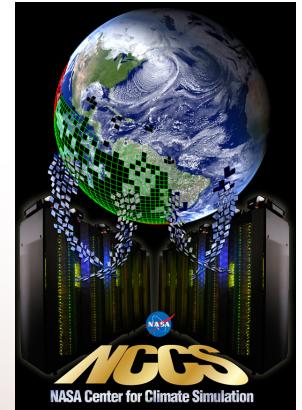


National Aeronautics and Space Administration



# SCIENCE

## Exploratory Climate Data Visualization and Analysis

by

Thomas Maxwell, Jerry Potter & Laura Carriere, NASA NCCS

and the UVCDAT Development Consortium



[www.sci.utah.edu](http://www.sci.utah.edu)



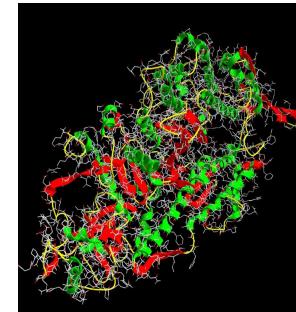
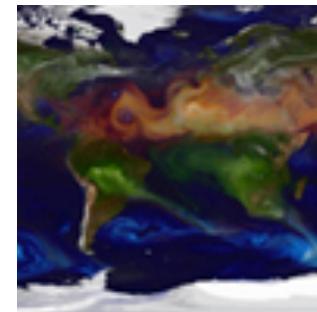
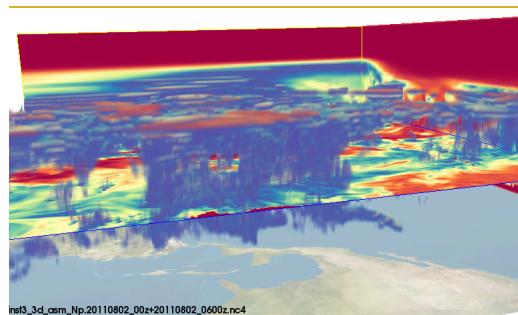
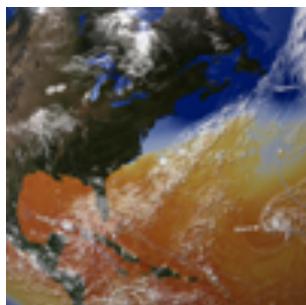
# Scientific Visualization

We process, understand, and analyze information visually.

- 30% of the cerebral cortex is devoted to visual information processing.
- The brain's capacity to detect visual patterns is invaluable in data analysis and understanding.
- Extremely difficult to emulate using statistical and machine learning approaches alone.

Visual representations are evolving

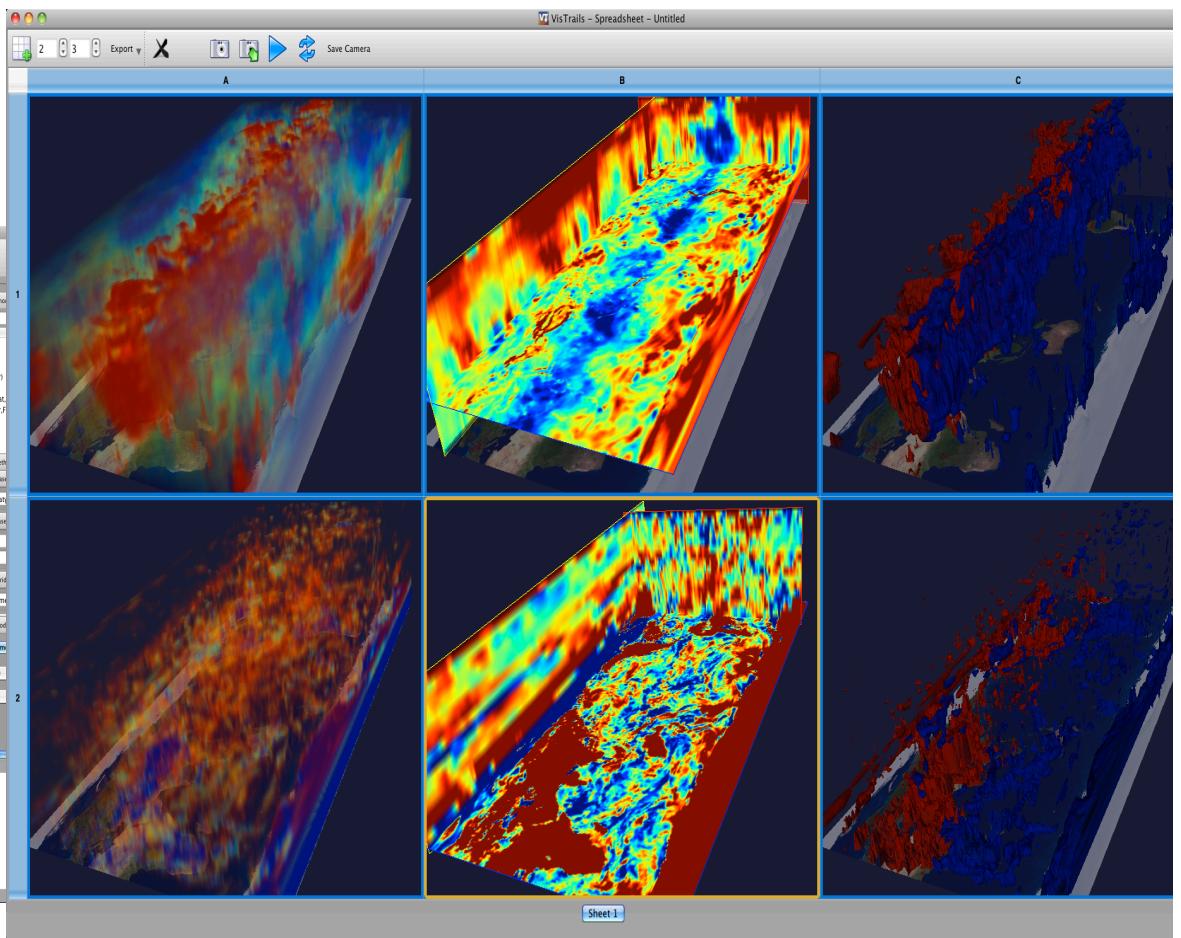
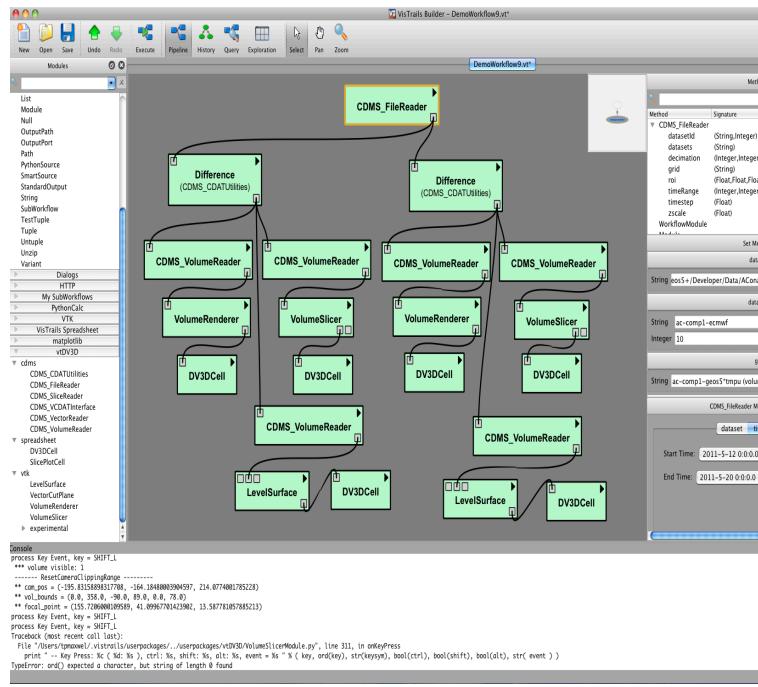
- Static 2D graphing and plotting => interactive 3D rendering.
- Driven by need to analyze complex multidimensional data.
- Calls for higher dimensional visuals and human perspective.
- Requires new tools for the scientific workbench.
- Must be seamlessly integrated and easy to use.



# DV3D (vcs3D): Interactive 3D Visualization



- High level analysis and visualization modules for Vistrails and vcs.
  - Encapsulates complex VTK visualization operations.
  - Greatly simplified API ( wrt raw VTK ).
  - Interactive configuration.
  - Tight UVCAT integration.



