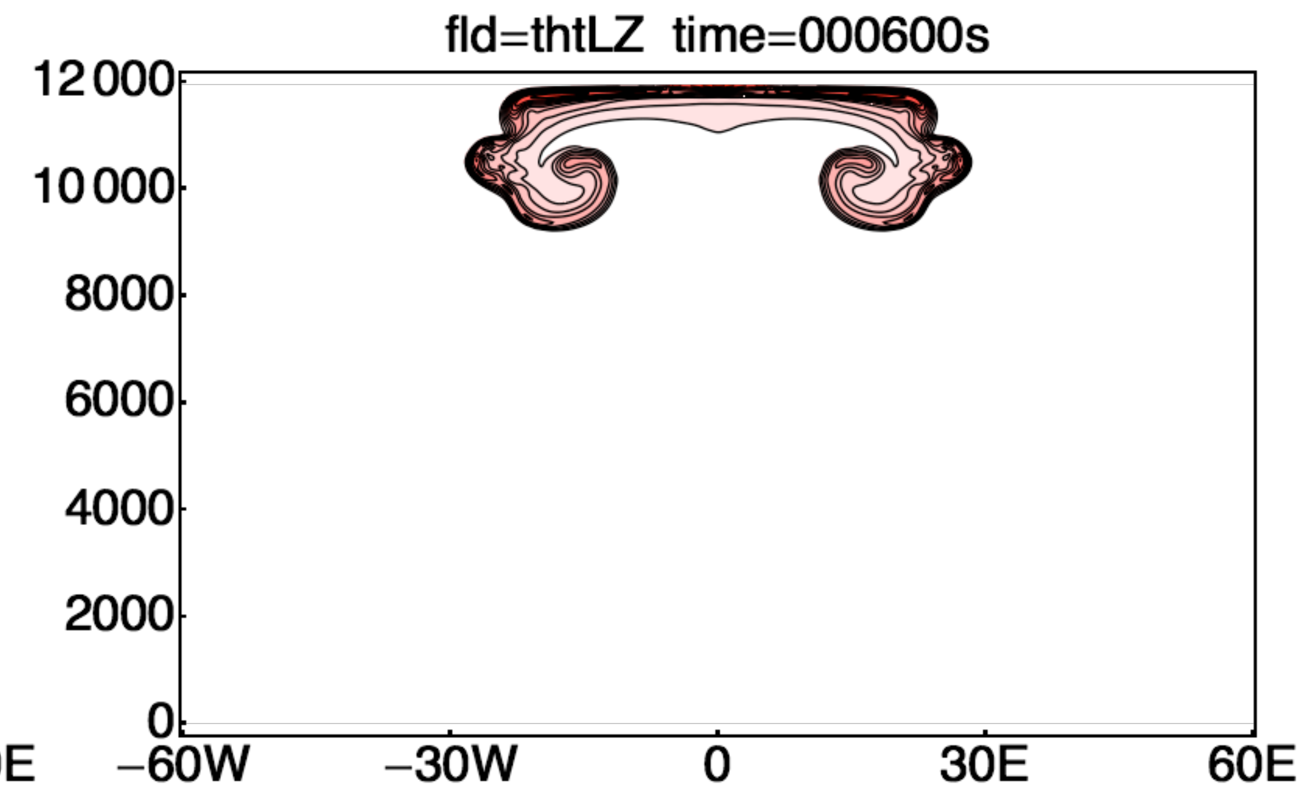


Example 1. Warm Bubble Test



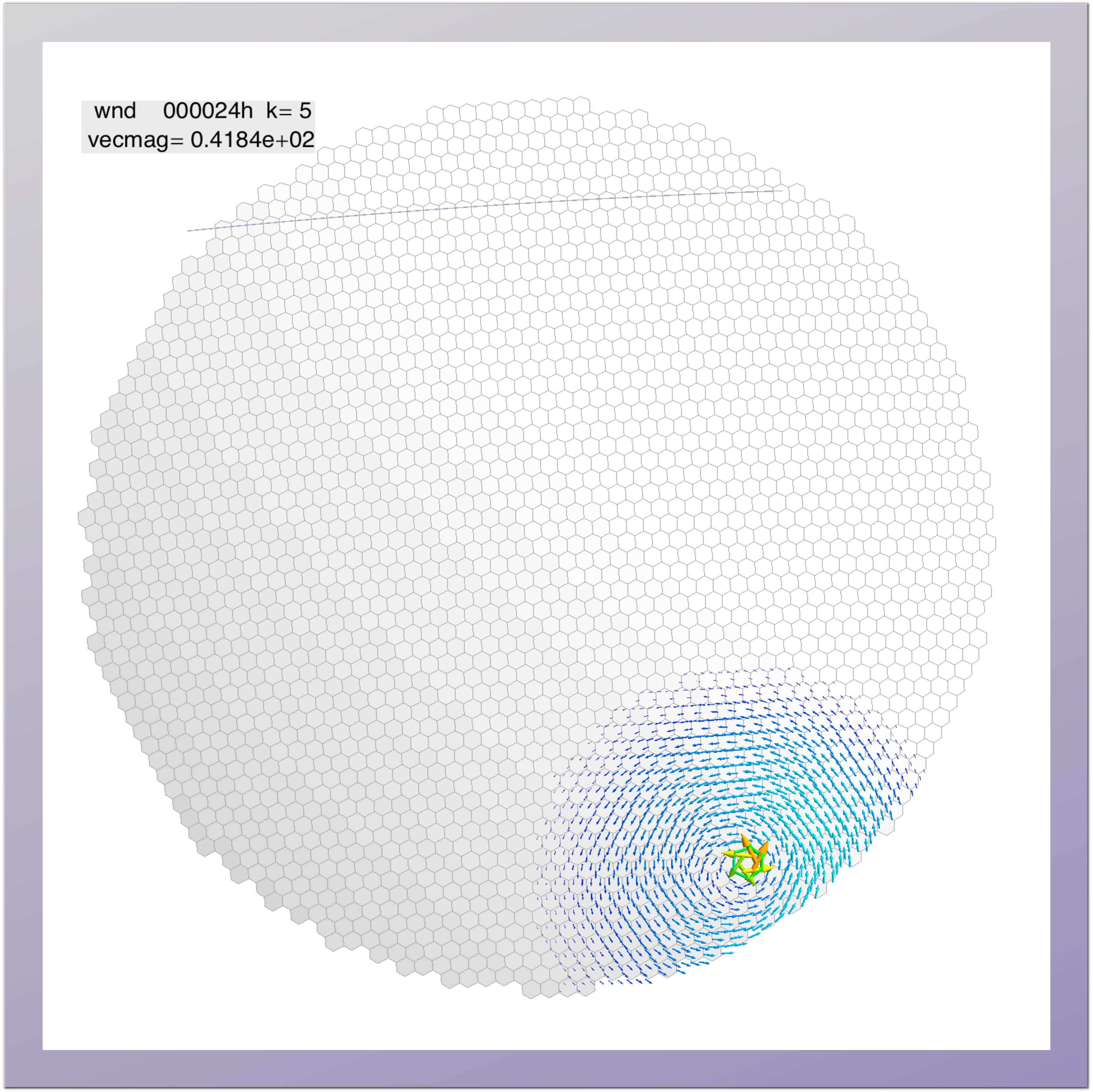






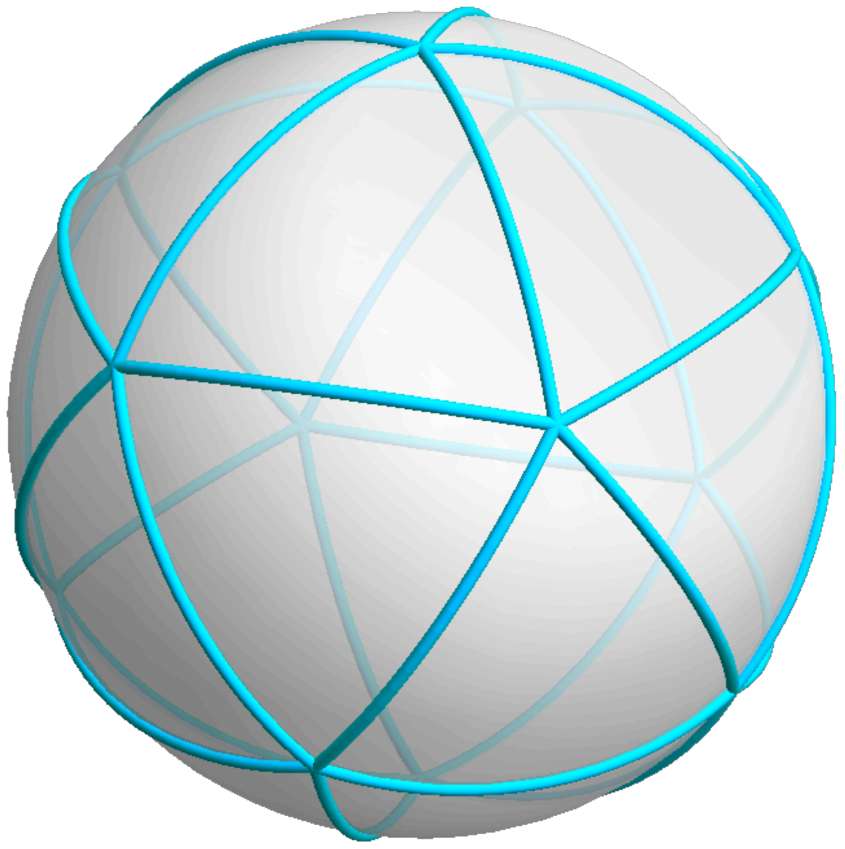
Example 2. Idealized tropical cyclone.

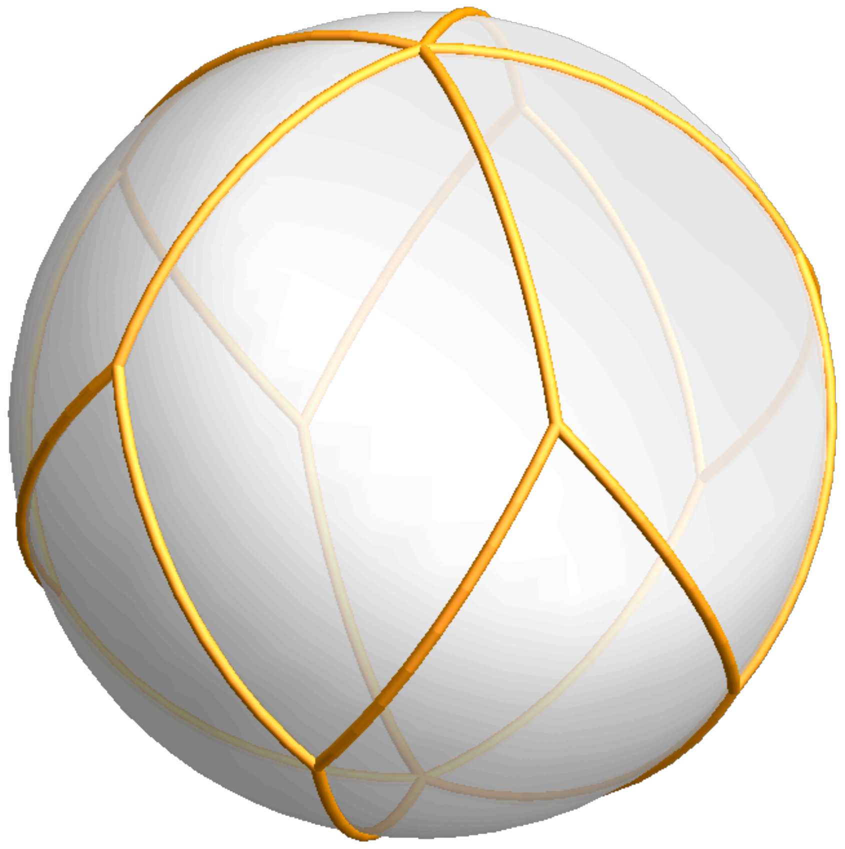
- Nonhydrostatic models of the atmosphere with moist physics.
- The animation shows the horizontal track of the cyclone.
