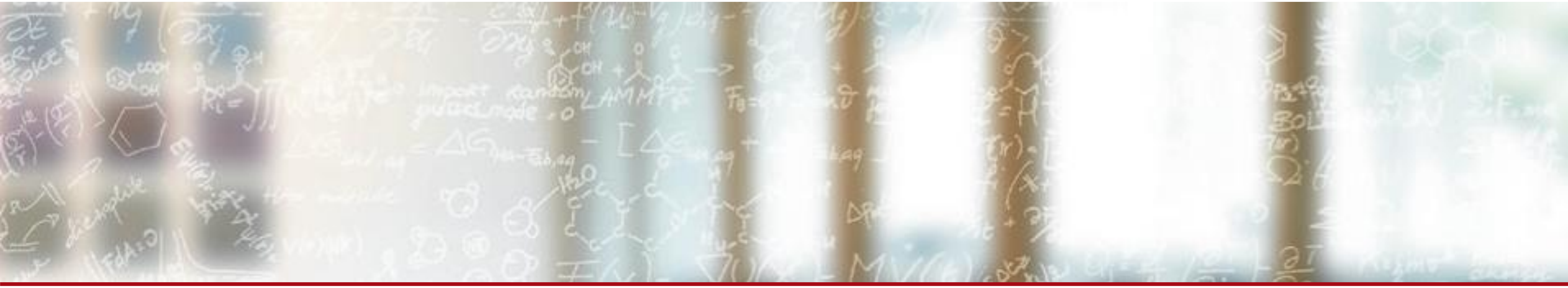
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Overview of applications/libraries for 3D Visualization

Dr. Jean M. Favre, CSCS

October 12, 2020

Preamble

All code and documentation is available here

- https://github.com/jfavre/Visualization-training

- All filenames for Python scripts or notebooks will be marked in purple, e.g. *ExploringLorentz.ipynb*

See also the User Portal

- https://user.cscs.ch/computing/visualisation/

Outline

- Present VisIt in 2 slides
- Present ParaView in 2 slides
- VNC remote desktops for OpenGL apps in 2 slides
- VTK?
- Jupyter notebooks
  - Introduce a ParaView display widget
  - Matplotlib Animation
  - ipywidgets
  - itkwidgets
    - Image viewer, line_profiler, 3D volumetric renderer
    - Exercise

Need help?

- If you have missed a point.., or if I have not understood you, or if I do not know the answer, or if I don’t have time to answer all on-line requests…

- Send email to help@cscs.ch and we will try to resolve your problem in the following days.

VisIt

- VisIt is a free, open source, platform independent, distributed, parallel, visualization tool for visualizing data defined on two- and three-dimensional structured and unstructured meshes. VisIt’s distributed architecture allows it to leverage both the compute power of a large parallel computer and the graphics acceleration hardware of a local workstation.

- https://visit-sphinx-github-user-manual.readthedocs.io/en/develop/index.html

VisIt (version 3.1.2)

https://user.cscs.ch/...../visit/

Install a binary pre-compiled on your desktop

Copy the host profile to $HOME/.visit/hosts

visit –cli –s connect_daint.py

On daint:

```
module load daint-mc Visit
echo $EBROOTVISIT
```

On desktop, edit connect_daint.py

```
install_dir =
"/apps/daint/UES/jenkins/7.0.UP02/mc/easybuild/software/Visit/3.1.2-CrayGNU-20.08"
args = ("-sshtunneling",
        "-np", "8", "-nn", "1", "-l", "srun",
        "-dir", install_dir,
        "-t", "00:05:00",
        "-la", "-Ausup -C mc -p debug")
OpenComputeEngine("daint101.cscs.ch", args)
```

ParaView

- ParaView is an open-source, multi-platform data analysis and visualization application. ParaView users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. The data exploration can be done interactively in 3D or programmatically using ParaView’s batch processing capabilities.

- ParaView was developed to analyze extremely large datasets using distributed memory computing resources. It can be run on supercomputers to analyze datasets of petascale size as well as on laptops for smaller data

ParaView (version 5.8.1)

https://user.cscs.ch/computing/visualisation/paraview/

Client-server connections are created in a Reverse-Connect fashion:

- The client initiates the call and creates an ssh tunnel to the compute node
- Once launched, the compute server connects back to the client

Connections are created via the GUI, or with a Python script.

For Linux users:

paraview –script=pvConnect_daint.py

VNC remote desktop

- Allocate a compute node with a GPU and an Xserver

- https://user.cscs.ch/computing/visualisation/vmd/#interactive-mode-with-a-remote-vnc-desktop

- The key ingredient is the SLURM option
  - #SBATCH -C gpu, startx

Once the job is started, LINUX users open an ssh tunnel to the node and connect their viewer to the localhost:

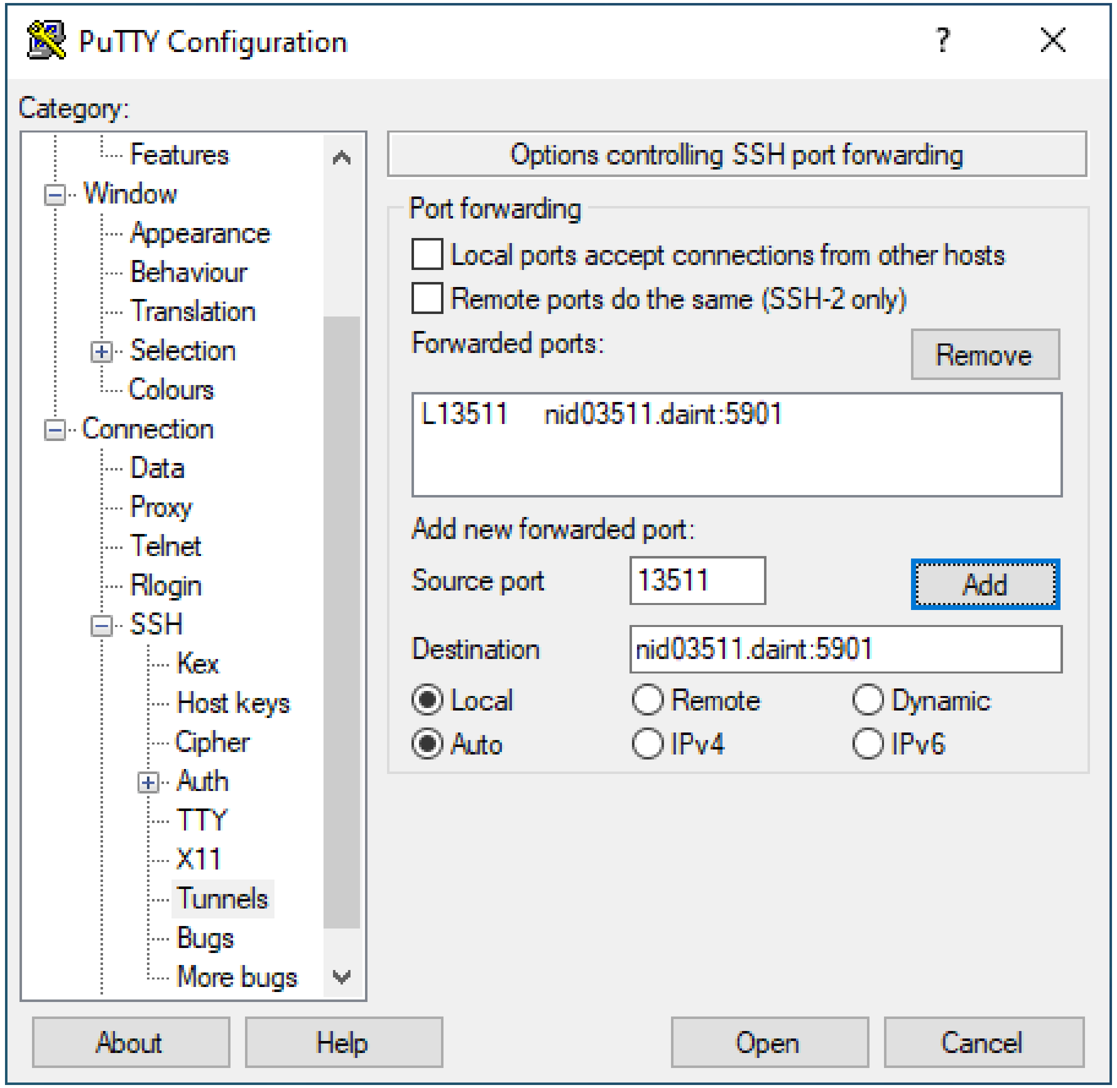
- ssh -f -L 13511:nid03511.daint:5901 <username>@ela.cscs.ch sleep 3600
- vncviewer localhost:13511

Windows users do something similar by adding a local ssh tunnel in a putty profile

- source port=13511
- destination=nid03511.daint:5901

VNC remote desktop

- See VNC_Remote/vnc_desktop.sbatch



Options controlling SSH port forwarding

Port forwarding

□ Local ports accept connections from other hosts

□ Remote ports do the same (SSH-2 only)

Forwarded ports:

```
L13511     nid03511.daint:5901
```

Add new forwarded port:

Source port 13511

Destination nid03511.daint:5901

● Local ○ Remote ○ Dynamic

● Auto ○ IPv4 ○ IPv6

VTK

- You are truly interested about Visualization since you are watching this, and you did not come for the free coffee and bagels.
- So you must have heard about VTK at least once in your life? If not…
- https://vtk.org/vtk-in-action/
- VTK is in VisIt
- VTK is in ParaView
- VTK is in jupyter notebooks
- https://pypi.org/project/itkwidgets/

vtkpython

module load ParaView

$EBROOTPARAVIEW/lib64/python3.8/site-packages has been added to your PYTHONPATH

mpi4py

paraview

vtkmodules

vtk.py

VTKpython

```
module load ParaView
python3
>>> import vtk
>>> vtk.VTK_VERSION
'8.90.0'
```

```
>>> s = vtk.vtkRTAnalyticSource()
>>> s.Update()
>>> s.GetOutput().GetBounds()
(-10.0, 10.0, -10.0, 10.0, -10.0, 10.0)
>>> s.GetOutput().GetDimensions()
(21, 21, 21)

>>> isosurface = vtk.vtkContourFilter()
>>> isosurface.SetInputConnection(s.GetOutputPort(0))
>>> isosurface.SetValue(0, 220)
>>> isosurface.Update()
>>> isosurface.GetOutput().GetNumberOfCells()
2072
```

