Trials and Tribulations (and Joys) of Developing with Alternative Parallel Frameworks

Development of the Intermediate Complexity Atmospheric Research model and others

Ethan Gutmann

National Center for Atmospheric Research

Soren Rasmussen, Damian Rouson, and many others

November 13, 2023

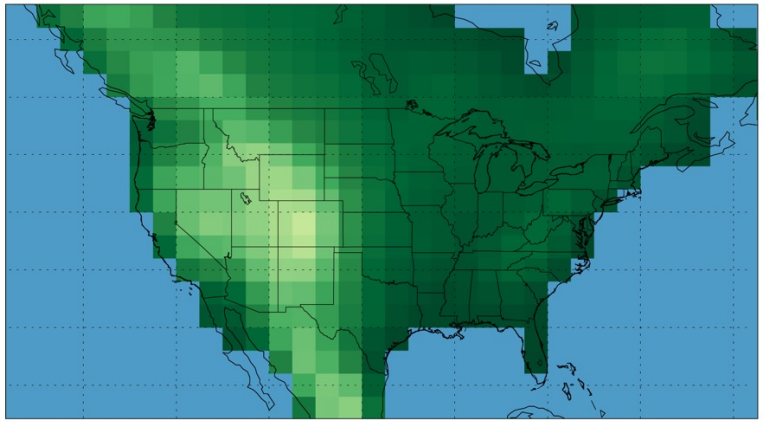
This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

What I'm going to say

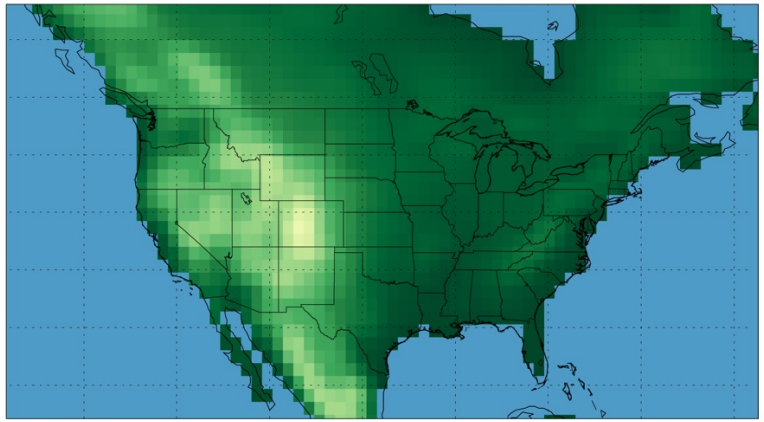
- Actionable use of Climate Projections
- Computing requirements / limitations
- Use of Coarray Fortran for rapid development
  - ICAR
  - SnowModel
- Scaling results
- Complications with Compilers and Computers
- Where might Parallel Applications be going?

Aren’t climate models useful already?

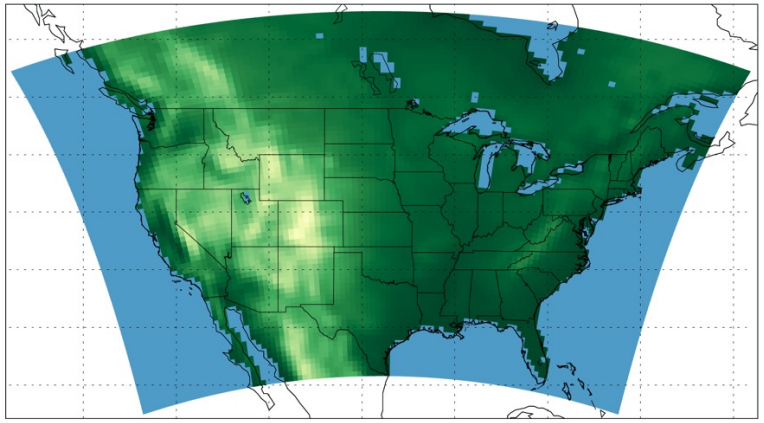
GFDL -CM3 2.5x2.0



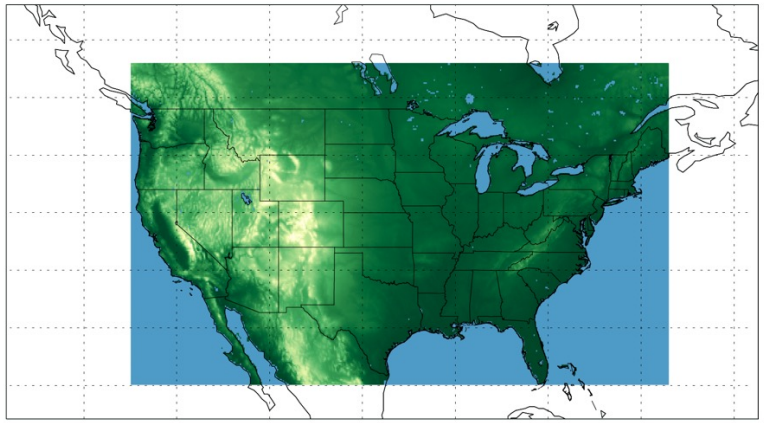
MRI-CGCM3 1.1x1.1



WRF 50x50 km



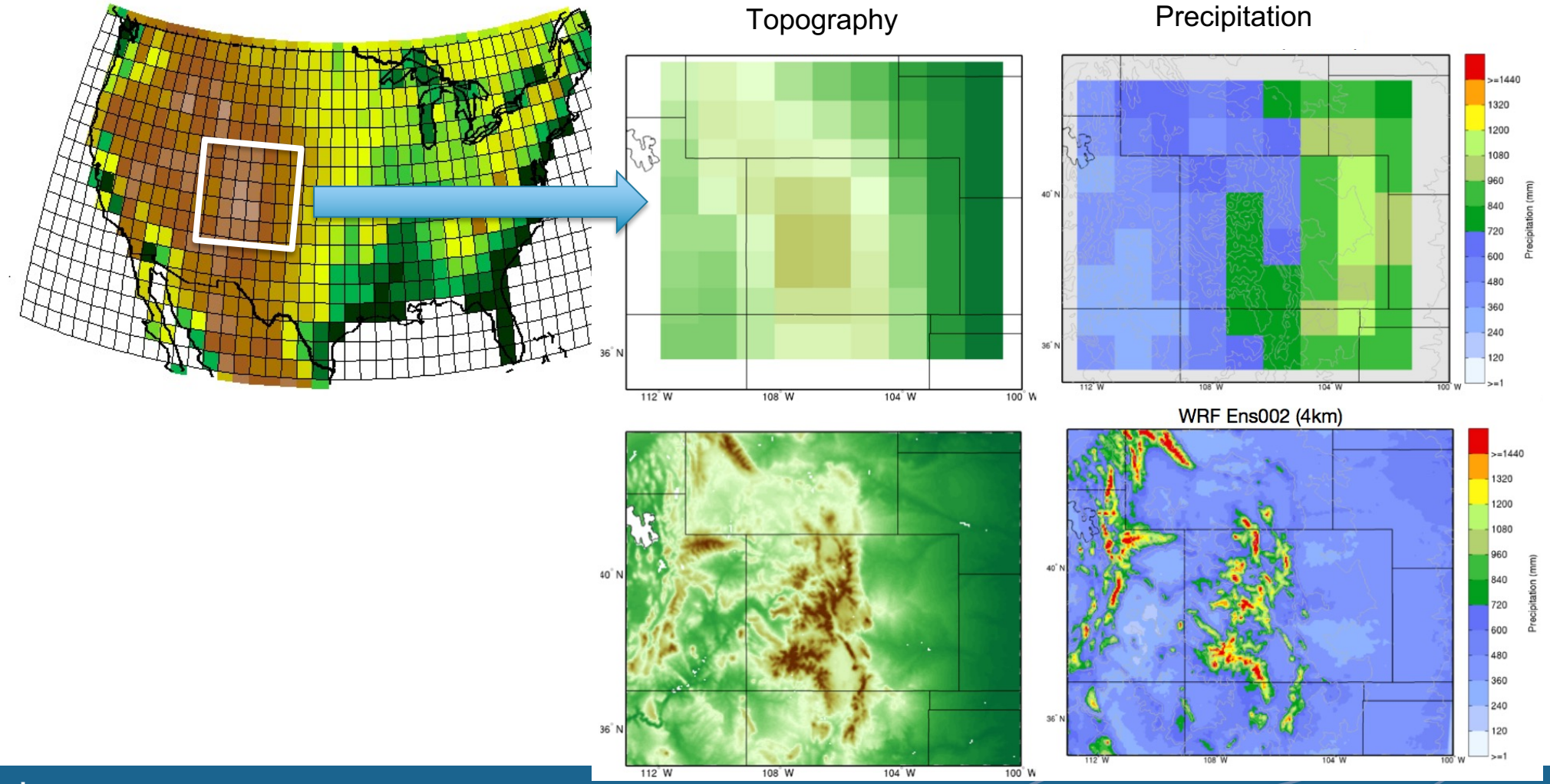
Reference 12x12 km



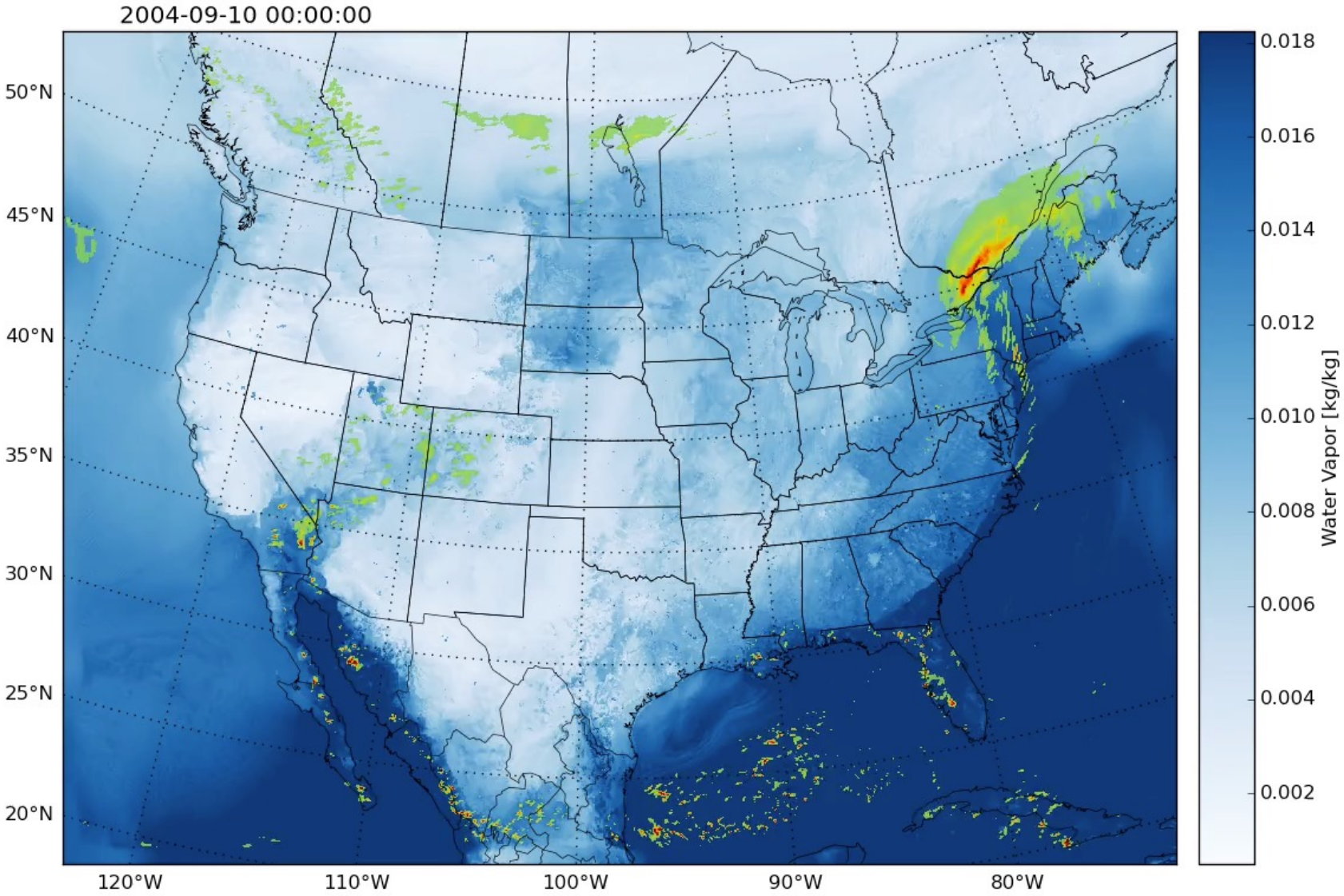
Terrain height (m)

NCAR | UCAR

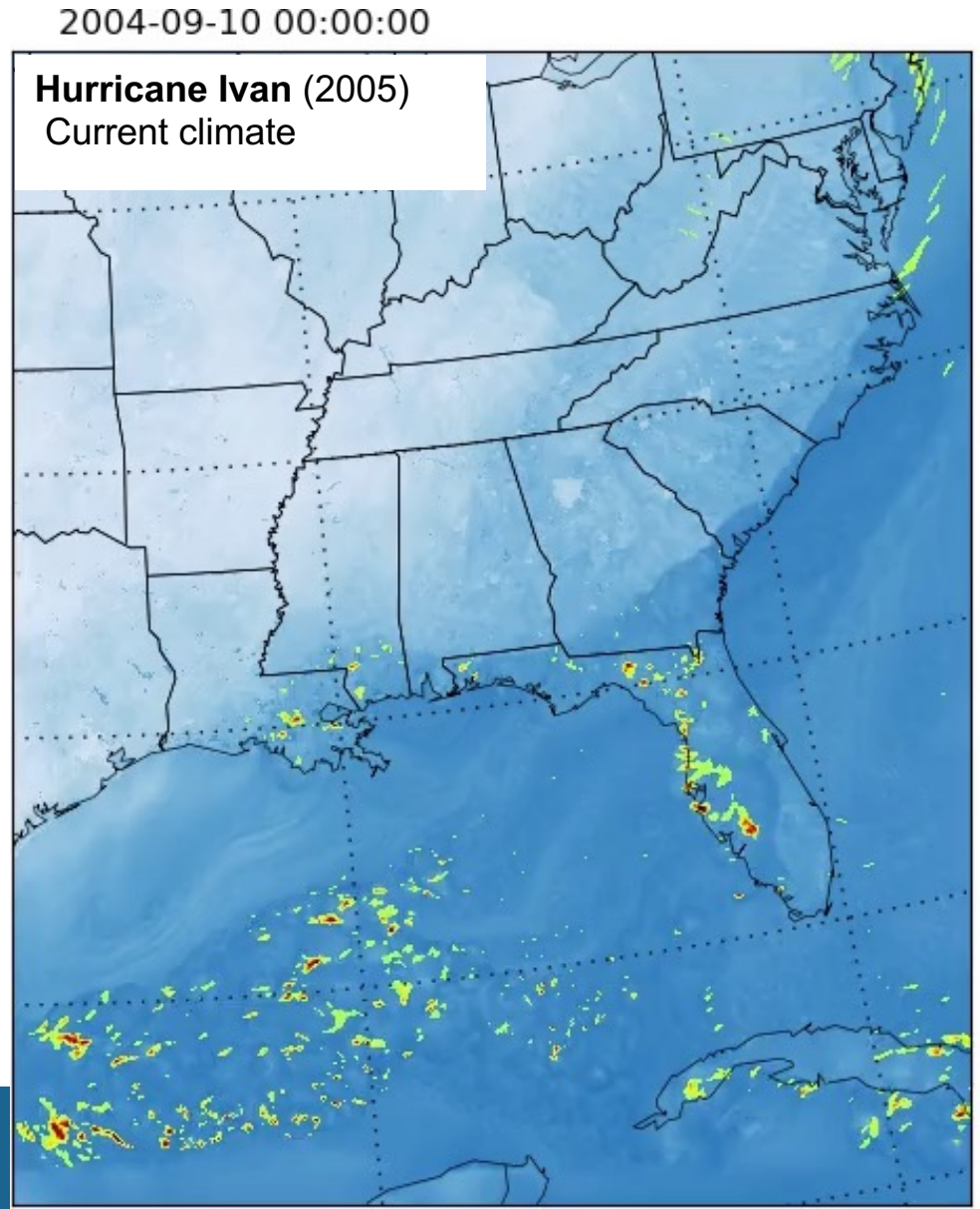
Climate Model Native Resolution and Application Resolution

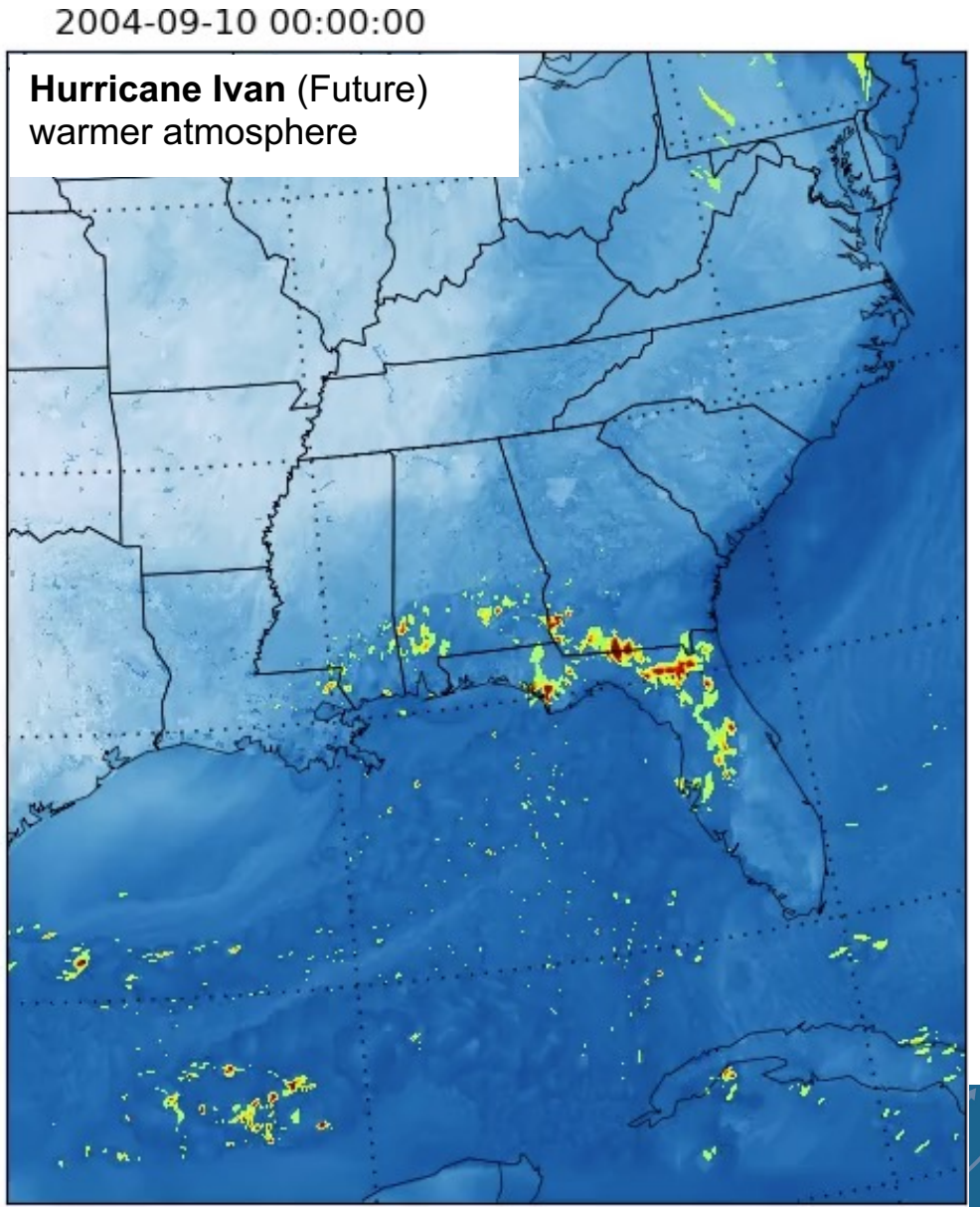


Dynamical Downscaling at Convection Permitting Scales



Hurricane Ivan





Gutmann et al. (2018)

A dichotomy of regional climate approaches

False

increasing physical representation

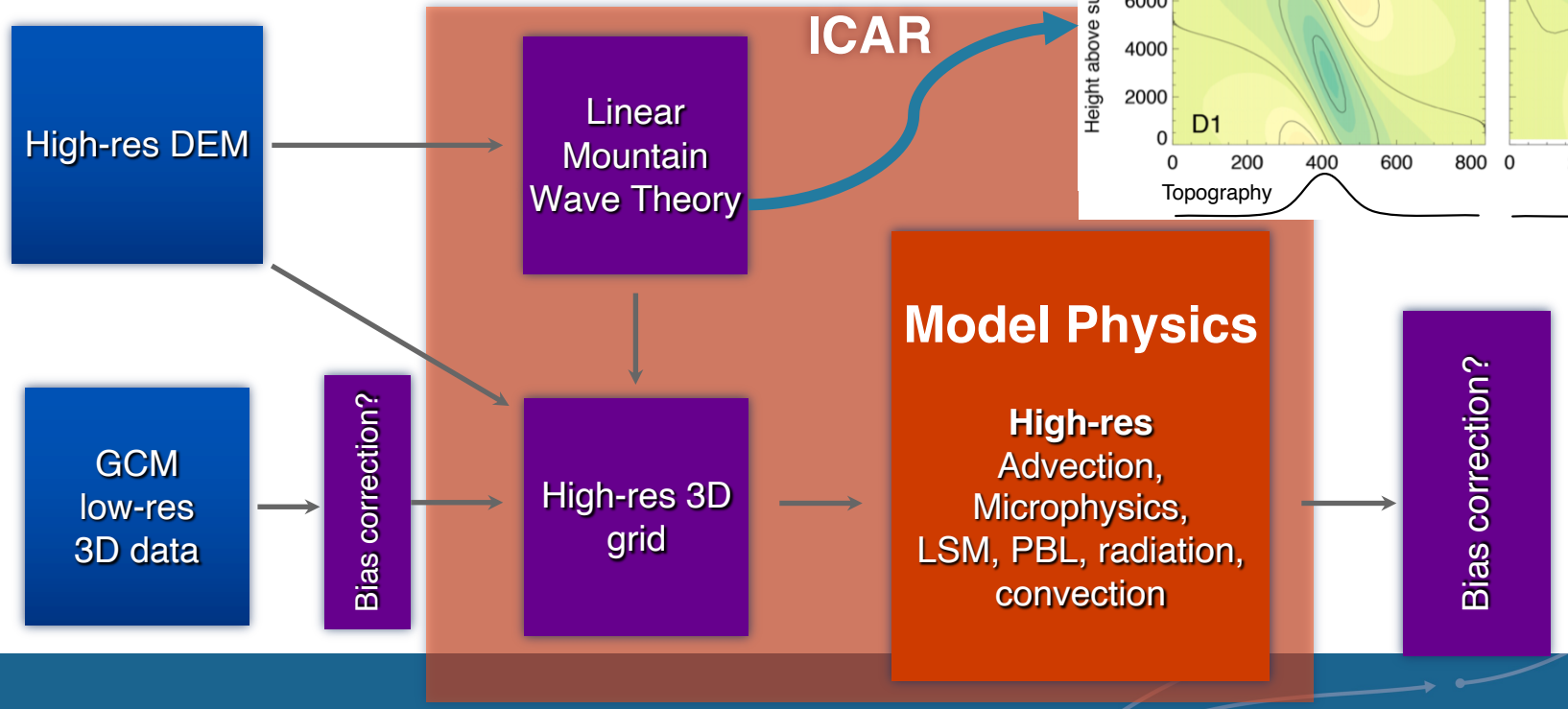
- Statistical downscaling based on rescaling GCM outputs
  - BCSD, BCCA, ARRM, LOCA

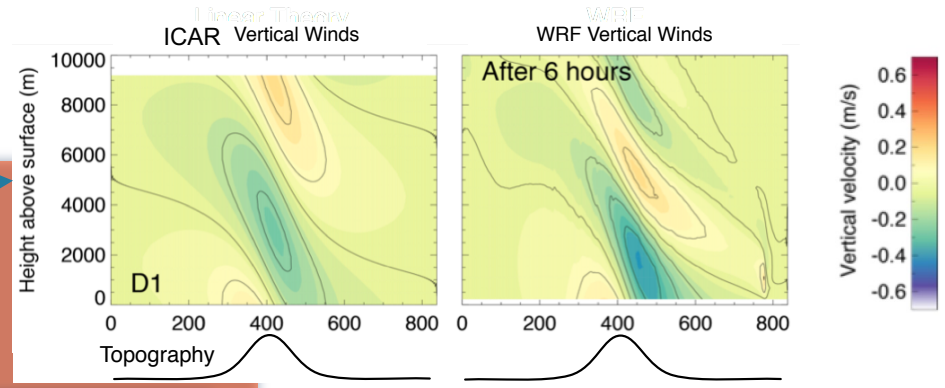
- Dynamical downscaling using state-of-the-art RCMs

