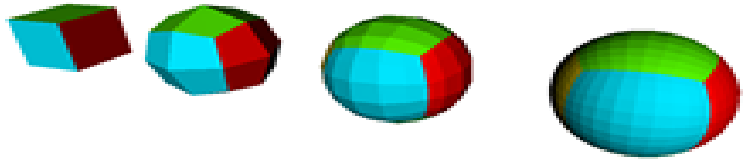
# Interpolating climate data using UV-CDAT




Alex Pletzer, Dave Kindig, and Srinath Vadlamani (Tech-X Corp.) – LibCF/GRIDSPEC

Paul Durack, Charles Doutriaux, Jeff Painter, and Dean Williams (LLNL) – UV-DAT, CMIP5

Ryan O'Kuinghttons, Bob Oehmke (NOAA) – ESMF

pletzer@txcorp.com


**Jan 23 2012, AMS, New Orleans**


**Work funded by MoDAVE: DOE/SBIR DE-FG02-08ER85153**


TECH

# UV-CDAT is the substrate upon which a software ecosystem can be built

- UV-CDAT builds many packages including scipy, ipython, Pmw, PyQt, .... Extending UV-CDAT can be as simple as typing "`python setup.py install`"

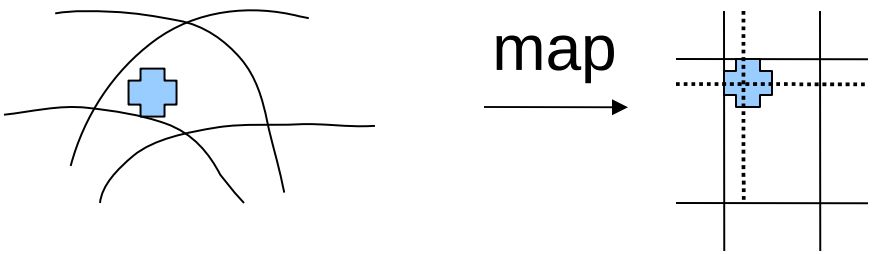
- Examples:
  - `mpi4py` (Message Passing Interface for Python)
  - `petsc4py` (sparse matrix solvers, non-linear equations, time steppers, ...) [Lisandro Dalcin]
  - `PyGNL`
  - `PyLog` (PROLOG engine)
  - `nltk` (Natural Language Toolkit)
  - Wavelets, database, web services, symbolic math, ....

# Currently available options for regridding in Python/UV-CDAT

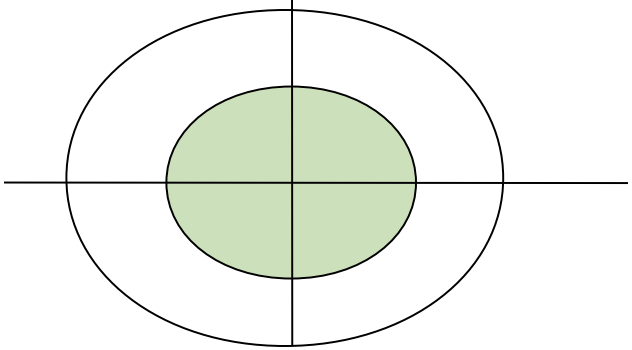
- **regrid2 (in CDAT 5.2)**
  - 2D, horizontal grid is a cross product of axes
- **SCRIP (in CDAT 5.2)**
  - 2D, curvilinear grids, conservative/linear/spline. Lacks documentation (was not able to use)
- **LibCF**
  - Multi-dimensional but only linear (in UV-CDAT 6.0). Interface to C library using `ctypes`.
- **ESMF/ESMP**
  - 2D/3D, option between linear, conservative, Python interface recently made available by Ryan O'Kuinghttons. Interface to C ESMF (ESMC) via `ctypes`.