- For example, Reed and Jablonowski (2011) idealized tropical cyclone test case.

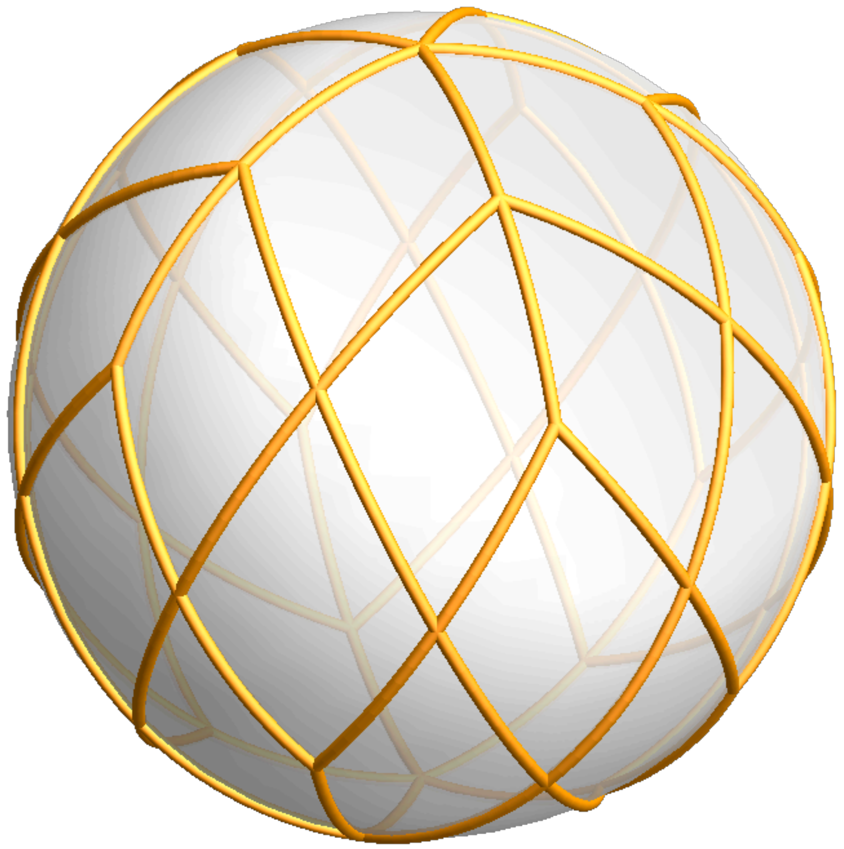


Parallel domain decomposition.

- An algorithm similar to the grid generation algorithm is used to partition the sphere into quadrilateral regions.
- This domain decomposition is used to assign pieces of the grid to MPI tasks.