# UVCDAT Interaction Layers

- Workflow
- Script
- GUI
- Remote

**Script**

```
vslicer = load_workflow_as_function('vtvd3d.vt','slicer')
vslicer(variable='Relative_humidity')
vrender = load_workflow_as_function('vtvd3d.vt','vr')
vrender(variable='Relative_humidity')
```

**Provenance**

**Workflow**

ESGF Data Archive

ParaView Cluster  
parallel processing

# Workflow and Provenance Support

The screenshot displays the VisTrails software environment. On the left, the 'VisTrails Builder' window shows a workflow graph titled 'untitled.vt'. The graph consists of several nodes connected by arrows: an 'open' node leads to a 'Variable' node, which then connects to a 'GraphicsMethod' node. The 'GraphicsMethod' node has two outputs, one of which connects to a 'CDATCell' node. The 'Basic Modules' panel on the left lists various Python modules. On the right, there are three windows: 1) 'VisTrails - Spreadsheet - Untitled' showing three panels (A, B, C) with global maps of total cloudiness; 2) 'The Visual Climate Data Analysis Tools - (VCDAT)' showing plot configuration for time, latitude, and longitude; 3) 'VisTrails Shell' showing a Python shell session. A red arrow points from the 'VisTrails Shell' window up towards the 'UV-CDAT' window, indicating a connection between them. A large orange callout box contains the text: 'Interacting with the UV-CDAT window or shell automatically generates provenance'.

VisTrails shell running Python 2.6.3 (r263:75184, Oct 2 2009, 07:56:03)  
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin.  
Type "copyright", "credits" or "license" for more information on Python.  
>>> import vcs

\*\*\* Description of Slab clt \*\*\*  
id: clt  
shape: (120, 46, 72)  
filename: /lsm/cdat/VT/Python.framework/Versions/2.7/sample\_data/clt.nc  
missing\_value: None  
comments: YONJAS5  
grid\_name: YONJAS5  
grid\_type: gaussian  
time\_statistic: average  
long\_name: Total cloudiness  
units: %  
Cgrid has Python id 0x1de30470.

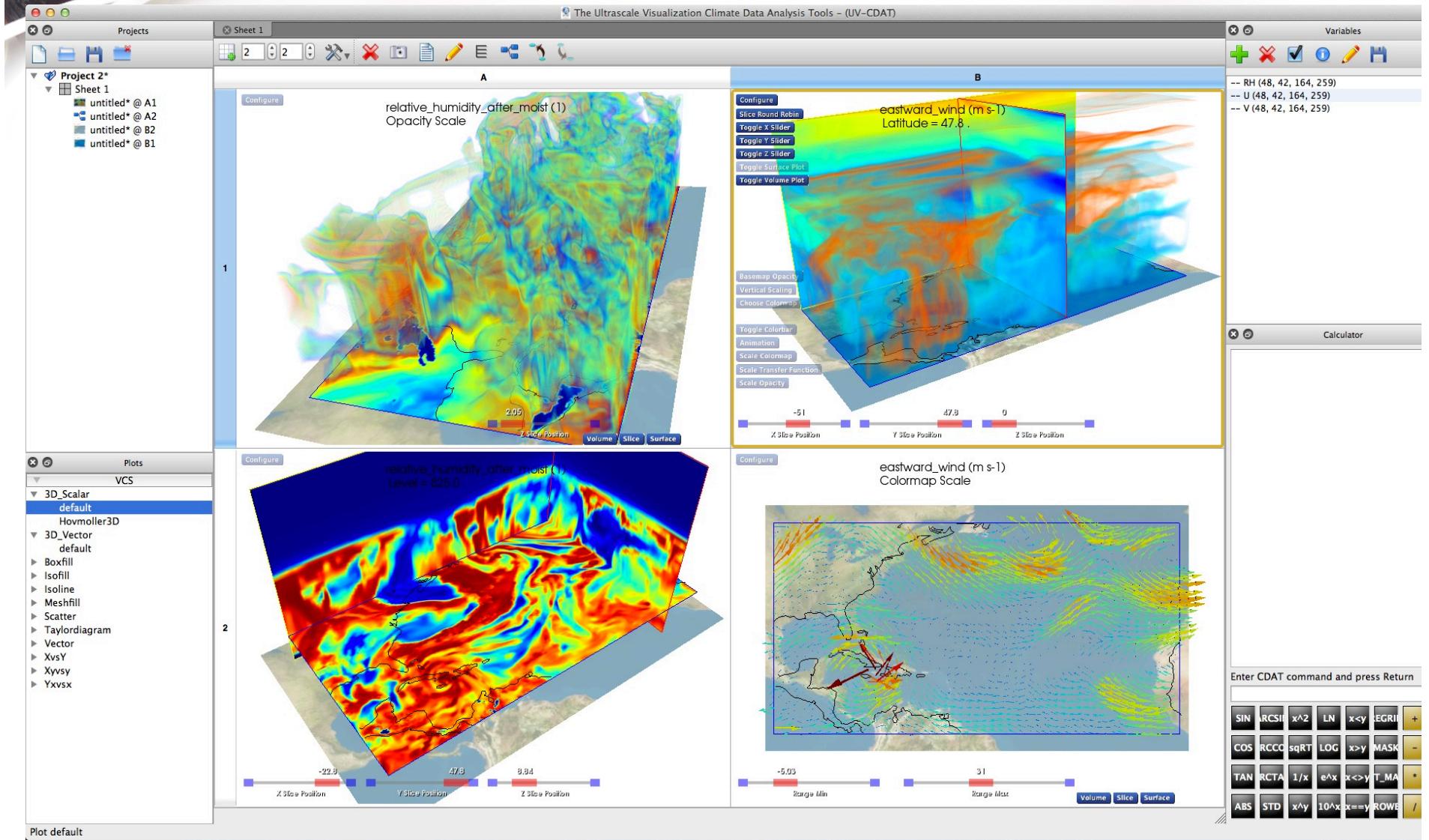
VisTrails Shell

\*\* Dimension 2 \*\*  
id: latitude  
Designated a latitude axis  
units: degrees\_north  
Length: 46