# LibCF regridding/interpolation

- Linear interpolation using nearest neighbors only
  - No over-shooting
  - Straightforward to parallelize
- Pseudo-Newton search of position in index space
  - Only one iteration required for uniform, rectilinear grids
- Line search to improve convergence
- Use previous index location as initial guess when regridding from structured to structured grids
- Handles dateline, can be anywhere
- Pole remains a problem
- Has support for masking



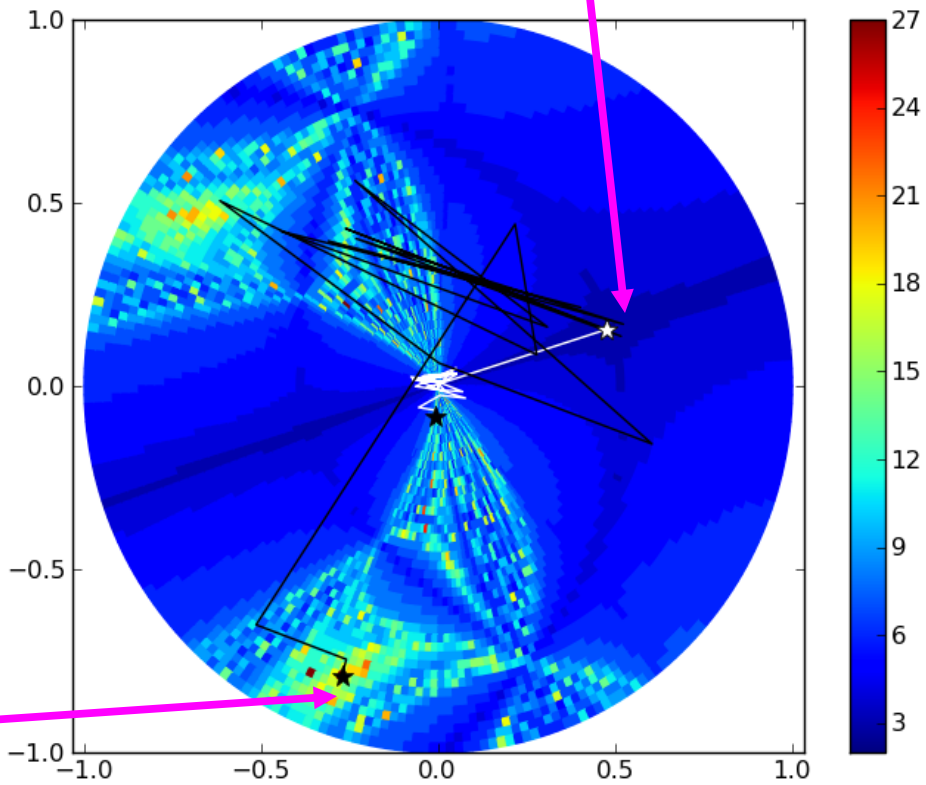

Pathological case has zero cell volume in lon-lat space



# Cell search = chaos

- **Number of iterations is a fractal**
  - **Limit cycles**
  - **Self-similarity**
  - **Basin of attraction**