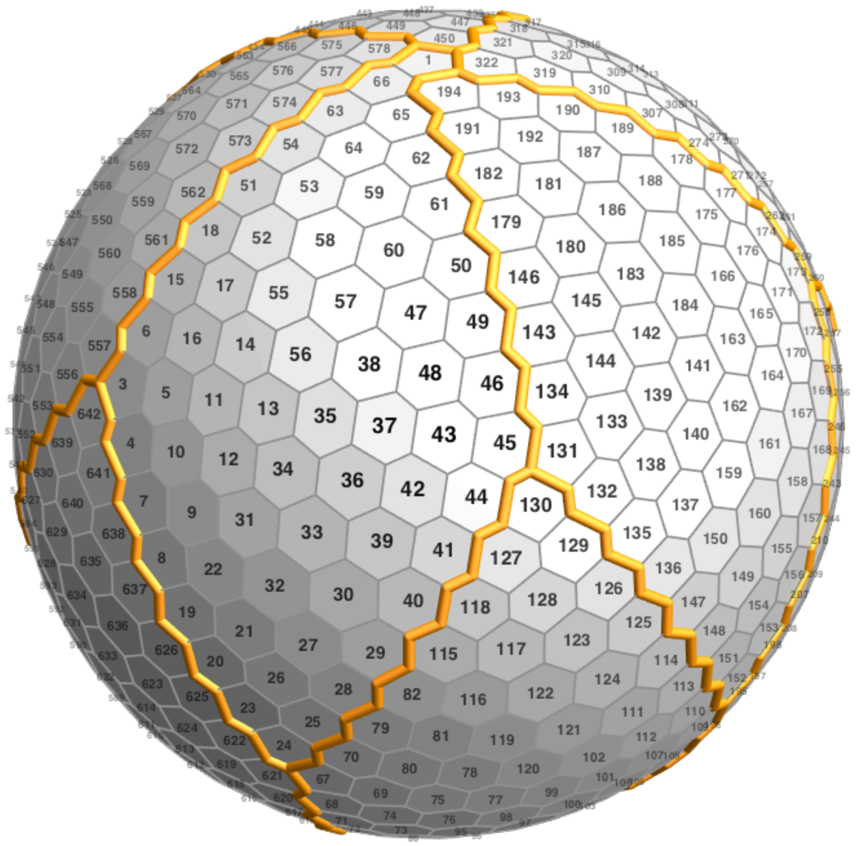

20 triangles

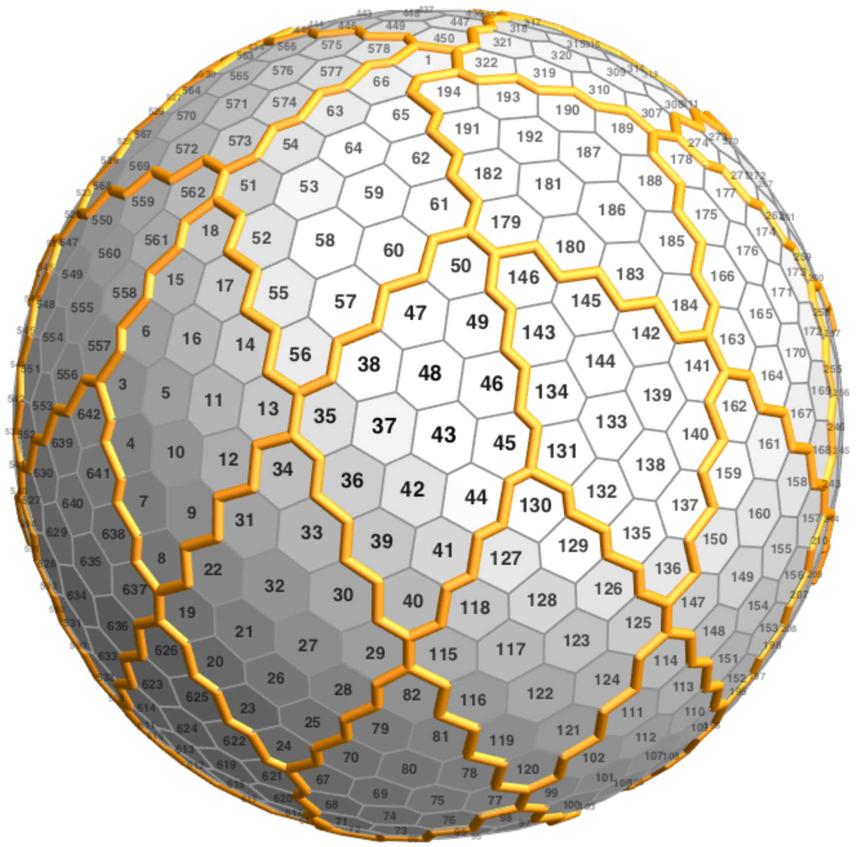

10 pieces

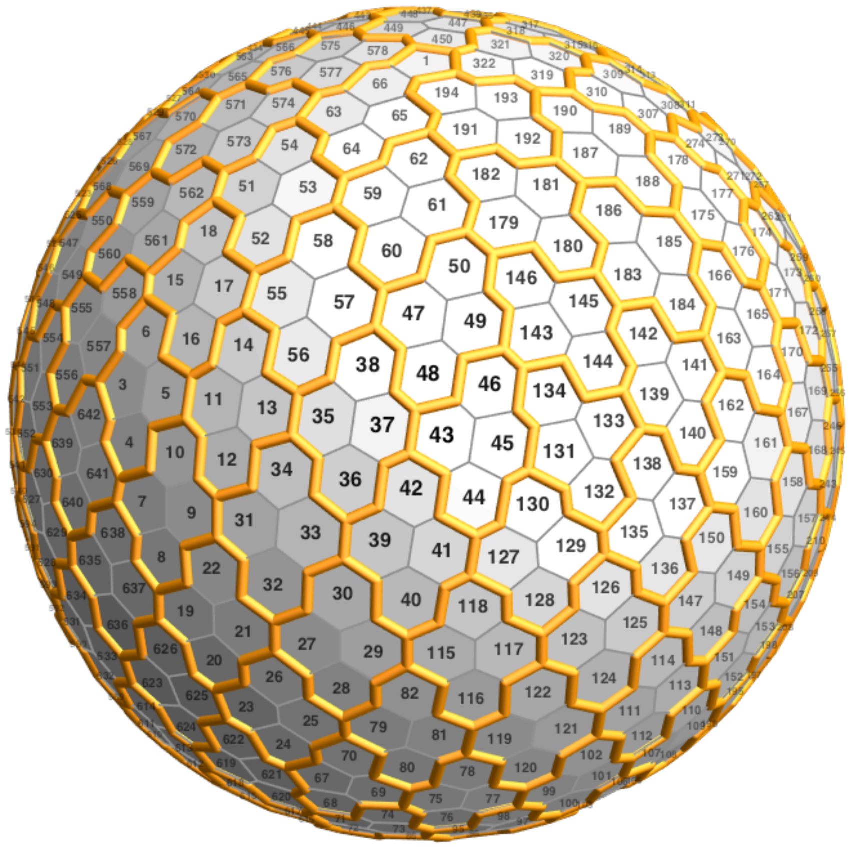

40 pieces

Icosahedral grid. Parallel domain decomposition. Distribution to MPI tasks.

- Pieces of the grid are assigned to MPI tasks.
- MPI non-blocking sends/receives are used to update ghost regions (halo regions) with data from neighboring processes.





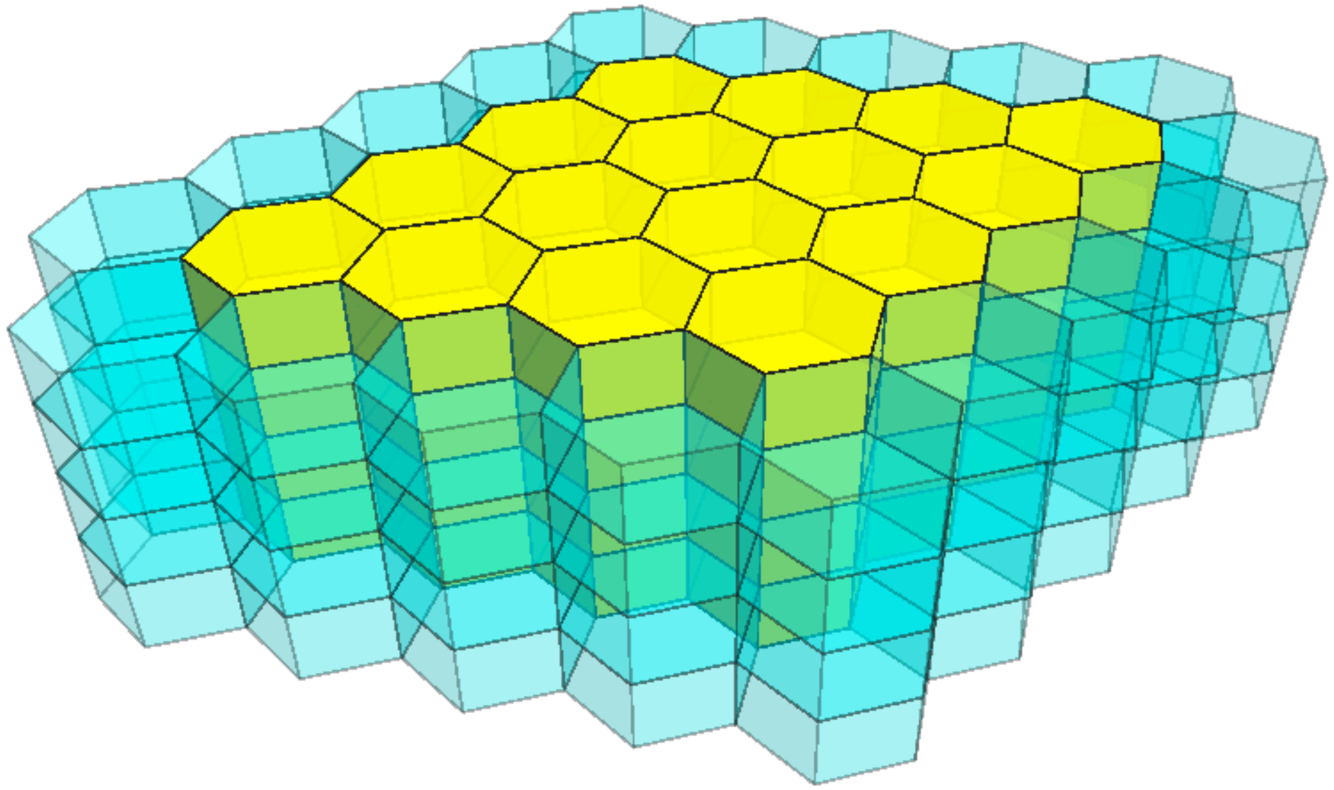


Parallel efficiency

- Each grid block requires information from neighboring subdomains to fill ghost cells.
- We can define **parallel efficiency** to be:

$$\text{parallel efficiency} \approx \frac{\text{number of local cells}}{\text{number of ghost cells}}$$

- **Larger parallel efficiency is better**. More useful work is done per ghost cells.