Visualization/Animation on jupyter.cscs.ch

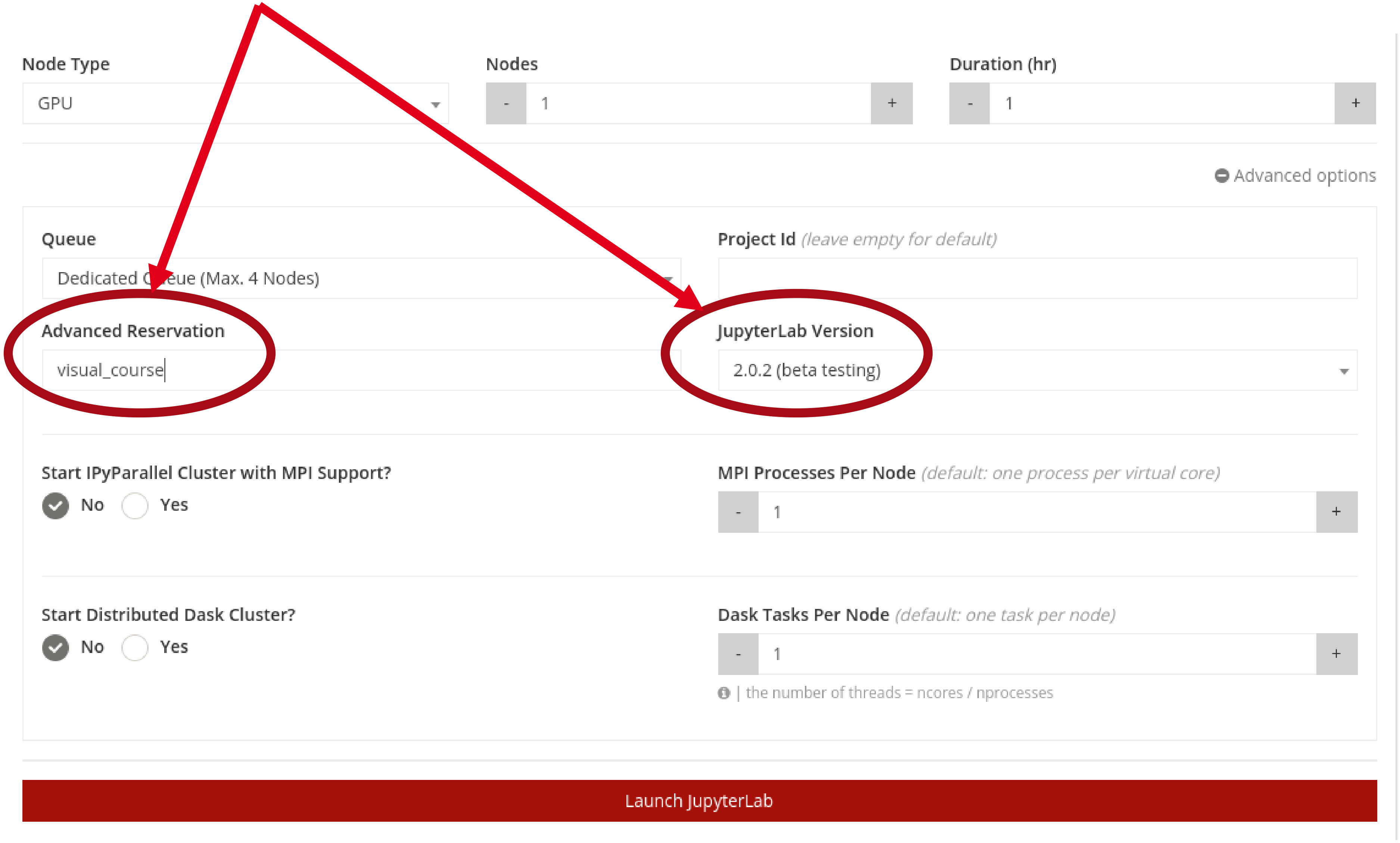
JupyterLab at CSCS

- CSCS supports the use of JupyterLab for interactive supercomputing. JupyterLab is the next-generation web-based user interface for Project Jupyter. Like the Jupyter Notebook, it is an open-source web application that allows creation and sharing of documents containing live code, equations, visualisations and narrative text.

- Credits go to my colleagues Tim Robinson and Rafael Sarmiento @ CSCS who have done the hard work of deploying JupyterLab

- Visit https://user.cscs.ch/tools/interactive/jupyterlab/
- And https://jupyter.cscs.ch/hub/spawn

How to on jupyter.cscs.ch



Node Type: GPU

Nodes: - 1 +

Duration (hr): - 1 +

Advanced options

Queue: Dedicated Queue (Max. 4 Nodes)

Project Id *(leave empty for default)*

Advanced Reservation: visual_course

JupyterLab Version: 2.0.2 (beta testing)

Start IPyParallel Cluster with MPI Support? ● No ○ Yes

MPI Processes Per Node *(default: one process per virtual core)*: - 1 +

Start Distributed Dask Cluster? ● No ○ Yes

Dask Tasks Per Node *(default: one task per node)*: - 1 +

| the number of threads = ncores / nprocesses

Jupyter widgets “ipywidgets”

- Widgets are eventful python objects that have a representation in the browser, often as a control like a slider, textbox, etc.
- You can use widgets to build interactive GUIs for your notebooks.

- https://ipywidgets.readthedocs.io/en/latest/examples/Widget%20Basics.html

- The interact function automatically creates user interface (UI) controls for exploring code and data interactively.

- Try out *ExploringLorentz.ipynb*

Pre-requisites

Edit your $HOME/.jupyterhub.env

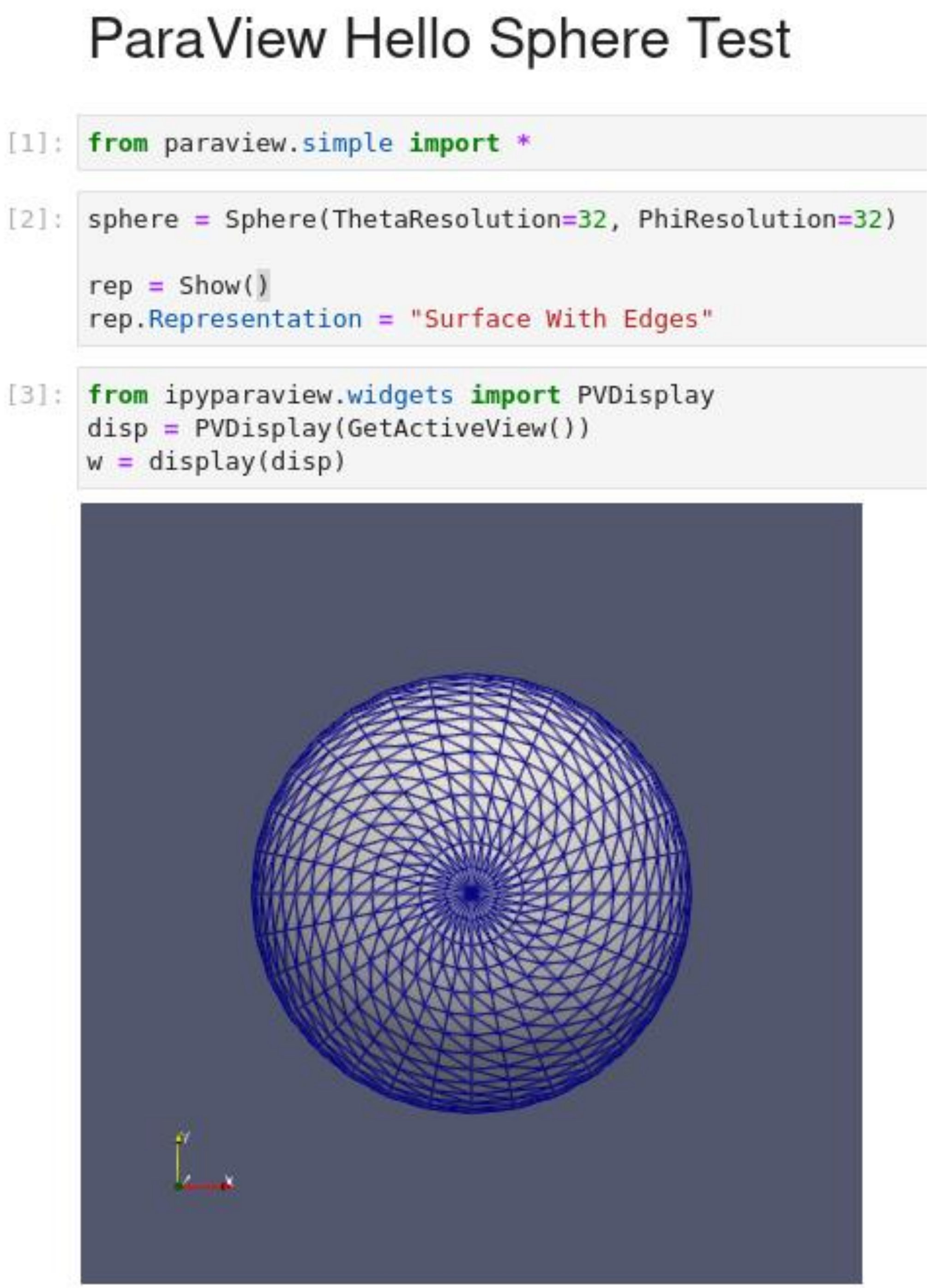
```
module load PyExtensions h5py/2.10.0-CrayGNU-20.08-python3-serial
module load ParaView
module load FFmpeg

# optionally…activate your virtual env to pick up additional modules
source $HOME/myvenv/bin/activate
```