Target

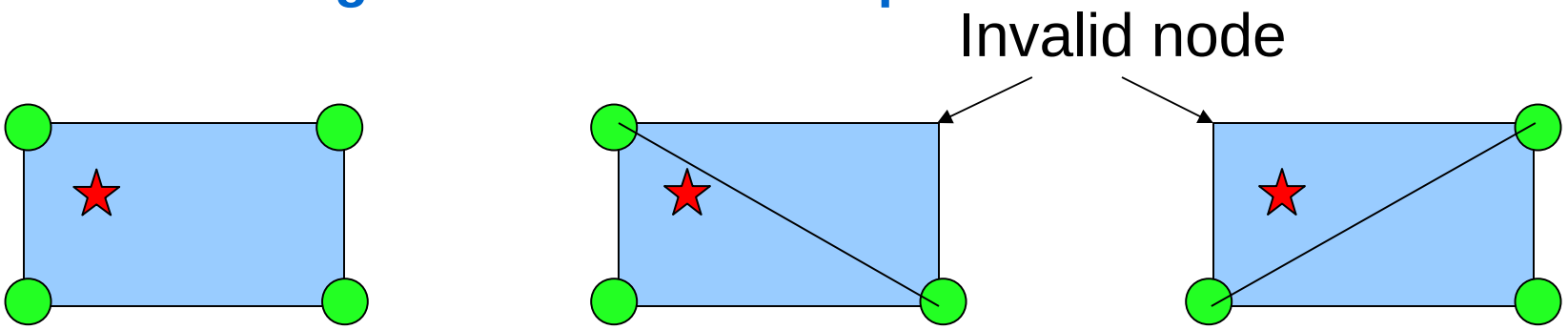



Number of iterations required to locate a target position for a polar to Cartesian map

Initial guess

# How LibCF deals with masking

- **Will do its best to interpolate in the presence of masked (or invalid) values**
- **3 cases:**