Yellow cells *belong to the local process*

Blue cells *are ghost cells filled from neighboring process*

Parallel domain decomposition and parallel efficiency

- We would like each MPI task to have a 32×32 cell block or a 64×64 cell block:
  - Smaller. The parallel efficiency is bad.
  - Bigger. Too much work per task
- For a given resolution increasing the number of tasks reduces parallel efficiency.

block size (parallel efficiency)		number of MPI tasks			
		640	**2560**	**10240**	**40960**
resolution (grid spacing)	**9** (14.99 km)	64×64 (15.7)	32×32 (7.76)	16×16 (3.76)	
	10 (7.495 km)	128×128 (31.7)	64×64 (15.7)	32×32 (7.76)	16×16 (3.76)
	11 (3.747 km)	256×256 (63.7)	128×128 (31.7)	64×64 (15.7)	32×32 (7.76)
	12 (1.874 km)		256×256 (63.7)	128×128 (31.7)	64×64 (15.7)

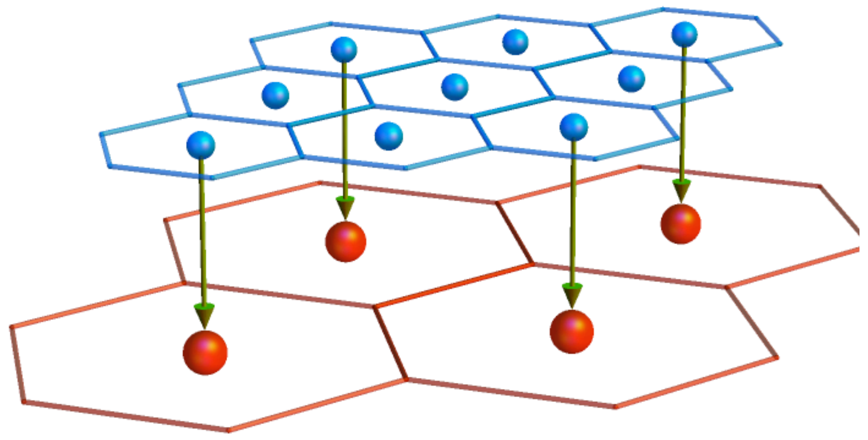
2D multigrid

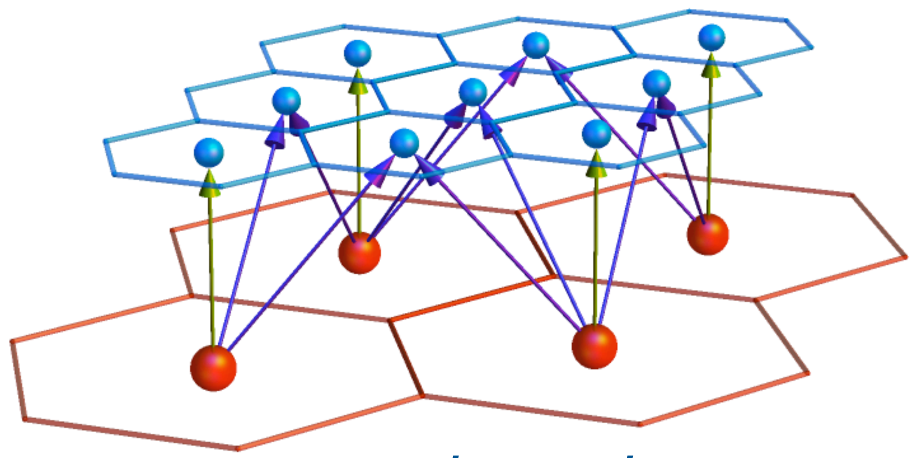
- The mathematical formulation of our prognostic equations requires solving Poisson's equation every time step in each model layer.
- The recursive structure of the grid facilitates the use of multigrid methods.
- This is most communication intensive portion of the model and challenging to parallelize. The lessons learned can be apply to other parts of the model.
- There are two main parts to the multigrid algorithm:

  (1) Relaxation sweep. Similar to a standard Jacobi iteration. Most expensive.

$$\alpha_i = \sum_j \omega_{i,j}\alpha_{i,j} - \omega_i \beta_i \quad \text{for all} \quad i = 1,2,\ldots,N$$

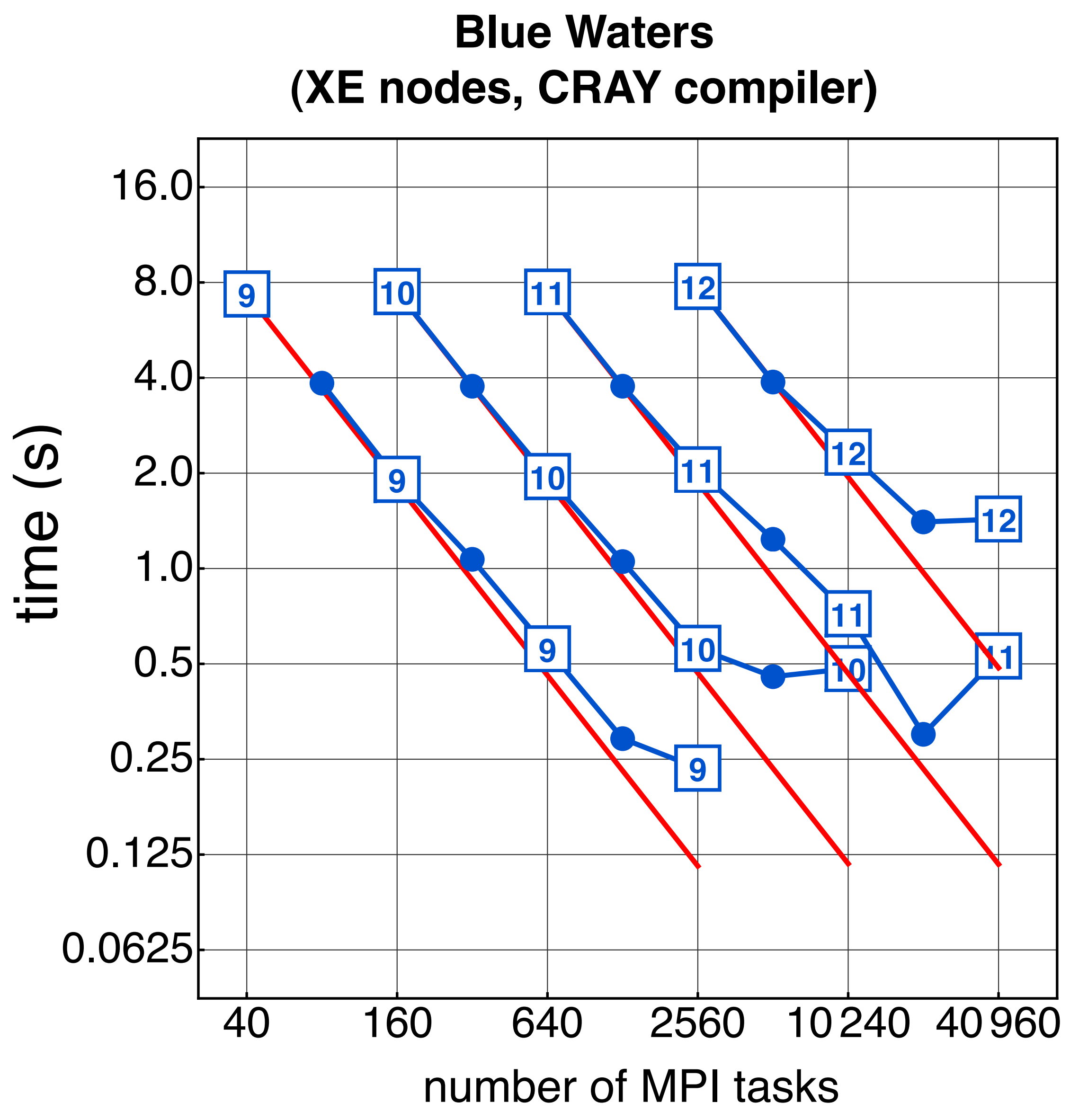
  (2) Transferring information between grid resolutions. Less expensive.