Interacting with the UV-CDAT window or shell automatically generates provenance

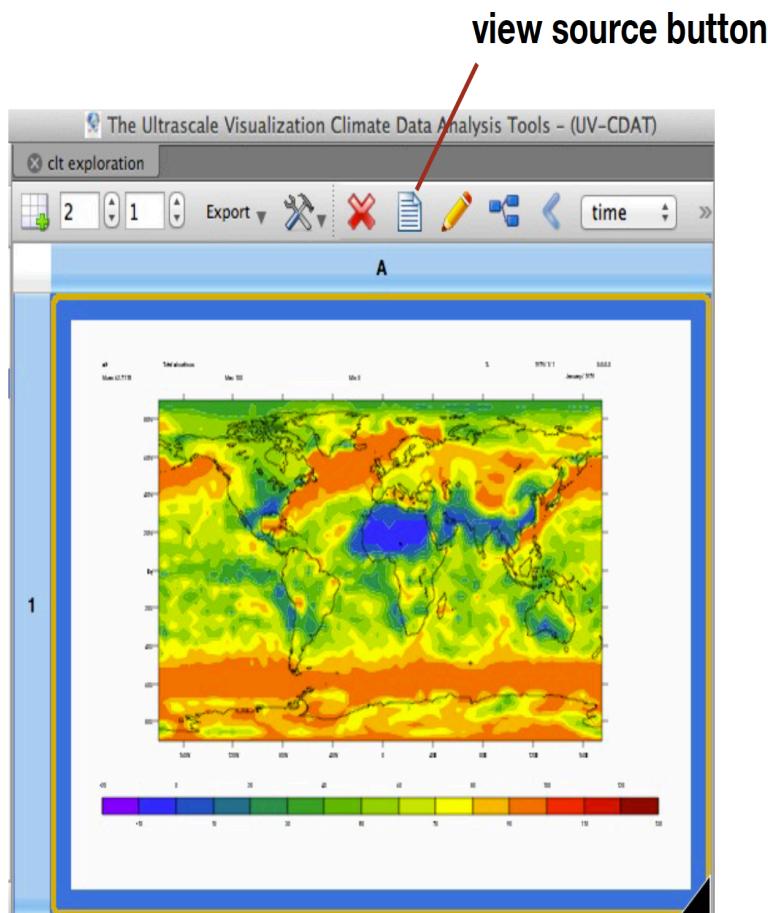


# vcs3D in the UVCDAT GUI



# Automating Workflows

- Workflows can be converted to python scripts
- Run outside of uvcdat in a python shell.



```
Visualization Source
clt exploration @ A1

from PyQt4 import QtCore, QtGui
import cdms2, cdutil, genutil
import vcs

if __name__ == '__main__':
    import sys
    app = QtGui.QApplication(sys.argv)
    cdmsfile = cdms2.open('/Volumes/home/emanuele/Desktop/data/clt.nc')
    clt = cdmsfile('clt')
    clt = clt(latitude=(-90.0, 90.0), squeeze=1, longitude=(-180.0, 175.0))
    axesOperations = eval("{'latitude': 'def', 'longitude': 'def', 'time': 'def'}")
    for axis in list(axesOperations):
        if axesOperations[axis] == 'sum':
            clt = cdutil.averager(clt, axis='(%s)'%axis, weight='equal')
        elif axesOperations[axis] == 'avg':
            clt = cdutil.averager(clt, axis='(%s)'%axis, weight='equal')
        elif axesOperations[axis] == 'wgt':
            clt = cdutil.averager(clt, axis='(%s)'%axis)
        elif axesOperations[axis] == 'gtm':
            clt = genutil.statistics.geometricmean(clt, axis='(%s)'%axis)
        elif axesOperations[axis] == 'std':
            clt = genutil.statistics.std(clt, axis='(%s)'%axis)
    canvas = vcs.init()
    gmIsosfill = canvas.getisofill('ASD')
    args = []
    args.append(clt)
    gmIsosfill.datawc_calendar = 135441
    gmIsosfill.datawc_timeunits = 'days since 2000'
    gmIsosfill.datawc_x1 = 1.00000002004e+20
    gmIsosfill.datawc_x2 = 1.00000002004e+20

Copy to clipboard Save to file
```

# vcs3D Scripting: Slicing

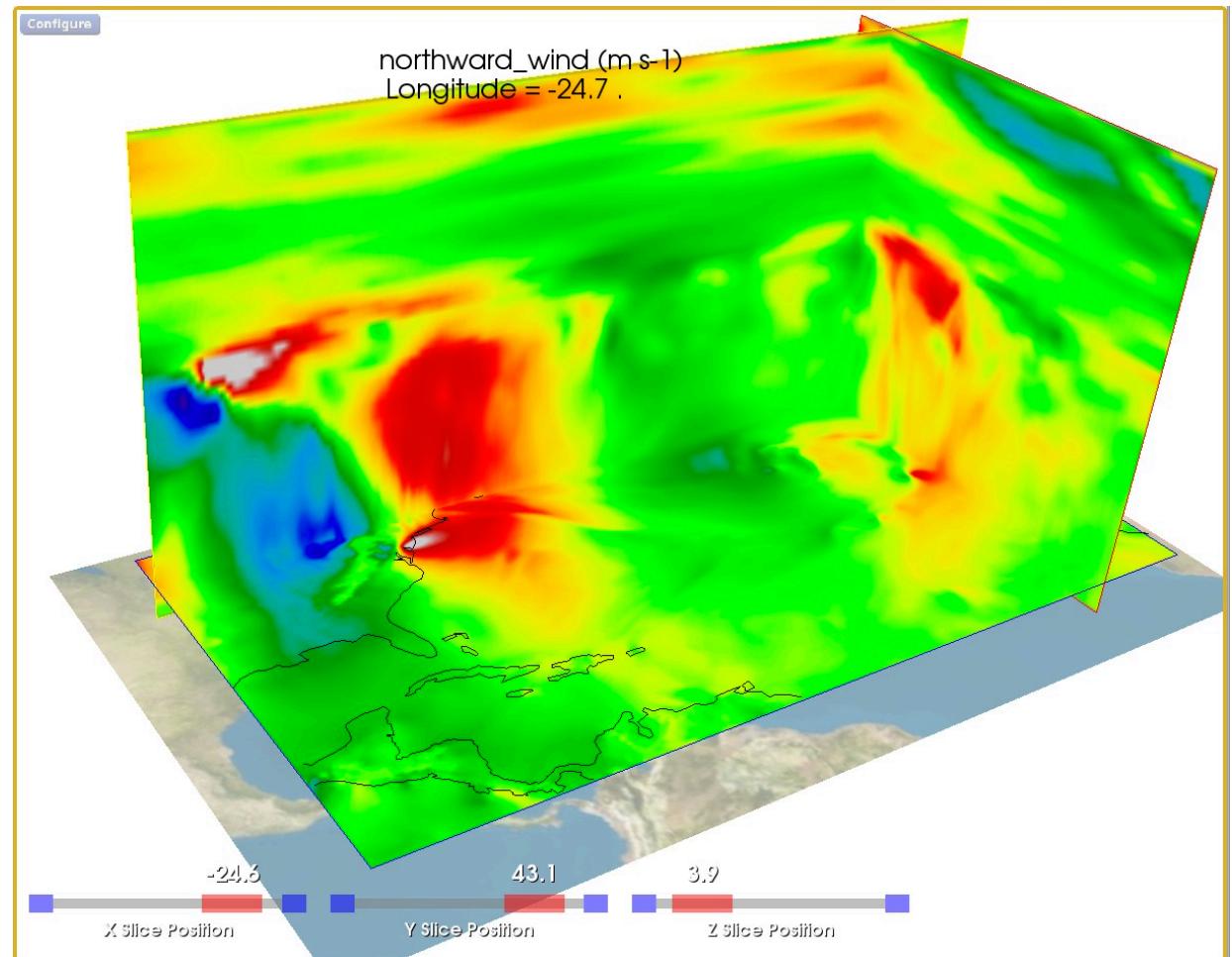
```
import sys, vcs, cdms2

x = vcs.init()
f = cdms2.open( "Sandy.xml" )
dv3d = vcs.get3d_scalar()

dv3d.Camera={
    'Position': (-136, -70, 86),
    'ViewUp': (.23, .46, .85),
    'FocalPoint': ( -57, 34, 7 ) }

dv3d.ScaleColormap= [ -45, 28 ]
dv3d.YSlider = ( 44, vcs.off)
dv3d.XSlider = ( -24, vcs.on )
dv3d.ZSlider = ( 0.07, vcs.on )
dv3d.ChooseColormap='spectral'

v = f[ "wnd" ]
x.plot( v, dv3d )
x.interact()
```