  - **All values in a cell are valid**
  - **Some invalid values**
    - **Switch from quadrilateral/hexahedron to triangle/tetrahedron interpolation**




All nodal values are valid

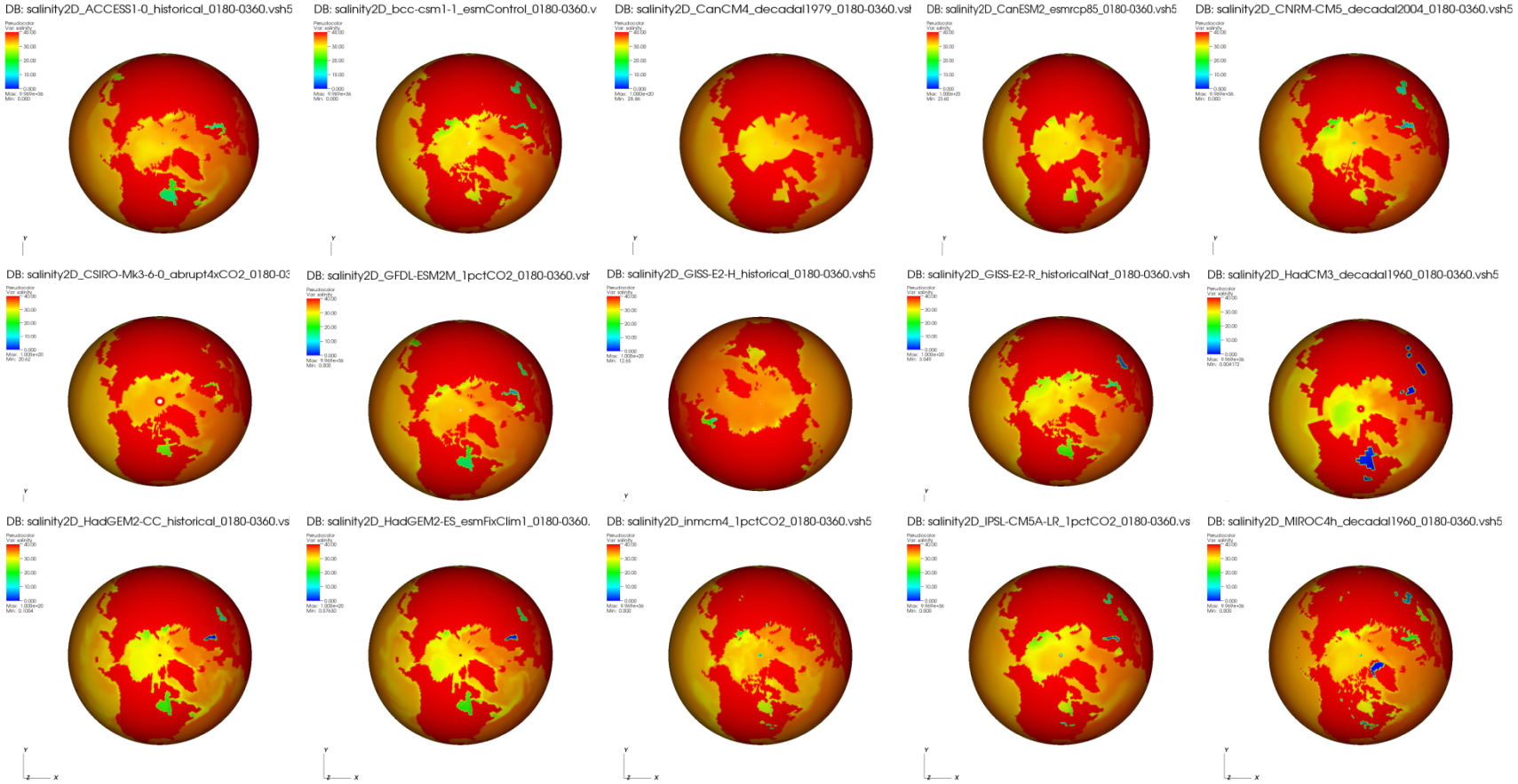
One missing value interpolation is still possible