Hello_Sphere-ParaView.0.ipynb

- Standard ParaView Python initialization
- Standard pipeline
  - ParaView Source
  - ParaView Representation
  - Render

+

- PVDisplay widget (contributed by NVIDIA)



ParaView Hello Sphere Test

```
[1]: from paraview.simple import *
```

```
[2]: sphere = Sphere(ThetaResolution=32, PhiResolution=32)

     rep = Show()
     rep.Representation = "Surface With Edges"
```

```
[3]: from ipyparaview.widgets import PVDisplay
     disp = PVDisplay(GetActiveView())
     w = display(disp)
```

Hello World (Sphere) augmented with ipywidgets

sphere.ListProperties()

Attach PhiResolution and ThetaResolution to an IntSlider

```
['Center',
'EndPhi',
'EndTheta',
'PhiResolution',
'PointData',
'Radius',
'StartPhi',
'StartTheta',
'ThetaResolution‘]
```

Hello World (Sphere) augmented with ipywidgets

```
sphere.ListProperties()
```

```
['Center',
'EndPhi',
'EndTheta',
'PhiResolution',
'PointData',
'Radius',
'StartPhi',
'StartTheta',
'ThetaResolution‘]
```

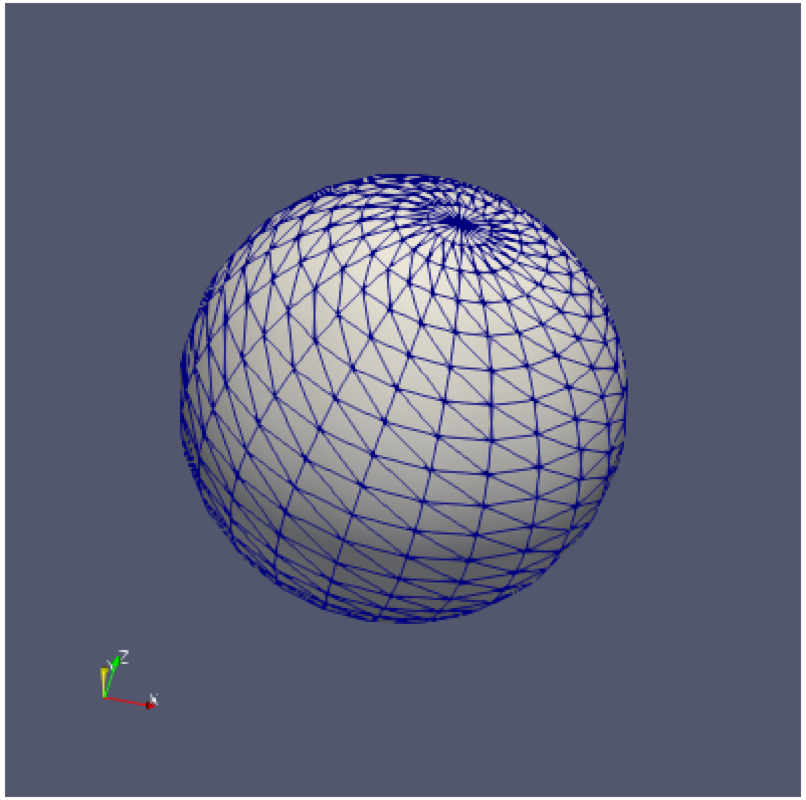
Attach PhiResolution and ThetaResolution to an IntSlider

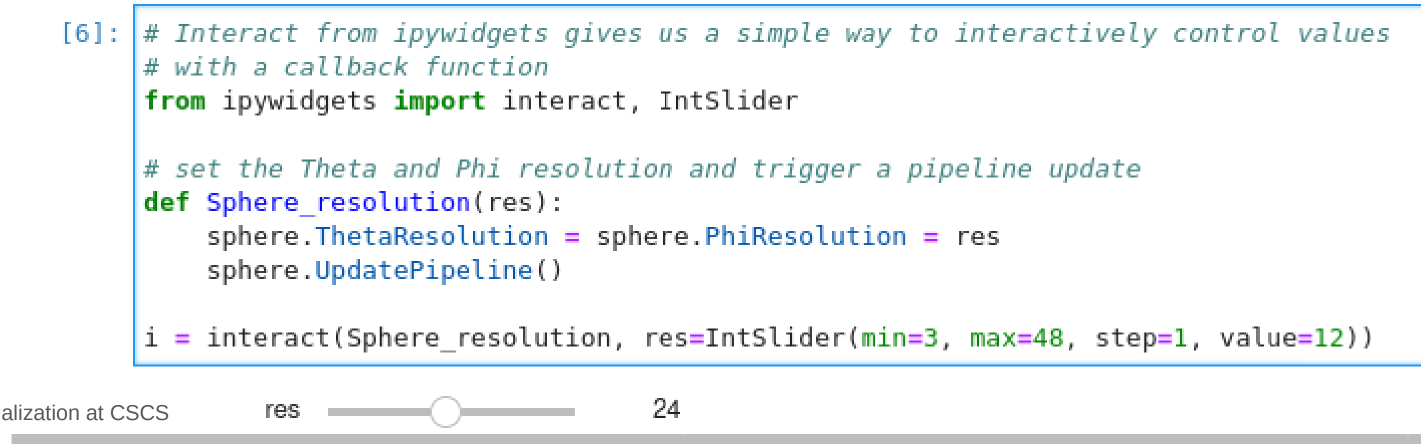
```
from ipywidgets import interact, IntSlider

# automatically triggers a pipeline update, and a render event
def Sphere_resolution(res):
    sphere.ThetaResolution = sphere.PhiResolution = res
    sphere.UpdatePipeline()

i = interact(Sphere_resolution,
                 res=IntSlider(min=3, max=48, step=1, value=12)
                )
```