# vcs3D Scripting: Simple Volume Render

```
import sys, vcs, cdms2

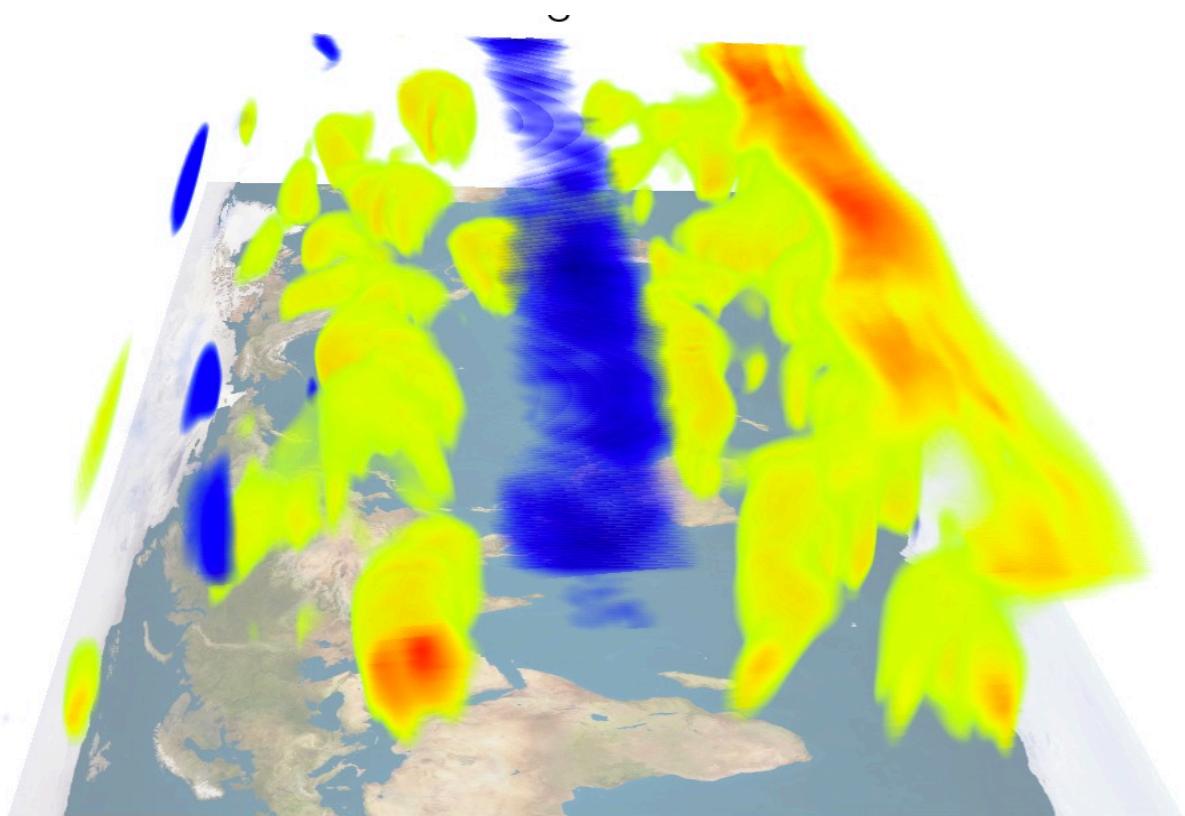
x = vcs.init()
f = cdms2.open( sys.prefix+{/sample_data/geos5-sample.nc" } )

dv3d = vcs.get3d_scalar()
dv3d.ToggleVolumePlot = vcs.on
dv3d.Camera={

    'Position': (-300, 0, 300),
    'ViewUp': (.3, .7, .7),
    'FocalPoint': ( 150, 0, 0 )}

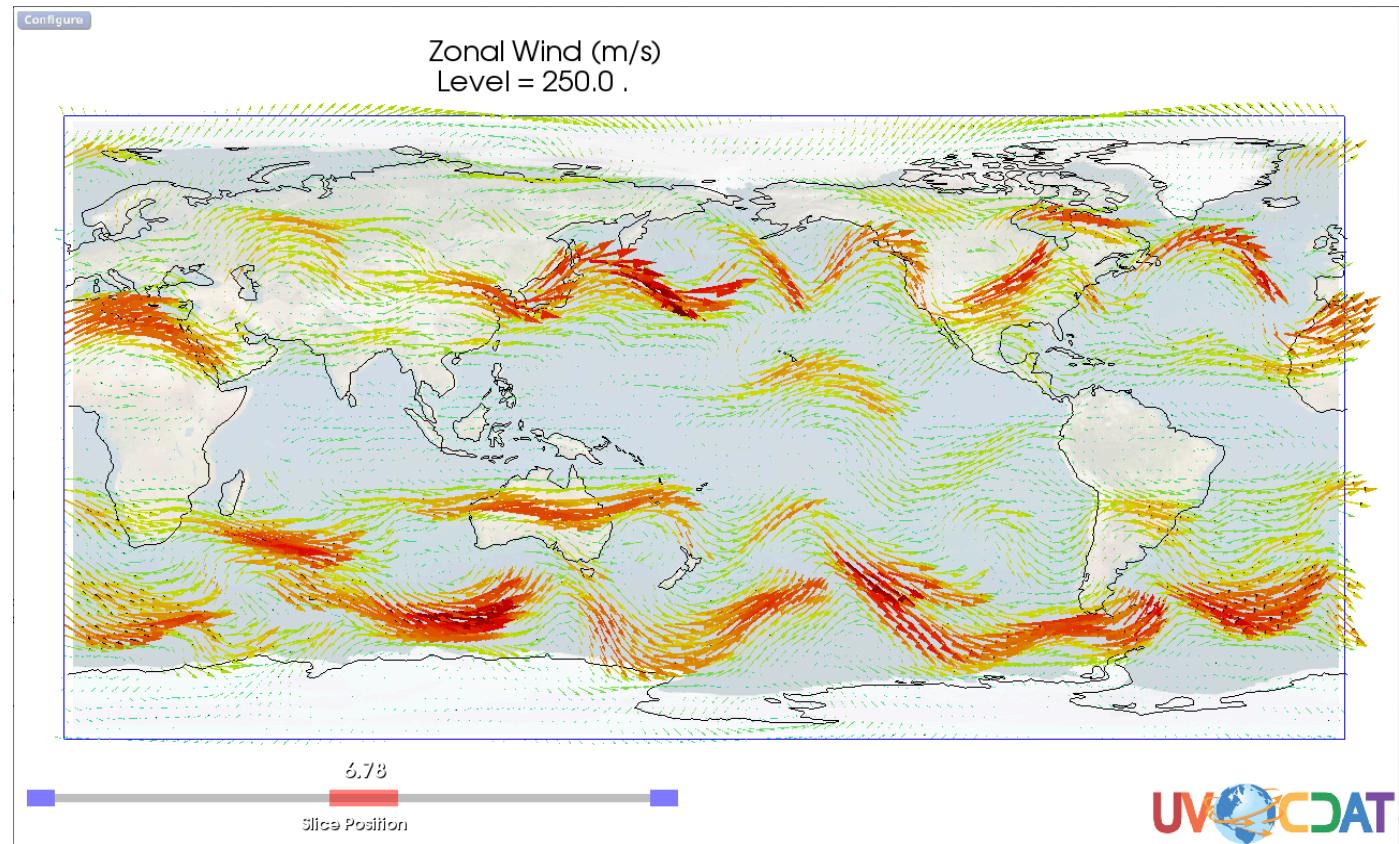
dv3d.VerticalScaling = 4.0
dv3d.ScaleTransferFunction = [10.0, 77.0]

v = f[ "uwnd"]
x.plot( v, dv3d )
x.interact()
```



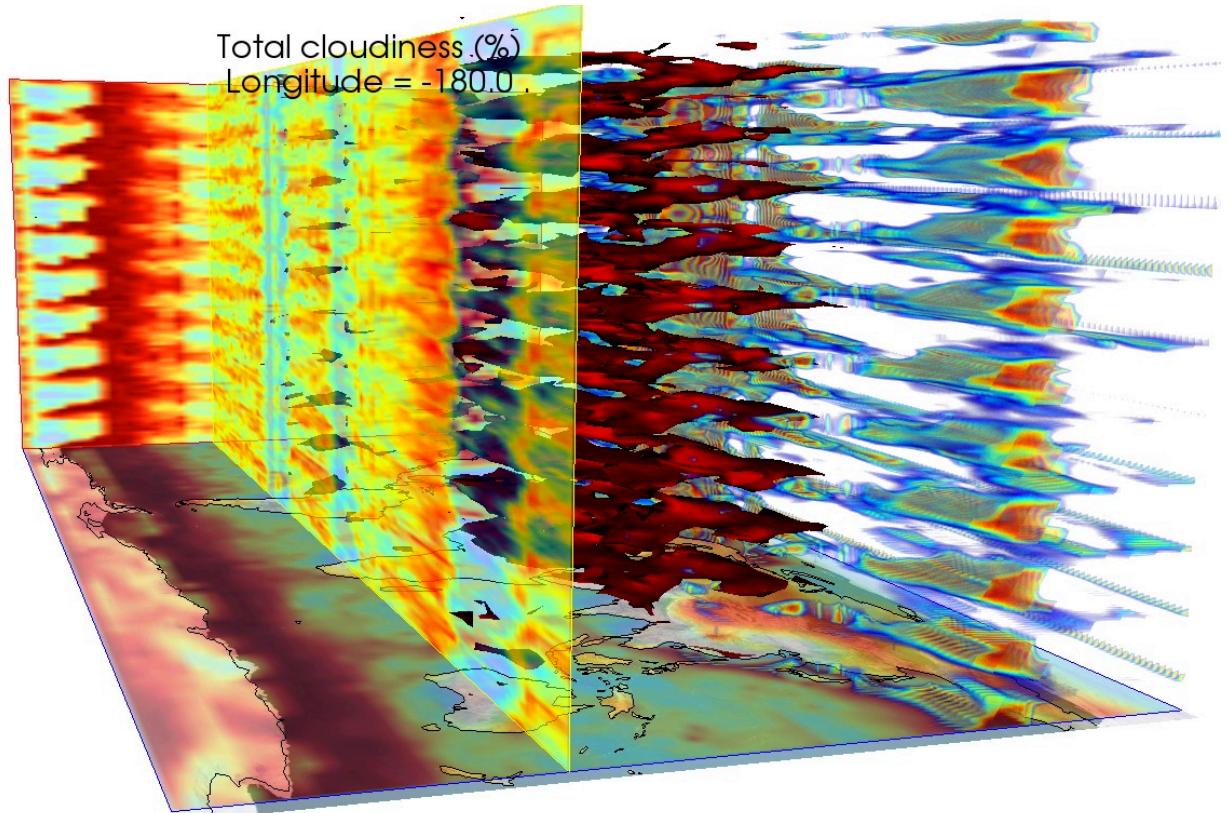
# vcs3D Scripting: Vector Plots

```
import vcs, cdms2, sys  
x = vcs.init()  
f = cdms2.open(sys.prefix+"/sample_data/geos5-sample.nc")  
v = f["vwnd"]  
u = f["uwnd"]  
dv3d = vcs.get3d_vector()  
dv3d.GlyphDensity = 3.5  
dv3d.GlyphSize = 0.27  
x.plot( u, v, dv3d )  
x.interact()
```