Not possible to interpolate

# How to call LibCF regrid from UV-CDAT

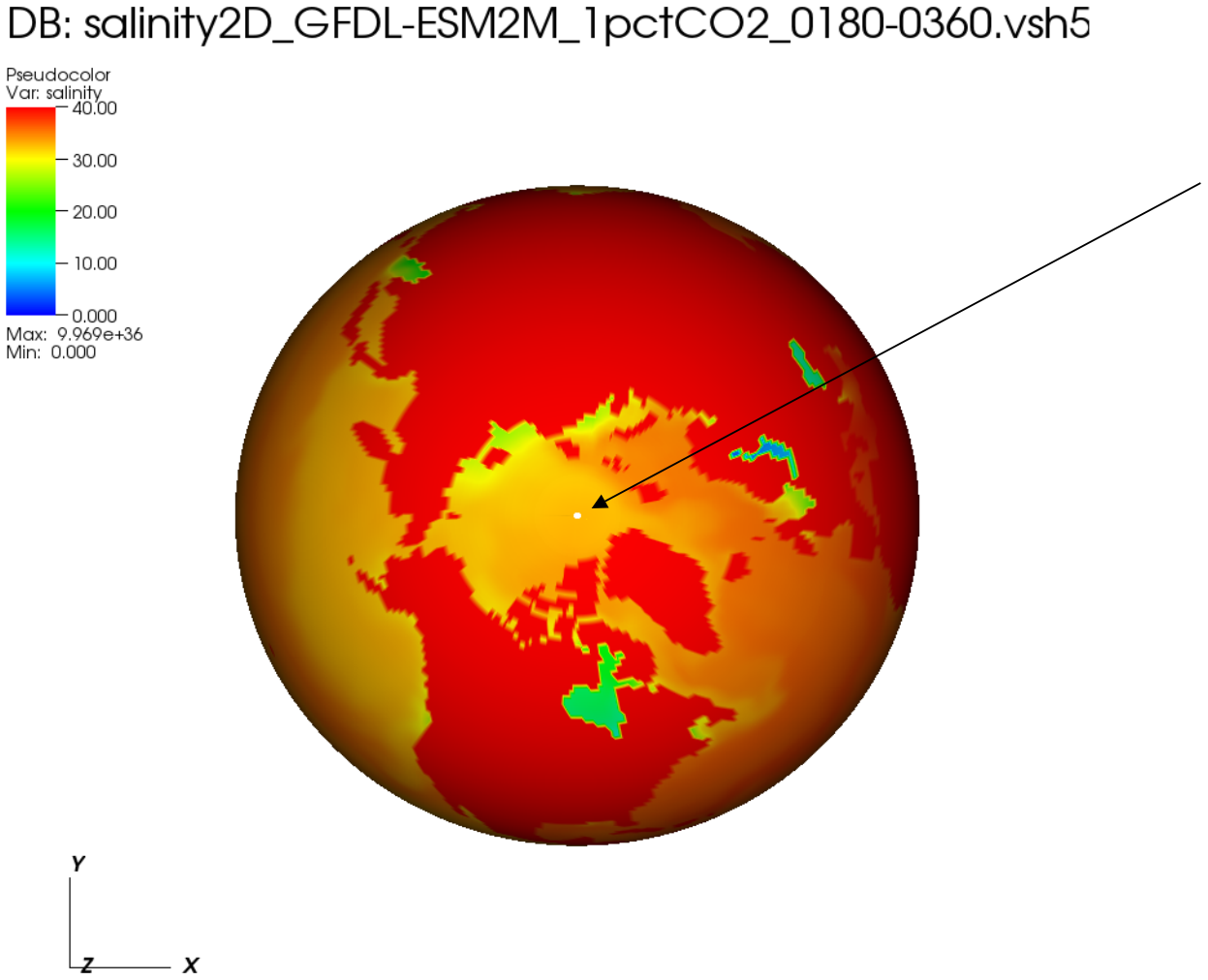
```
from cdms2 import gsRegrid
...
# .... src_y, src_x can be curvilinear coordinates
# or axes, ditto for dst_y, dst_x, ....
# takes numpy or cdat cdms2 type variables
src_grd = [..., src_y, src_x]
dst_grd = [..., dst_y, dst_x]

# constructor
rg = gsRegrid.Regrid(src_grd, dst_grid,
                     mkCyclic = False,
                     handleCut = False,
                     src_bounds = None)

# compute interpolation weights
rg.computeWeights(nitermax=20, tolpos=0.01)

# interpolate src_field, result is dst_field
rg(src_var, dst_var)
```