Hello_Sphere-ParaView.1.ipynb

Exercise:

- Modify the callback to have two sliders, one for Theta, and one fo Phi





[6]:

```
# Interact from ipywidgets gives us a simple way to interactively control values
# with a callback function
from ipywidgets import interact, IntSlider

# set the Theta and Phi resolution and trigger a pipeline update
def Sphere_resolution(res):
    sphere.ThetaResolution = sphere.PhiResolution = res
    sphere.UpdatePipeline()

i = interact(Sphere_resolution, res=IntSlider(min=3, max=48, step=1, value=12))
```

res 24

Matplotlib Animation

Some really cool animation widget

- https://matplotlib.org/3.2.2/api/_as_gen/matplotlib.animation.FuncAnimation.html

- On Piz Daint, we can do the animation in-line, or we can write it to disk with FFmpeg

- Try out *MatplotlibAnimation.ipynb*

A case study. Hurricane Isabel Data

ipywidgets

itkwidgets

Exercise

- https://en.wikipedia.org/wiki/Hurricane_Isabel

- Hurricane Isabel data produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR and the U.S. National Science Foundation (NSF).

48 timesteps are available in a binary file

- big-endian format and compressed via GNU zip (gzip) for storage.

- Since the data in question simulates an actual event (a hurricane), each computational data point (voxel) corresponds to an actual physical point. The surface topology (for the actual ground) is in a special 500x500x1 data file (HGTdata.bin.gz). In the other files, where there is ground, no data is recorded. The special value for this "no data" value is 1.0000000e+35.

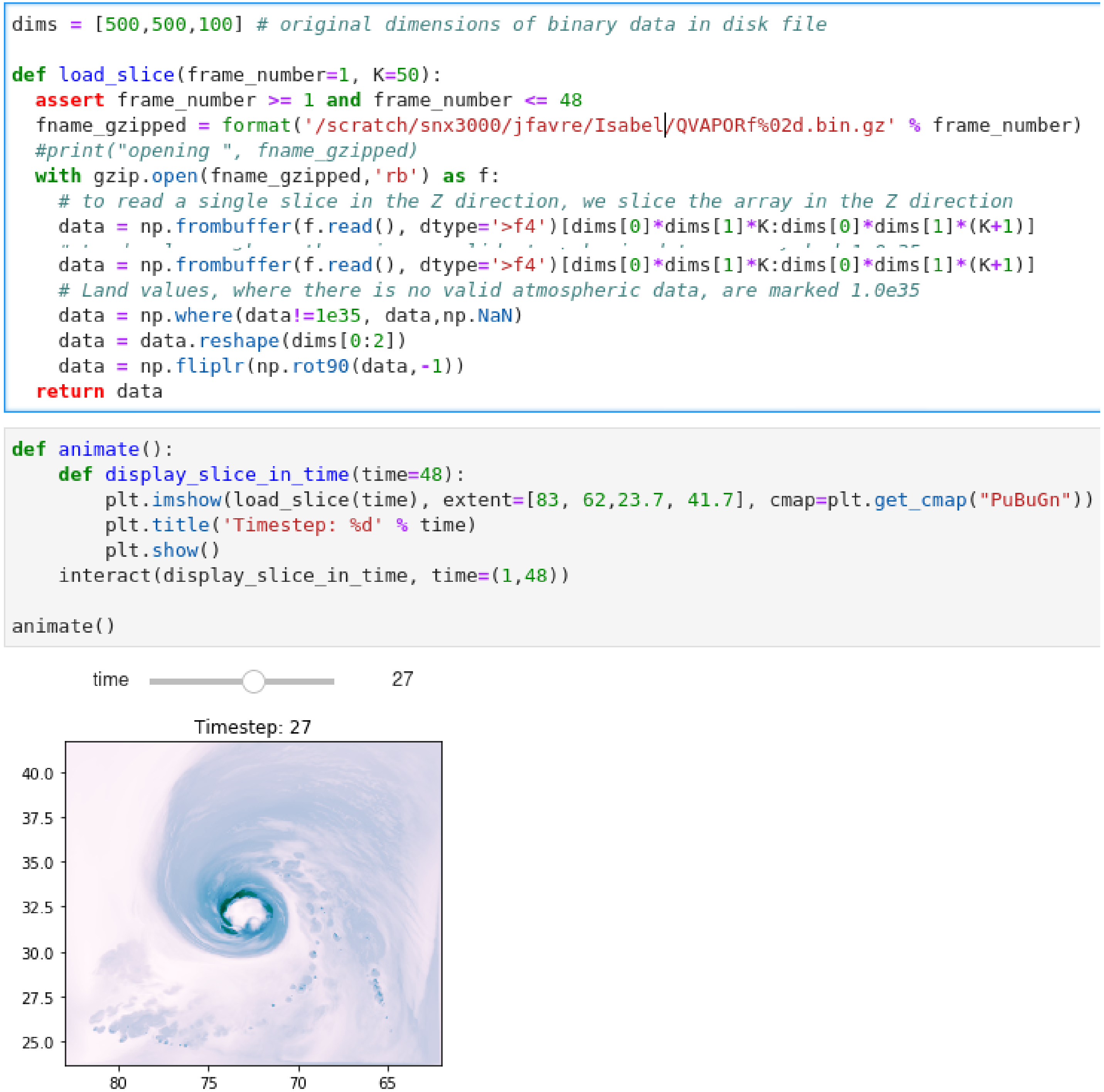
- Find all 48 files for the “QVAPOR” variable in /scratch/snx3000/jfavre/Isabel