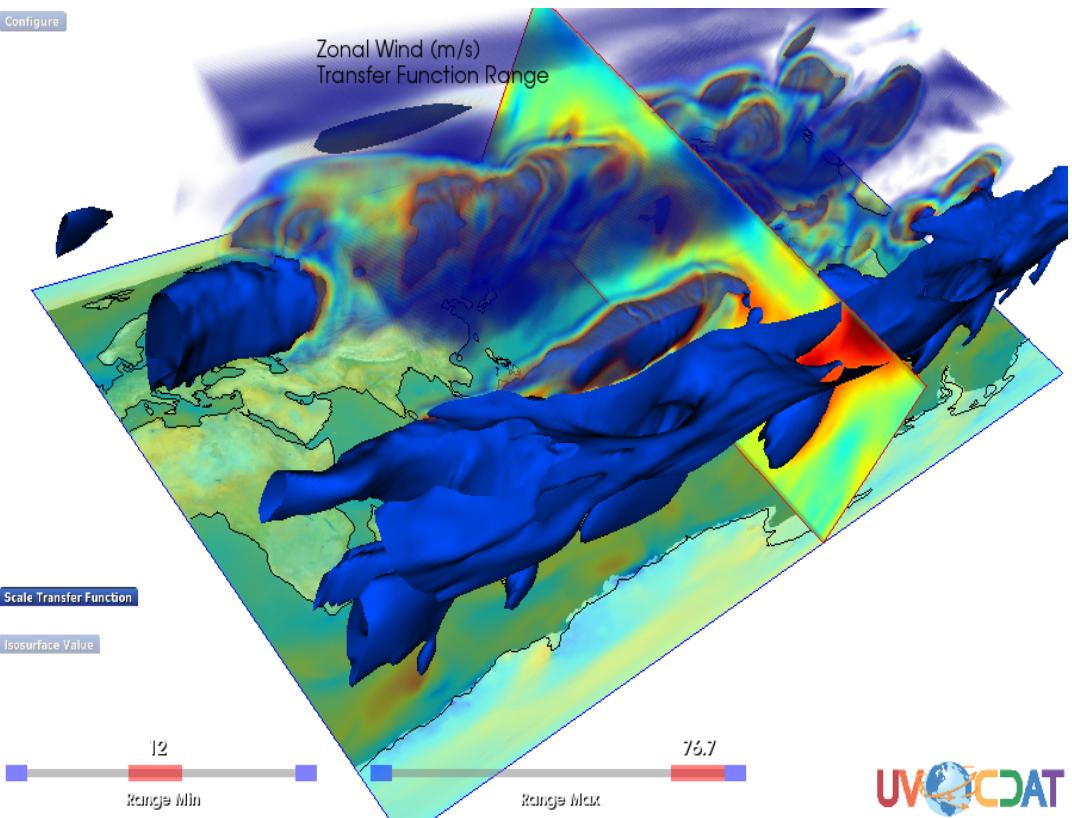
# vcs3D Scripting: Hovmoller Plots

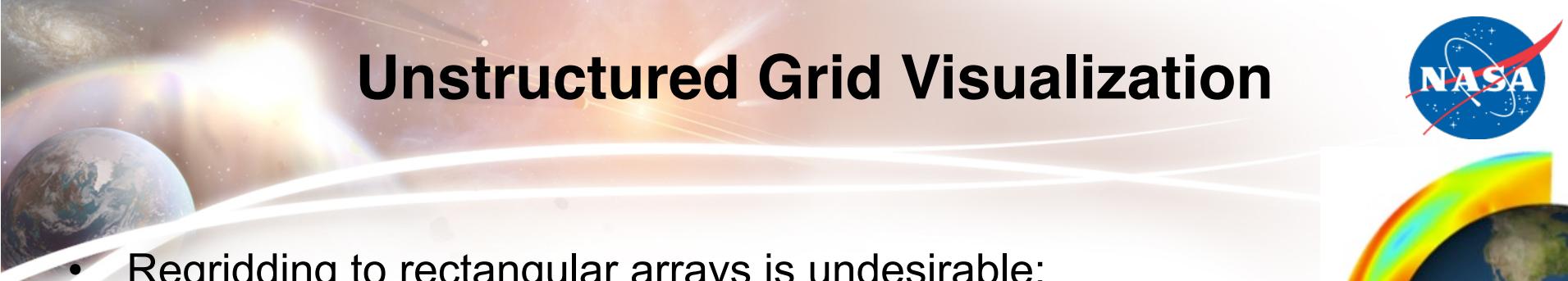
```
import vcs, cdms2, sys
x = vcs.init()
f = cdms2.open(sys.prefix+"/sample_data/clt.nc")
v = f["clt"]
dv3d = vcs.get3d_scalar('Hovmoller3D')
dv3d.ToggleSurfacePlot = vcs.on
dv3d.ToggleVolumePlot = vcs.on
dv3d.IsolurfaceValue = [ 10.0 ]
dv3d.ToggleClipping =
    [-180.0, 175.0, -22.0, 90.0, 0.0, 119.0 ]
dv3d.ScaleTransferFunction = [80, 100, 1]
dv3d.ScaleOpacity={ 'Volume': [0.0, 0.2] }
dv3d.YSlider = ( 20.0, vcs.on )
dv3d.Camera={
    'Position': (436.8, -126.3, 285.2),
    'ViewUp': (-0.5, 0.25, 0.83),
    'FocalPoint': (9.6, 19.9, -3.2) }
x.plot( v, dv3d )
x.interact()
```



# vcs3D Scripting: Combined Plots

```
import vcs, cdms2, sys
x = vcs.init()
f = cdms2.open( sys.prefix+ "/sample_data/geos5-sample.nc" )
v = f["uwnd"]
dv3d = vcs.get3d_scalar()
dv3d.ToggleClipping = ( 40, 360, -28, 90 )
dv3d.YSlider = ( 0.0, vcs.off)
dv3d.XSlider = ( 180.0, vcs.on )
dv3d.ZSlider = ( 0.0, vcs.on )
dv3d.ToggleVolumePlot = vcs.on
dv3d.ToggleSurfacePlot = vcs.on
dv3d.IsolurfaceValue = 31.0
dv3d.ScaleOpacity = { 'Volume': [0.0, 0.3] }
dv3d.BasemapOpacity = 0.5
dv3d.Camera={ 'Position': (-161, -171, 279),
              'ViewUp': (.29, 0.67, 0.68),
              'FocalPoint': (146.7, 8.5, -28.6) }
dv3d.VerticalScaling = 5.0
dv3d.ScaleColormap = [ -46.0, 48.0 ], {
    'Volume': [ 16.0, 30.0 ],
    'Surface': [ 30.0, 35.0 ] }
dv3d.ScaleTransferFunction = [ 12.0, 77.0 ]
x.plot( v, dv3d )
x.interact()
```