# LibCF: 2D interpolation was tested on 23 ocean models




... etc.

# LibCF: GFDL model was made cyclic and additional row was added to fill in gap

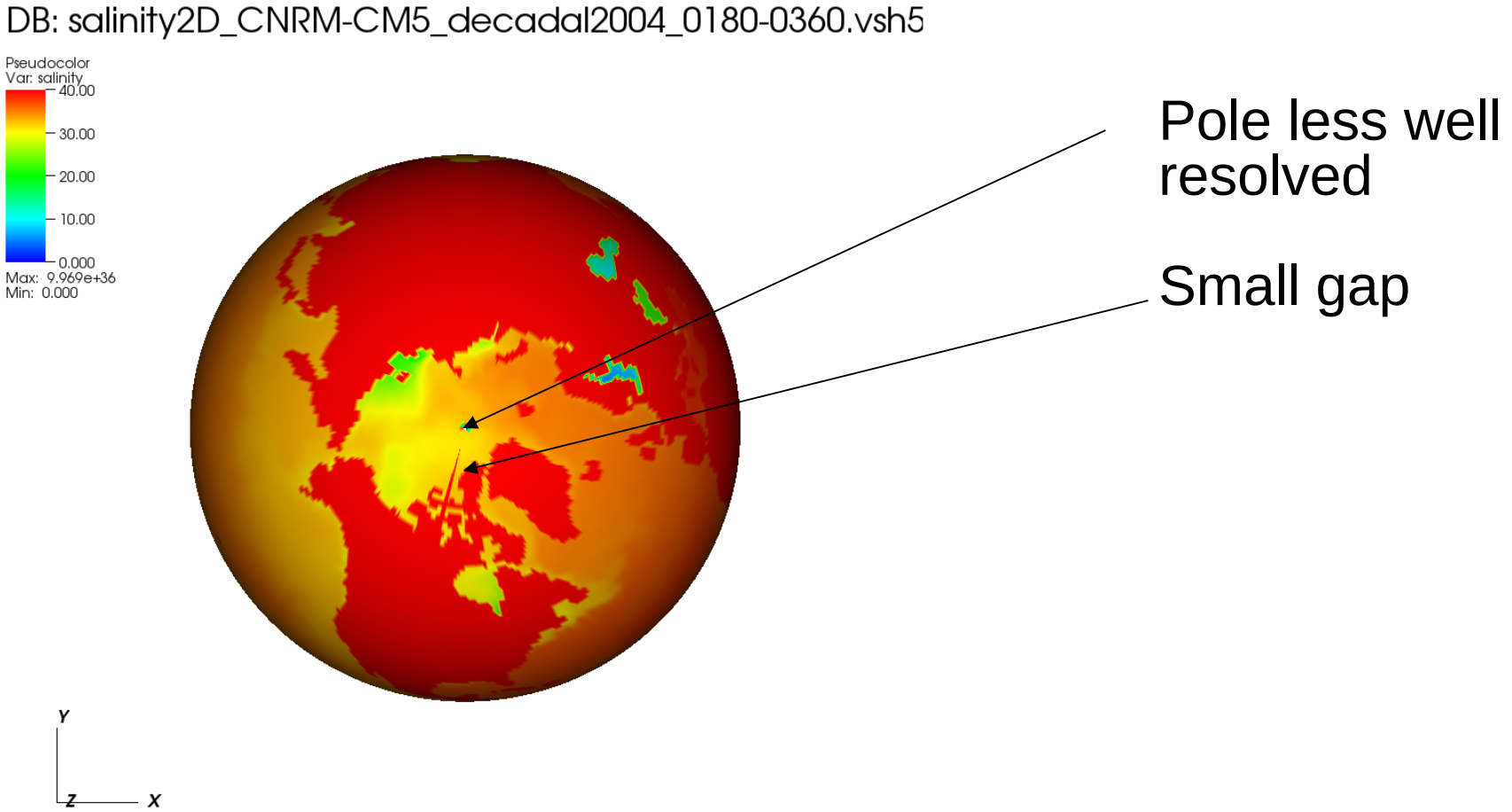



Pole is well resolved

Tripolar grid, no Gap