Intermediate Complexity Atmospheric Research model (ICAR)

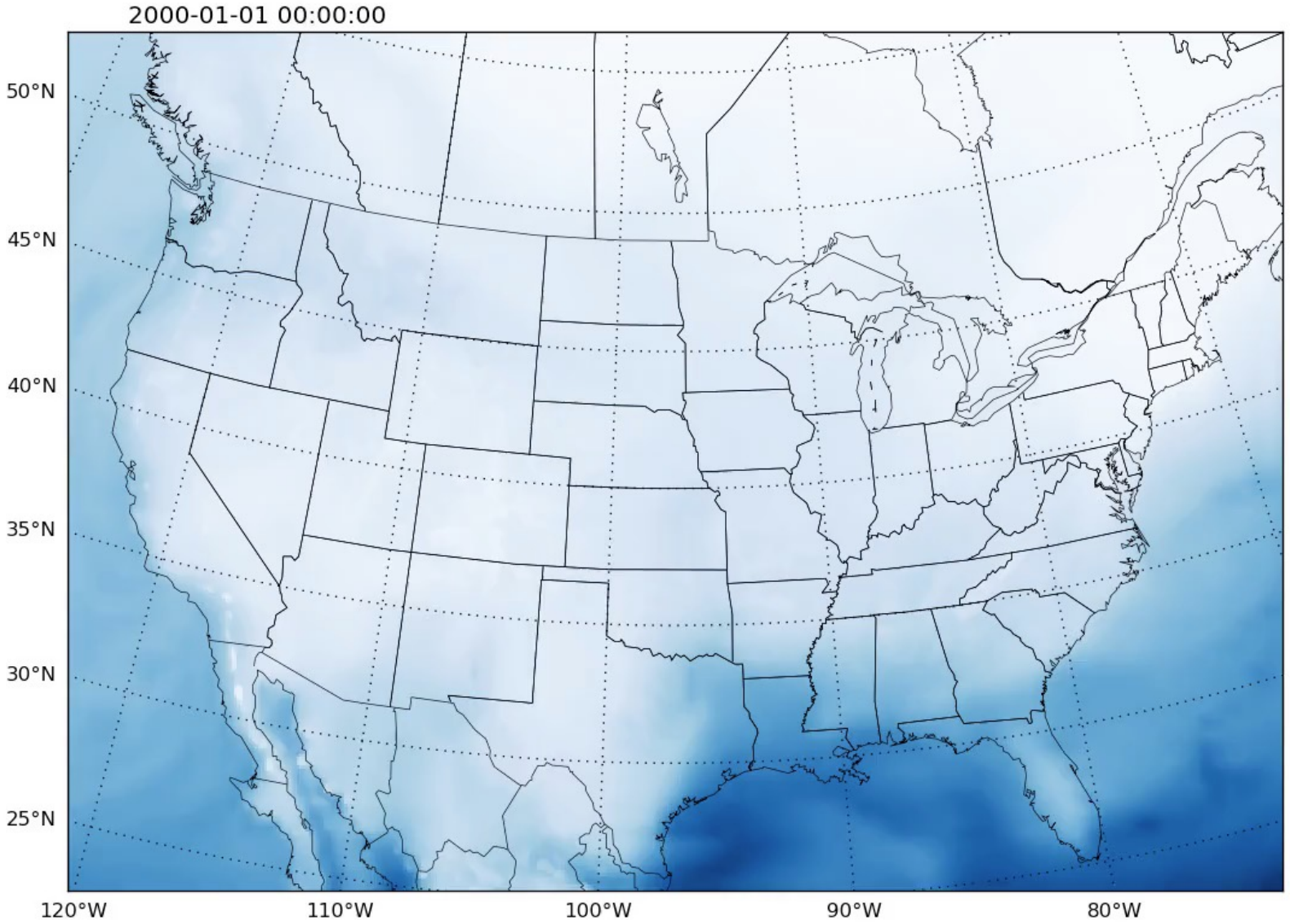
Identify the key physics and develop a simple model
GOAL: >90% of the information for <1% of the cost

http://github.com/NCAR/icar





ICAR simulation

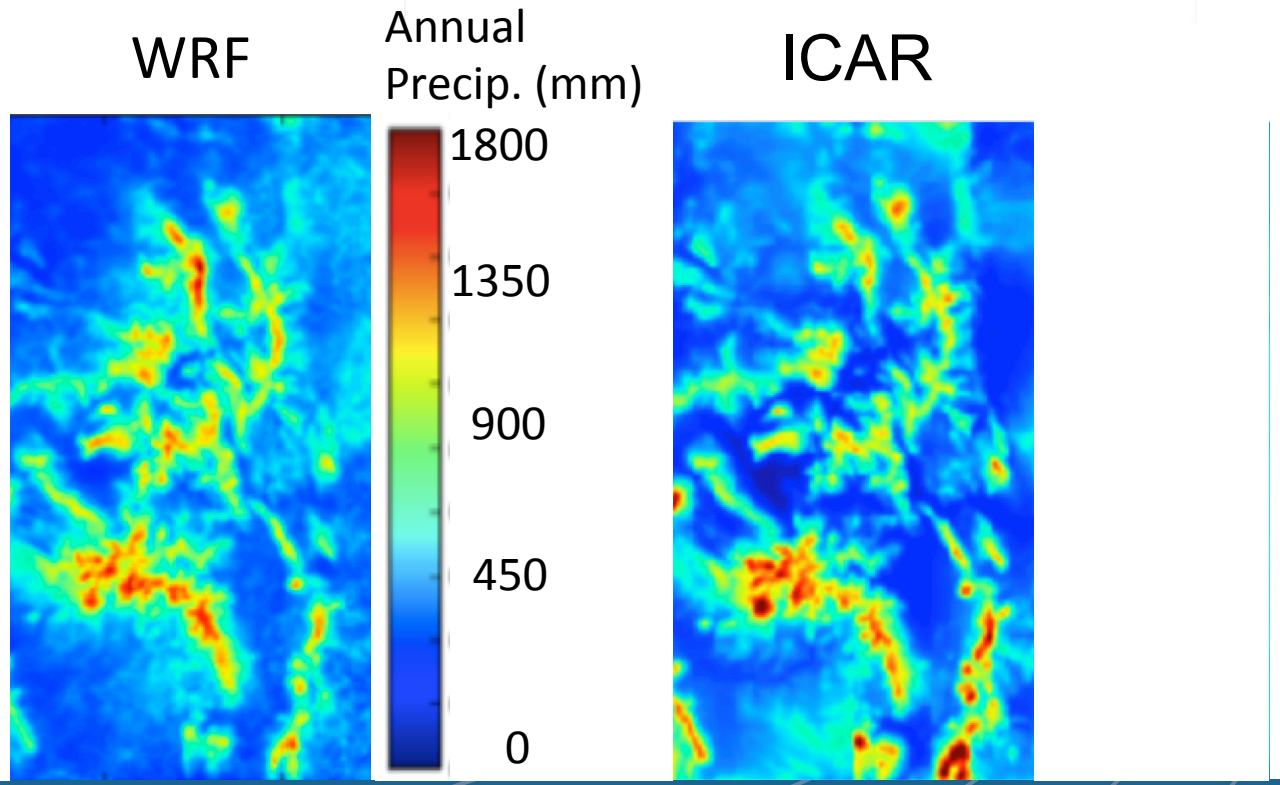


ICAR Precipitation

WRF and ICAR have very similar precipitation distributions.

ICAR requires ~1% of the computational effort of WRF.

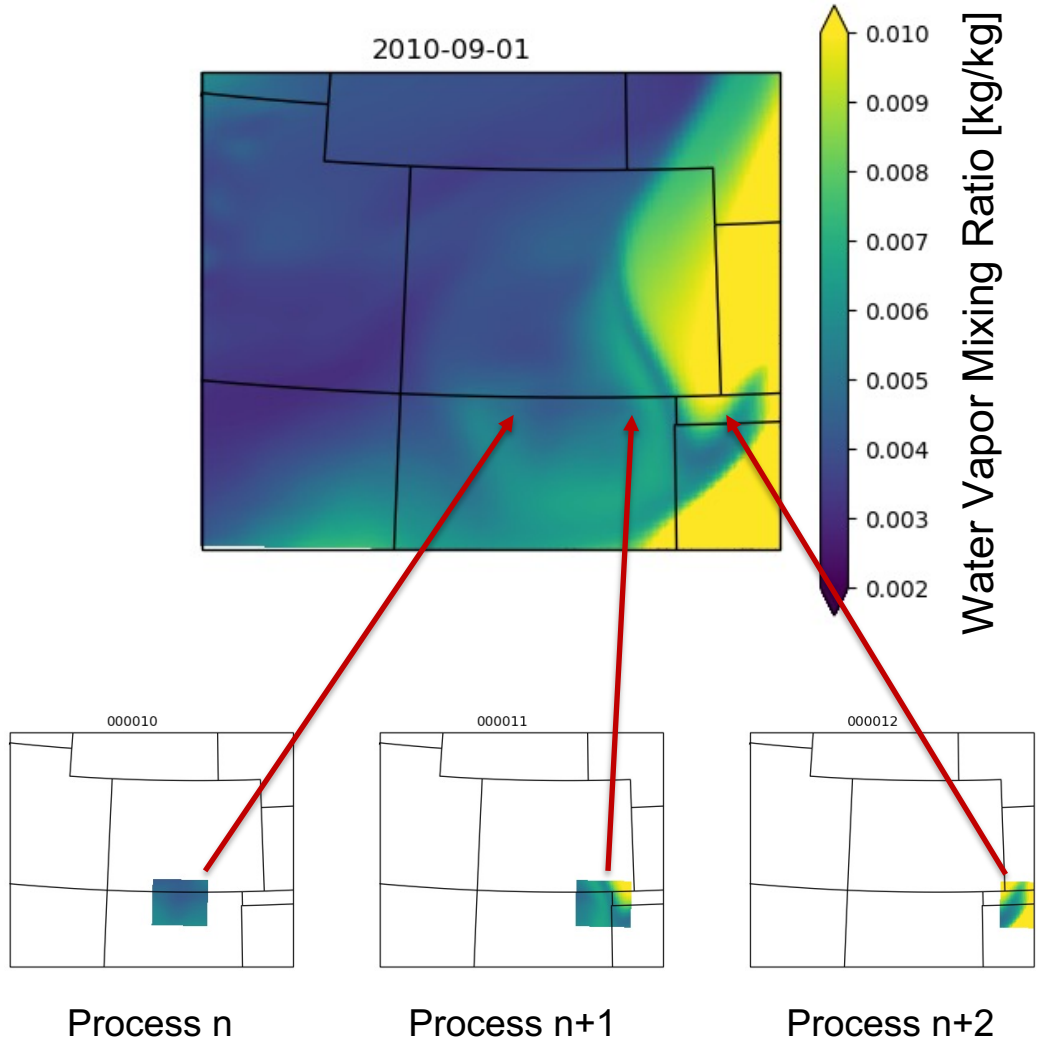
This enables a quasi-dynamical downscaling for a wide variety of GCM / scenario combinations



Partitioned Global Address Space

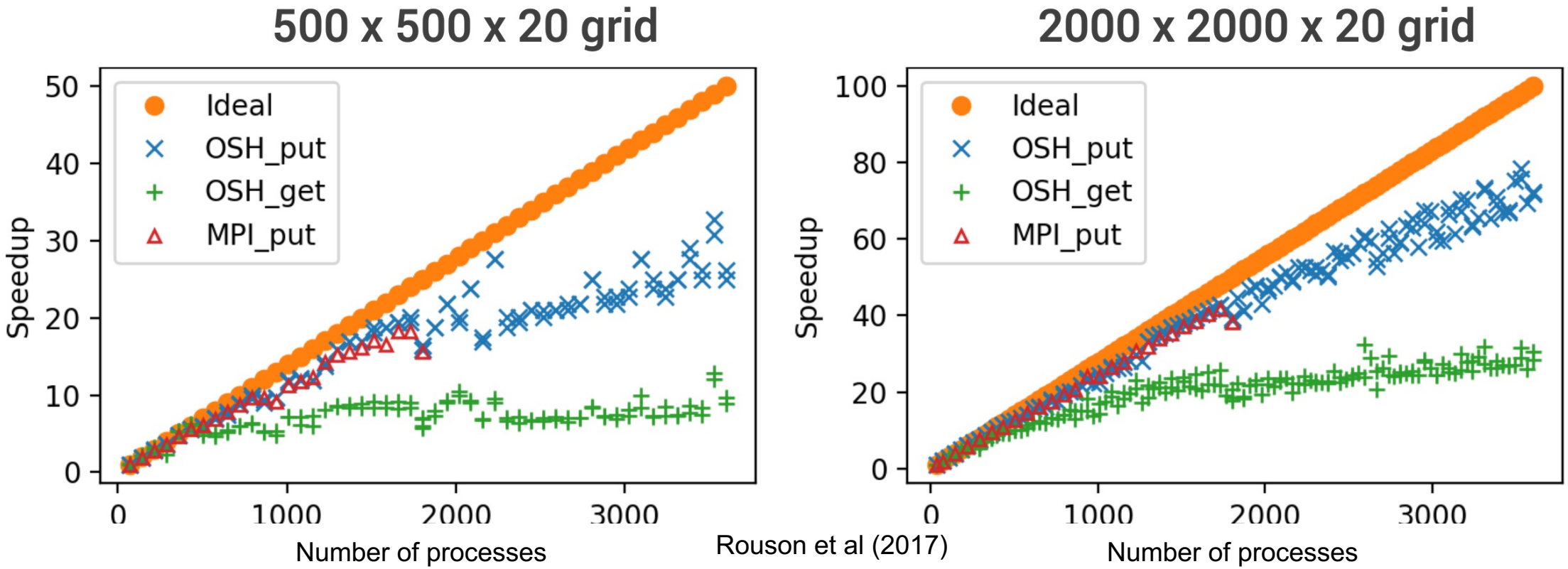
- Ideally, all variables would be coarrays
- Coarrays have costs and restrictions
- “Exchangeable” objects instead contain local data arrays and PGAS buffer coarrays

- Per time step:
  - Process edge grid cells
  - Send edge grid cells to neighbors **asynchronously**
  - Process inner grid cells
  - Sync with neighbors to update neighboring edges



Simple Scaling Results

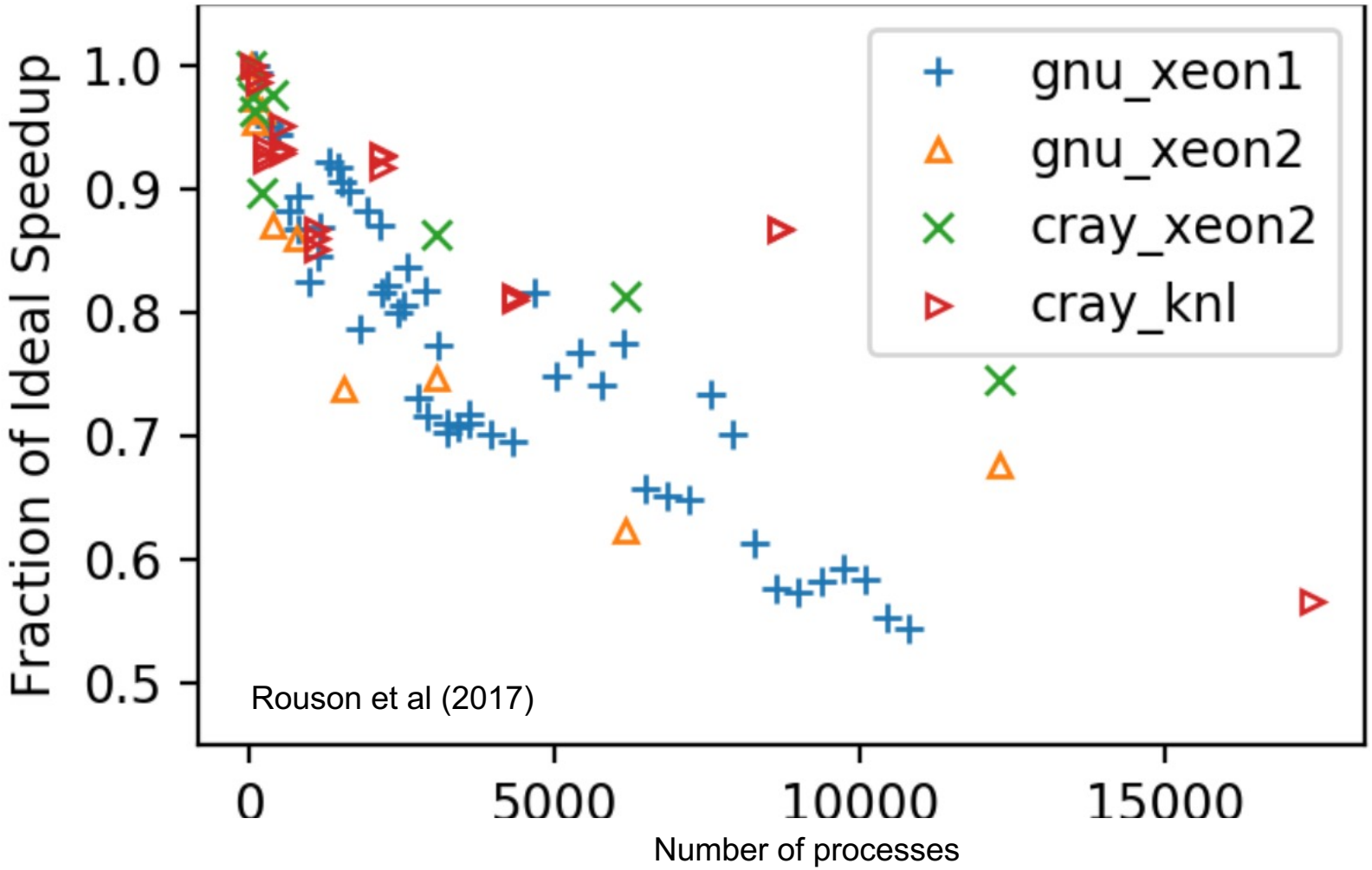
- Reduced syncs speeds up code
- Coarray fortran can leverage MPI, OpenSHMEM, GasNet,…



Rouson et al (2017)

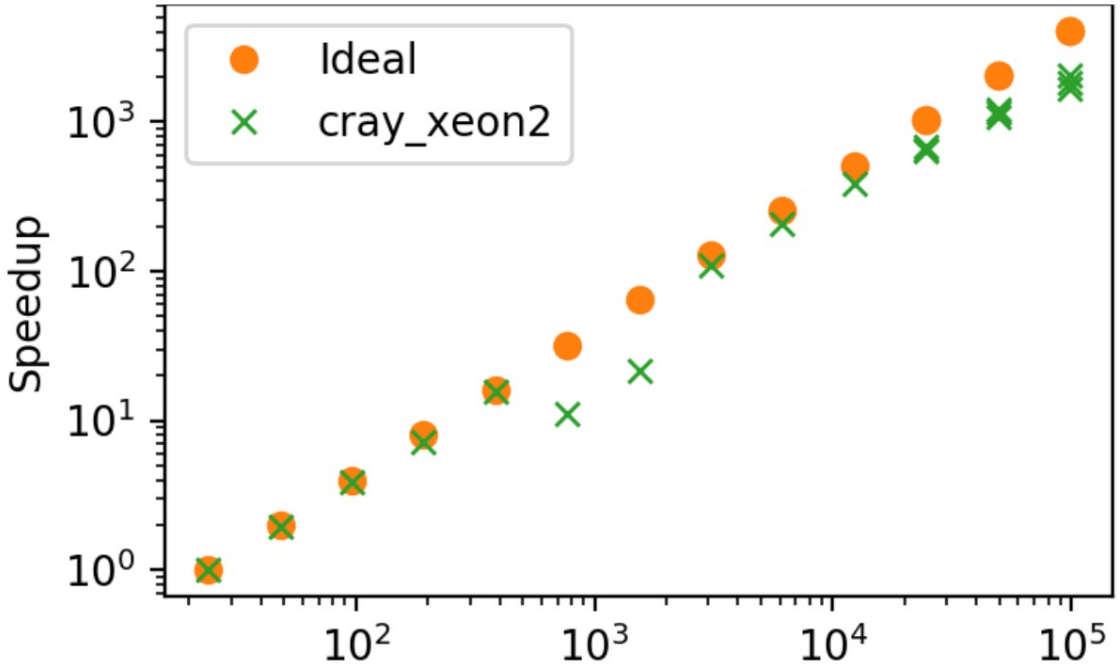
Scaling Across Computers

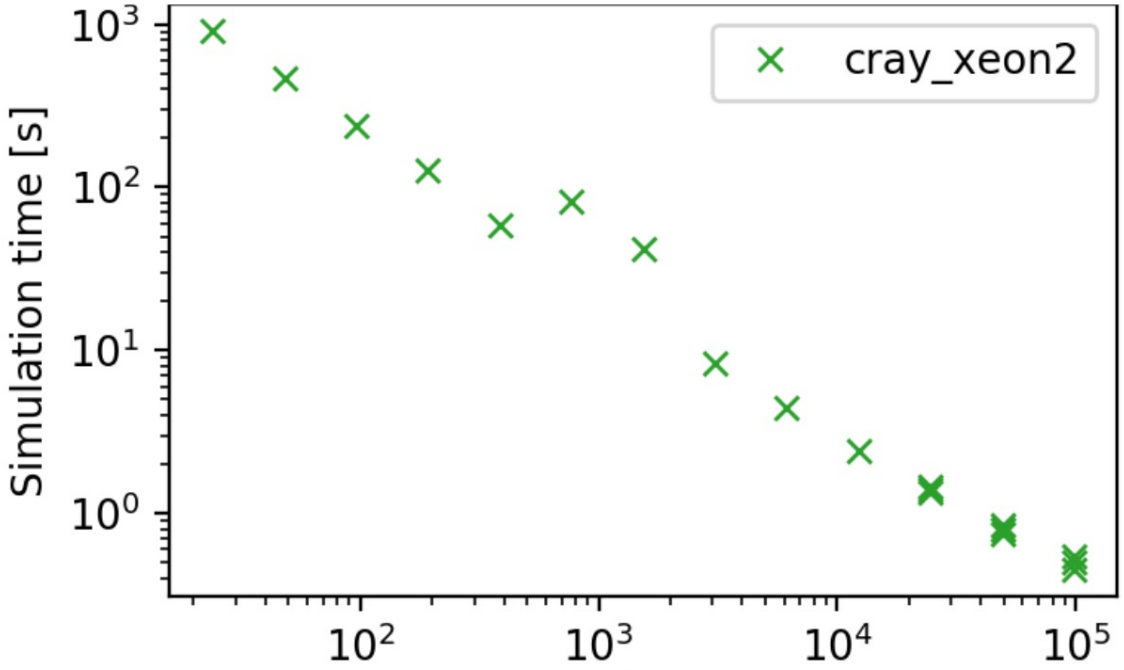
- Ideal = 1
- Cray compiler consistently outperforms GNU / opencoarrays
- Differences between computers (orange vs blue) smaller than between compilers (orange



Performance at Scale!

