NC STATE UNIVERSITY

Henry2: Scientific Visualization with ParaView

ParaView is an open-source, multi-platform data analysis and visualization application.

A small subset of supported file types:

ANSYS	**EnSight**	**FLASH**
FVCOM	**Fluent**	**LAMMPS**
MPAS	**NetCDF**	**OpenFOAM**
SAS	**VASP**	**VTK**
Xdmf	**Tecplot**	**Protein Data Bank**

Users may write new readers for additional file types.

A small subset of supported operators:

Contour	**Histogram**	**Slice**
Scatter Plot	**Clip**	**Surface Flow**
Threshold	**Plot over Time**	**Resample**
K Means	**Stream Tracing**	**Isosurfaces**
Particle Tracer	**Interpolate**	**Convert to Molecule**

Users may write additional operators in Python.

Difficulties with Visualization on HPC

Visualizing directly on HPC:

- Speed of rendering a window

Visualizing on a local computer:

- Speed of transferring files
- Space limitations
- Compute limitations

Mitigating the difficulties of using HPC for visualization

HPC-VCL:

- Remote Desktop Protocol (RDP) renders faster than SSH X11 tunnelling
- Calculations done on a VCL node (8 GB RAM) or an interactive compute node (up to 500 GB RAM)
- No need to transfer files

- Rendering may still be slow compared to a local window

Too slow to render: Another option

NC State University Libraries:

- High-end workstations with attached GPUs
  https://www.lib.ncsu.edu/spaces/dataspace

- You still need to transfer data from HPC

Too slow to render, too much data to transfer

PvBatch Server - Paraview without a GUI

- Transfer only a single file and visualize with the GUI
- Record the commands to a Python script
- Use LSF to submit the script as a ParaView job using PvBatch Server
- No need for data transfer massive amounts of data

What's the catch?

- *You need to learn a teeny bit of Python*

Video Demo Notes:

The video shows

- Rendering of a dataset on the HPC-VCL
- Saving a state file of the final graphic
- Recording the steps of loading the state and exporting graphics into a Python script
- Running the Python script using LSF

Basic Steps for a compute intensive visualization workflow

Render and record locally:

- Use the Libraries computer or a local computer.
- Transfer a single file from the HPC to that computer.
- Use the interactive GUI to create the desired visualizations.
- Save the final state of the session or record the steps taken with Python using the ParaView automatic tools.

Run on HPC:

- Use the HPC provided example to add a loop over datafiles to the recorded workflow (a Python script).
- Do a small test batch - only loop over a few files or timesteps and check the results.
- After a successful test, expand your loop to include all your data.
- Submit to LSF.

Workflow:

- To follow the tutorial, get the files from Henry2 and copy to your /share directory

```
cp /usr/local/apps/examples/video_tutorials/paraview_demo.tar .
tar -xvf paraview_demo.tar
cd paraview_demo
```

- To follow the tutorial on a local computer, get the files with scp

```
scp unityid@login.hpc.ncsu.edu:/usr/local/apps/examples/video_tutorials/paraview_demo.tar .
```

- The directory has the LSF submit script, submit.csh, the required input files, and the saved files that were generated during the demo. You will only need the input files and submit script. You will generate the output, the ParaView state file and the Python script as you follow the demo.

- Make a reservation of the HPC-VCL. Use the instructions from the website.
- Log in to the HPC-VCL.
- Load ParaView and start the GUI

```
module load paraview
paraview --mesa
```

- In the ParaView GUI, do

  File:Open

  File name: (...)/paraview_demo/input/t1d_1.nc

  Open Data With: NetCDF Reader

  Apply

  Choose Surface

  Set Range: 0 - 16.1276, Rescale and disable automatic rescaling

- Continue in the ParaView GUI, do

    Choose Preset:Rainbow Desaturated:Apply:Close

    Choose Preset:Blue Orange:Apply:Close

    Undo

    Undo

    Redo

    Add Filter:Common:Contour

    Isosurfaces:Add Range:From 0, To 16, Steps 17:ok:Apply

    Clear error

    Click t1d_1.nc in Pipeline Browser to reactivate

    Click Contour1

    Coloring:Edit:black

- Continue in the ParaView GUI, do

    (If the Point Size and Line Width is not visible, click the Settings wheel to see all options.)

    Styling:Line Width:3

    File:Save State:(..)/paraview_demo/t1d

    Exit

- Load ParaView and start the GUI (again)

```
module load paraview
paraview --mesa
```

- Check the State File - In the ParaView GUI, do

    Load State File:(...)/paraview_demo/t1d.pvsm

    Load State Options:Choose File Names

    Clear error messages

    Close ParaView

- Record the Script - In the ParaView GUI, do

  Tools:Start Trace

  Load State File:(...)/paraview_demo/t1d.pvsm

  Load State Options:Choose File Names

  Clear error messages

  Save Screenshot:t1d

  Export View:SVG:t1d

  Tools:Stop Trace

  File:Save As:(...)/paraview_demo/t1d_script.py

- On the HPC, if the pvsm file and script files are names as defined above, submitting the script should be successful without modification:

```
bsub < submit.csh
```

- On the HPC, modify the Python script by changing the input files and output files. Replace “t1d_1” with “t1d_4” and resubmit the job:

```
bsub < submit.csh
```

- Look at the output files with the display command:

```
display output/t1d_1.png
```

Automate batch processing

- The demo package includes a sample Python code that adds a loop over a number of input files. There are 5 input files in this demo.
- To run:

```
cp /usr/local/apps/examples/video_tutorials/paraview_demo.tar .
tar -xvf paraview_demo.tar
cd paraview_demo
cp loop_files/* .
```

- Look at the file t1d_script_loop.py. Compare it with the autogenerated one (demo_files/t1d_script.py). Change the user modified section to reflect where your copy of paraview_demo is located, then run the script with LSF.

```
bsub < submit.csh
```

- This should generate 10 files, two for each available input file.