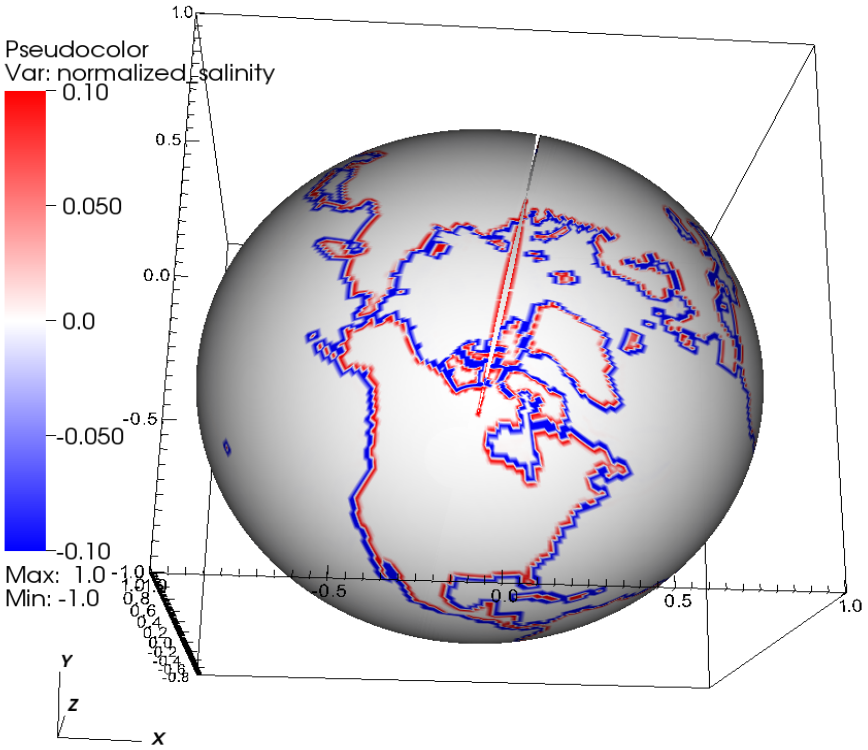
No dateline problem

# LibCF: interpolation of CNRM model shows small gap



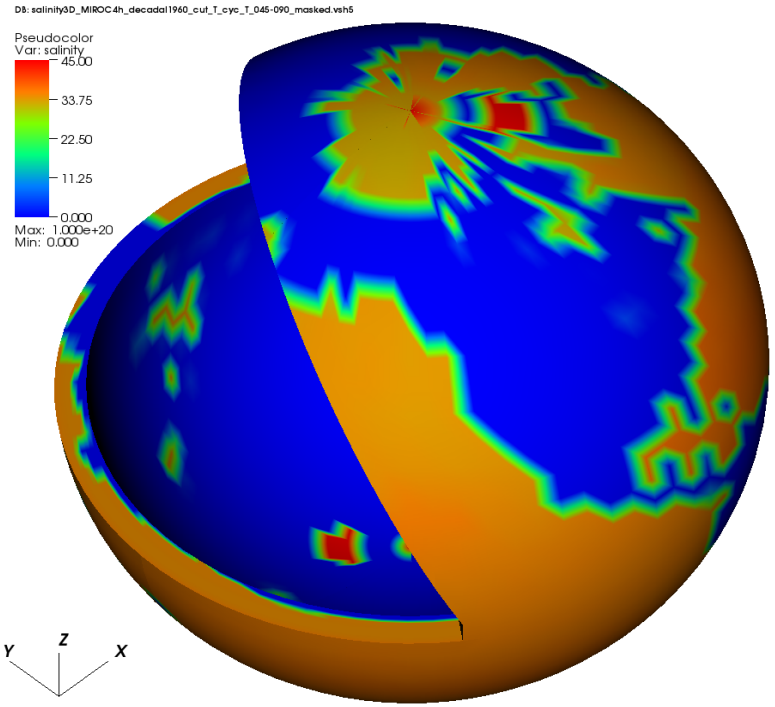
# Interpolation error after interpolating back onto the source grid