- Coarray fortran can scale extremely well
- Simplified code, physics only, no I/O…

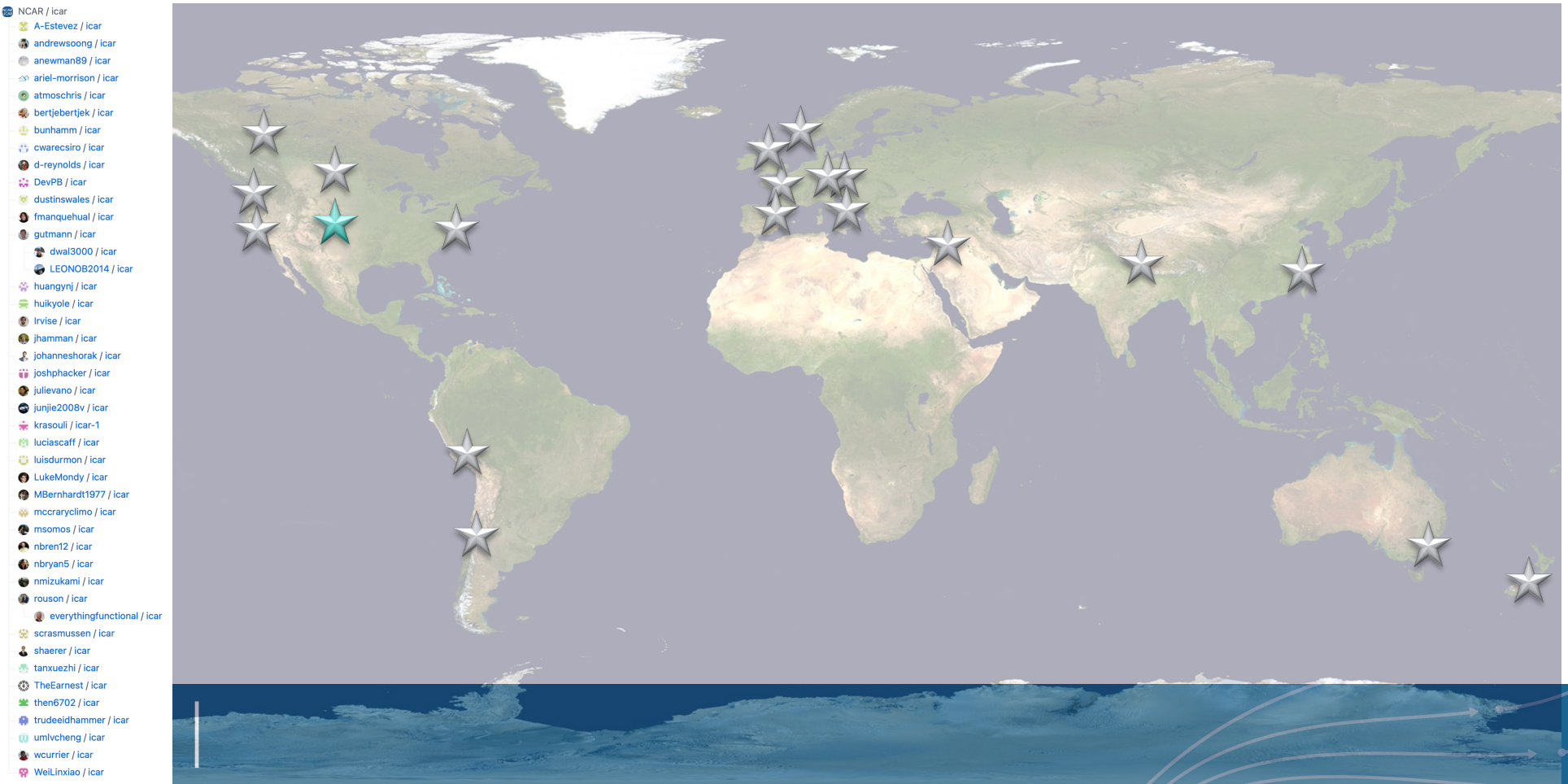




Rouson et al (2017)

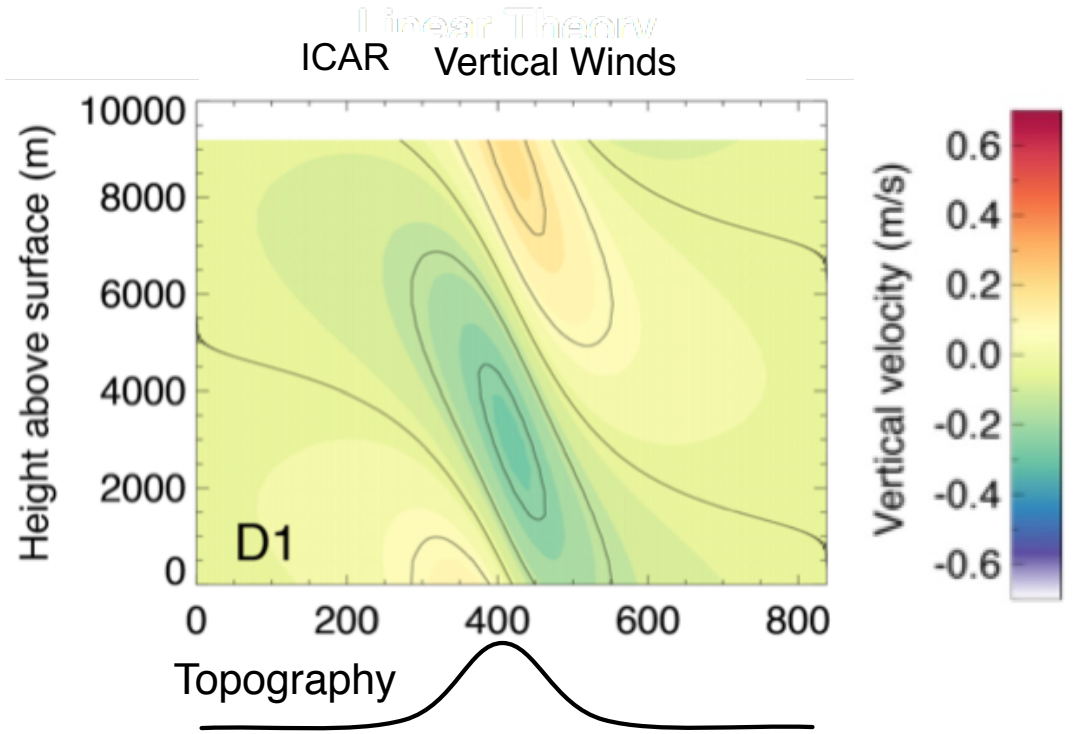
ICAR Users vs their Computers

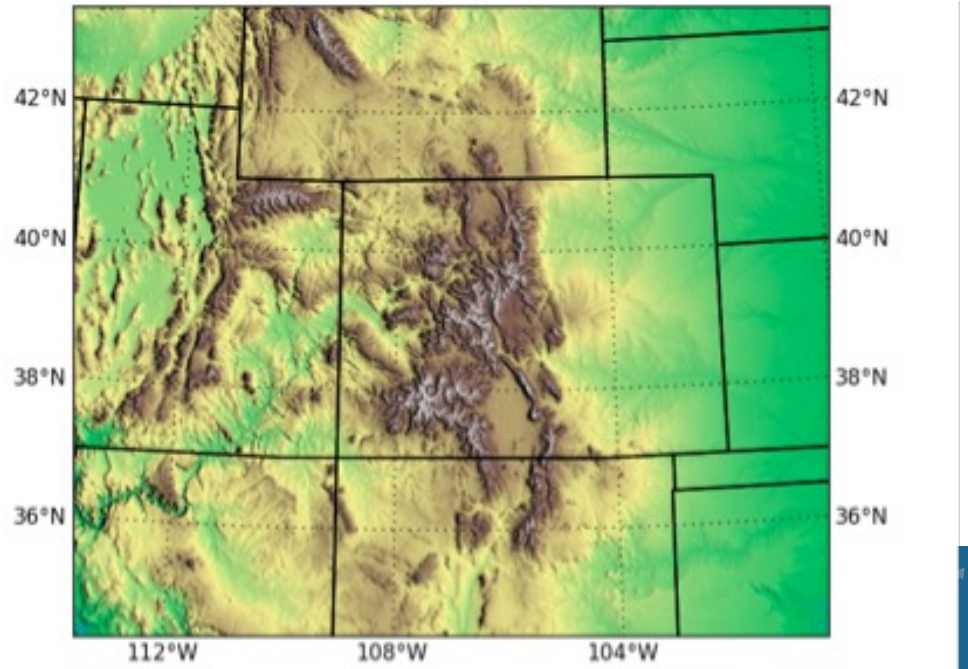
Ethan Gutmann
National Center for Atmospheric Research



Linear Wind Analytical Solution

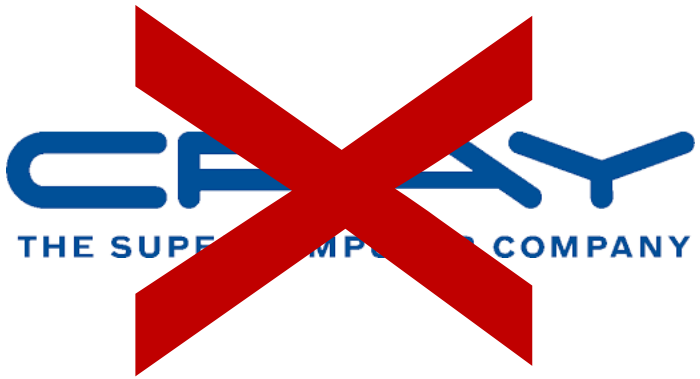
- Linear theory solution uses a global domain FFT
- Still expensive at run-time
- Create an enormous Look-up-Table
  - Typical LUT =
  - 500 x 500 x 20 x 2 x 36 x 10 x 10 = 134GB
  - nx * ny * nz * 2-vectors * ndirections * nspeeds * nstabilities
- In serial creation of LUT exceeds wall clock
- Great case for PGAS
  - Each process calculates a portion of the LUT over the global domain and sends the sub-domains to all other processes

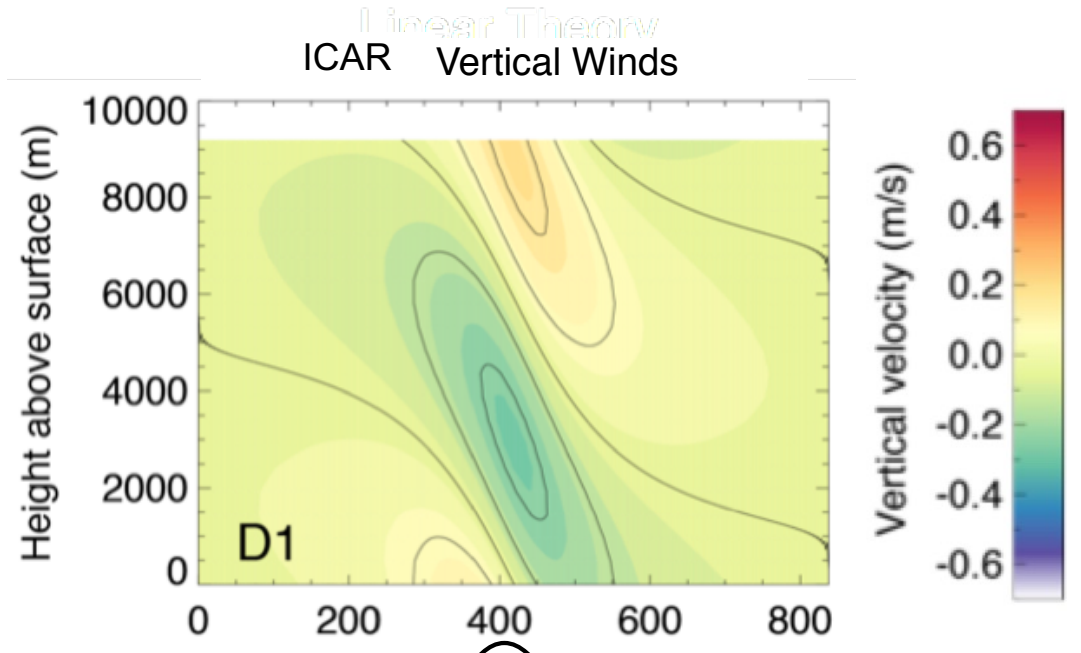


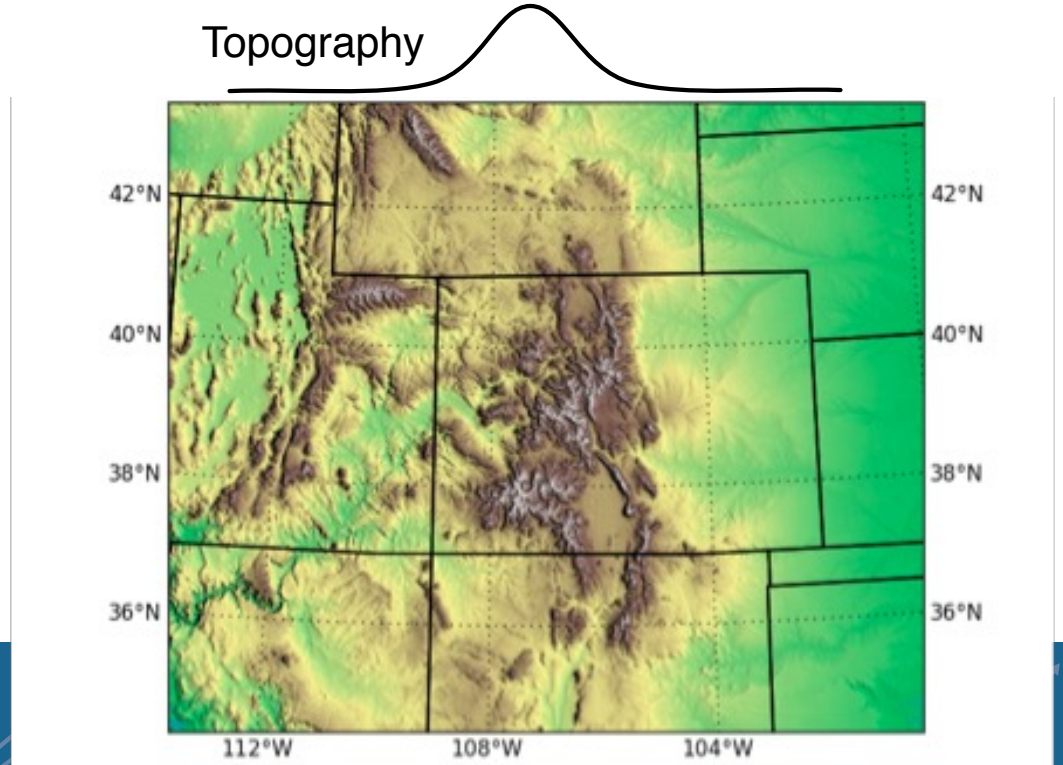


Linear Wind Analytical Solution

- Coarray LUT works great on GNU
- Crashes on allocation with Cray







Original Lookup Table (LUT)

```
subroutine copy_data_to_remote(wind, grids, LUT, i, j, k, z)
    implicit none
    real,           intent(in)  :: wind(:,:)
    type(grid_t),   intent(in)  :: grids(:)
    real,           intent(inout):: LUT(:,:,:,:,:,:)[*]
    integer,        intent(in)  :: i, j, k, z
    integer :: img

    do img = 1, num_images()
        associate(ims => grids(img)%ims, &
                  ime => grids(img)%ime, &
                  jms => grids(img)%jms, &
                  jme => grids(img)%jme)
        LUT(k,i,j, 1:ime-ims+1, z, 1:jme-jms+1)[img] = wind(ims:ime,jms:jme)
        end associate
    enddo
end subroutine copy_data_to_remote
```

No synchronization required!

Issue: one large coarray requires too much memory crashed Cray compiled code
Does Cray use special memory for coarrays allocated on NIC or other?

Non-Coarray LUT: step 1

Split large coarray into smaller buffer. Also need to send LUT indicies for each
First send then sync

```
! each image will communicate the wind values it has calculated for an ijkz
! location to the other images, the ijkz values are also sent so the receiving
! images know where to put the incoming values
do img = 1, num_images()
    associate(ims => grids(img)%ims, &
              ime => grids(img)%ime, &
              jms => grids(img)%jms, &
              jme => grids(img)%jme  &
        )
    LUT_co(1:ime-ims+1, 1:jme-jms+1, this_image())[img] = wind(ims:ime,jms:jme)

    LUT_index_co(this_image(), 1)[img] = i
    LUT_index_co(this_image(), 2)[img] = j
    LUT_index_co(this_image(), 3)[img] = k
    LUT_index_co(this_image(), 4)[img] = z
   end associate
enddo

sync all  Synchronization required!
```

Non-Coarray LUT: step 2

After sync, copy smaller coarray lookup table to large **non-coarray** table.
Need to use LUT_index variable to know where to copy

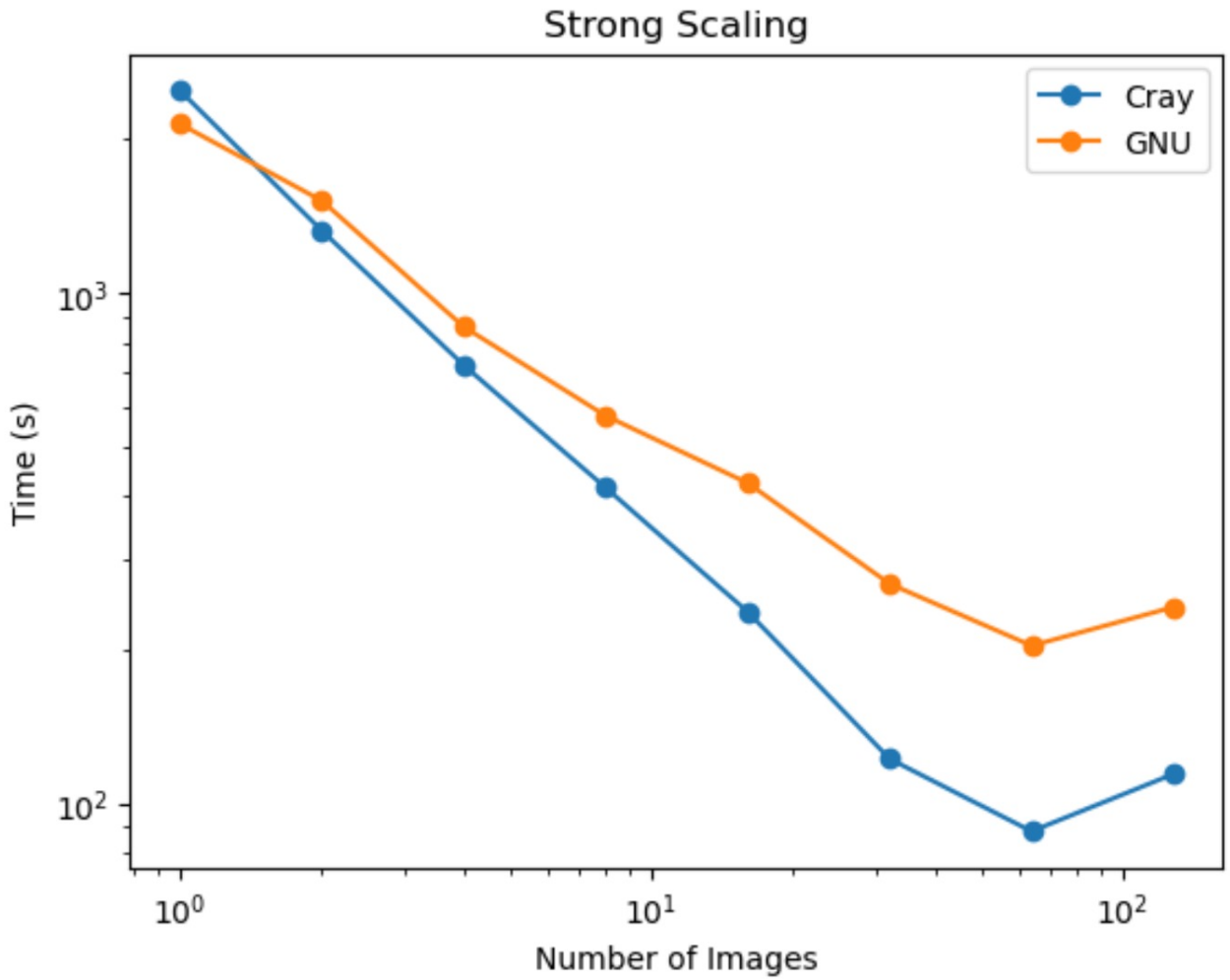
```
sync all

associate(ims => grids(this_image())%ims, &
          ime => grids(this_image())%ime, &
          jms => grids(this_image())%jms, &
          jme => grids(this_image())%jme)
    do img = 1, num_images()
        LUT_i = LUT_index_co(img, 1)
        LUT_j = LUT_index_co(img, 2)
        LUT_k = LUT_index_co(img, 3)
        LUT_z = LUT_index_co(img, 4)
        LUT(LUT_k, LUT_i, LUT_j, 1:ime-ims+1, LUT_z, 1:jme-jms+1) = &
             LUT_co(1:ime-ims+1, 1:jme-jms+1, img)
    end do
end associate
sync all
```

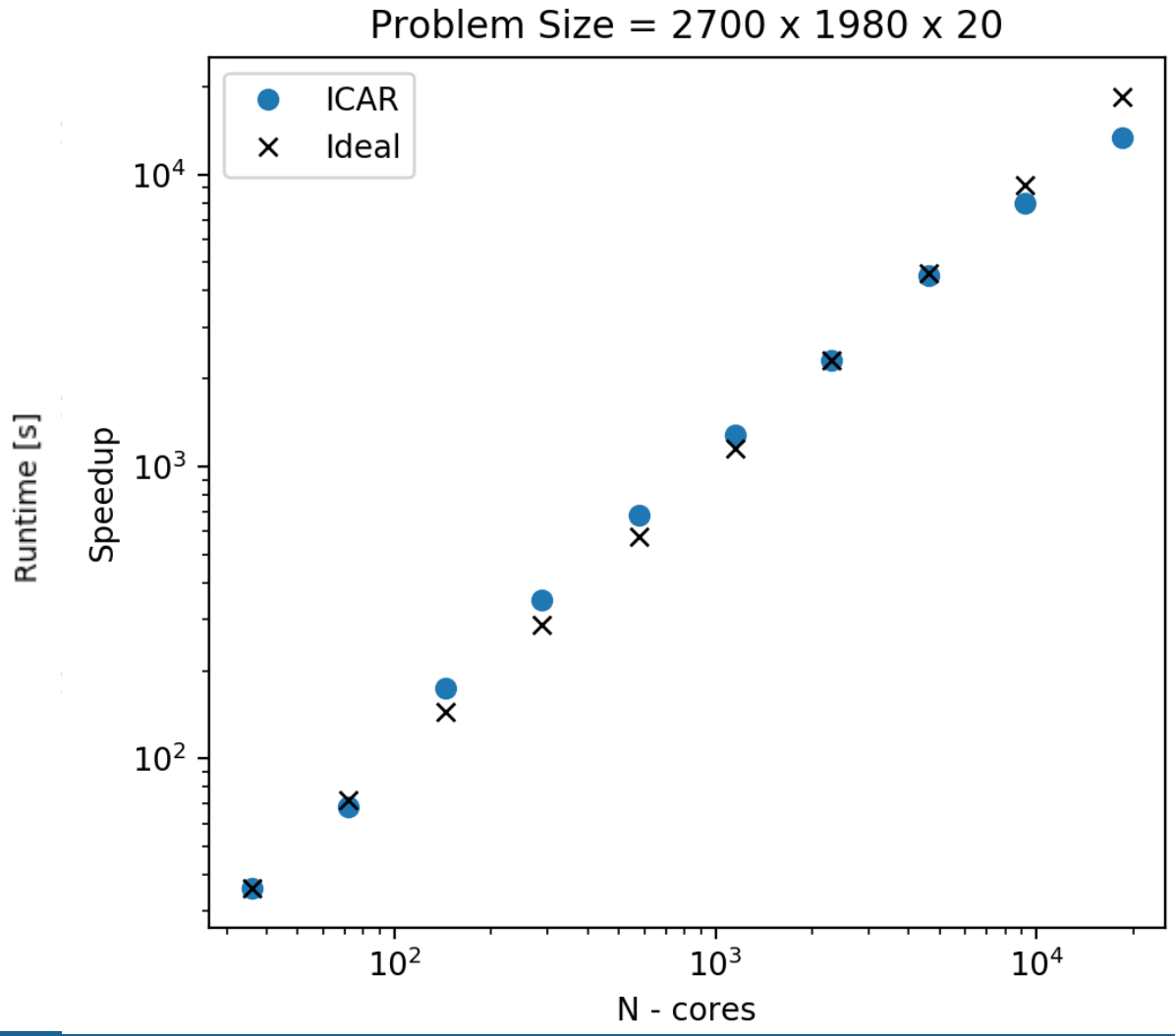
Two synchronizations required! (other approaches possible)