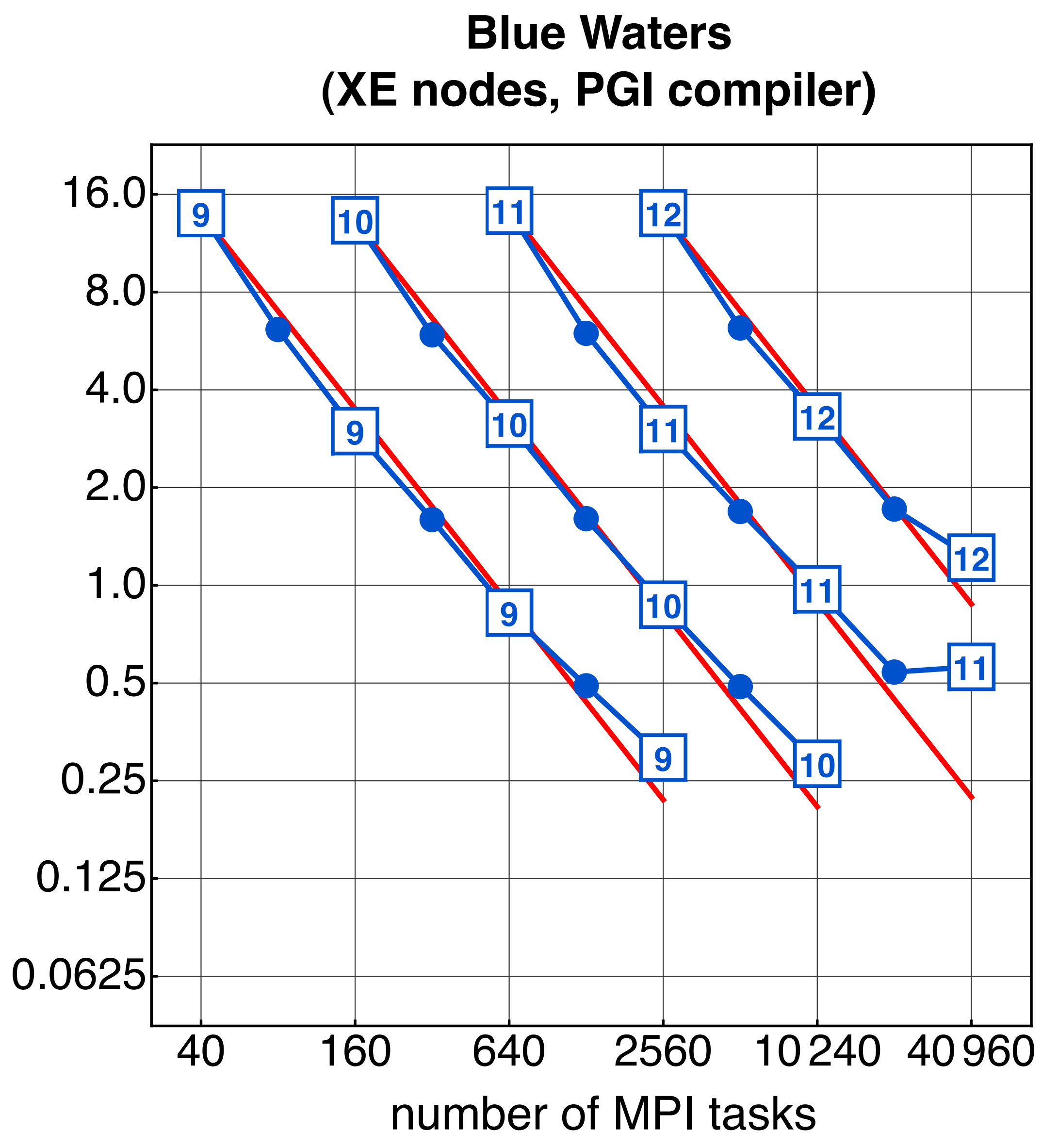

restriction


prolongation

Parallel scaling with MPI on Blue Waters

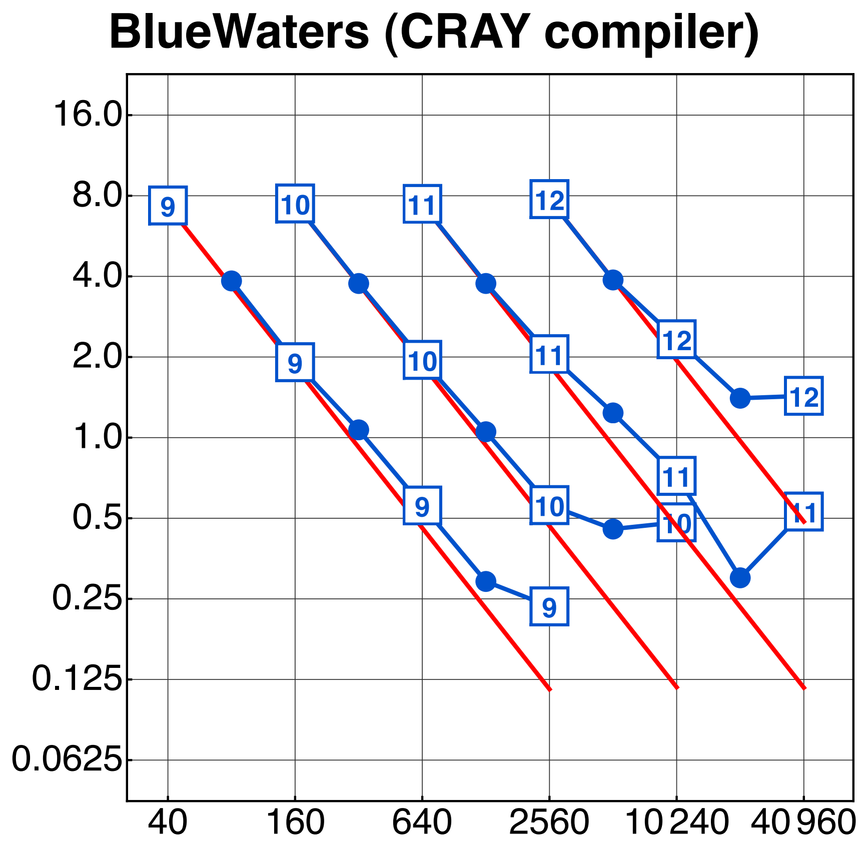
- Plot show the time to do 10 multigrid v-cycles
- X-axis is number of MPI tasks. Y-axis is time. Both are log scale.
- Each blue line indicates a particular grid resolution. Grids 09, 10, 11 and 12.
- The red line is the idealized speed-up.
- For each resolution the red line and the blue line should be coincident.