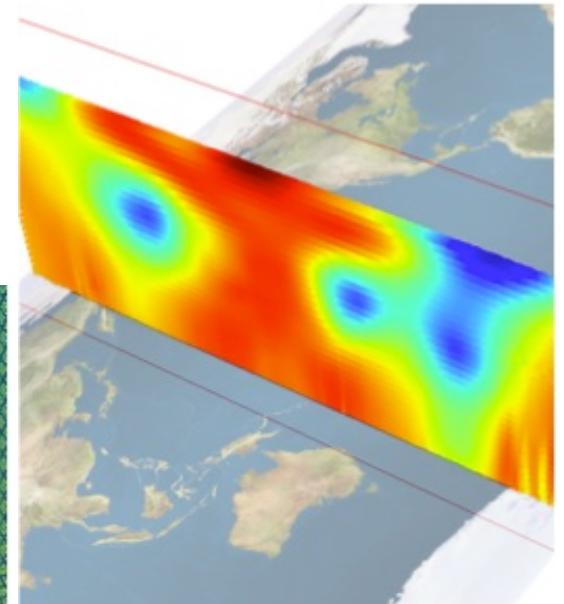
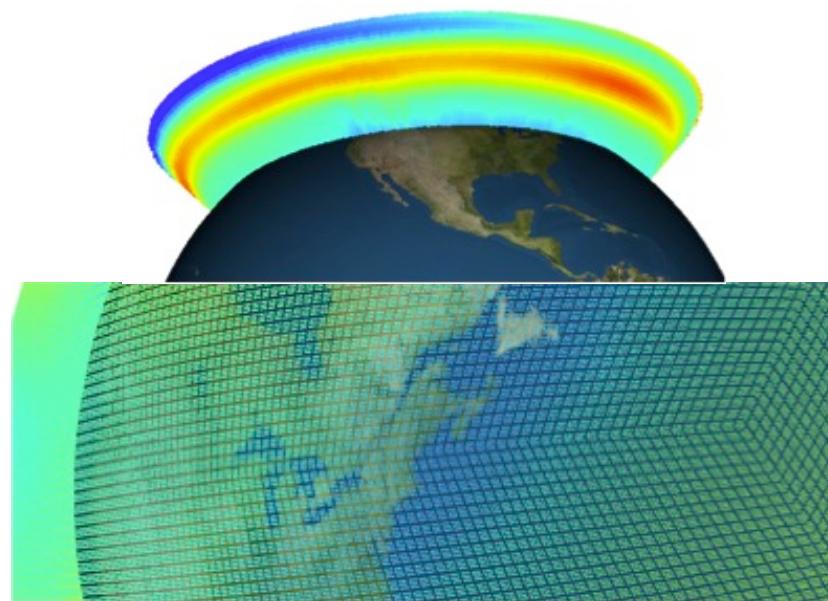
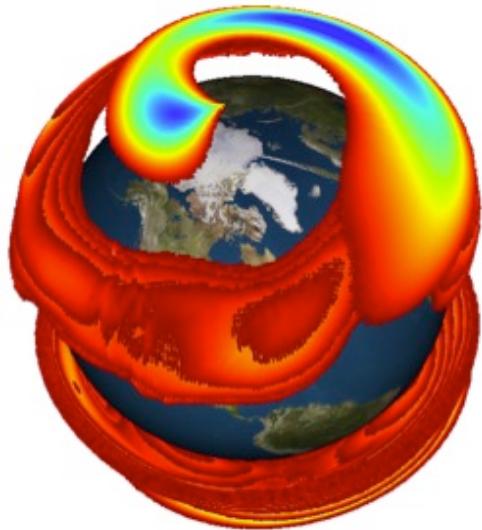
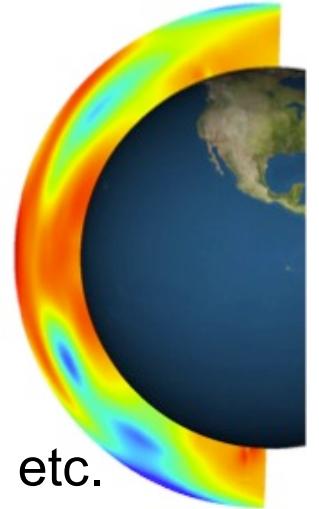


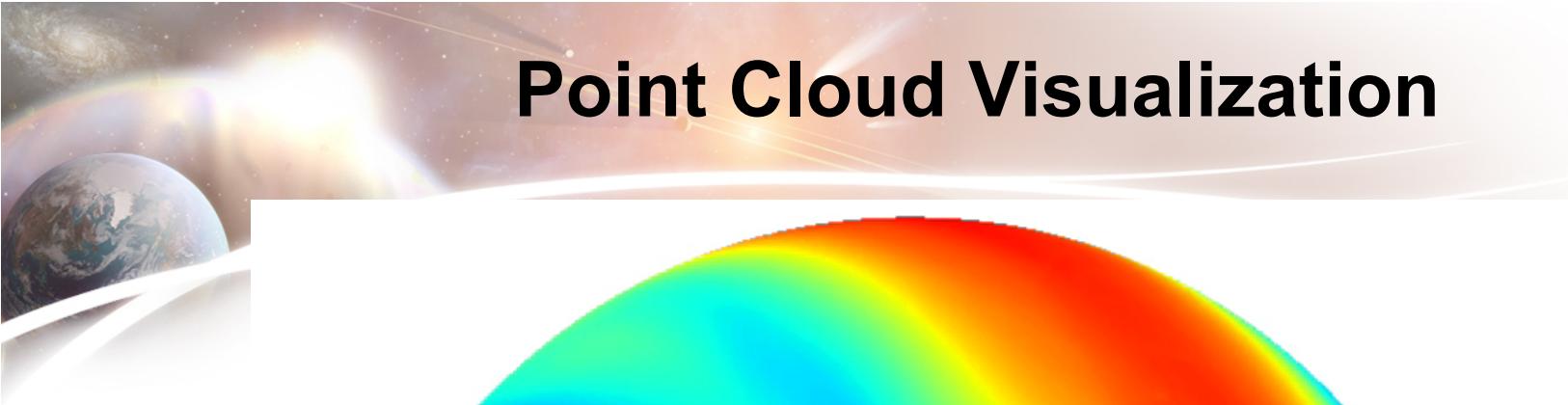


# Unstructured Grid Visualization

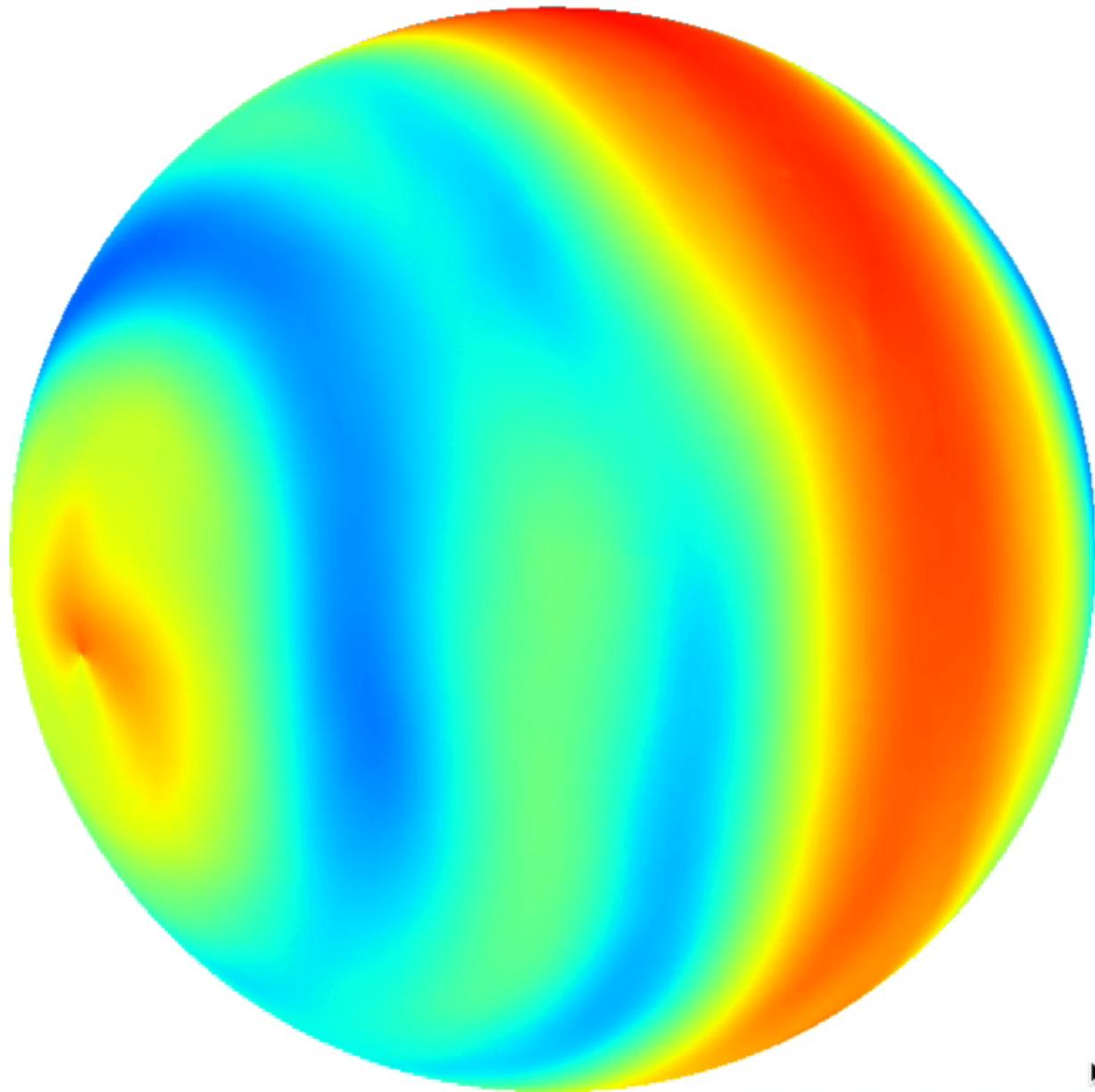


- Regridding to rectangular arrays is undesirable:
  - Computationally expensive.
  - Distorts data- may mask features of interest.
- vcs3D plots raw data without regridding:
  - Renders point clouds- requires no geometry.
  - Subsetting operation generates slices, surfaces, volumes, etc.
  - Multi-resolution, multi-core operations enable responsivity.
  - Easily switch between different projections & vertical coordinates.