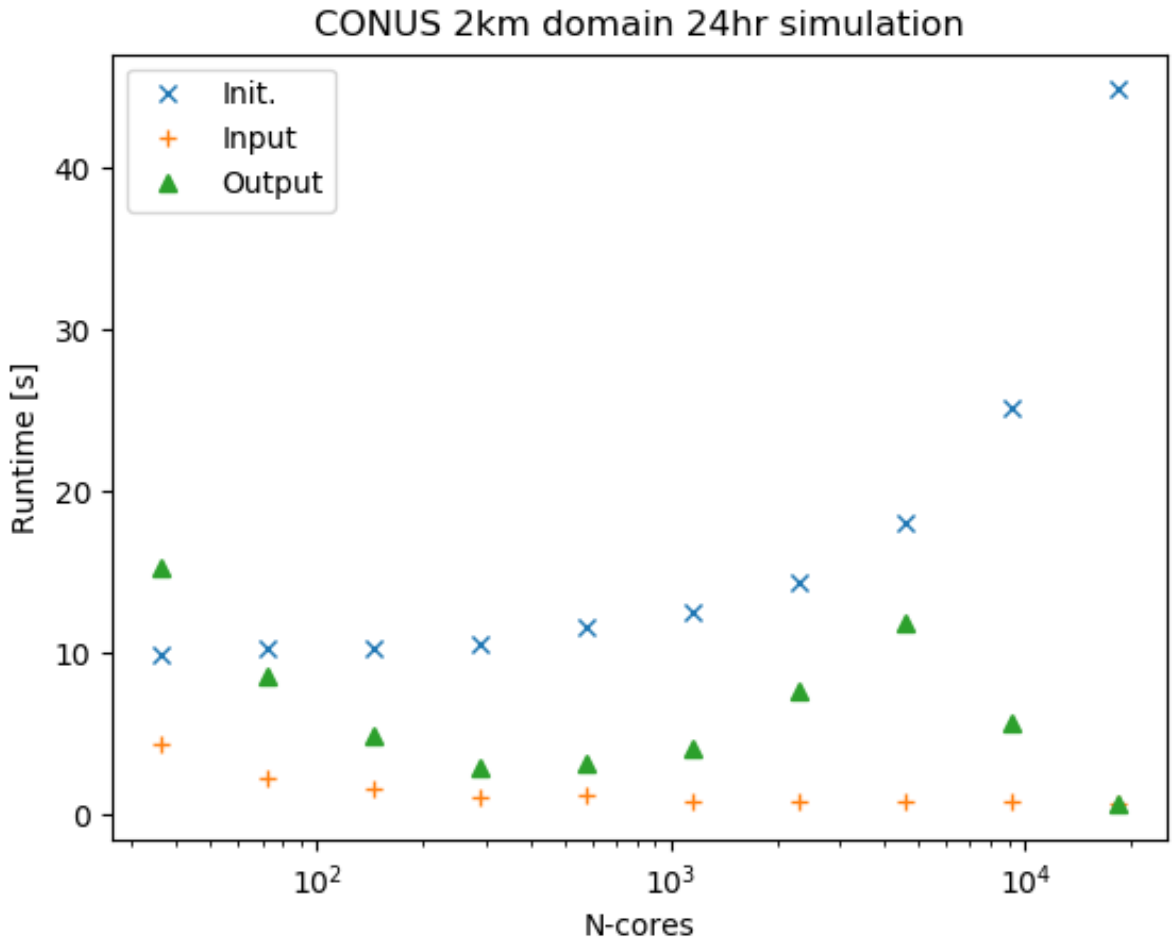
Strong scaling on smaller domains

- 210 x 160 grid cell domain
- Cray ~2.5x faster than GNU! (90s v 220s)
- Primarily due to coarray communication, GNU is faster than Cray on a single process
- Odd performance drop when node is full



Real Applications

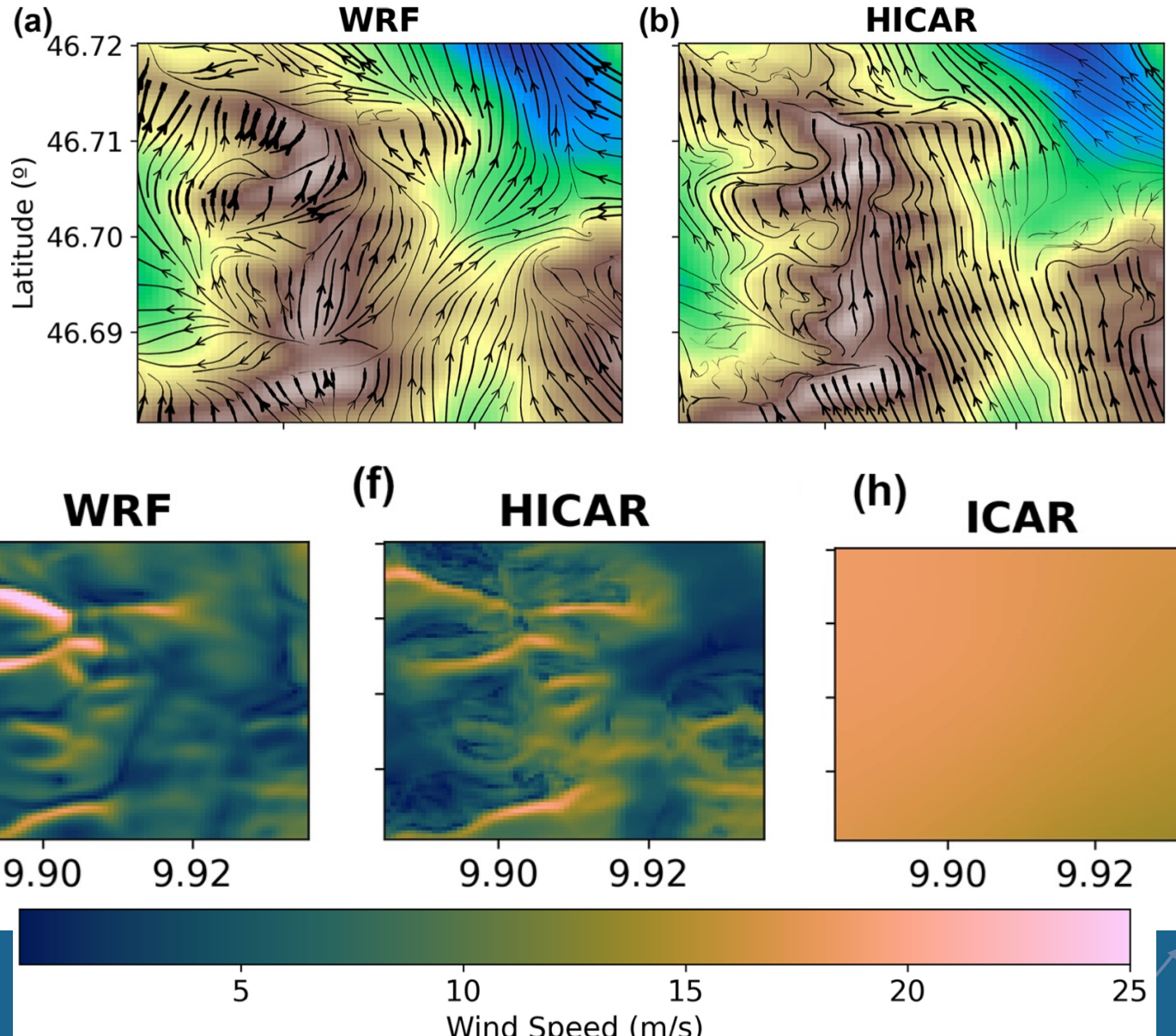




Coarrays makes parallel easy!

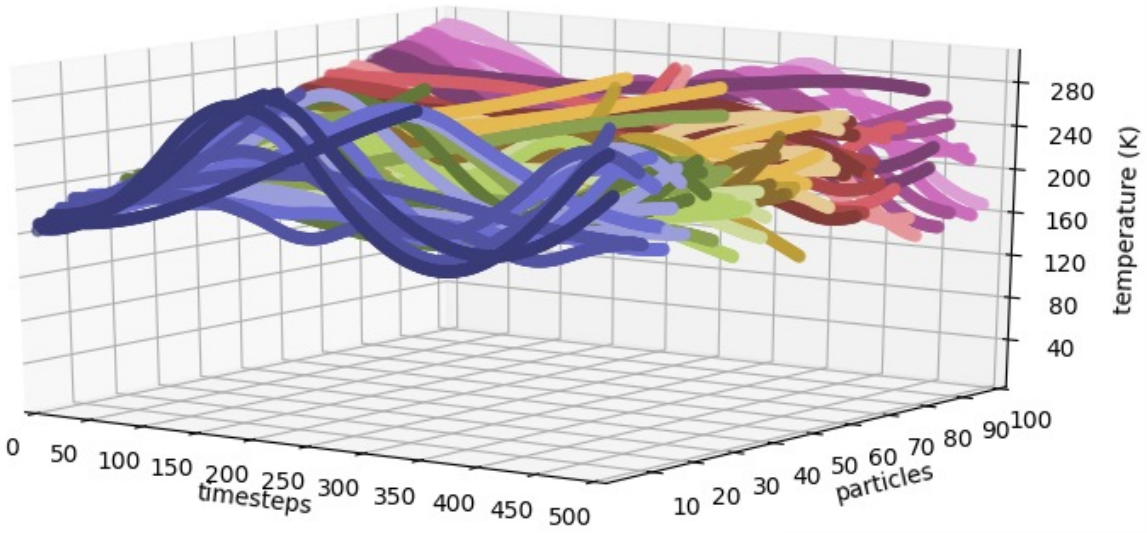
- New iterative wind solver added to ICAR by grad student
  - Dylan Reynolds at SLF
  - no computer science or atmospheric science background!

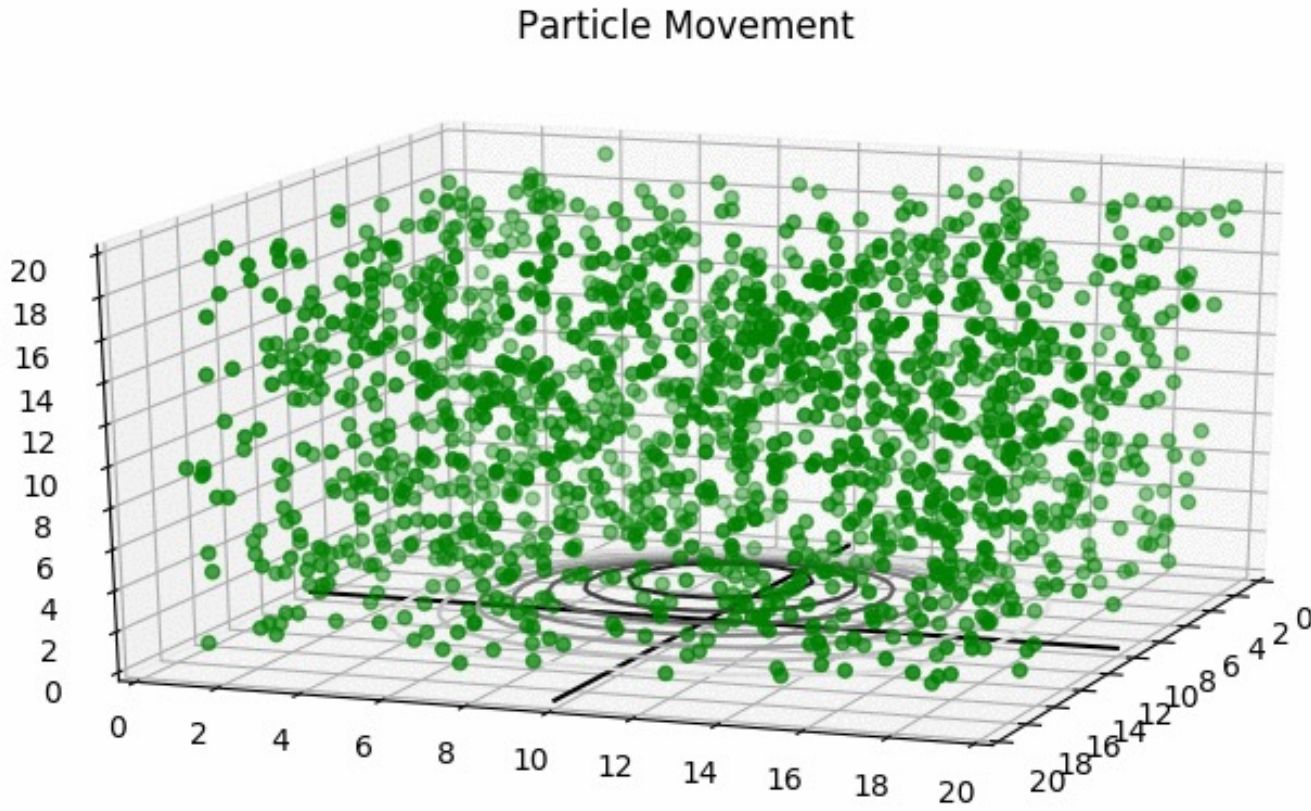
- Adds surface wind adjustments
- ICAR linear theory only adjusts upper level winds



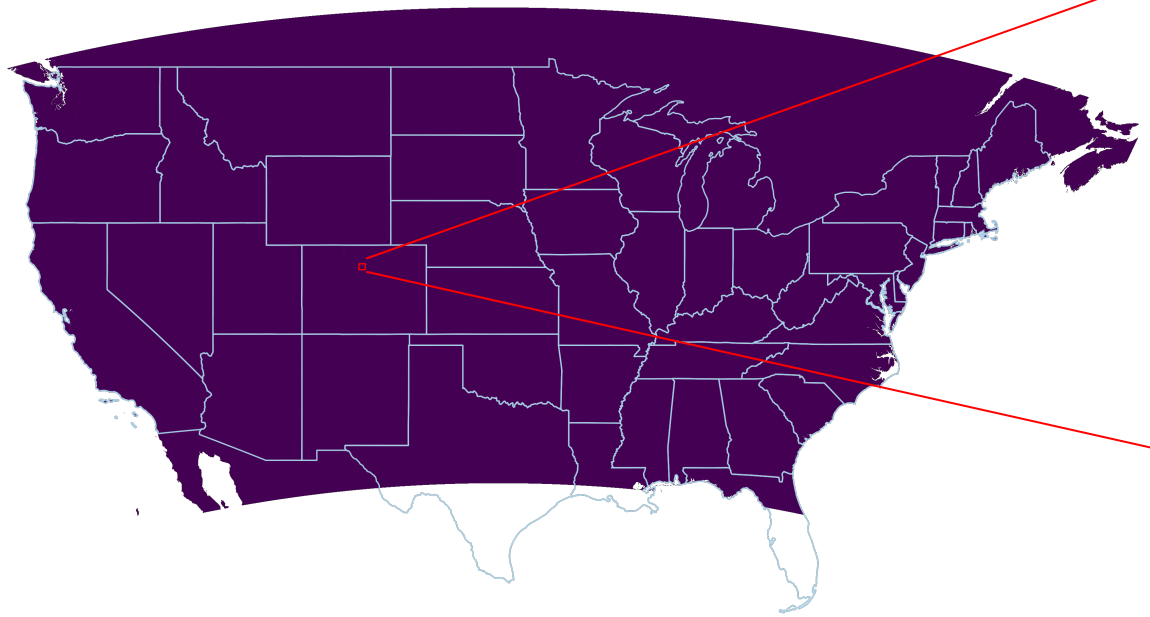
Coarrays makes parallel easy!

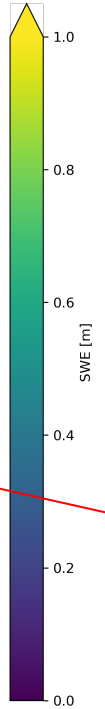
- New lagrangian air parcels added by Grad Student
  - Soren Rasmussen, Cranfield (now NCAR)
  - Fortran objects as coarrays
  - Passing entire object between processes