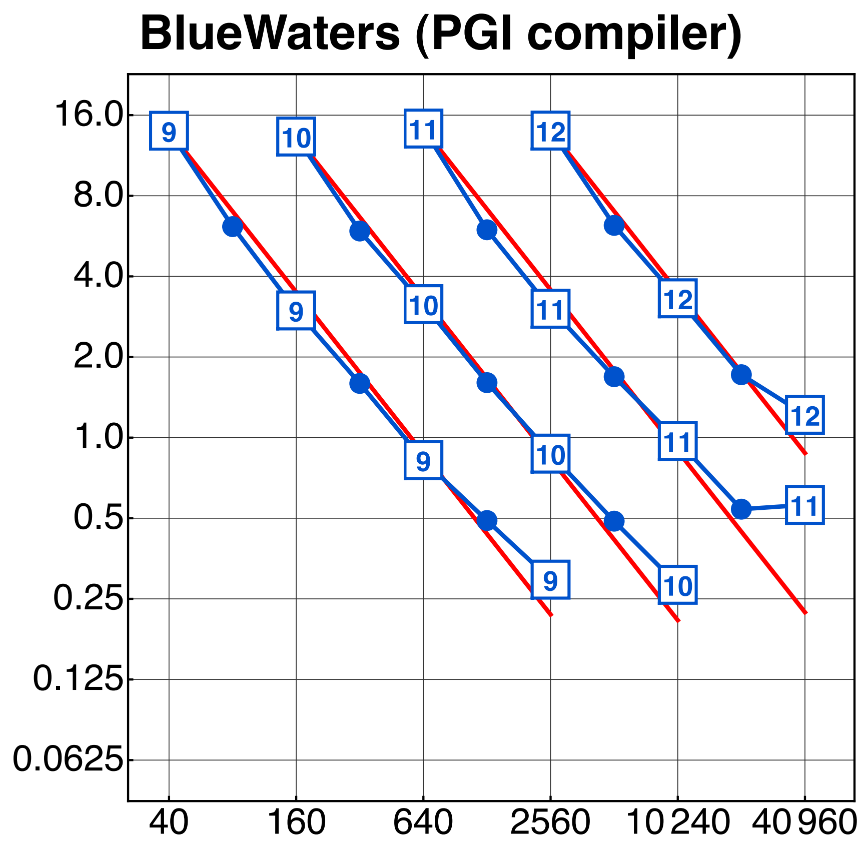


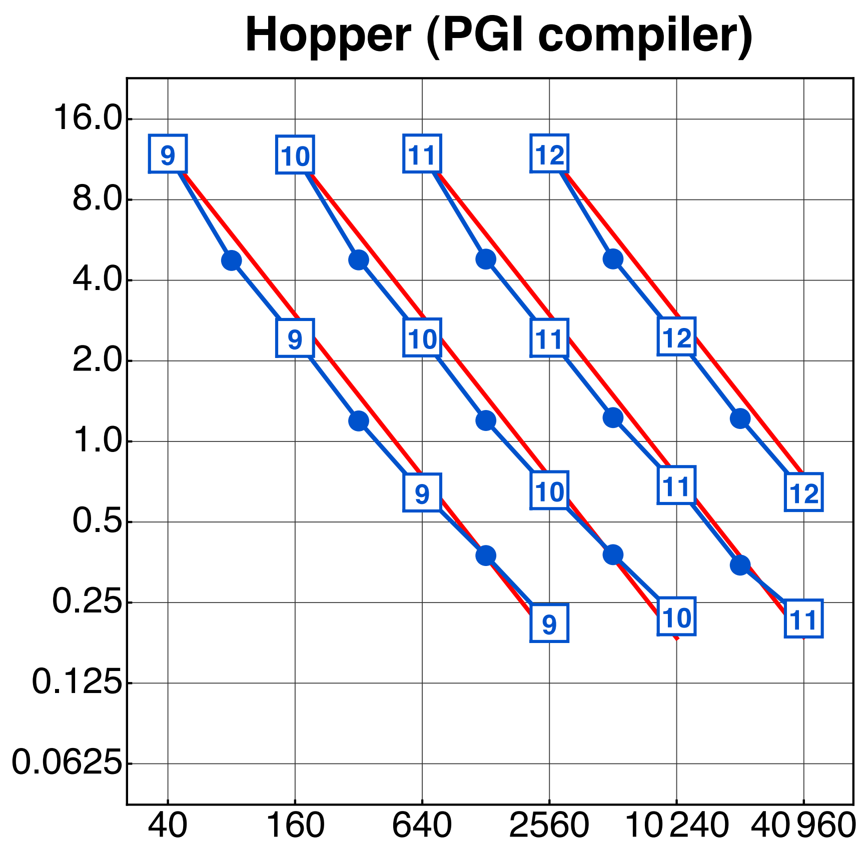


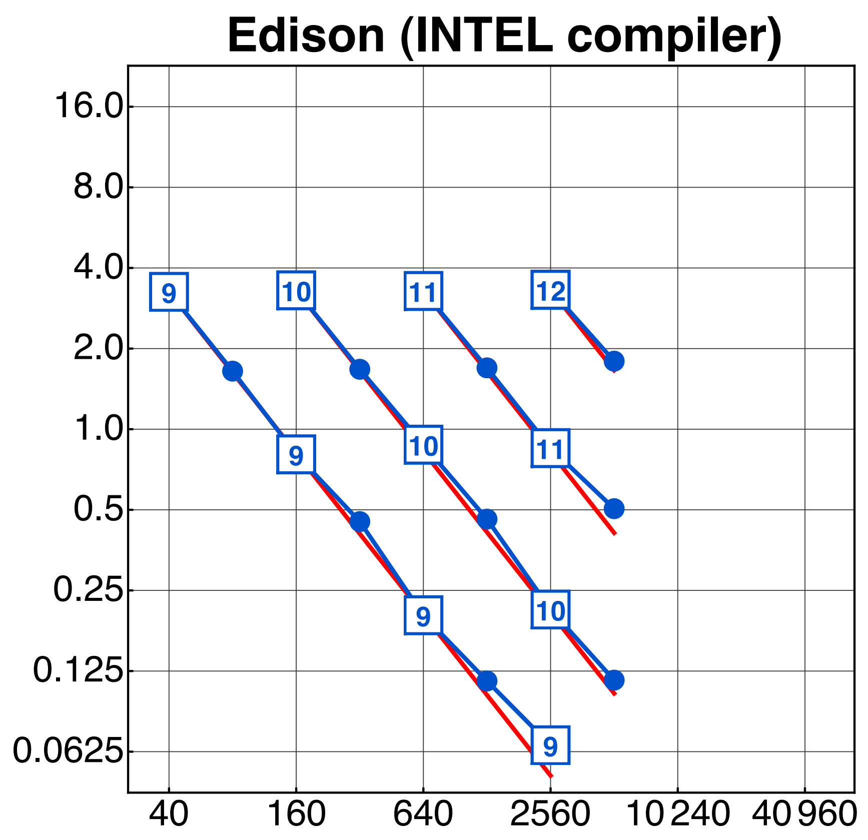
Parallel scaling with MPI. Comparisons.

- All the same code with no heroic optimization.
- We can see:
  - CRAY 2X faster than PGI on BW.
  - PGI scales better than CRAY on BW.
  - BW (PGI) and Hopper (PGI) have similar time
  - Hopper scales well.
  - Edison scales well (but with a relatively low number of cores).
  - Edison is pretty fast.









Multigrid on the accelerators.

- We are very interested in modifying the code to use the accelerators.
- We focus on the relaxation sweep portion of the multigrid algorithm. Experiments show this is the most expensive part of the code.
- The lessons learned can be apply to other parts of the model since the form of the code mimics other finite-difference operators in the model.
- Schematically the pure MPI code looks like this:

```
SUBROUTINE mltgrd2D_rlx (lvl,itermax,im0,jm0,km0,nsdm0,area,wght,beta,alph)

 DO iter = 1,itermax ! number of sweeps
      MPI communication

      Relaxation Sweep

 ENDDO ! iter

 END SUBROUTINE mltgrd2D_rlx
```