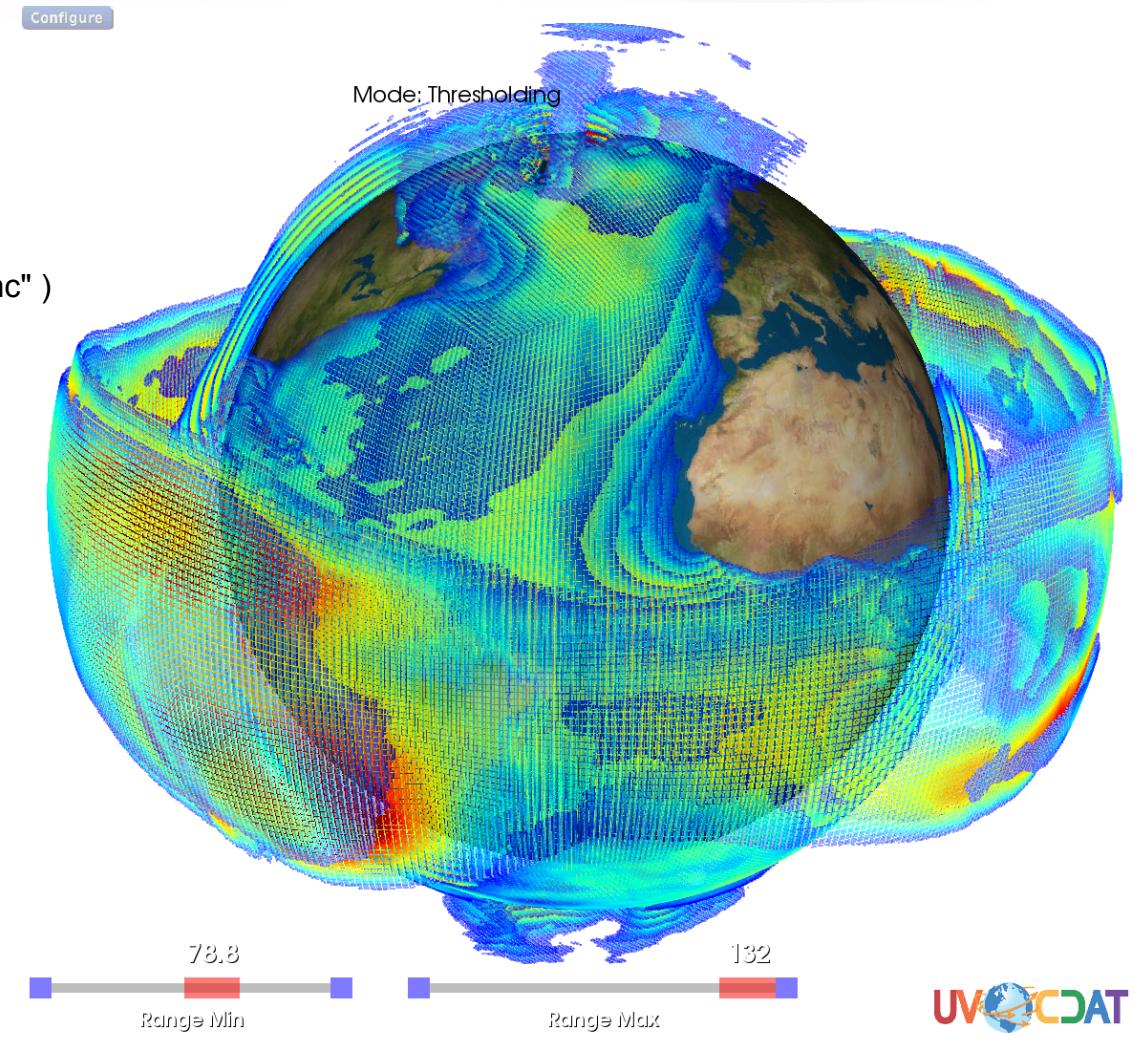
# Point Cloud Visualization



# Unstructured Grid Plot

Plotting CAM cubed sphere data without regridding.

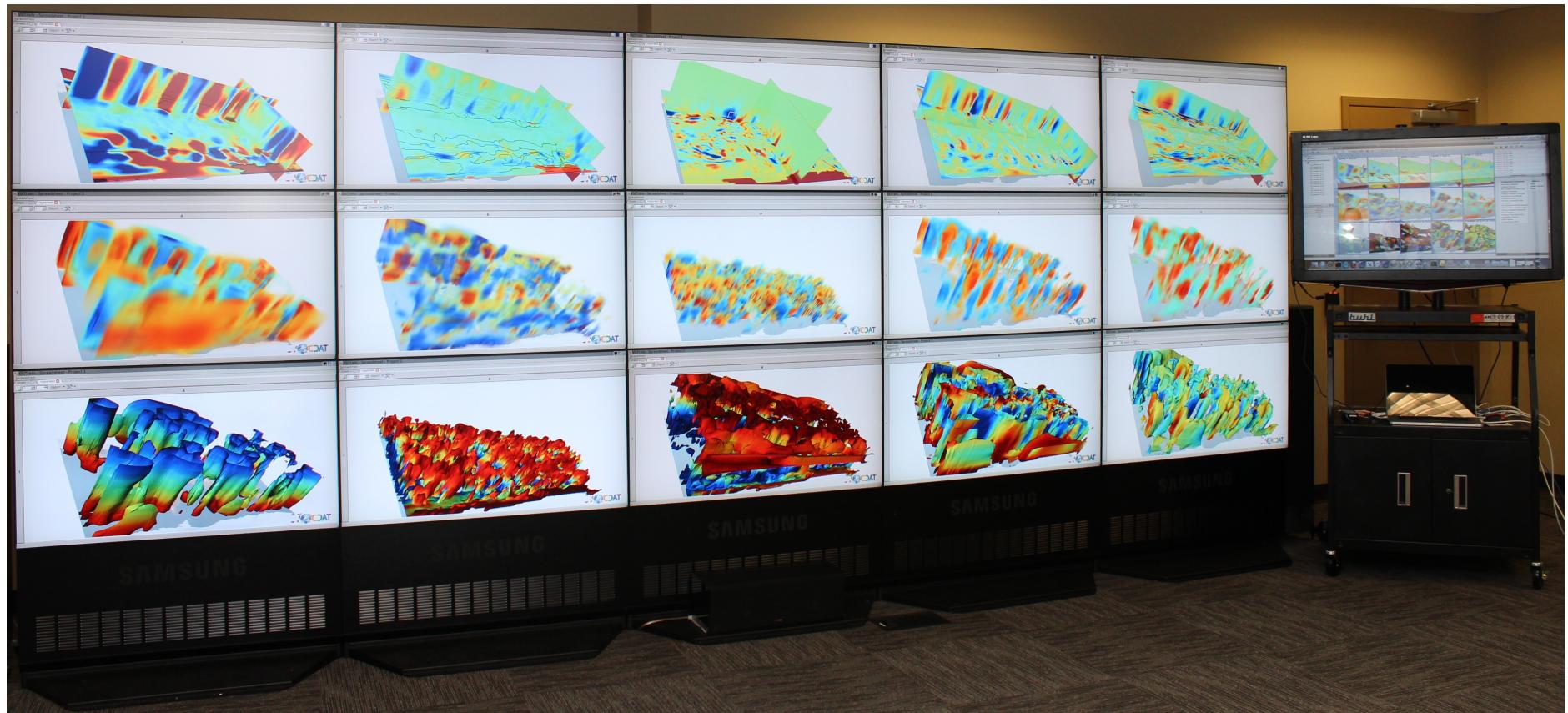
```
import vcs, cdms2, sys  
  
x = vcs.init()  
f = cdms2.open( "f1850c5_t2_ANN_climo-native.nc" )  
v = f["RELHUM"]  
dv3d = vcs.get3d_scalar()  
dv3d.ScaleTransferFunction = [ 78.7, 132.0 ]  
dv3d.ScaleColormap = [ 75.4, 100.0 ]  
dv3d.ScaleOpacity = [0.27, 1.0]  
dv3d.PointSize = [5, 2]  
dv3d.VerticalScaling = [ 1.7 ]  
dv3d.ToggleVolumePlot = vcs.on  
dv3d.ToggleSphericalProj = vcs.on  
dv3d.Camera = {  
    'Position': (-108, -477, 85),  
    'ViewUp': ( 0.0, 0.0, 1.0),  
    'FocalPoint': (0, 0, 0) }  
x.plot( v, dv3d, grid_file="ne120np4_latlon.nc" )  
x.interact()
```