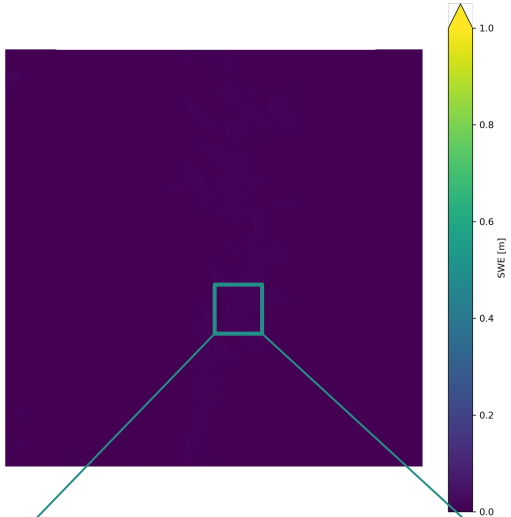


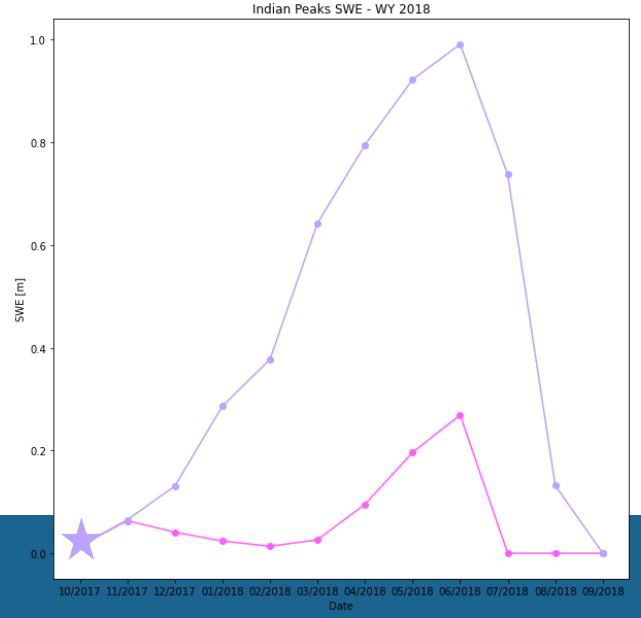
CONUS SWE October 1st , 2017



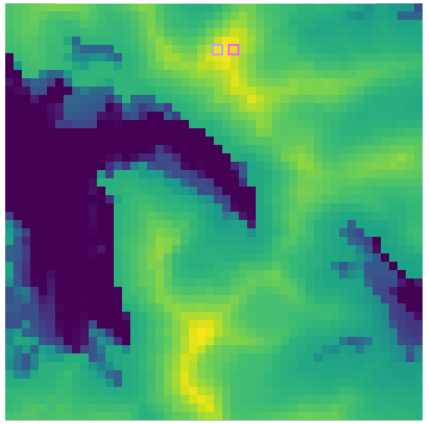


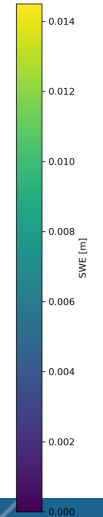
Domain Zoom 2500 km²





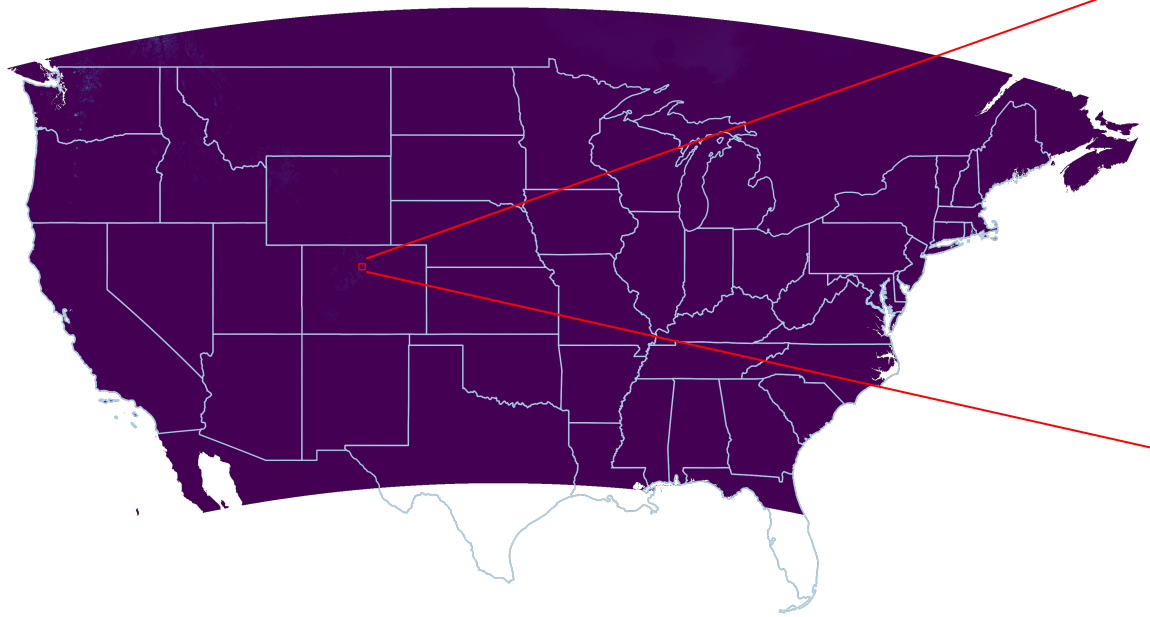
Domain Zoom 25 km²

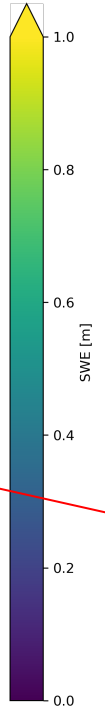


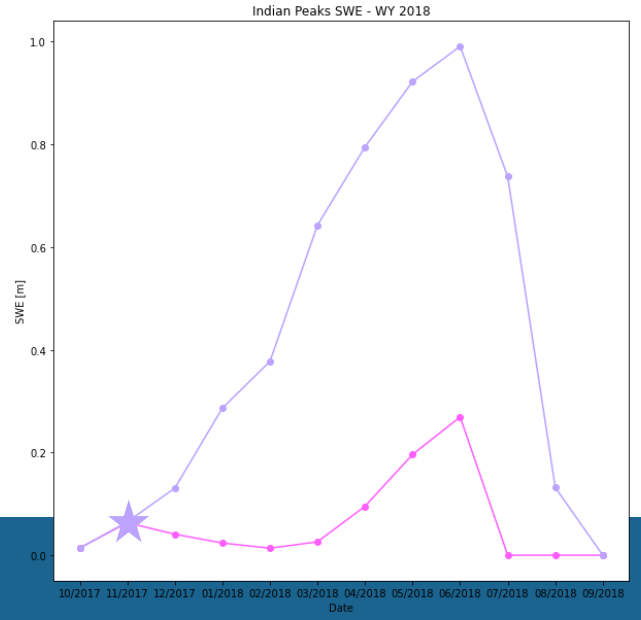


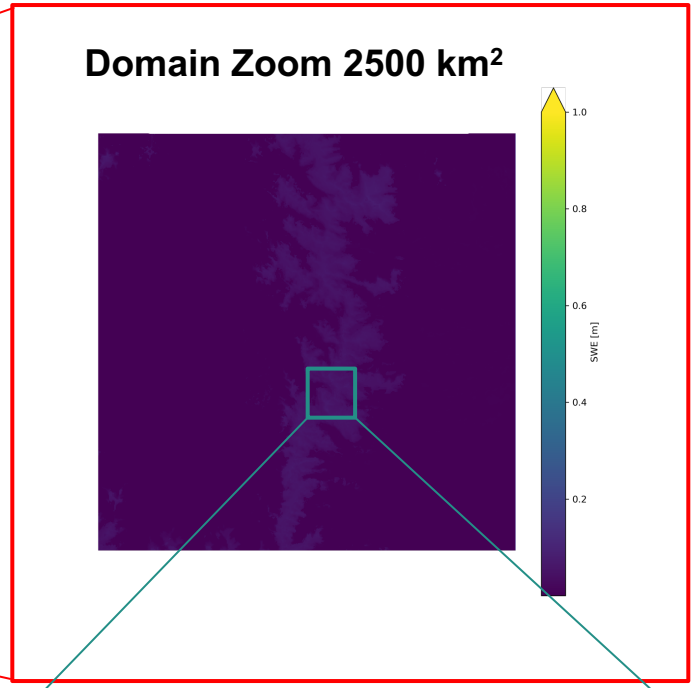
NCAR
UCAR

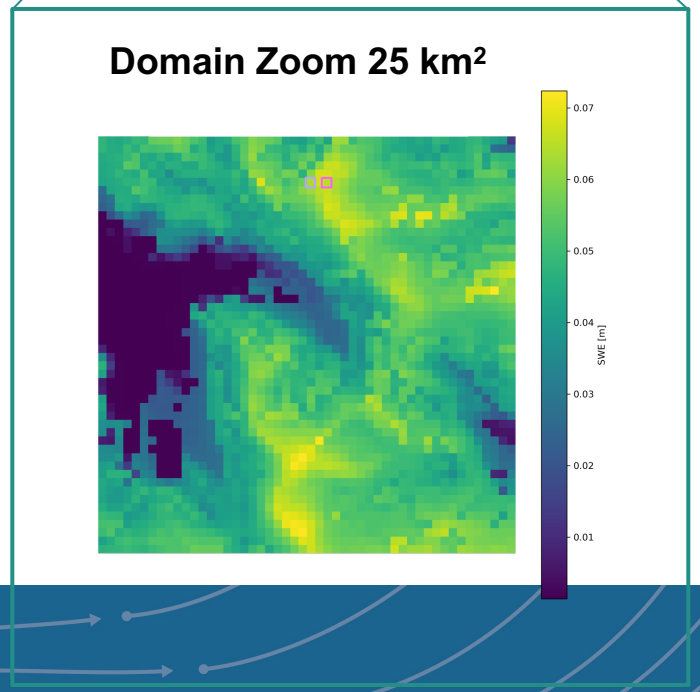
CONUS SWE November 1st , 2017





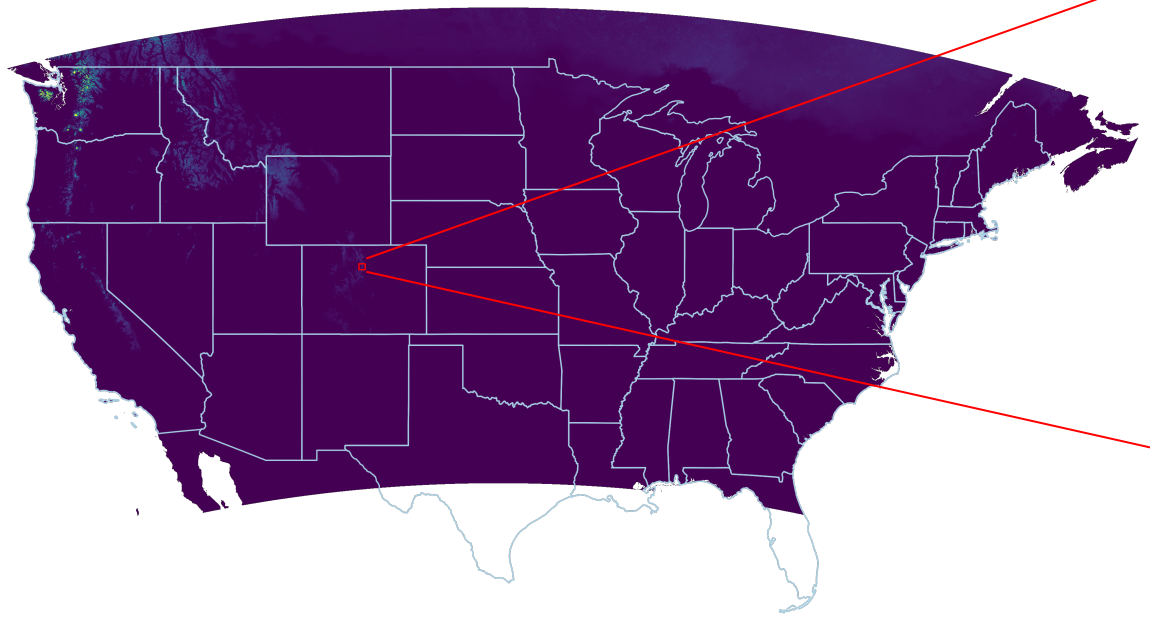


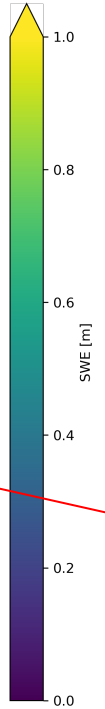




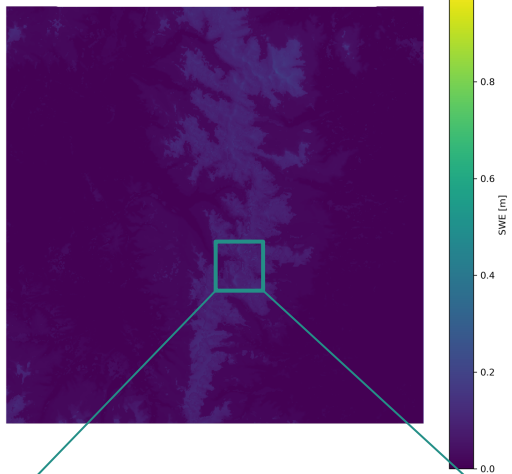
NCAR
UCAR

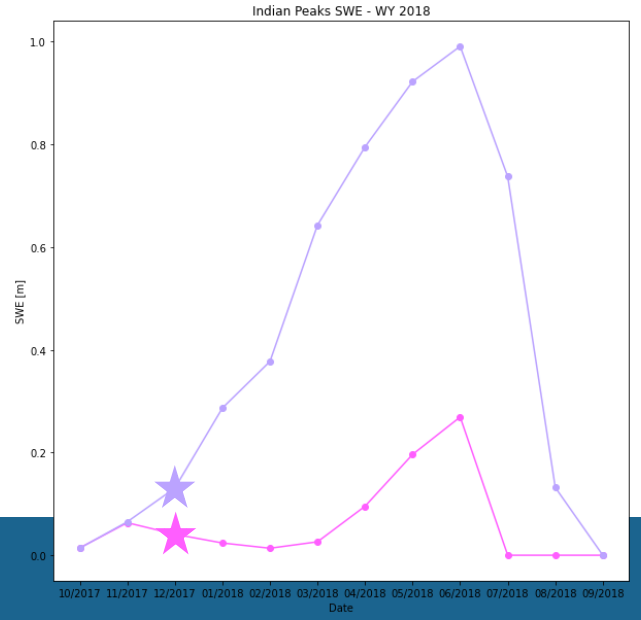
CONUS SWE December 1st , 2017



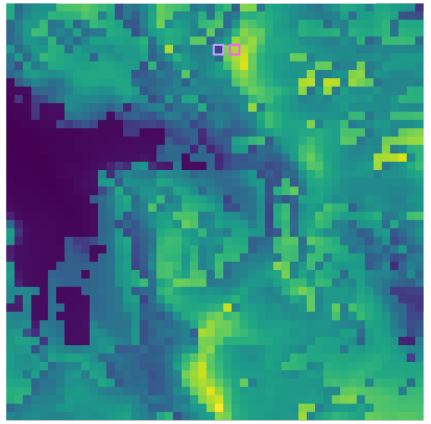


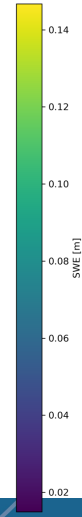
Domain Zoom 2500 km²





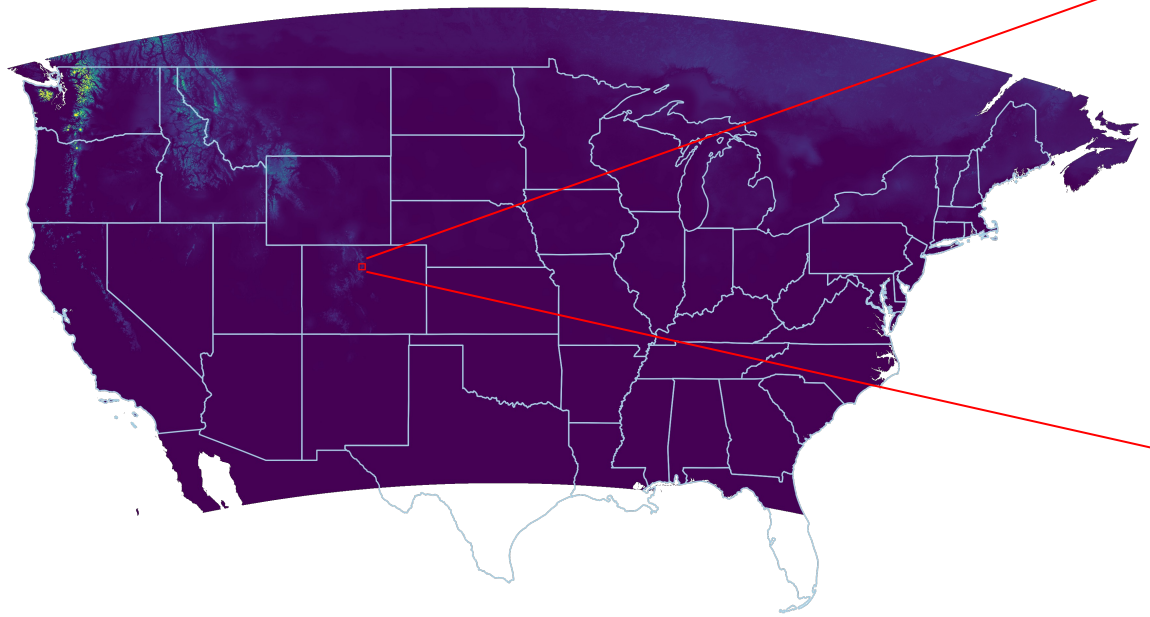
Domain Zoom 25 km²

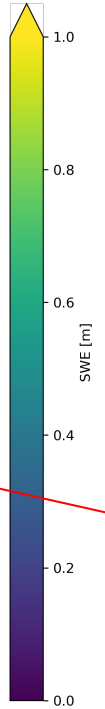


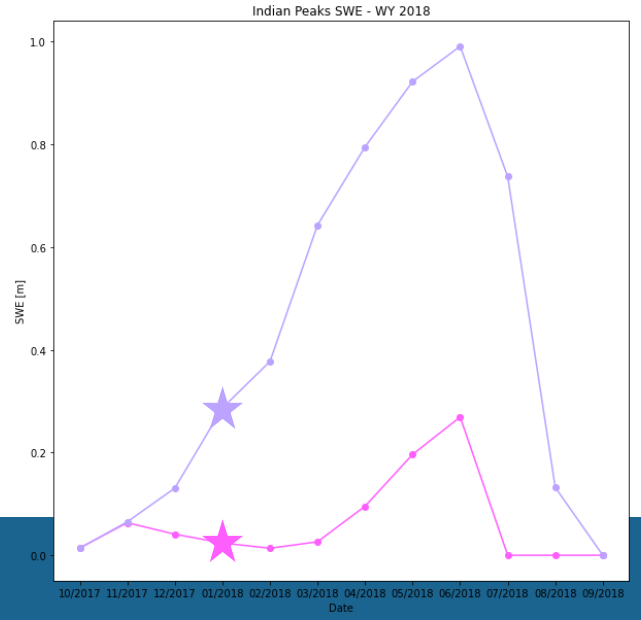


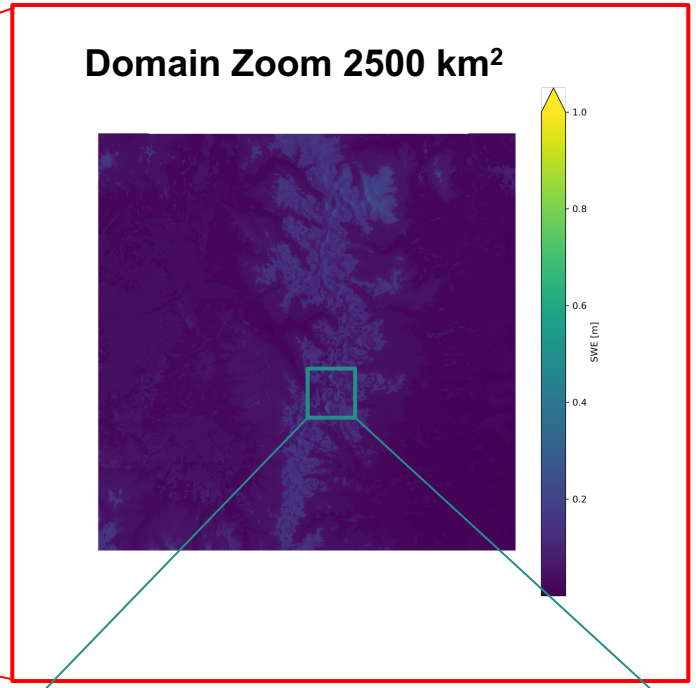
NCAR
UCAR

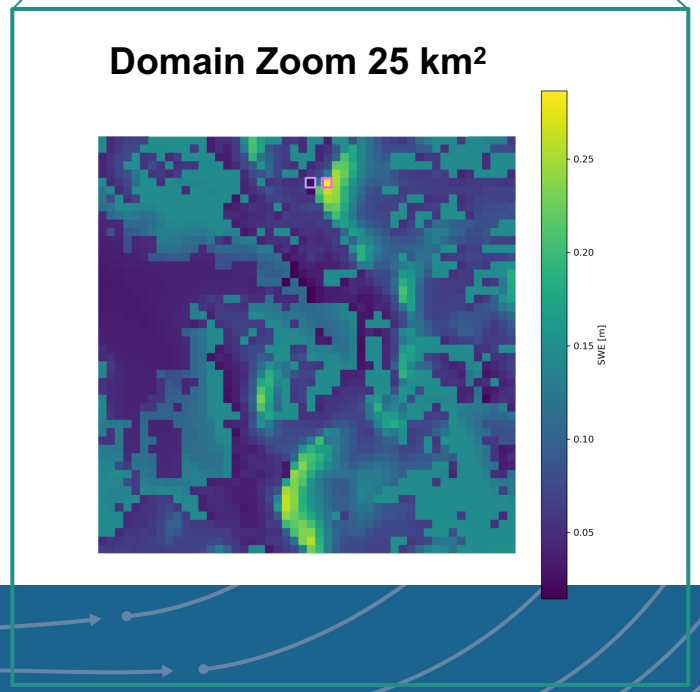
CONUS SWE January 1st , 2018





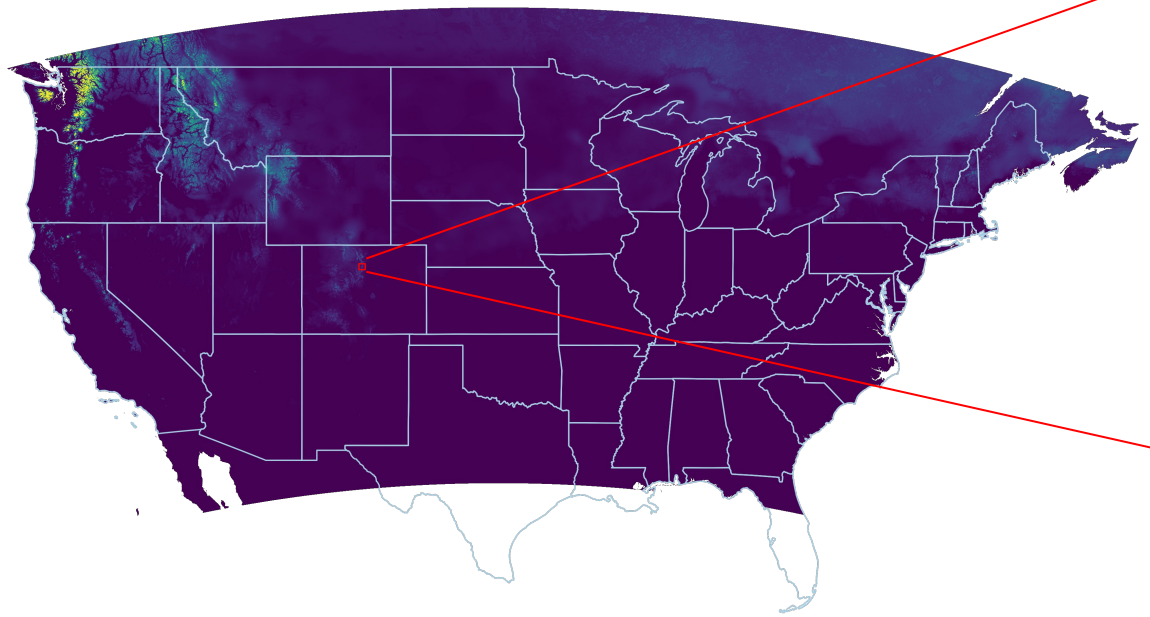


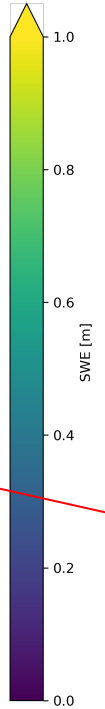




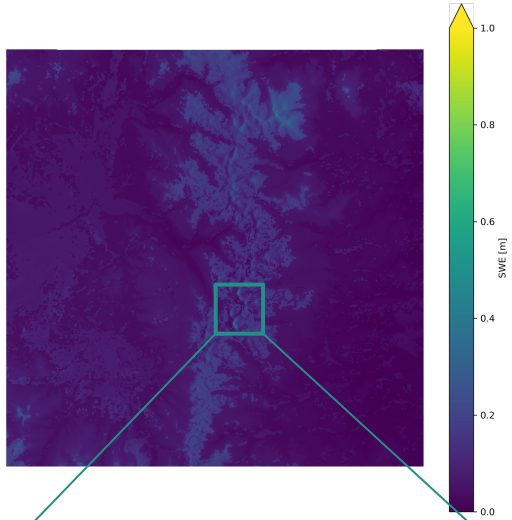
NCAR
UCAR

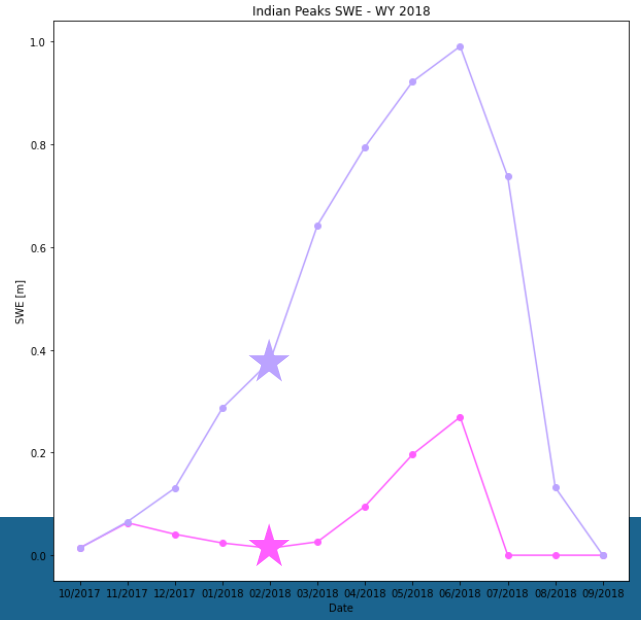
CONUS SWE February 1st , 2018



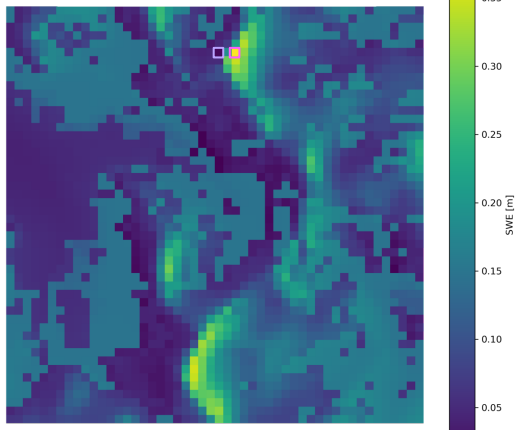


Domain Zoom 2500 km²



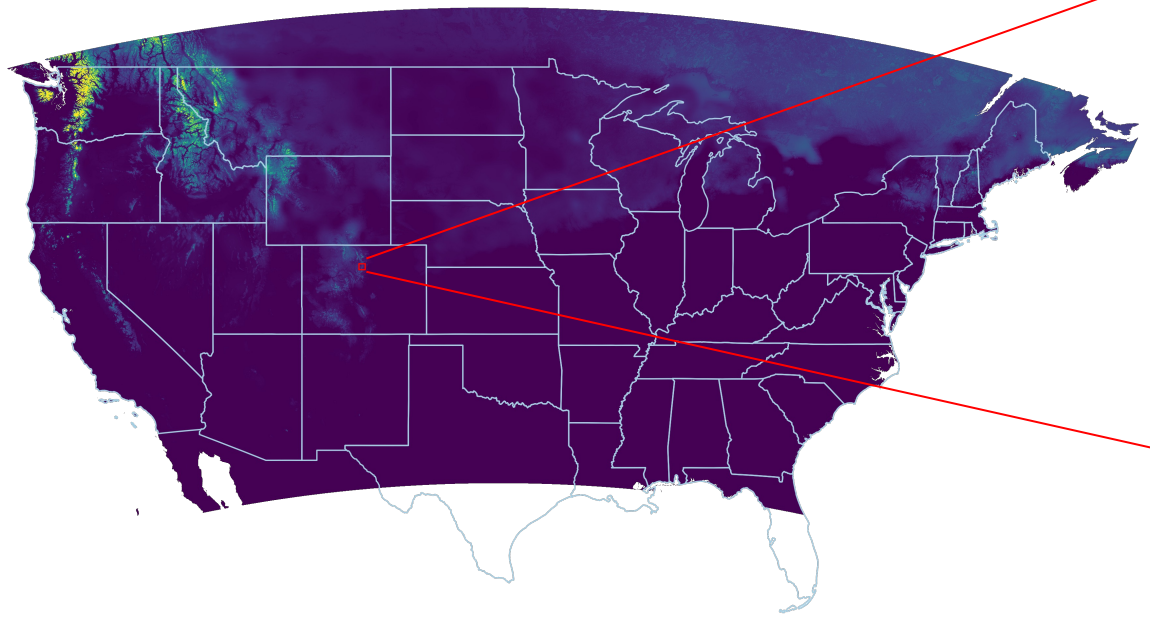


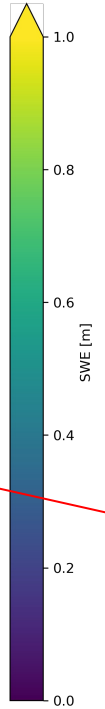
Domain Zoom 25 km²