Extracting a single slice from the 500x500x100 volume

```
dims = [500,500,100] # original dimensions of binary data in disk file


def load_slice(frame_number=1, K=50):
  fname = format('QVAPORf%02d.bin.gz' % frame_number)
  with gzip.open(fname,'rb') as f:
    # read a single slice in the Z direction by slicing the volume
    data = np.frombuffer(f.read(), dtype='>f4') [dims[0]*dims[1]*K :
                                                  dims[0]*dims[1]*(K+1)]
    # Land values, where there is no valid data, are marked 1.0e35
    data = np.where(data!=1e35, data, np.NaN)
    data = data.reshape(dims[0:2])
    data = np.fliplr(np.rot90(data,-1))
  return data
```

ipywidgets' interact

Try out *ipywidgets.Isabel.ipynb*



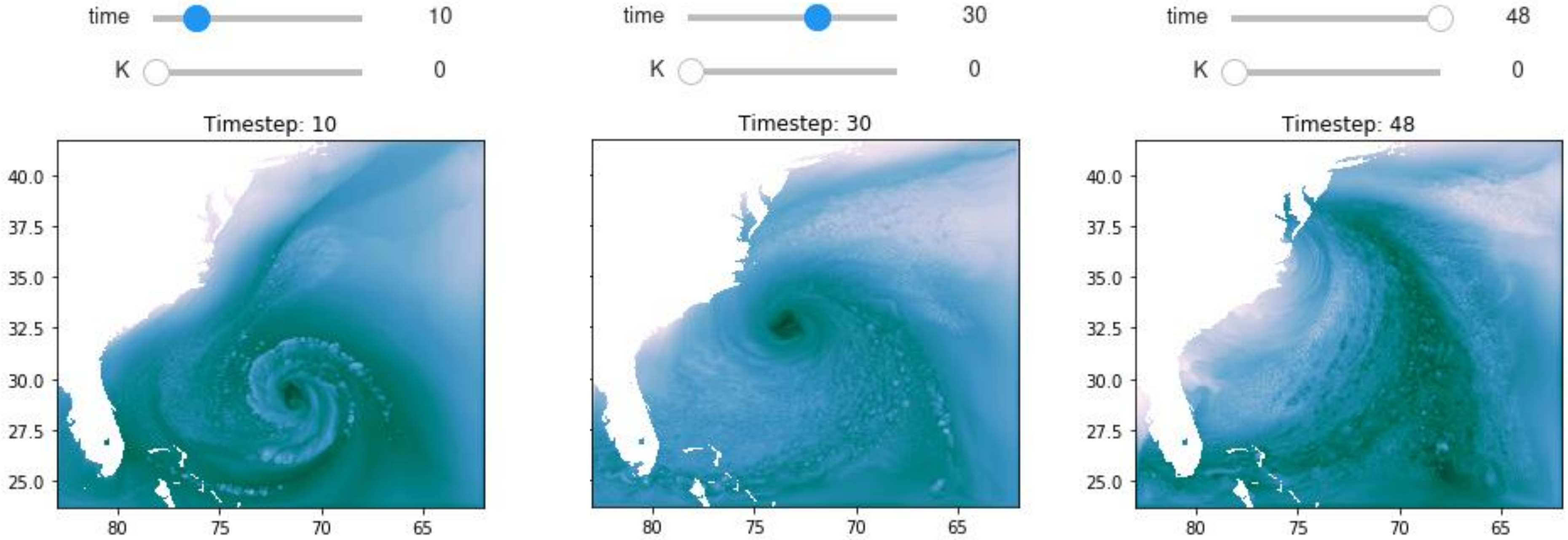
```
dims = [500,500,100] # original dimensions of binary data in disk file

def load_slice(frame_number=1, K=50):
  assert frame_number >= 1 and frame_number <= 48
  fname_gzipped = format('/scratch/snx3000/jfavre/Isabel/QVAPORf%02d.bin.gz' % frame_number)
  #print("opening ", fname_gzipped)
  with gzip.open(fname_gzipped,'rb') as f:
    # to read a single slice in the Z direction, we slice the array in the Z direction
    data = np.frombuffer(f.read(), dtype='>f4')[dims[0]*dims[1]*K:dims[0]*dims[1]*(K+1)]
    data = np.frombuffer(f.read(), dtype='>f4')[dims[0]*dims[1]*K:dims[0]*dims[1]*(K+1)]
    # Land values, where there is no valid atmospheric data, are marked 1.0e35
    data = np.where(data!=1e35, data,np.NaN)
    data = data.reshape(dims[0:2])
    data = np.fliplr(np.rot90(data,-1))
  return data
```

```
def animate():
    def display_slice_in_time(time=48):
        plt.imshow(load_slice(time), extent=[83, 62,23.7, 41.7], cmap=plt.get_cmap("PuBuGn"))
        plt.title('Timestep: %d' % time)
        plt.show()
    interact(display_slice_in_time, time=(1,48))

animate()
```

Exercise

Modify *ipywidgets.Isabel.ipynb* to add a second slider to slice the volume at different height (an index between 0 and 99).

N.B. Showing the K=0 plane will let you see the geographic profile of Florida



time 10
K 0

time 30
K 0

time 48
K 0

itkwidgets, view 2D images

- Try out *itkwidgets.Isabel.ipynb*

```
from itkwidgets import view, line_profile
```