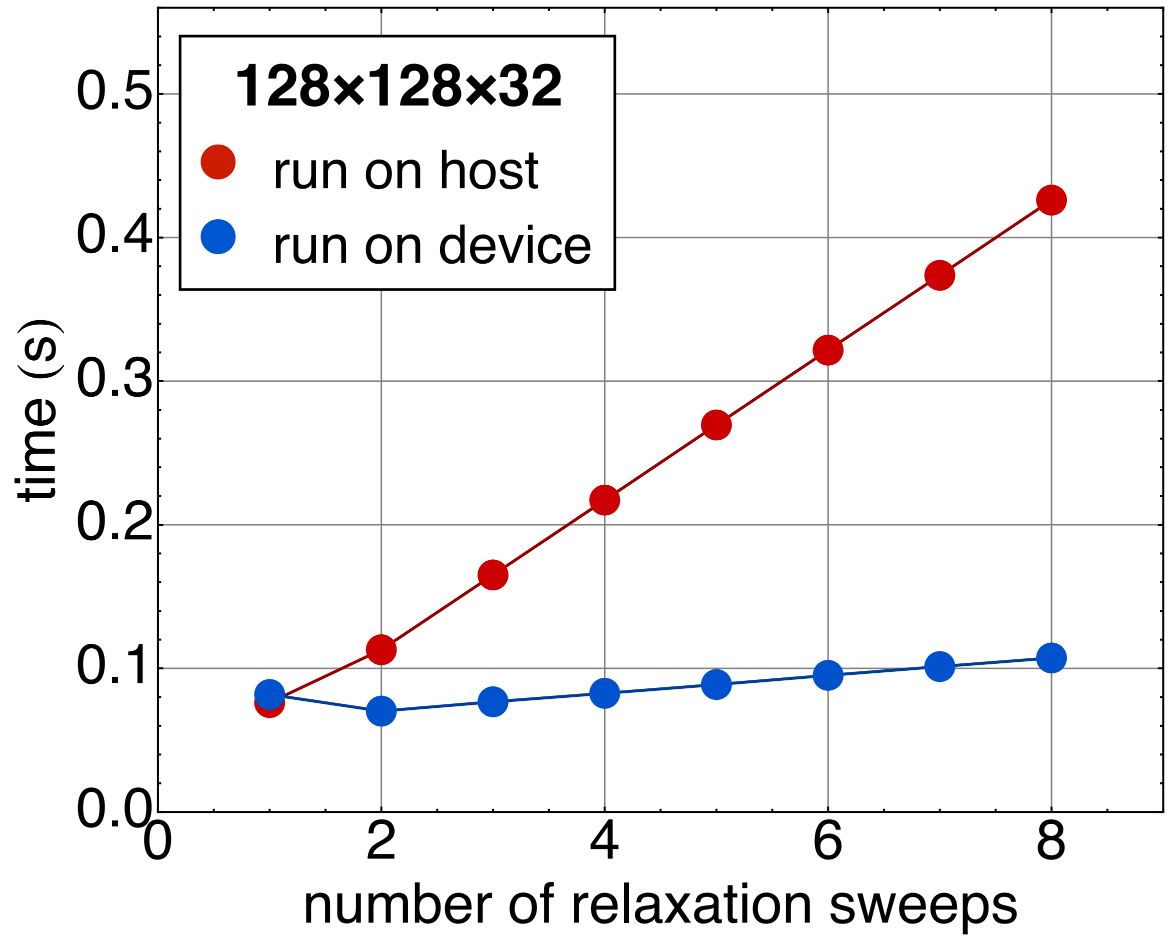
Multigrid on the accelerators. The ideal best case with no MPI communication.

- Initially we can suppose no MPI communication was necessary. (Note that this gives the wrong answer.) Add a few OpenACC directives.
- What speed-up can we expect running code on host vs. accelerator?

```
   SUBROUTINE mltgrd2D_rlx (lvl,itermax,im0,jm0,km0,nsdm0,area,wght,beta,alph)

!$acc data copyin (om1,om2,area,wght,beta) create (tmpry,work) copy (alph)

   DO iter = 1,itermax ! number of sweeps

!$acc kernels

      Relaxation Sweep

!$acc end kernels

   ENDDO ! iter

!$acc end data

   END SUBROUTINE mltgrd2D_rlx
```

Multigrid on the accelerators. The ideal case with no MPI communication.

- Loading (unloading) the appropriate modules on the xk nodes, we can toggle to run on host or accelerator.
- We can see the latency associated with transfer of data from the host to the accelerator through the PCI express.
- But, when data is on the accelerator, it is very fast. Blue line very flat. Amazing.
- Typically 3 or 4 sweeps are optimal. So, 2.5× speed-up.



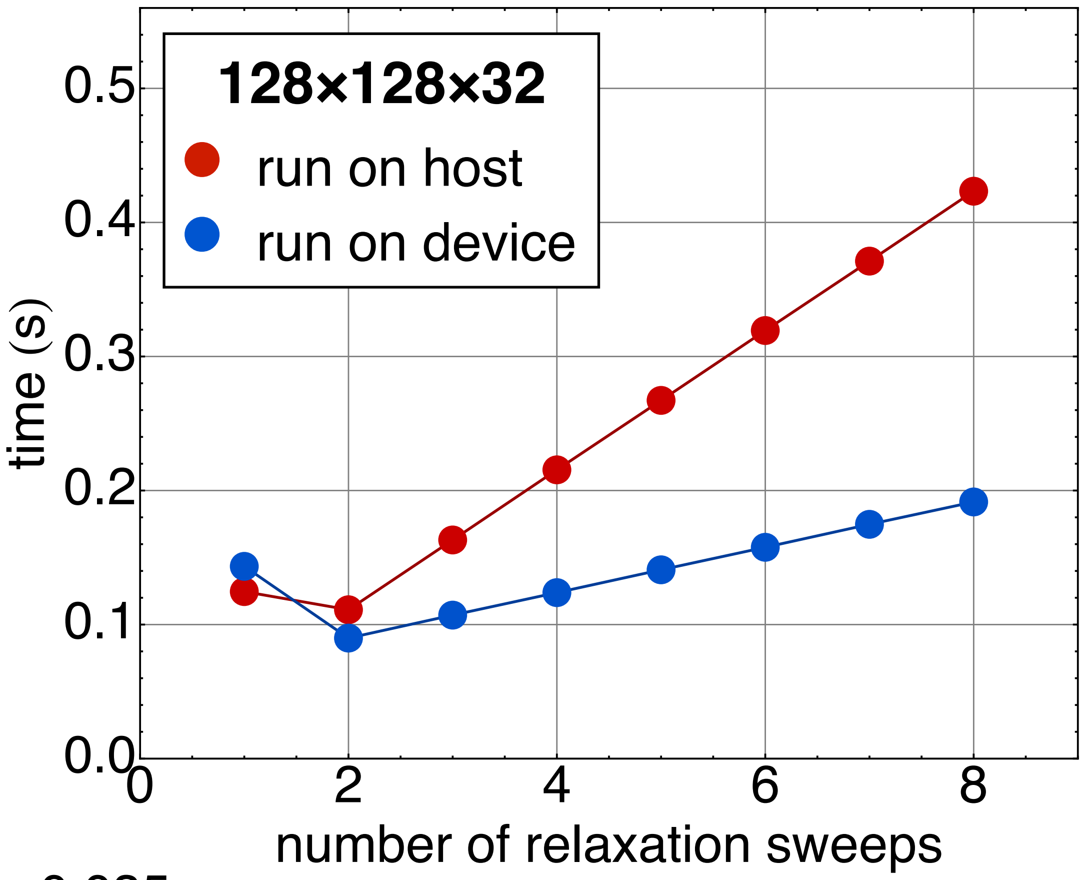
Multigrid on the accelerators. With MPI communication.

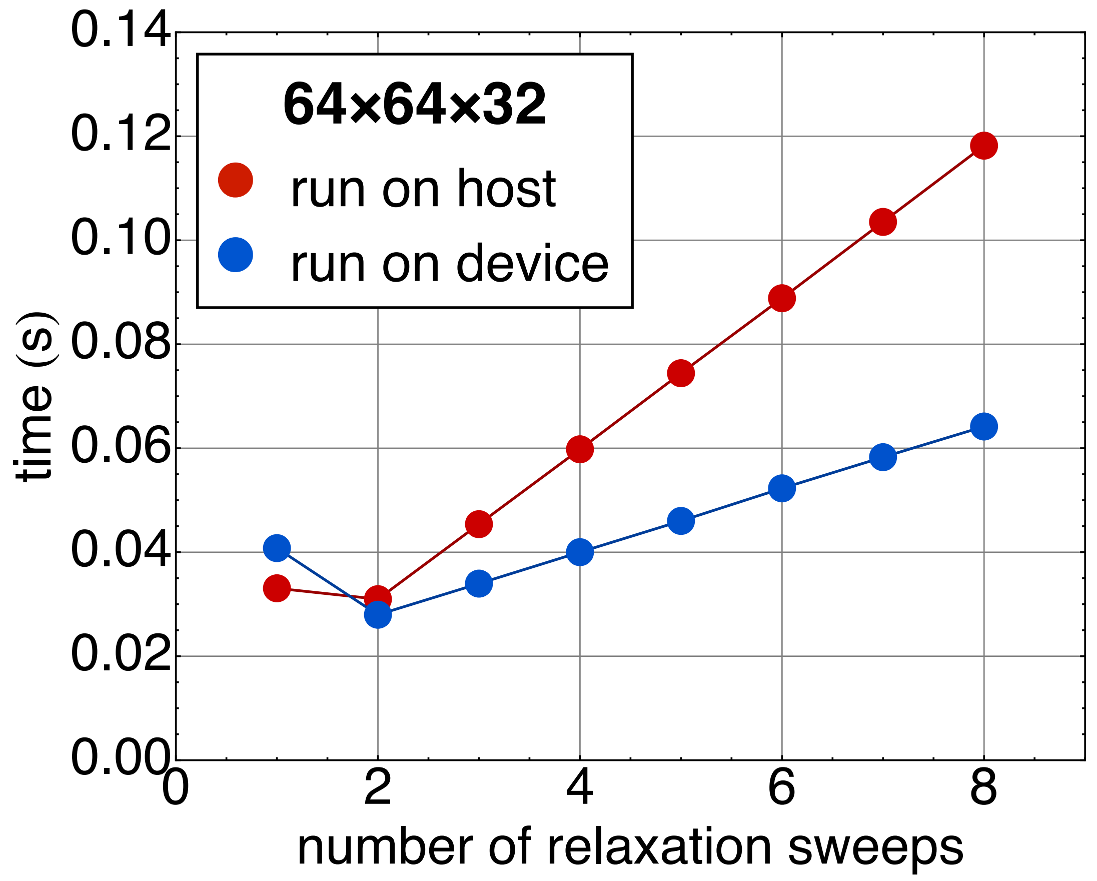
- Now we include the MPI and use the **`!$acc update`** directive:

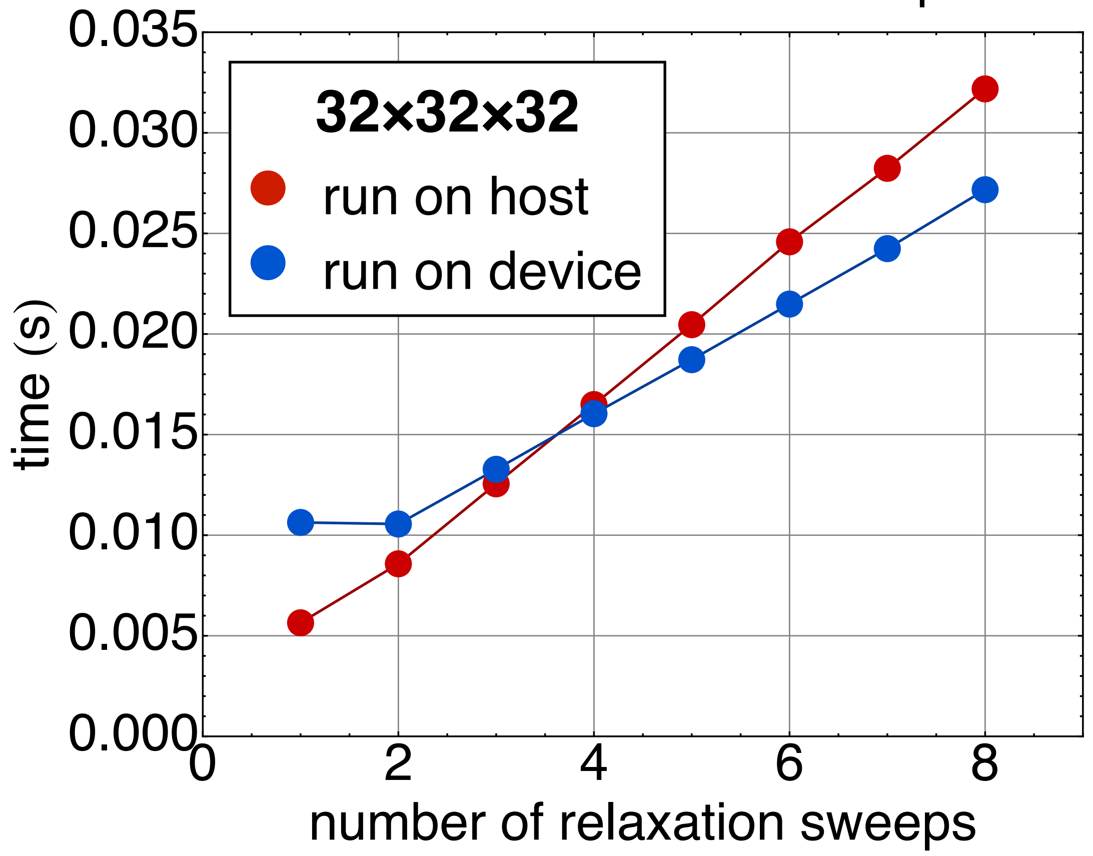
```
      SUBROUTINE mltgrd2D_rlx (lvl,itermax,im0,jm0,km0,nsdm0,area,wght,beta,alph)

!$acc data copyin (om1,om2,area,wght,beta) create (tmpry,work) copy (alph)

   DO iter = 1,itermax ! number of sweeps

        MPI communication

!$acc update device (alph(1:im0-1, 1      ,:,:))
!$acc update device (alph(  im0   ,1:jm0-1,:,:))
!$acc update device (alph(2:im0   ,  jm0   ,:,:))
!$acc update device (alph(   1    ,2:jm0   ,:,:))

!$acc kernels

        Relaxation Sweep

!$acc end kernels

!$acc update host (alph(2:im0-2, 2      ,:,:))
!$acc update host (alph(  im0-1,2:jm0-2,:,:))
!$acc update host (alph(3:im0-1,  jm0-1,:,:))
!$acc update host (alph(   2   ,3:jm0-1,:,:))

   ENDDO ! iter

!$acc end data

   END SUBROUTINE mltgrd2D_rlx
```

Multigrid on the accelerators. With MPI communication.

- The speed-up depends on the block size. Less speed-up on smaller blocks.
- This will become an issue on coarser grid resolution within the multigrid v-cycle.
- Coarser grids may run exclusively on the host

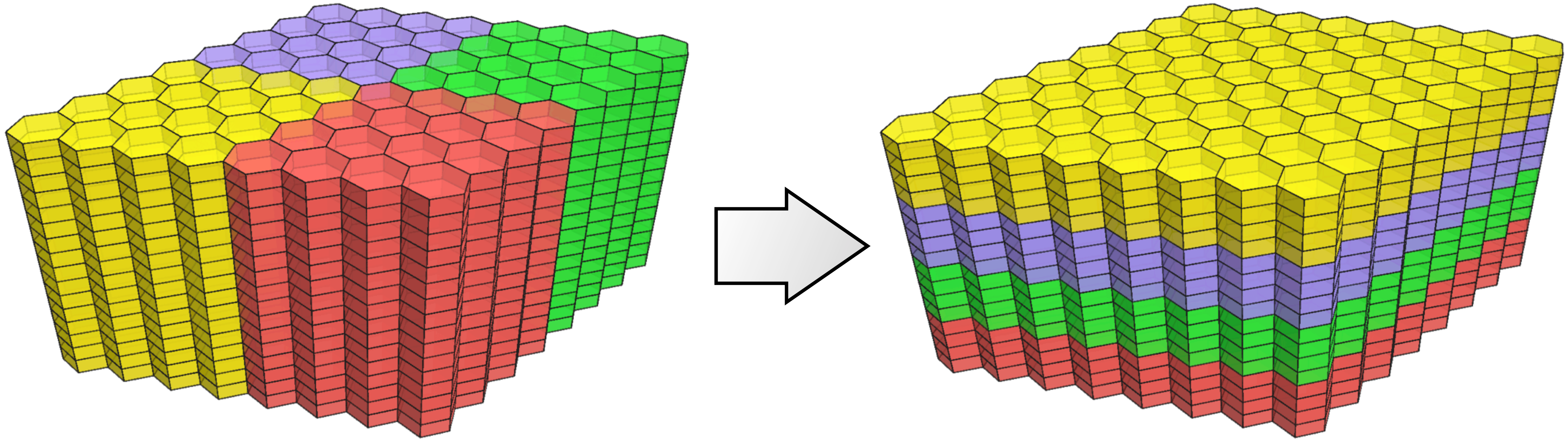






Multigrid on the accelerators. Possible solution.

- Transpose the model blocks and columns so that blocks become bigger.
- Hopefully the speed-up will outweigh the additional communication.



Each task has a 4×4×16 block

Each task has a 16×16×4 block

Summary.

- Still some things to figure out with MPI scaling.
- The multigrid algorithm is somewhat limited by the amount of useful work done per MPI communication.   Possible solutions:
  - Data transpose
  - Asynchronous work. Host performs MPI communication while simultaneously the accelerator is doing relaxation.
  - Duplicate some calculation on the host and accelerator to avoid the need for OpenACC updates every relaxation sweep.