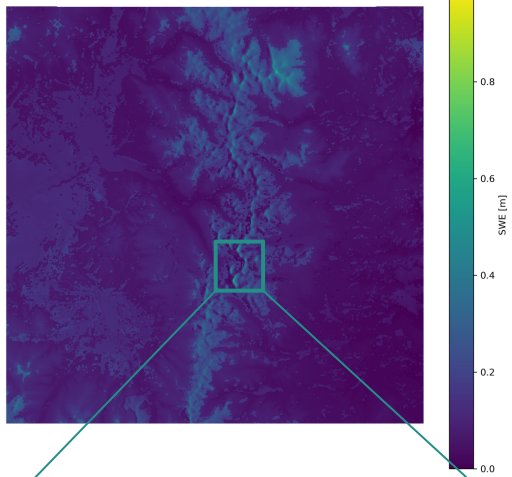
NCAR
UCAR

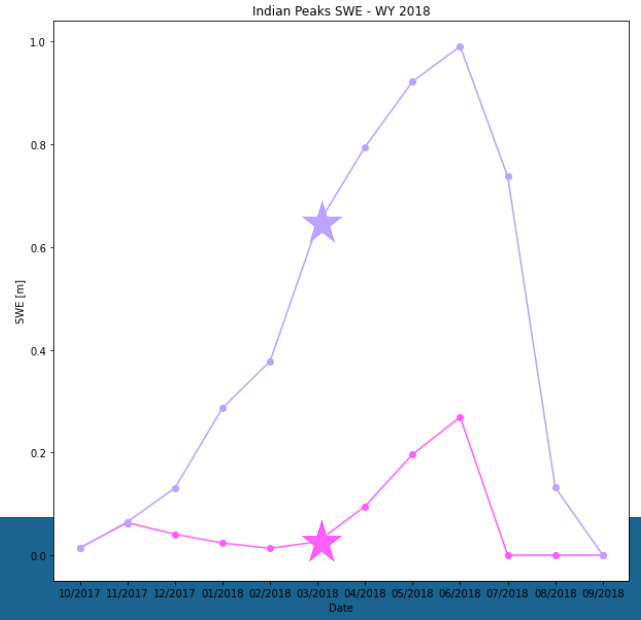
CONUS SWE March 1st , 2018



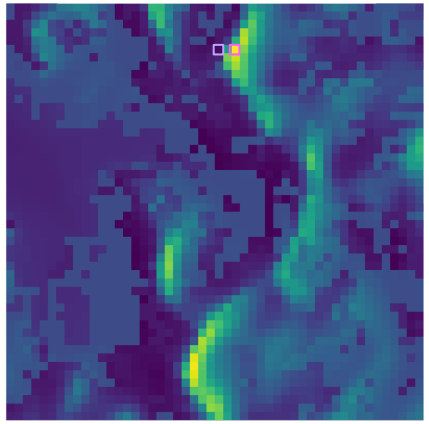


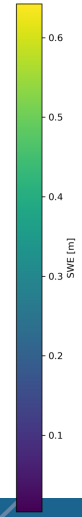
Domain Zoom 2500 km²



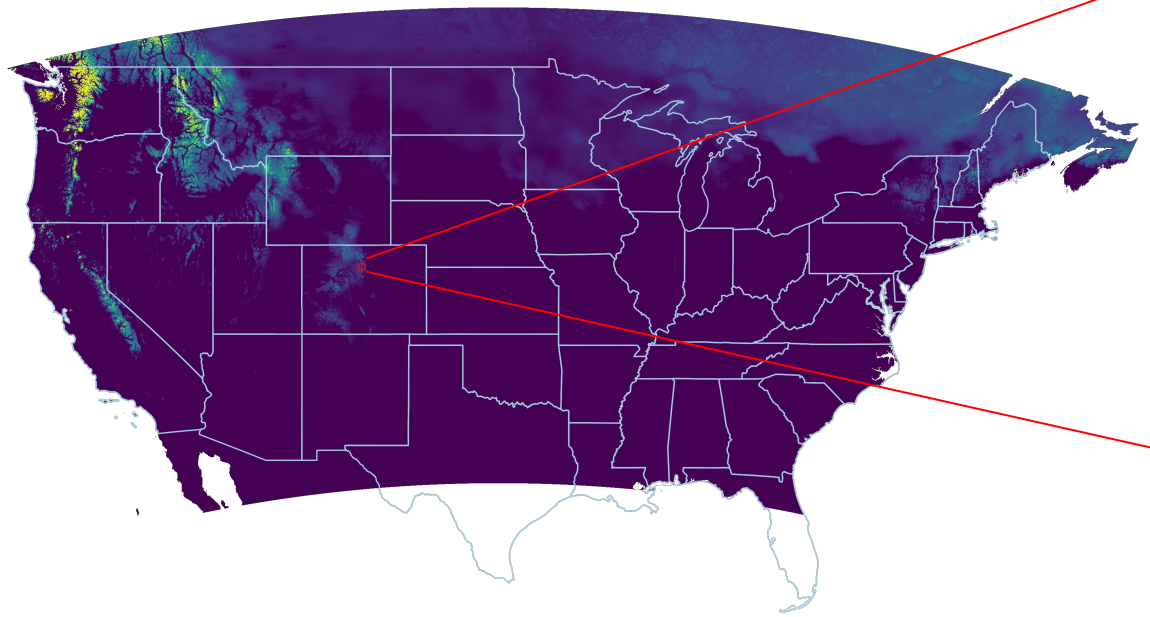


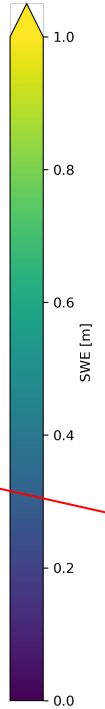
Domain Zoom 25 km²



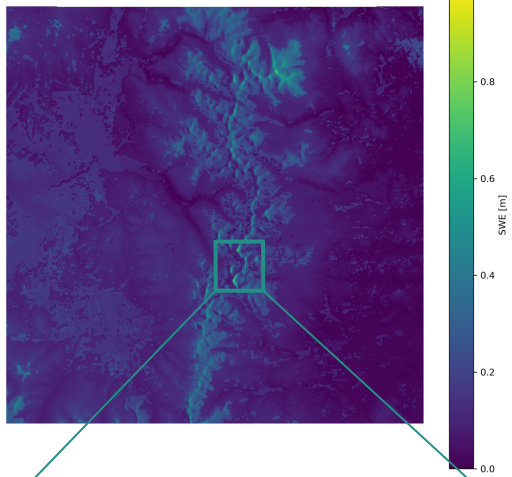


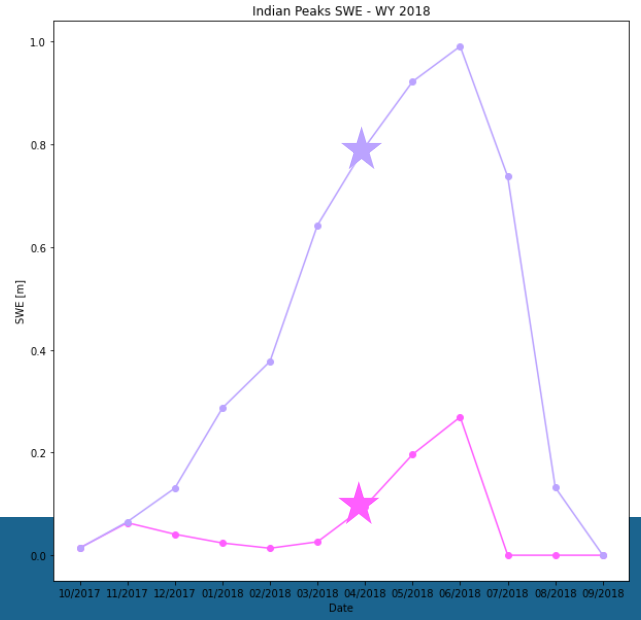
CONUS SWE April 1st , 2018



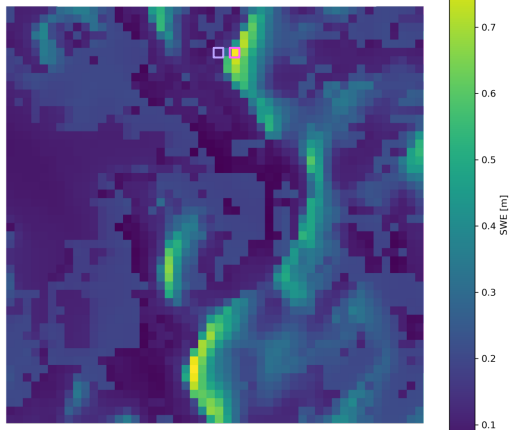


Domain Zoom 2500 km²



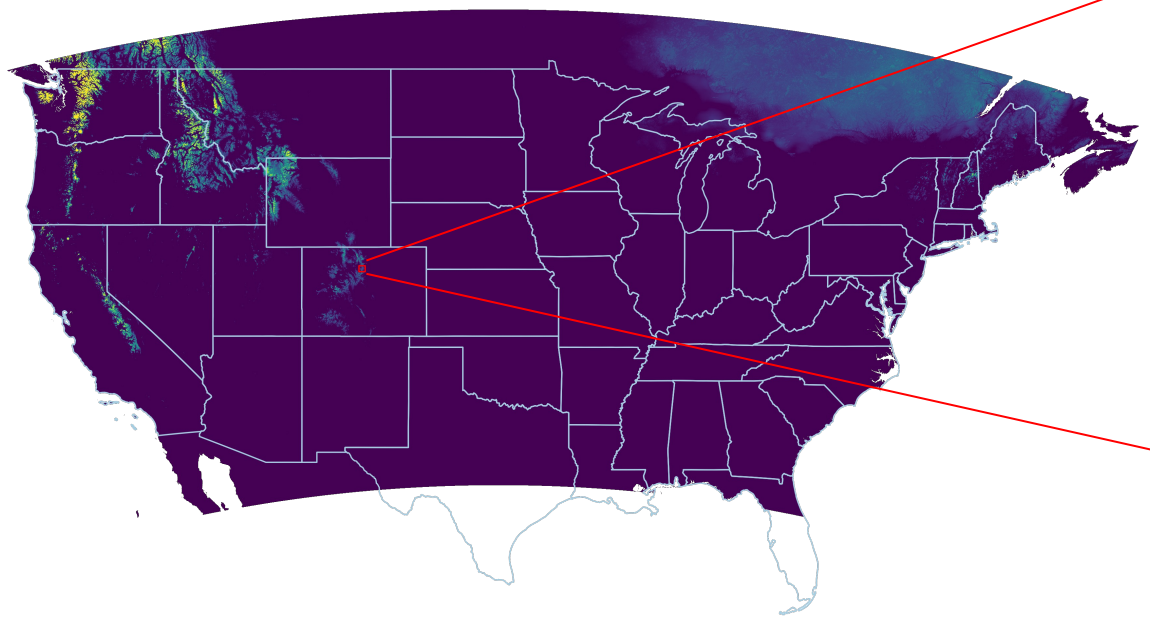


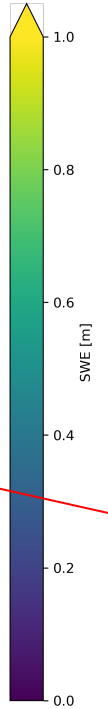
Domain Zoom 25 km²



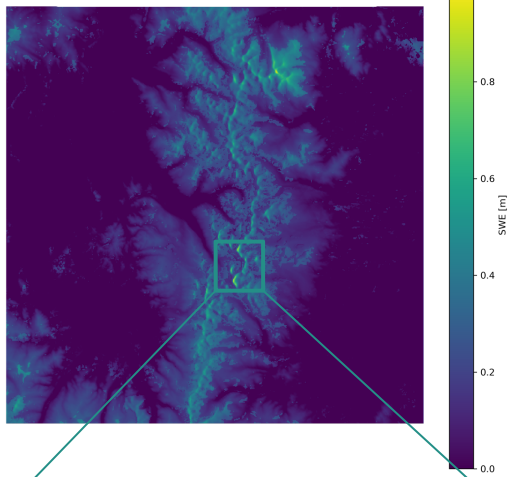
NCAR
UCAR

CONUS SWE May 1st , 2018

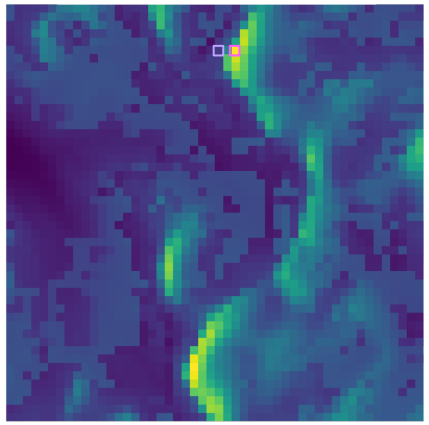


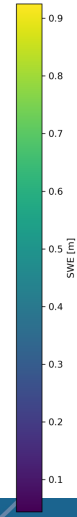


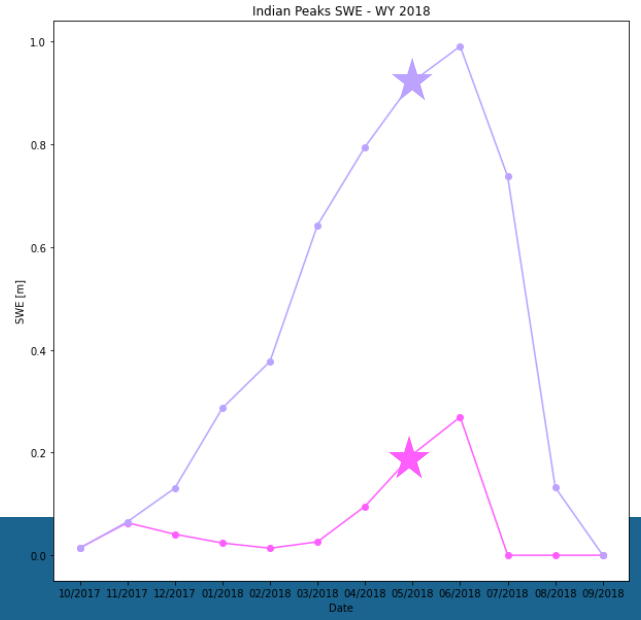
Domain Zoom 2500 km²



Domain Zoom 25 km²

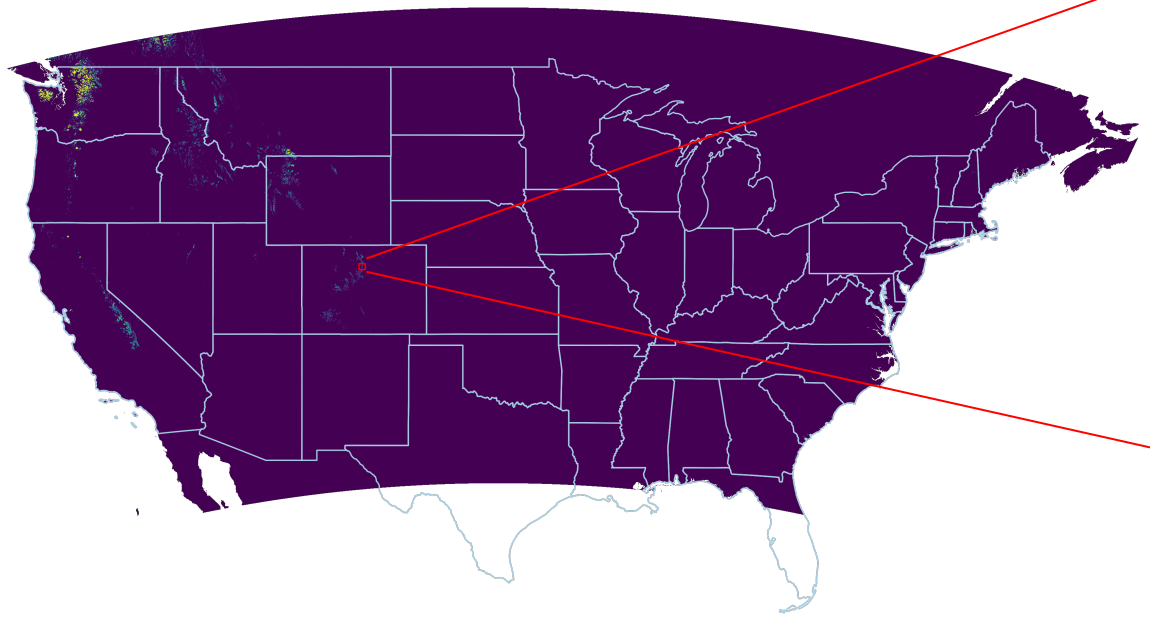


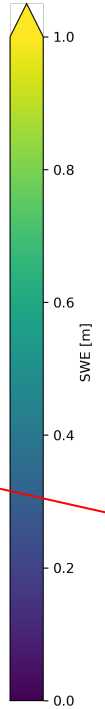




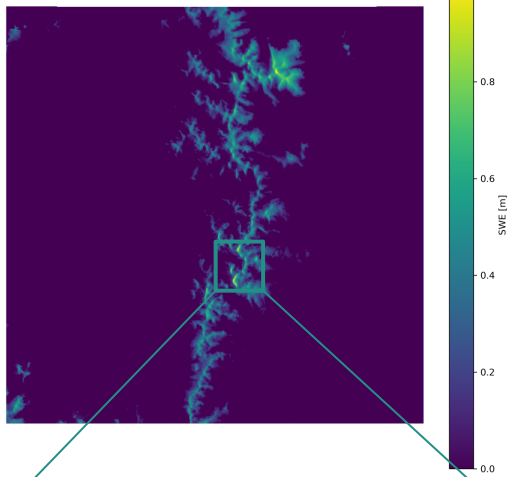
NCAR
UCAR

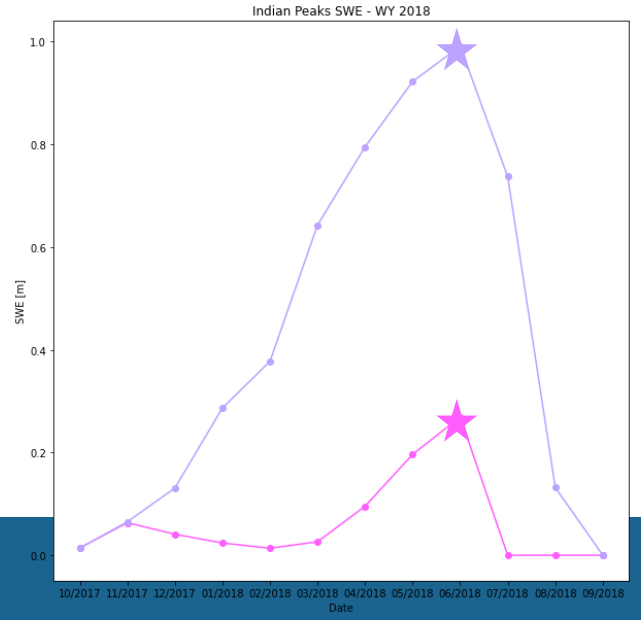
CONUS SWE June 1st , 2018



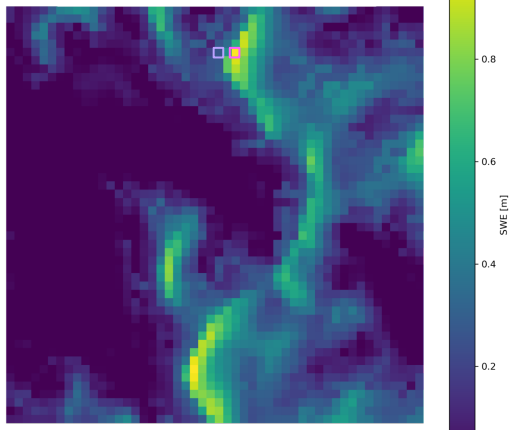


Domain Zoom 2500 km²



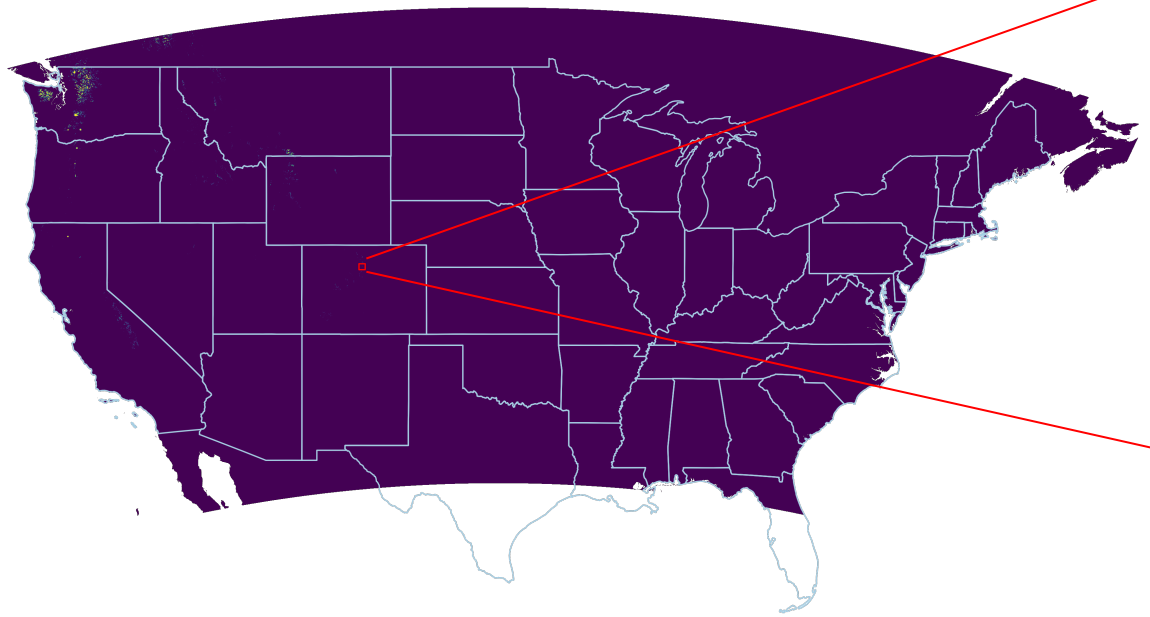


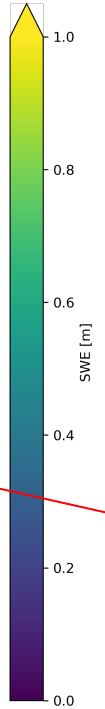
Domain Zoom 25 km²



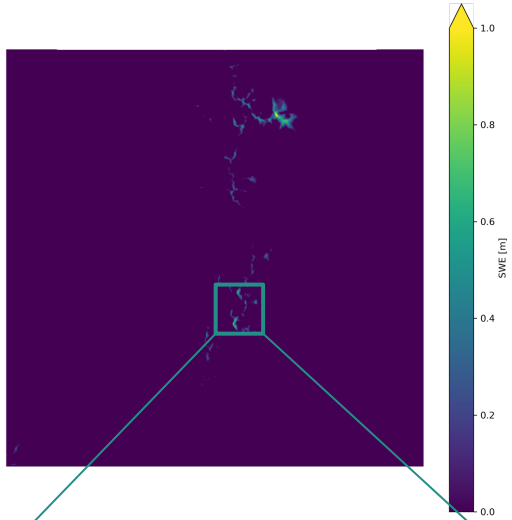
NCAR
UCAR

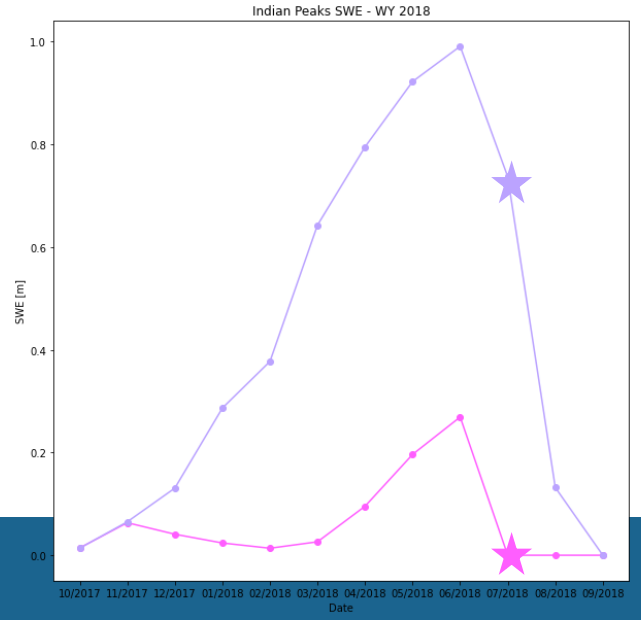
CONUS SWE July 1st , 2018



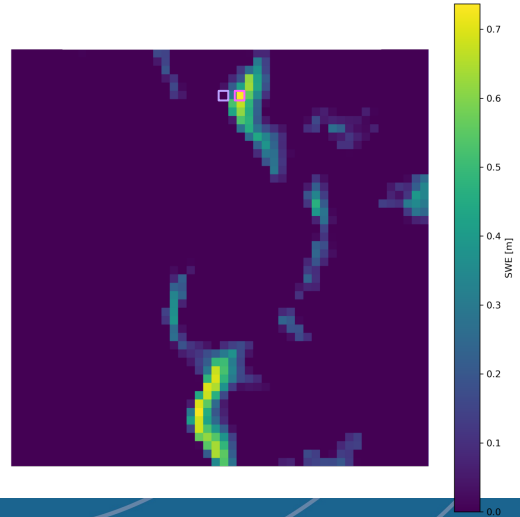


Domain Zoom 2500 km²



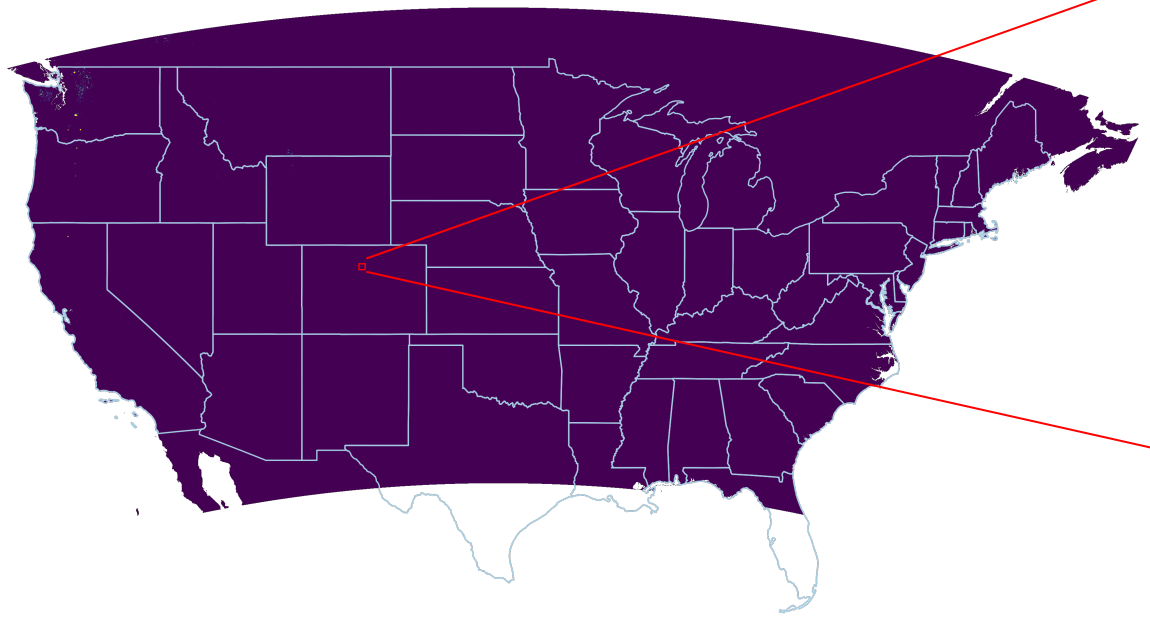