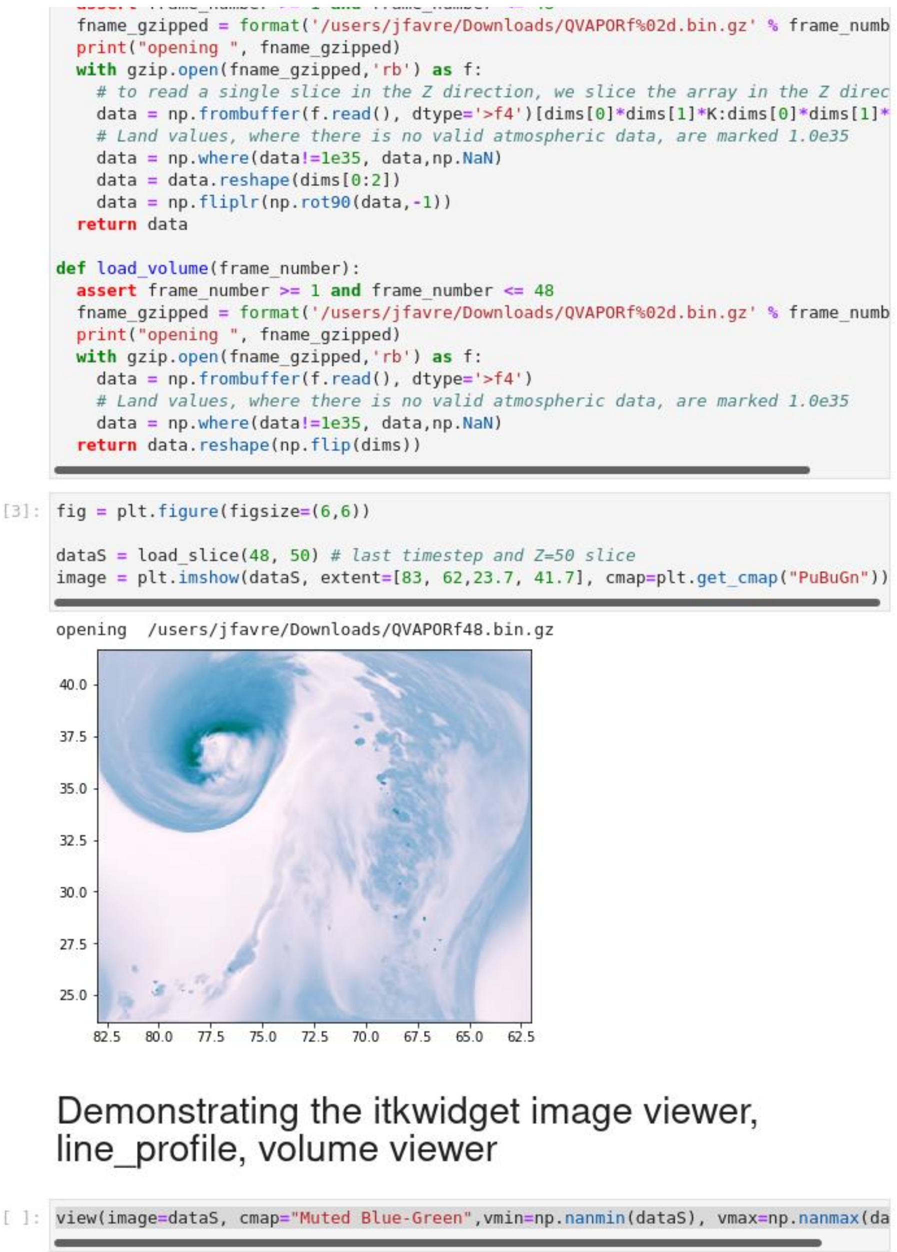
```
view(image=dataS, cmap="Muted Blue-Green", vmin=np.nanmin(dataS),
vmax=np.nanmax(dataS))
```

Itkwidgets, view 2D images and query/plot a line

Try out
itkwidgets.Isabel.ipynb

from itkwidgets import line_profile

Checkout the colormap improved perception



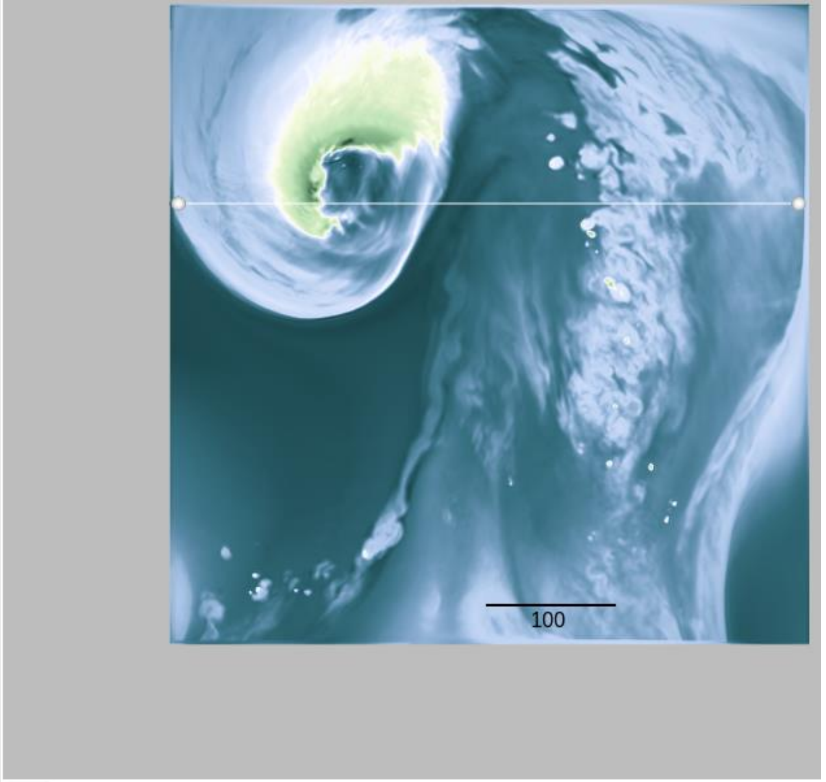
```
  fname_gzipped = format('/users/jfavre/Downloads/QVAPORf%02d.bin.gz' % frame_numb
  print("opening ", fname_gzipped)
  with gzip.open(fname_gzipped,'rb') as f:
    # to read a single slice in the Z direction, we slice the array in the Z direc
    data = np.frombuffer(f.read(), dtype='>f4')[dims[0]*dims[1]*K:dims[0]*dims[1]*
    # Land values, where there is no valid atmospheric data, are marked 1.0e35
    data = np.where(data!=1e35, data,np.NaN)
    data = data.reshape(dims[0:2])
    data = np.fliplr(np.rot90(data,-1))
  return data

def load_volume(frame_number):
  assert frame_number >= 1 and frame_number <= 48
  fname_gzipped = format('/users/jfavre/Downloads/QVAPORf%02d.bin.gz' % frame_numb
  print("opening ", fname_gzipped)
  with gzip.open(fname_gzipped,'rb') as f:
    data = np.frombuffer(f.read(), dtype='>f4')
    # Land values, where there is no valid atmospheric data, are marked 1.0e35
    data = np.where(data!=1e35, data,np.NaN)
  return data.reshape(np.flip(dims))
```