- Error is mostly near the coast line



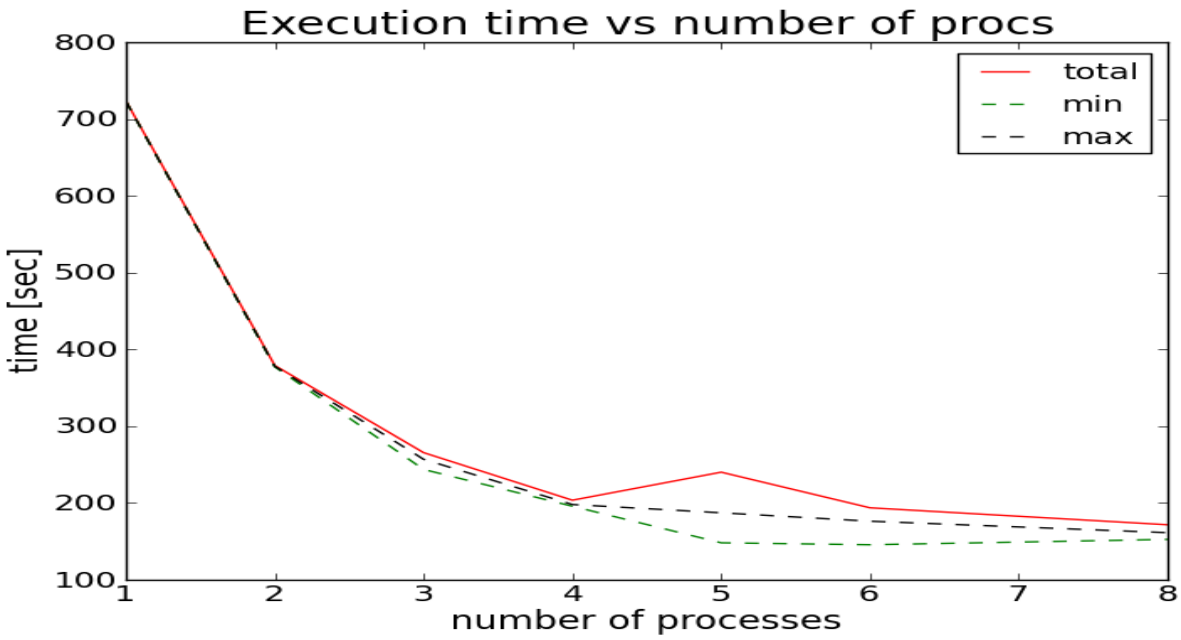
# LibCF: 3D test cases

- Takes ~ 20-60 seconds (only 10 levels)
- MIROC hi-res model



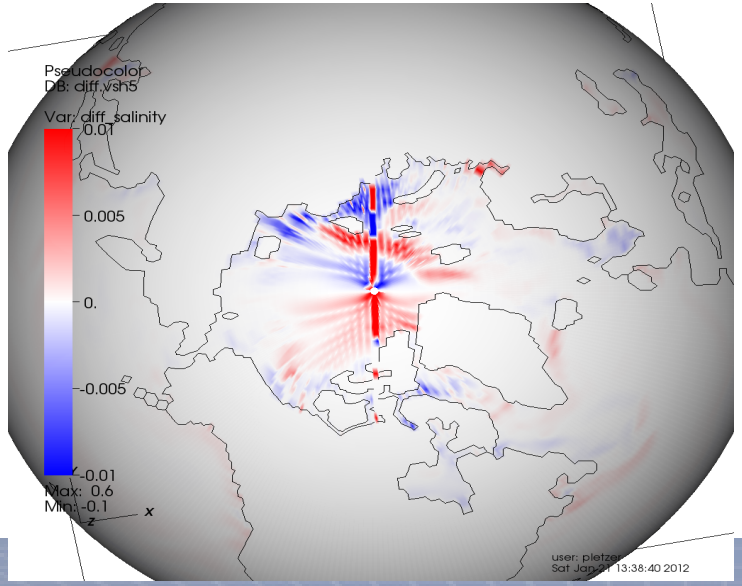
# Can be easily parallelized using (e.g.) mpi4py

- Embarrassingly parallel, use for instance a decomposition in longitudes
- Example of speedup on a 8-core workstation (3D)
- Load balancing is the limit



# How does ESMF/ESMP regridding compare to LibCF?

- Full interface (multi-linear, conservative, and patch) available in Fortran 90 and C. First order and higher order supported for nodal, conservative and patch interpolation.
- Python interface currently restricted to multi-linear interpolation and 8-byte floats (likely a temporary limitation)
- Distinguishes between topological (parametric) and space dimensions. This allows ESMF regridding to overcome the problem at the pole.




Difference between LibCF and ESMF Interp. ~ 1% or less

# Summary

- Highly distorted lat-lon grids present challenges for interpolation software
  - Cuts
  - Jump in longitude
  - Pole
- LibCF interpolation has benefited from being exposed to “real” datasets
- Timings: `gsRegrid` takes ~ few seconds for 2D, ~40 seconds for 3D
- Can apply domain decomposition and MPI parallelization to accelerate weight computation (embarrassingly parallel)
- Lack of conservation ~ 2%. Can be “fixed” globally by multiplying weights by a constant factor

# Summary (2)

- ESMF interpolation likely to offer best solution when conservation is required
  - Actively working with ESMF developers to extend Python API
  - Work by Peggy Li [ESMF Offline Regrid Generator Performance Comparison with SCRIP] shows good scalability and accuracy for atmospheric model