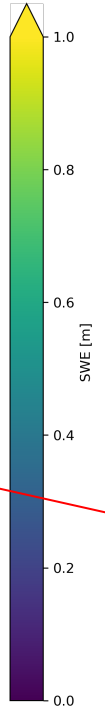
Domain Zoom 25 km²

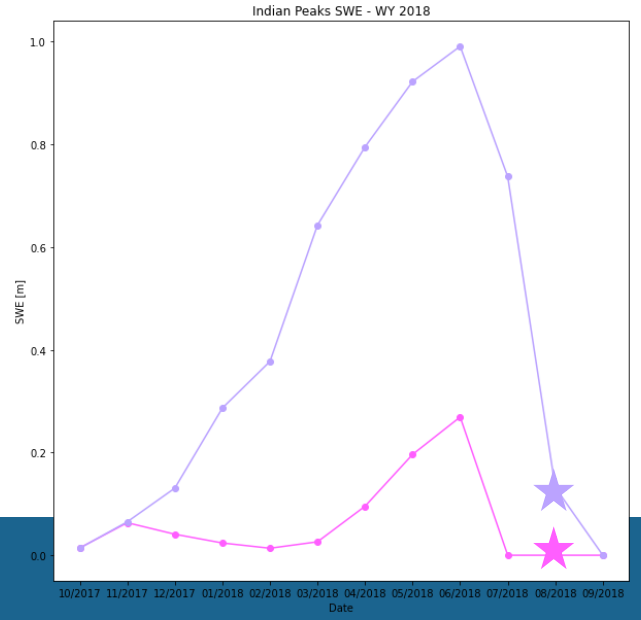


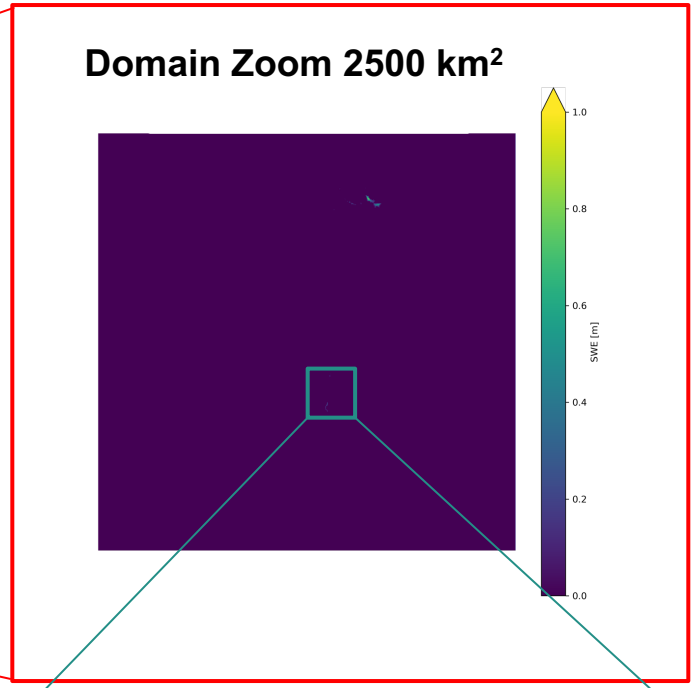
NCAR
UCAR

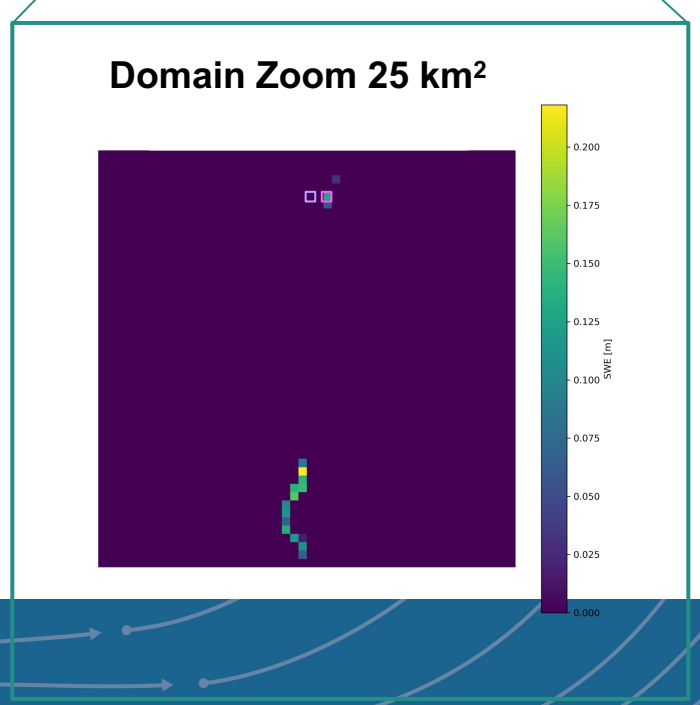
CONUS SWE August 1st , 2018



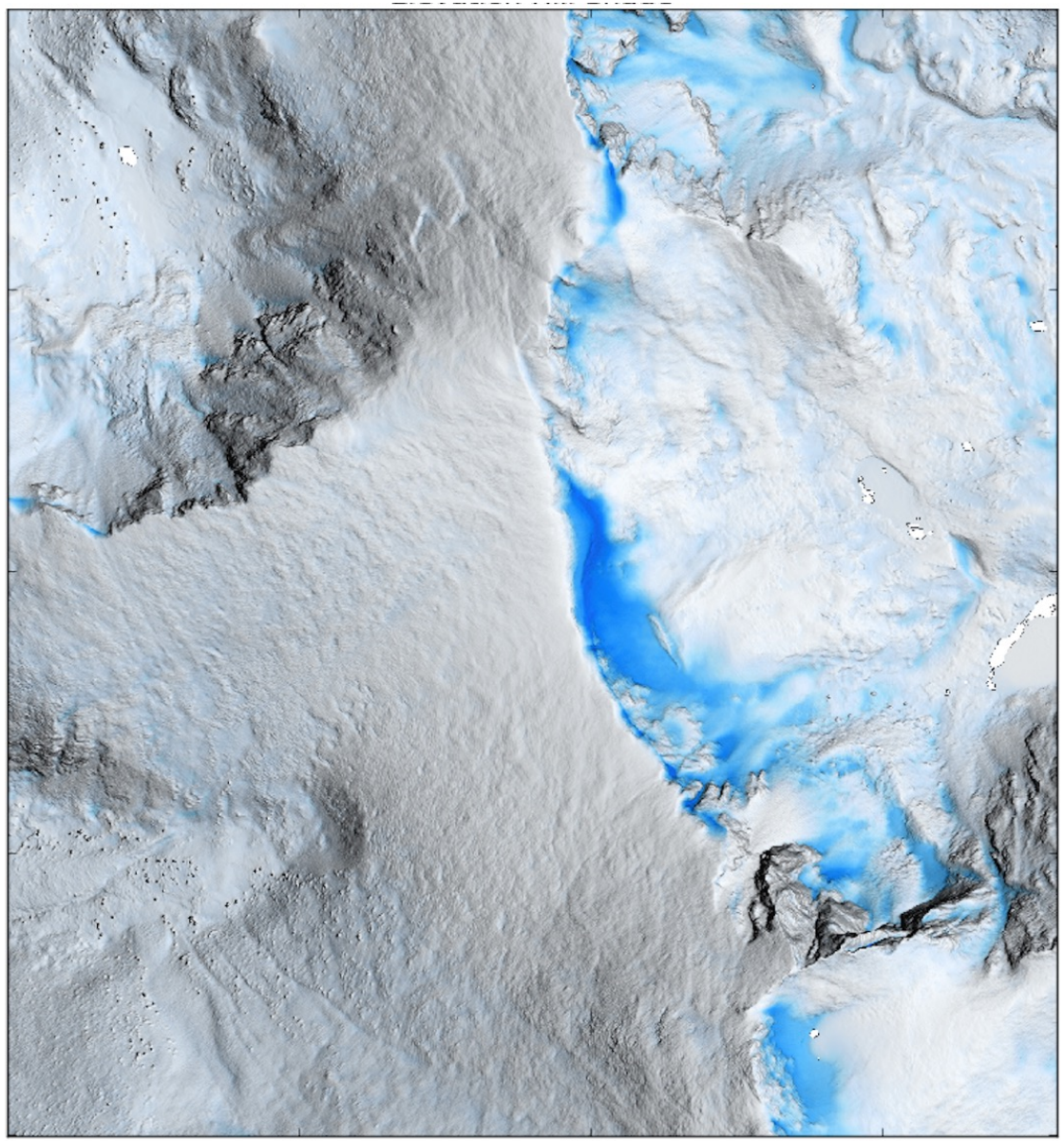


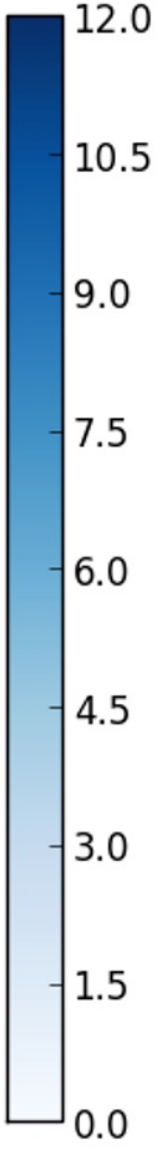


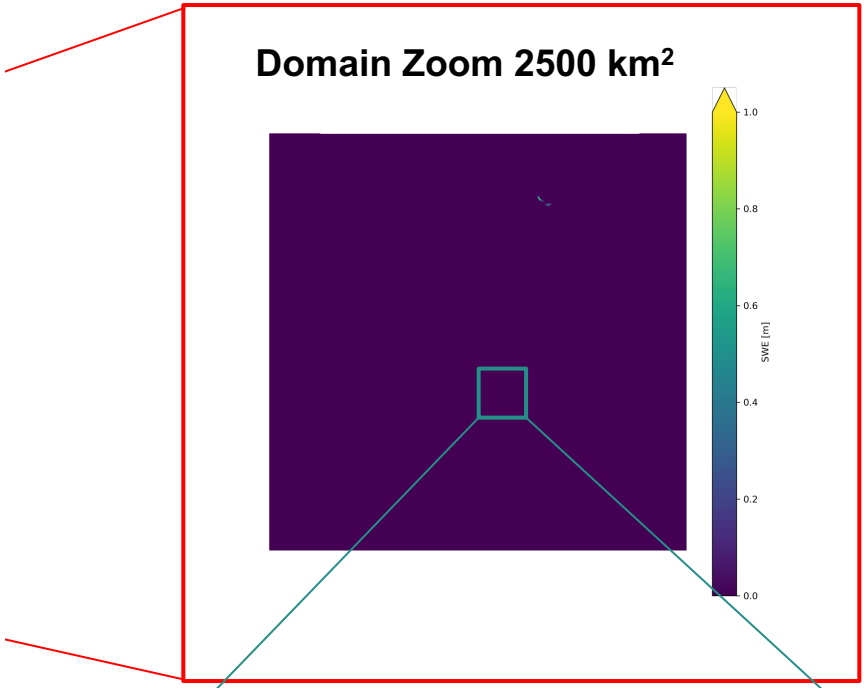


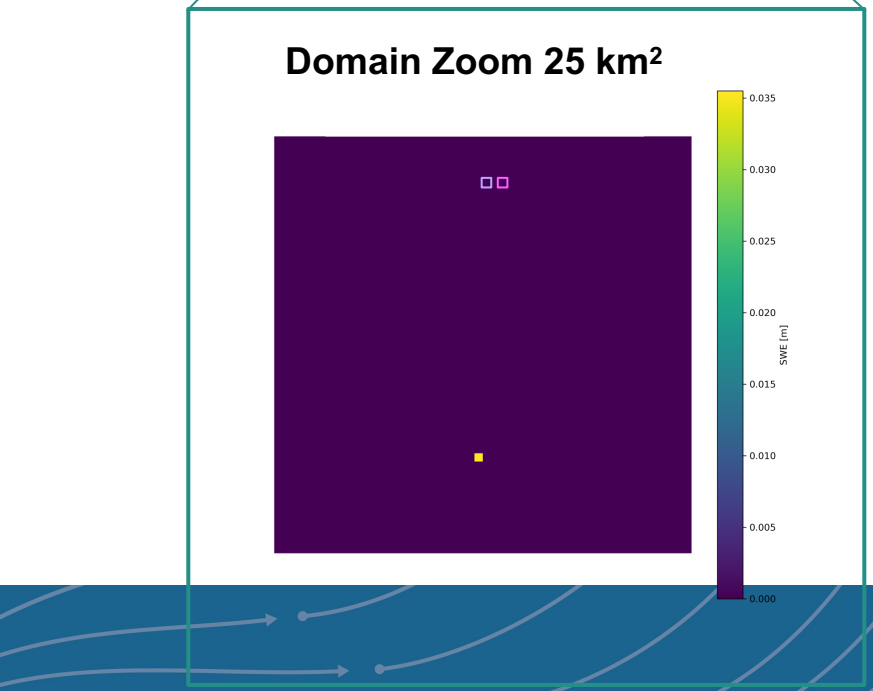


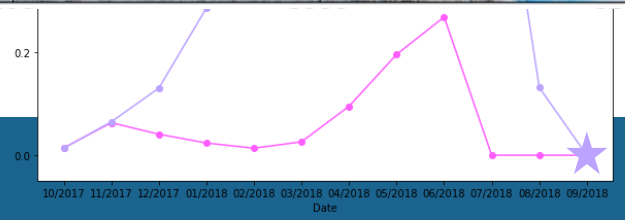
NCAR
UCAR




12.0
10.5
9.0
7.5
6.0
4.5
3.0
1.5
0.0
Snow Depth [m]


Domain Zoom 2500 km²
1.0
0.8
0.6
0.4
0.2
0.0
SWE [m]

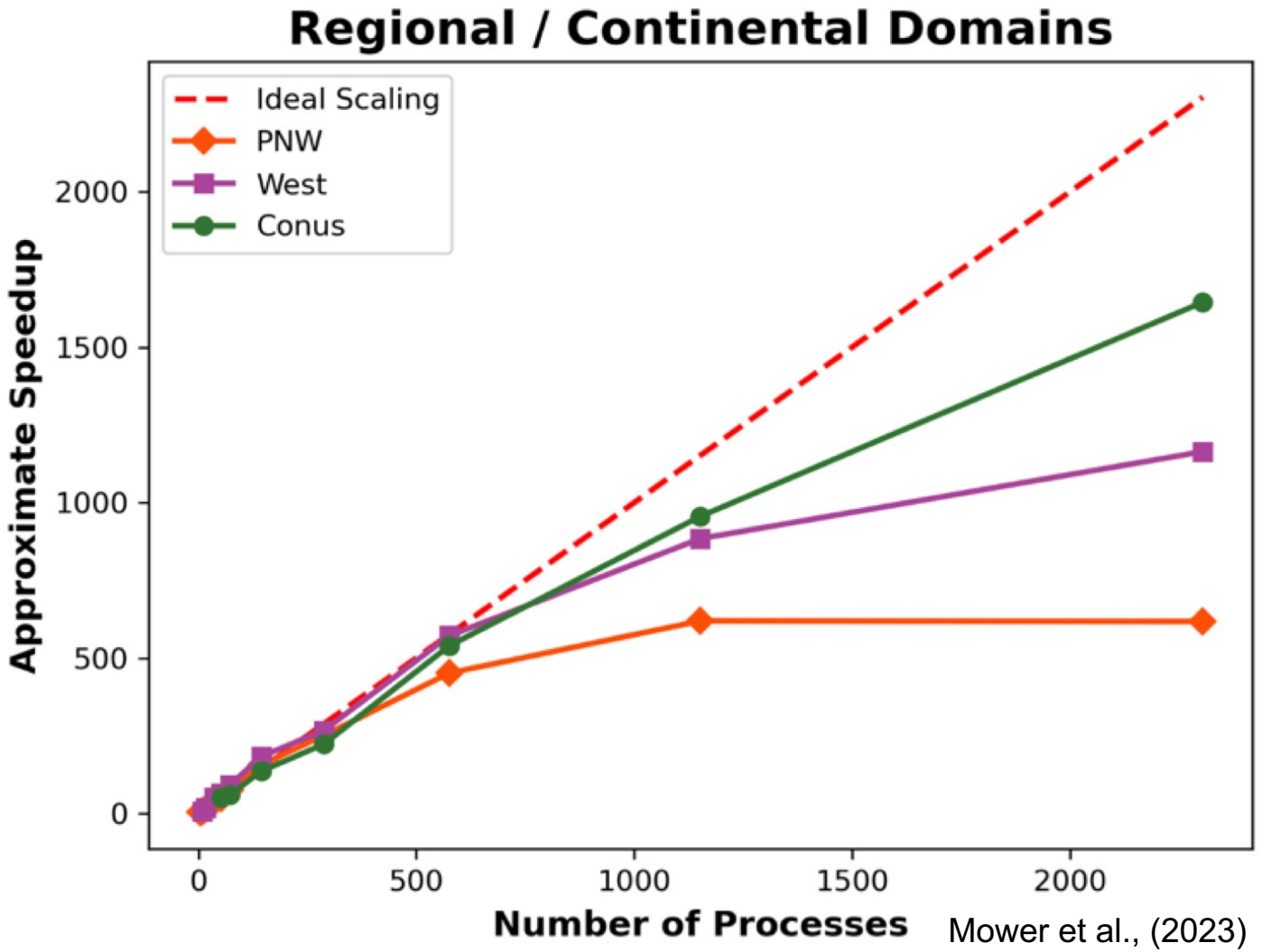

Domain Zoom 25 km²
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
SWE [m]


0.2
0.0
10/2017 11/2017 12/2017 01/2018 02/2018 03/2018 04/2018 05/2018 06/2018 07/2018 08/2018 09/2018
Date

NCAR
UCAR

Parallelization of SnowModel

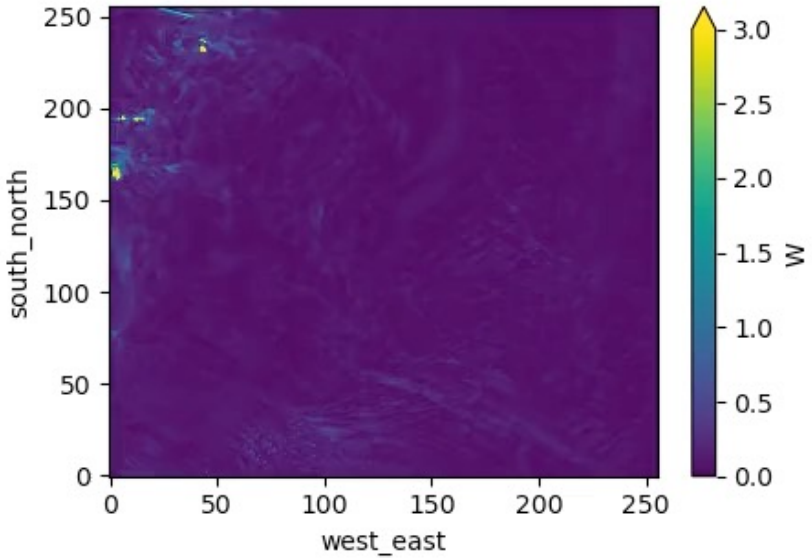
- One of the best high-res snow models in the world
- One of the only snow models that transport snow in the wind
- Parallelization of legacy fortran code
- Mix of F77-F90
- Use of coarray fortran enabled an early career scientist (and now PhD student) to parallelize code efficiently



Mower et al., (2023)

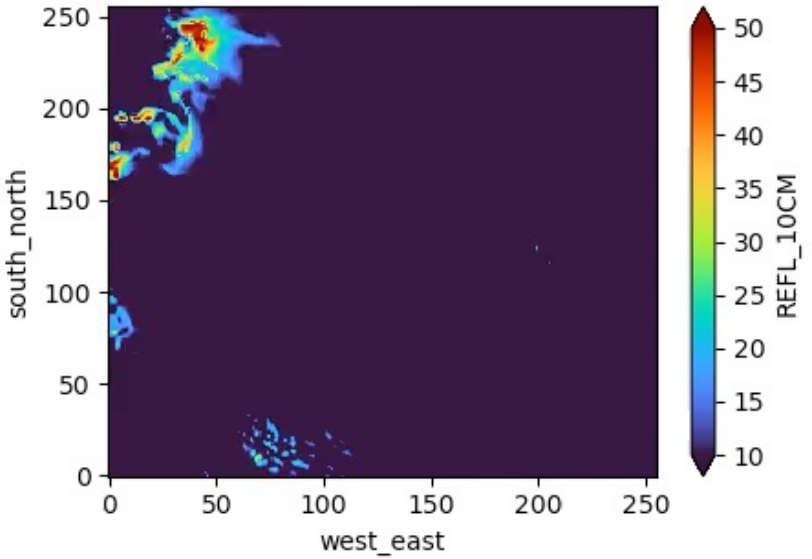
Learning from Radar data

- Radar spatial patterns contain information about storm dynamics
- Radar observations are assimilated into state of the art atmospheric models
  - But they don't know what the patterns mean
- Leveraging HPC or ML should enable greater information



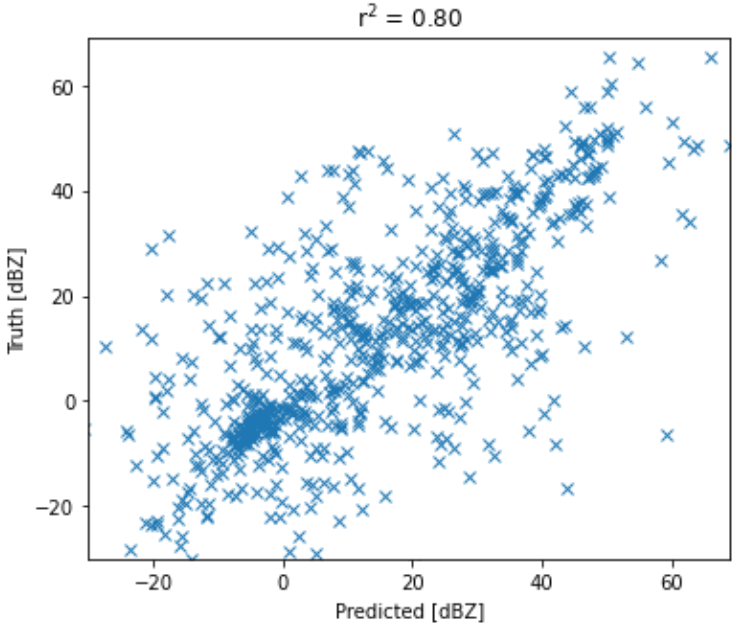
Simulated Radar Reflectivity

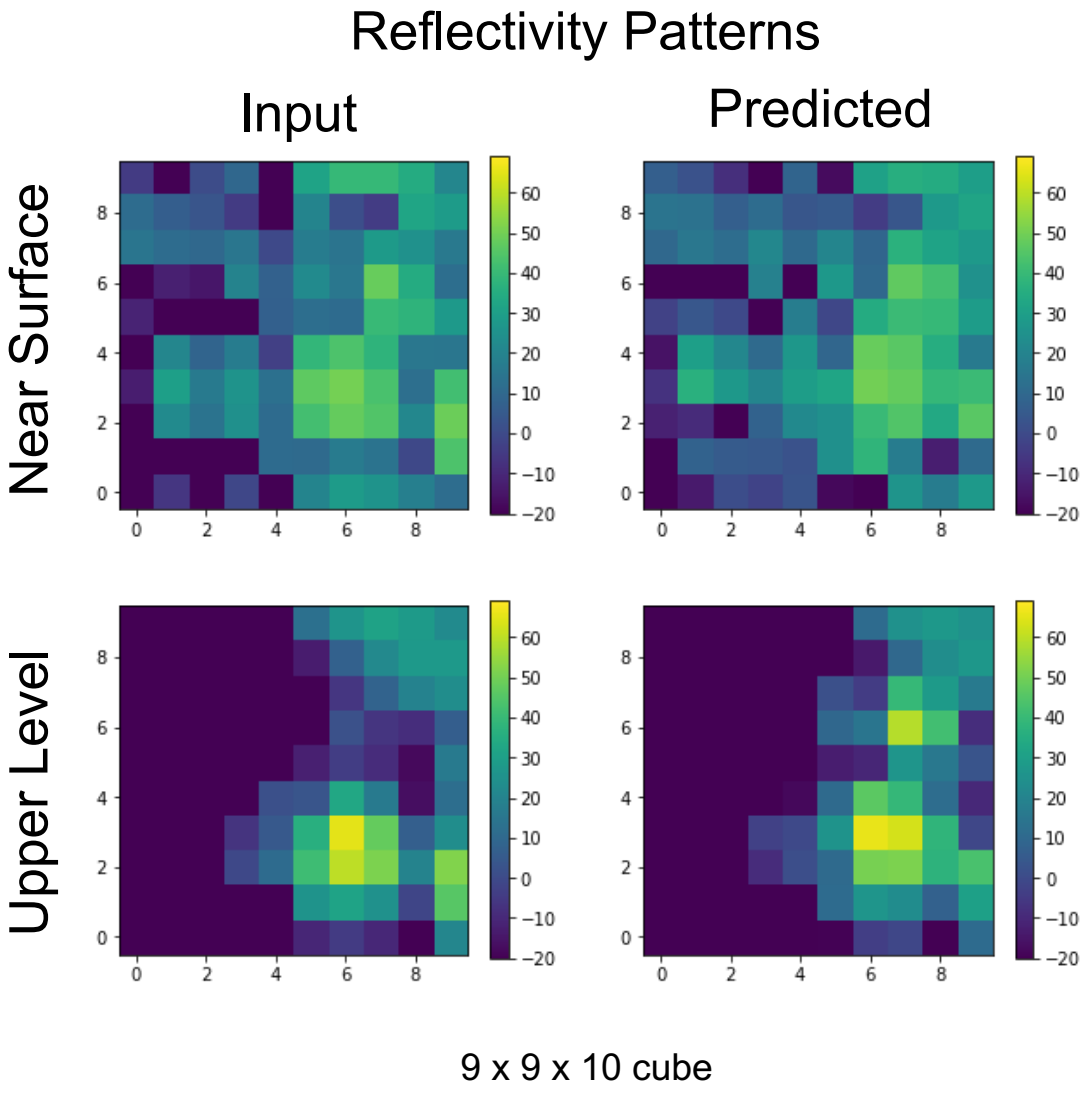
Simulated Vertical Wind Speed



Spatial Analog Machine-Learning Approach

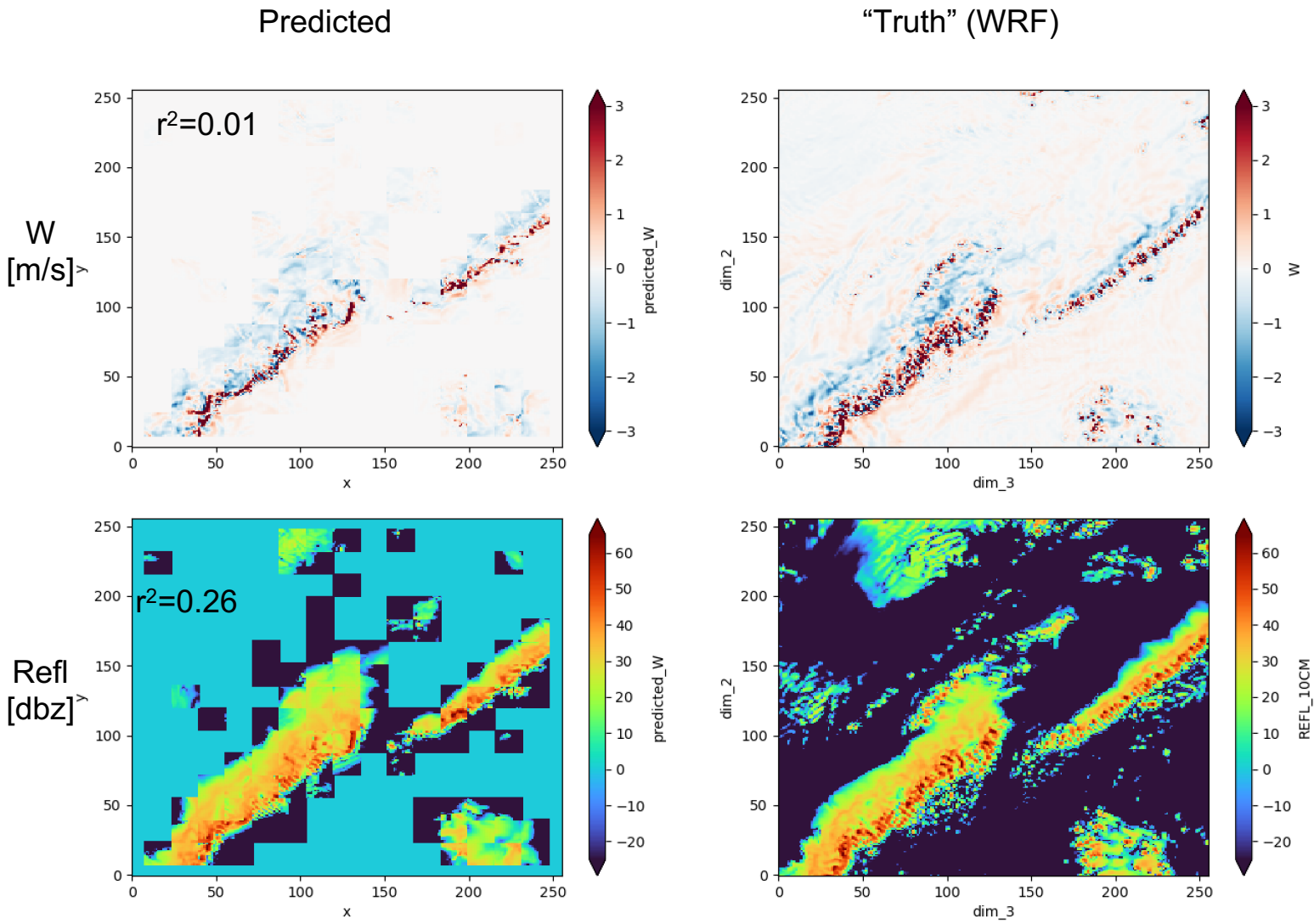
- Benefits from **big** data
- Selecting using 3-D spatial patterns
- Chop domain into 9x9x10 data cubes
- Search for best matches from archive
- Single cases promising but scaling is challenging





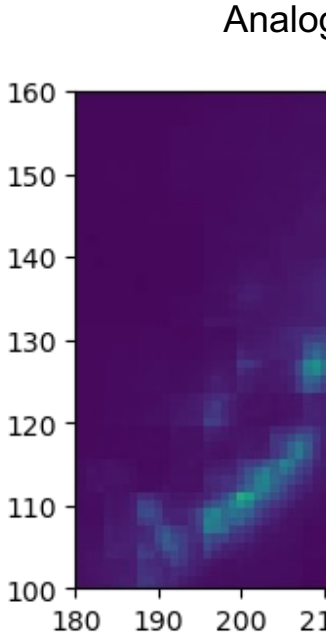
Scaled up Shallow Learning Approach

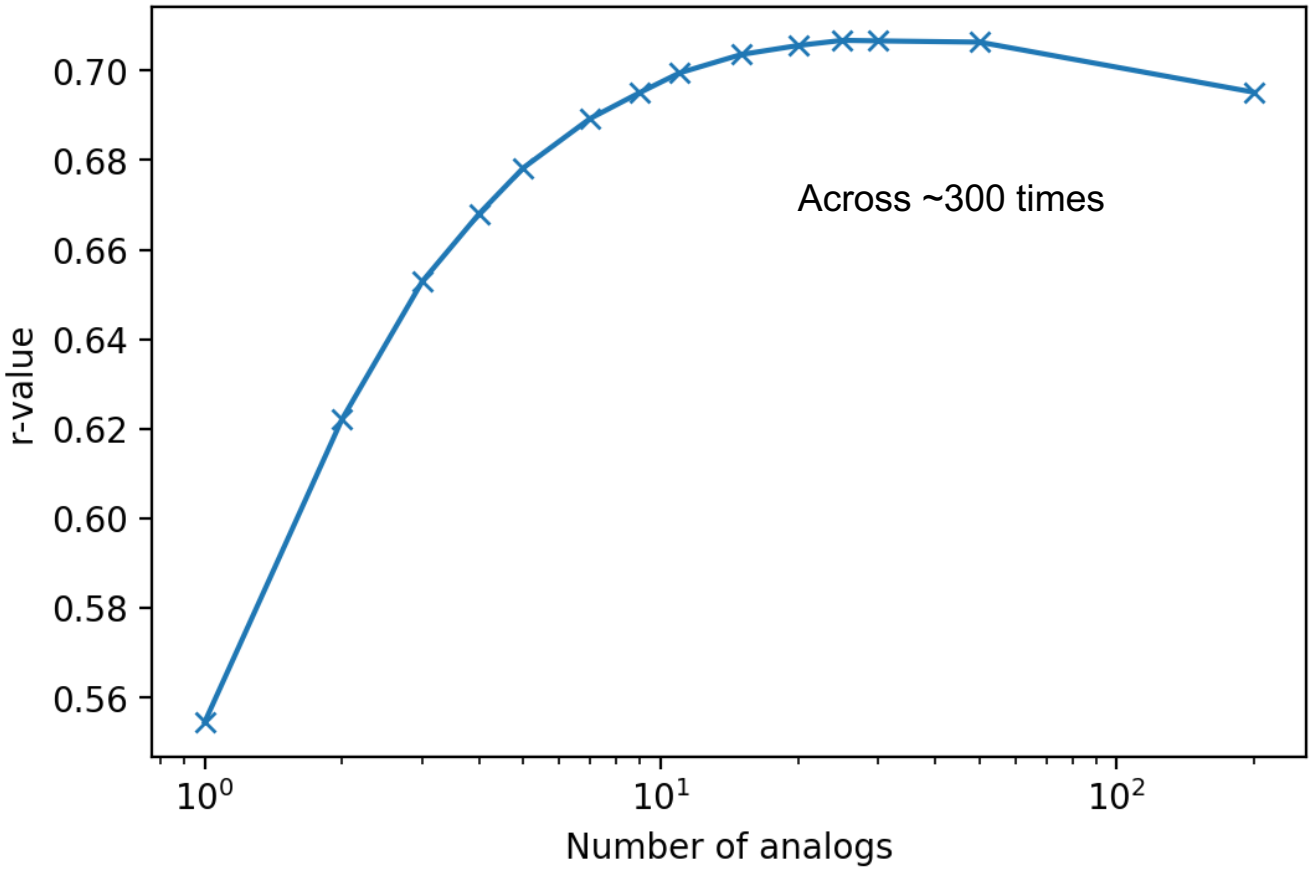
- Scaling across multiple nodes to process
  - Searching ~500,000 possible matches for each of ~500,000 16x16x5 patches
- Initial analog patch investigation
  - Note “blocky” nature
  - No predictions outside of blocks
  - W-field has strong isolated updrafts with associated trailing subsidence
- Error derived from goodness of fit in reflectivity only
  - Even so, where errors are larger, W-fields disagree more
- Need to scale up further… now running ~100,000,000 possible patches

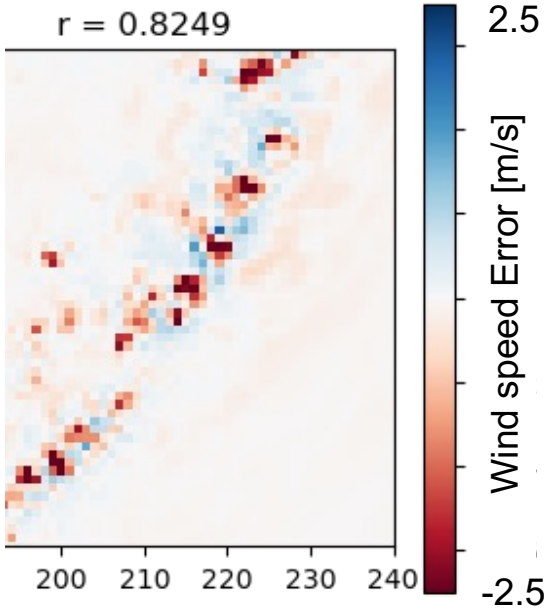


Skill of spatial analogs

- Increasing
- Increasing magnitudes

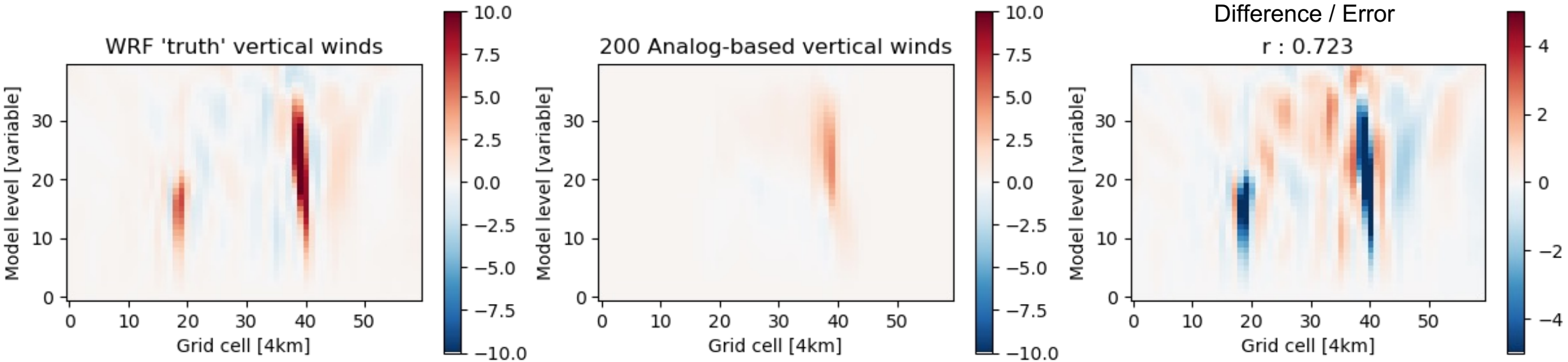


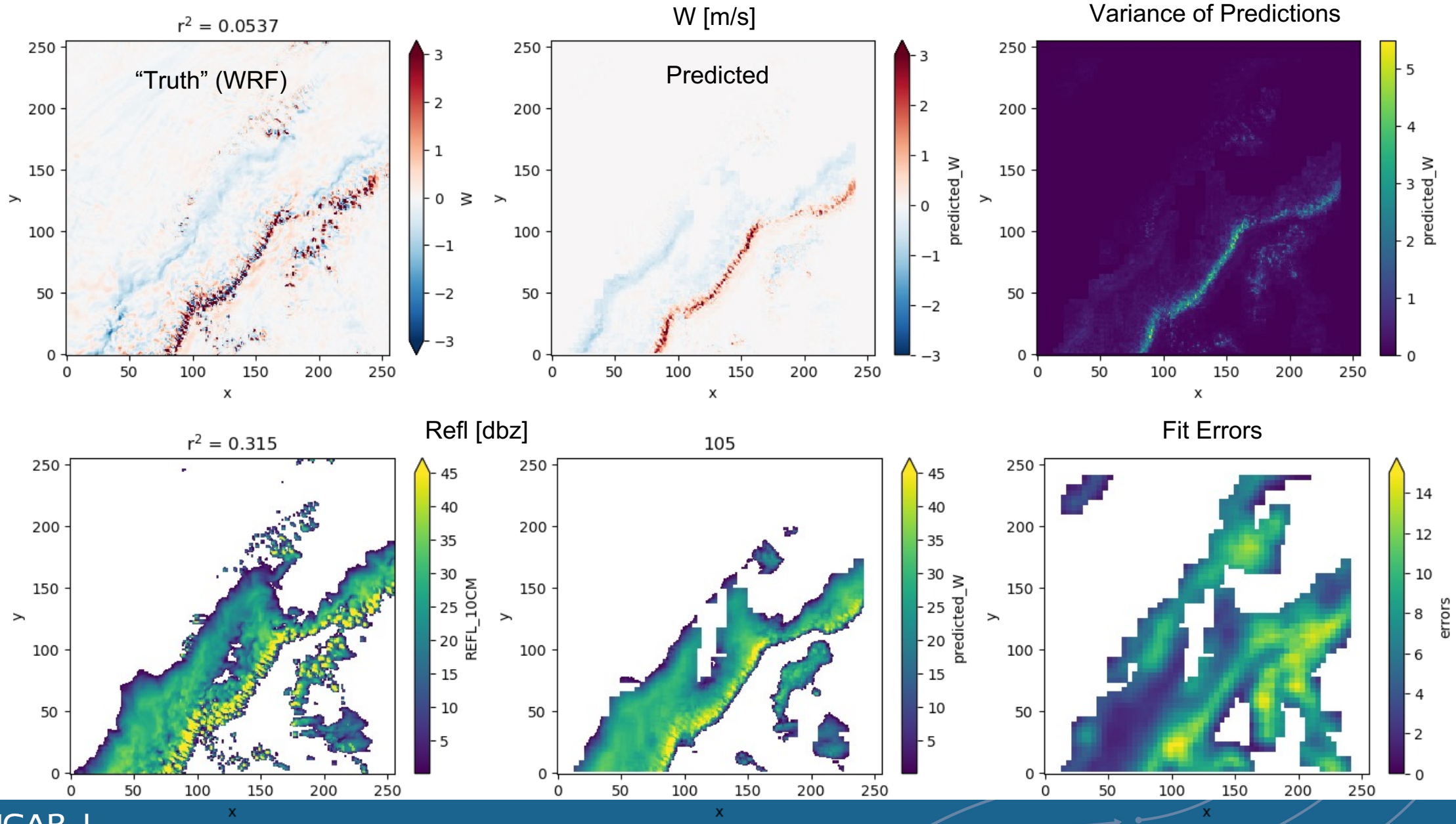




Spatial Analog 3D wind field

- Spatial analogs can provide 3D wind fields
- But DA tests with 3D w from WRF did bad things…

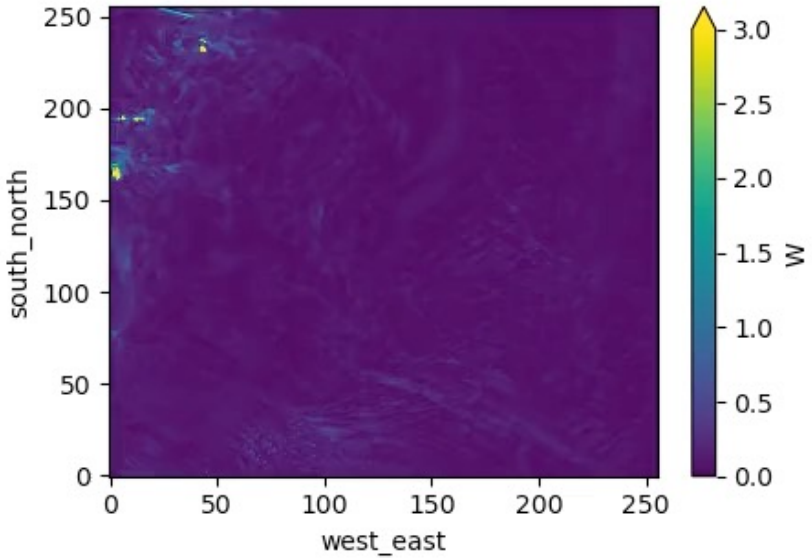



W [m/s]
Variance of Predictions
r^2 = 0.0537
"Truth" (WRF)
Predicted
Refl [dbz]
Fit Errors
r^2 = 0.315
105
x
y
W
predicted_W
REFL_10CM
errors

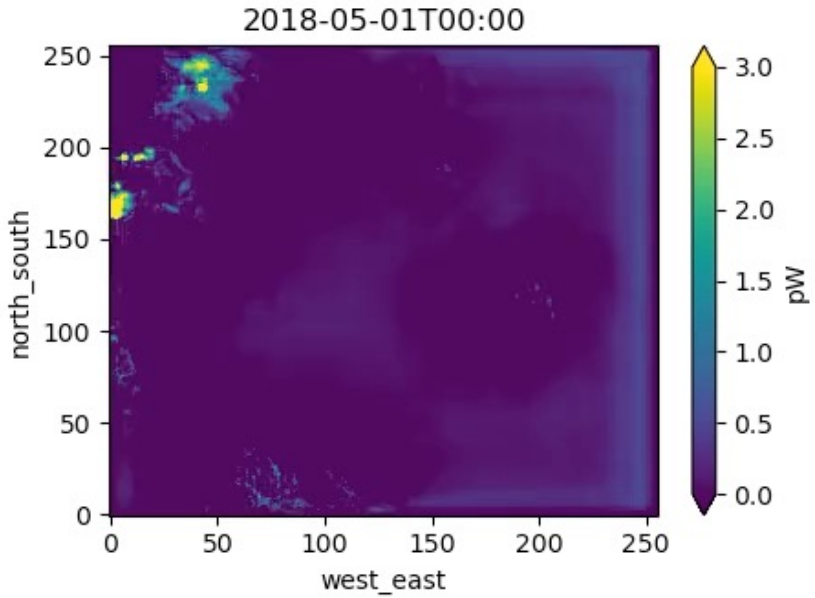
Unet results

- WRF extremes are larger
- UNET is smoother (larger areas with “high” values)
  - Same problem as shallow ML model

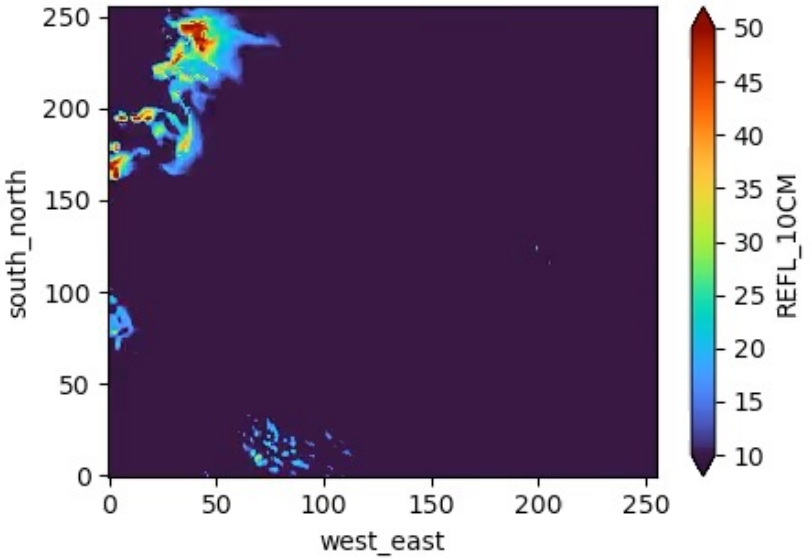
WRF w



UNET predicted-w



WRF reflectivity



Whither PGAS / ATM in an AI world?

- As ML advances, more HPC applications are “written” in Pytorch / Keras
- ML applications “easy” to port to GPUs, TPUs, WaferscaleEngine, and other accelerators
- Multi-node GPU utilization can remain challenging
- ~~Young~~ programmers are lazy
  - Adapt ATM techniques for their world
- ~~Young~~ ~~ML~~ programmers like python
  - Push ATM paradigms for ~~fortran, c++, chapel,~~ python?
  - Think outside the box, no longer “just” communication, e.g. Dask task-graphs

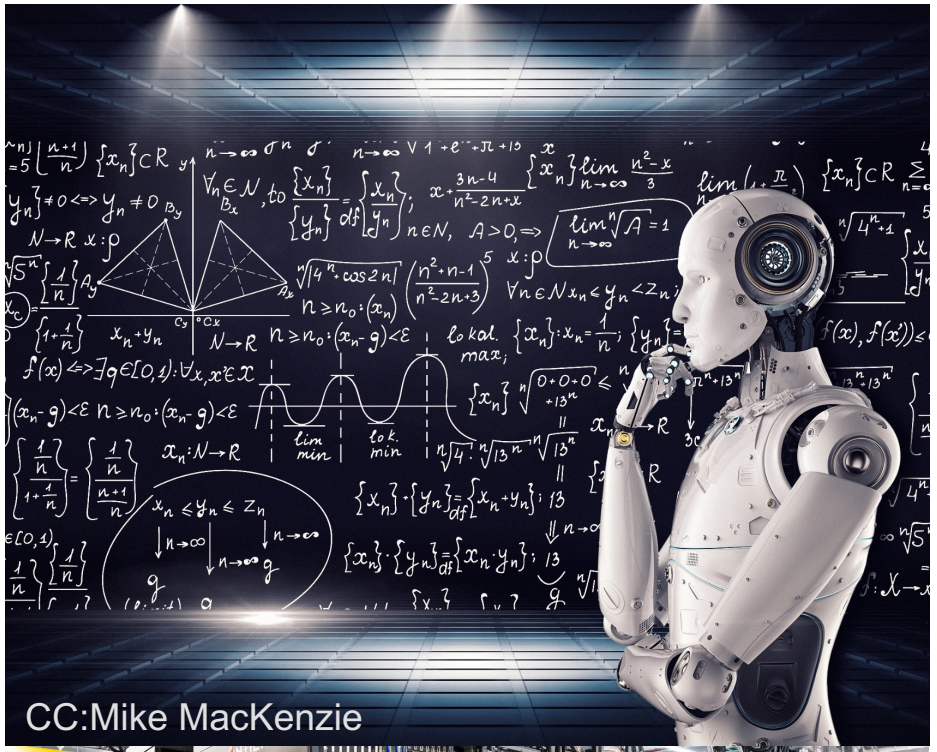
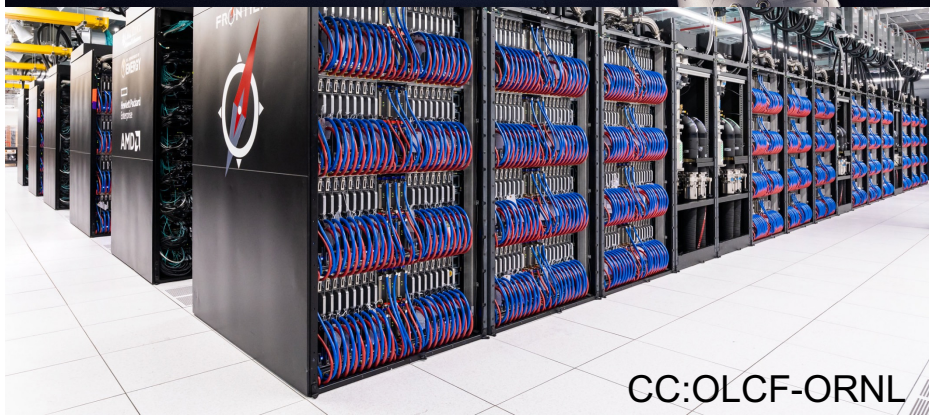


Image via www.vpnsrus.com



Trials and Tribulations (and Joys) of Developing with Coarray Fortran

- Simple communication API
- Early career scientists / students have parallelized real-world applications that perform at scale (thousands of cores)
- Not all compilers support coarrays
- Those that support coarrays have issues
  - Memory limits
  - Speed
  - Documentation
- Issues getting support from HPC centers leaves a bad taste in the mouth