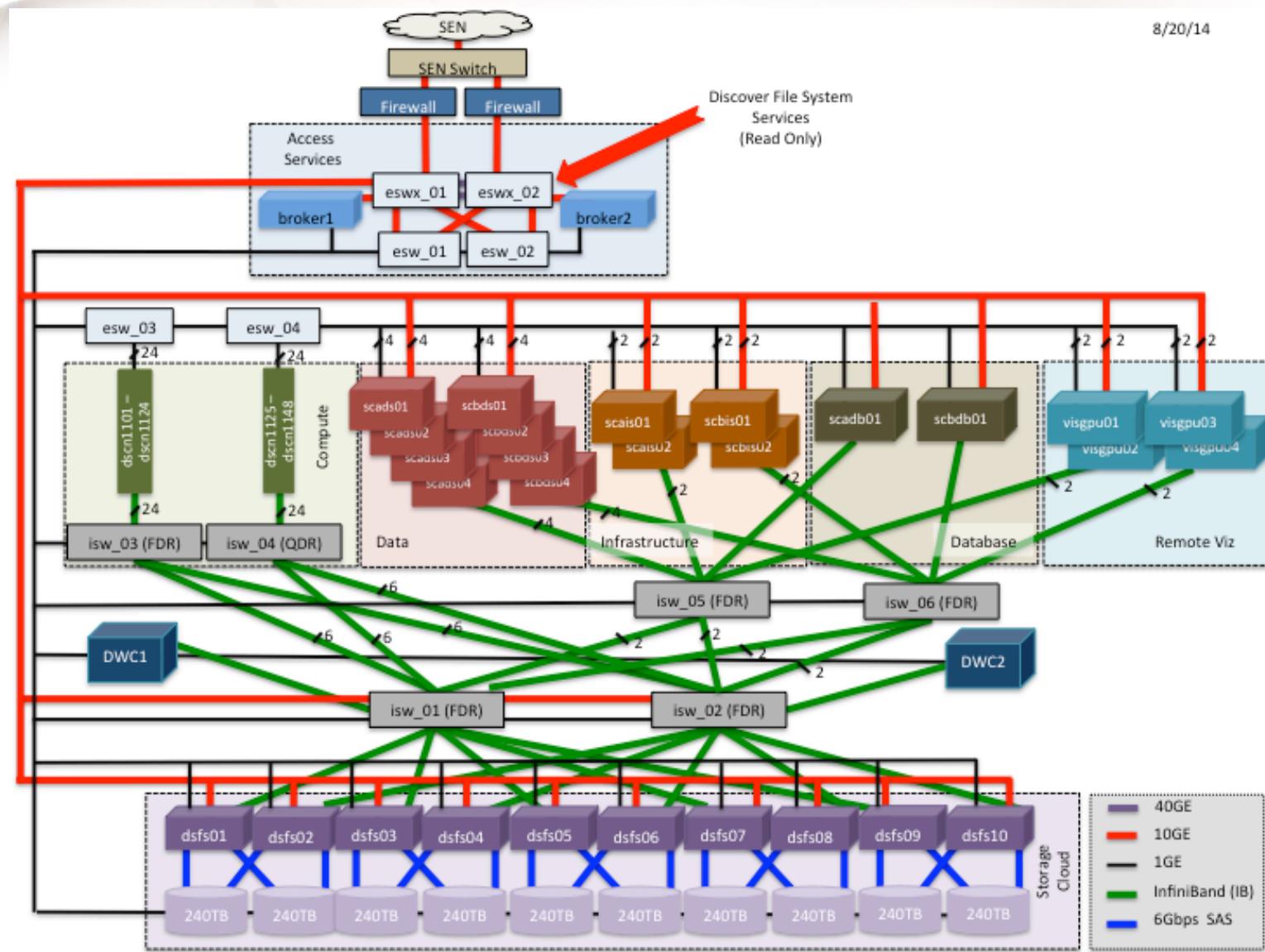
# Interactive Hyperwall Visualization

- Uses parallelism to address data complexity
- UVCDAT runs on each display node (full-res 1-cell hyperwall display)
- UVCDAT runs on control node (low-res 15-cell touchscreen display)
- Control node interactions broadcast to all hyperwall nodes



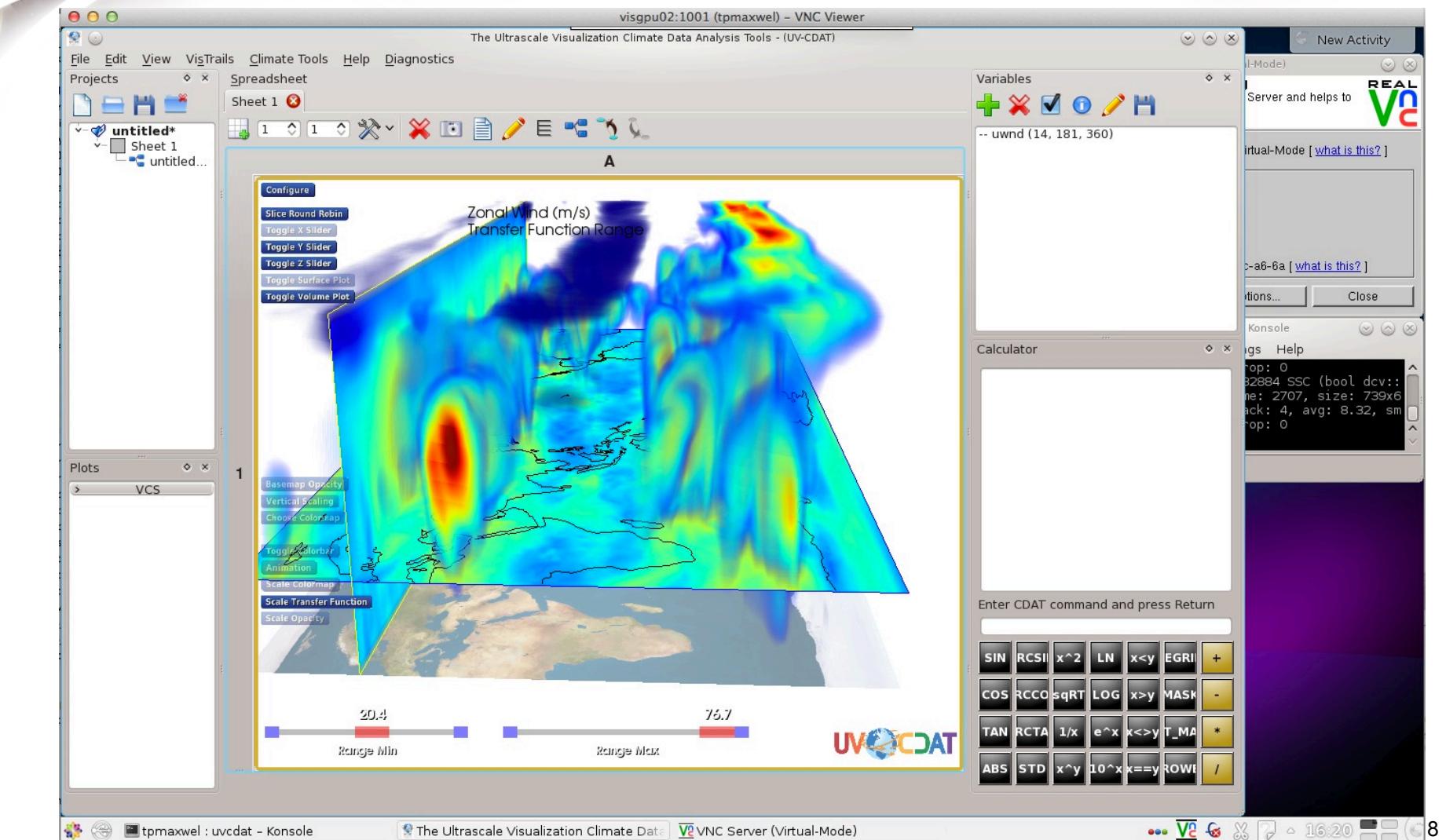
# Remote Visualization UVCDAT in NCCS Science Cloud

8/20/14



# Remote Visualization Deployment

Utilize NICE Visualization (VNC) software for interactive remote visualization.





# Advantages of Feature Rich Higher Dimensional Visualization

- Provides a more expansive window into the dataset.
- Many layers simultaneously visible “at a glance”.
- Interactivity enables iterative exploration.
- Provides overviews facilitating gestalt understanding.
- Facilitates serendipitous discoveries:
  - Unexpected features can “pop out”.