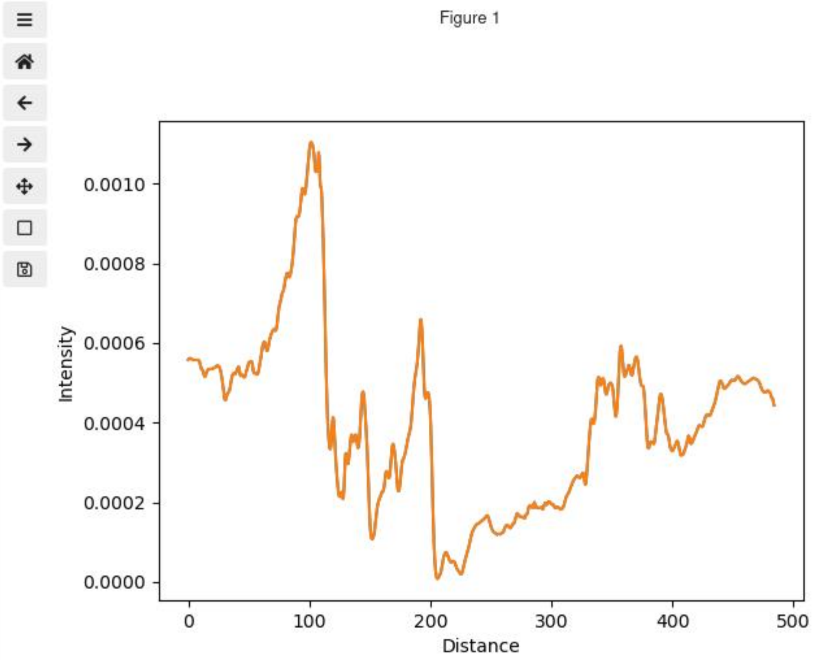
```
[3]: fig = plt.figure(figsize=(6,6))

     dataS = load_slice(48, 50) # last timestep and Z=50 slice
     image = plt.imshow(dataS, extent=[83, 62,23.7, 41.7], cmap=plt.get_cmap("PuBuGn"))
```

```
opening  /users/jfavre/Downloads/QVAPORf48.bin.gz
```

Demonstrating the itkwidget image viewer, line_profile, volume viewer

```
[ ]: view(image=dataS, cmap="Muted Blue-Green",vmin=np.nanmin(dataS), vmax=np.nanmax(da
```



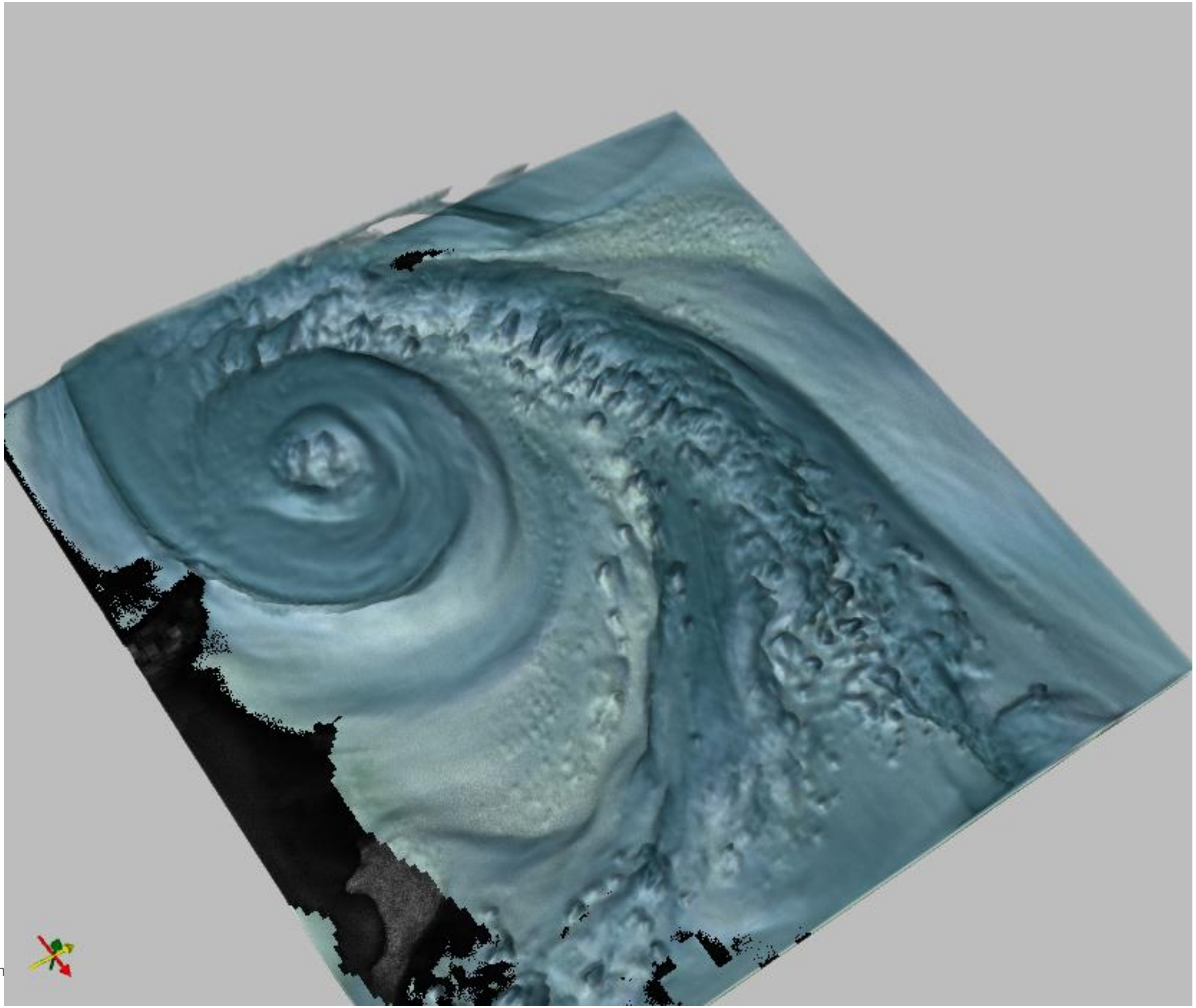


Itkwidgets, 3D views

Try out
itkwidgets.Isabel.ipynb

from itkwidgets import view

An interactive “volume rendering” is shown



Exercise

- Modify *ipywidgets.Isabel.ipynb* to replace the slider “manual” interaction, by an automatic animation writer (both javascript, and mpeg), using matplotlib.Animation

- See *Matplotlib.Animation.ipynb* as a hint

- Solution in *Animation.Isabel.ipynb (to be shared at end of session)*

Batch-mode matplotlib animation

```
module load daint-gpu  FFmpeg    PyExtensions    jupyterlab
jupyter nbconvert --to=python Animation.Isabel.ipynb
python Animation.Isabel.py
```

Summary

- Traditional 3D “heavy-weight” applications such as VisIt, ParaView
- VNC desktop for any OpenGL application requiring an X server
- New style JupyterLab notebooks
  - A ParaView display widget
  - ipywidgets for quick UI prototyping
  - itkwidgets for advanced viewing

Next, after the break

- We’ll discuss data formats for VTK-centered Visualization
  - Big data require parallel I/O
- Introduction to ParaView
  - Importing numpy arrays
  - Client-server connections
  - A bit of everything

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich