Cornell University
Center for Advanced Computing

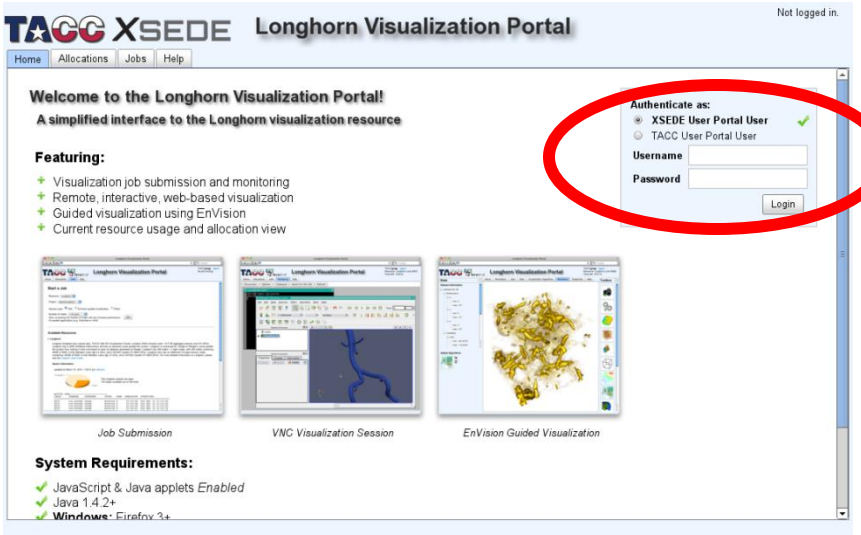
Parallel Lab

Aaron Birkland
Cornell Center for Advanced Computing

Data Analysis on Ranger
January 2012

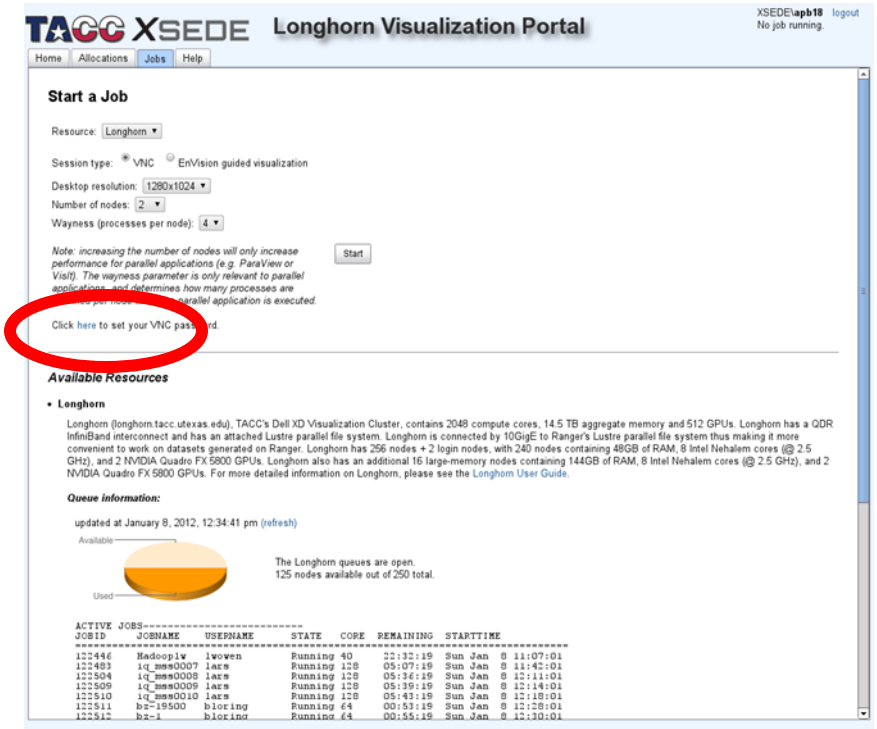
Getting started

- Go to Longhorn portal: http://portal.longhorn.tacc.utexas.edu
- Sign in using your XSEDE credentials



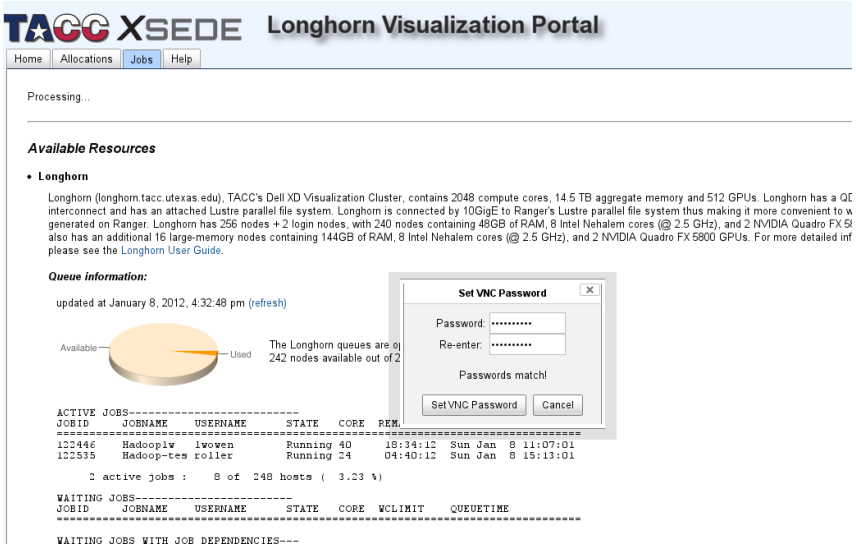
Set your VNC password

- Click “here” to set/change your VNC password
  - This is exactly equivalent to using `vncpasswd` on the command line
  - If you forget to do this, and nave never set the vnc password before, you will be prompted. Don’t worry



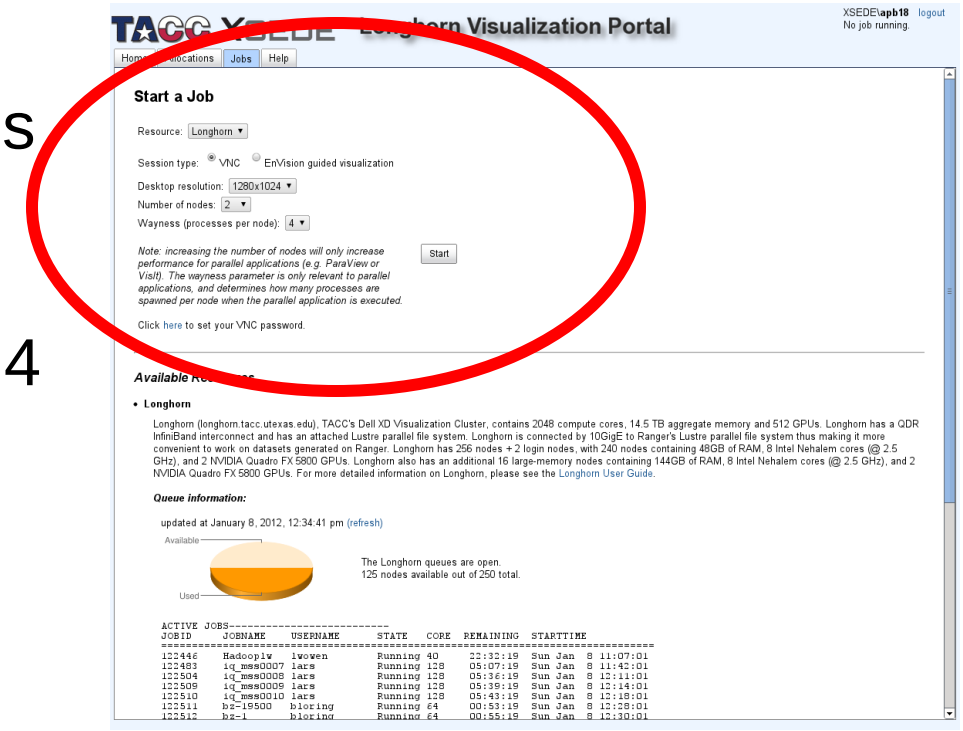
Set your VNC password

- Enter a password in the window that pops up. Click “Set VNC password” button, then close
  - Do NOT use an important (e.g. login) password for this!! Remember, VNC passwords protect a running session only, and might be shared with others



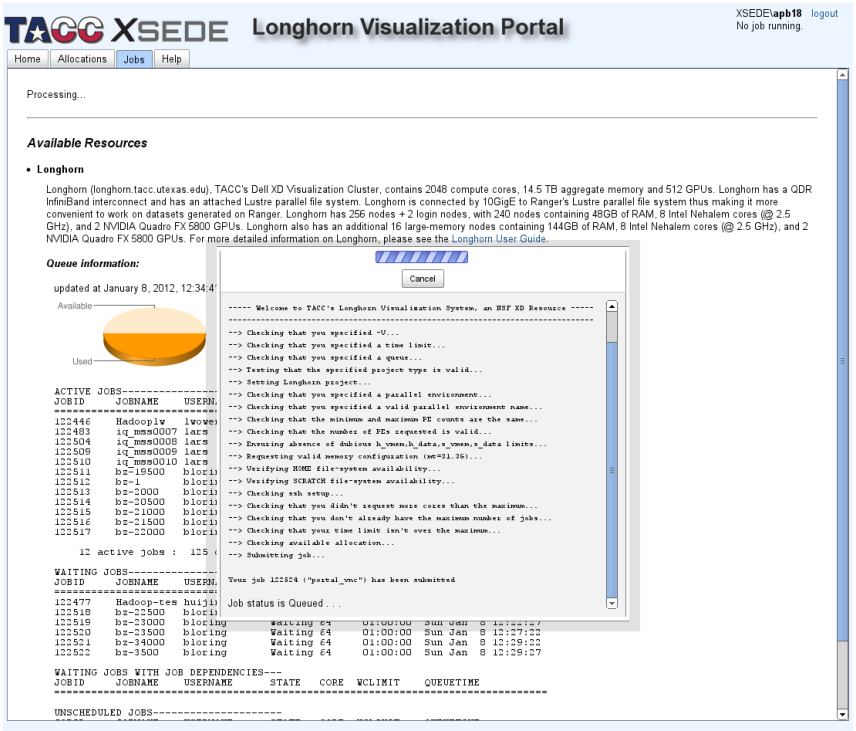
Start a VNC session

- Make sure VNC session type is selected
- Specify number of nodes = 2
- Specify wayness (processes per node) = 4
- Click “Start”



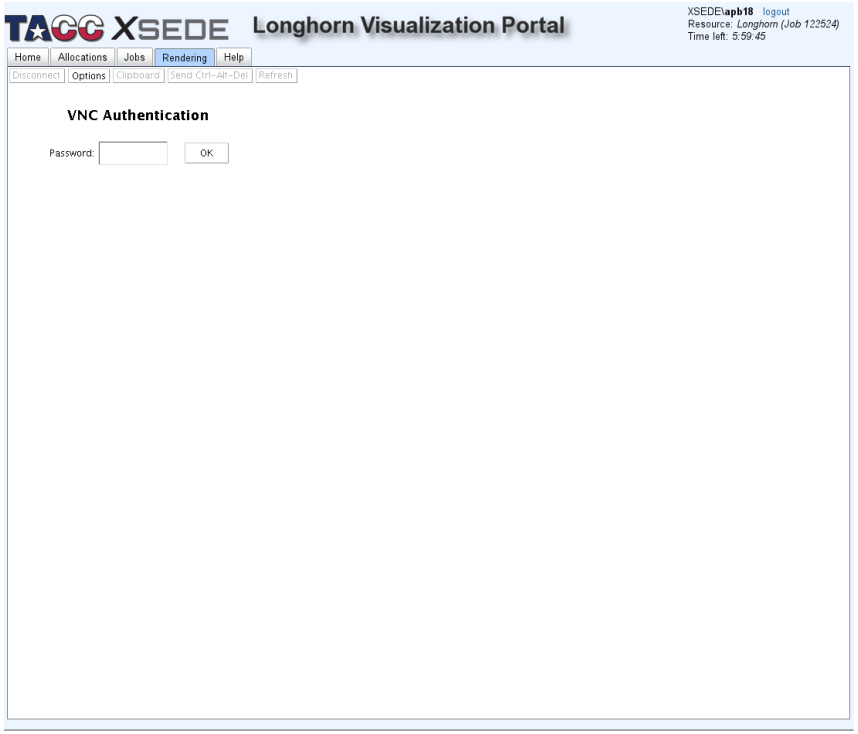
Start a VNC session

- When “Start” is clicked, the portal will automatically submit a job via qsub.
- You will see status on screen. It may take a little while to go from Queued to Running.



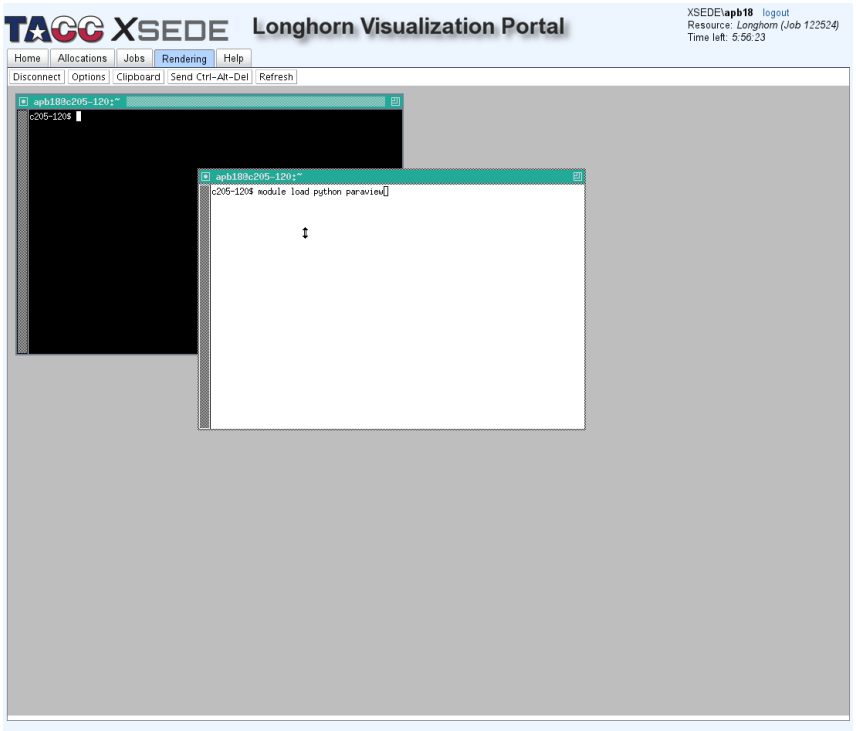
Start a VNC session

- A VNC session is now running. By default, you are brought to a "rendering" tab which has a built-in VNC viewer. You can use this if you wish, or just use your own VNC client.
- Type in your VNC password to access the session.



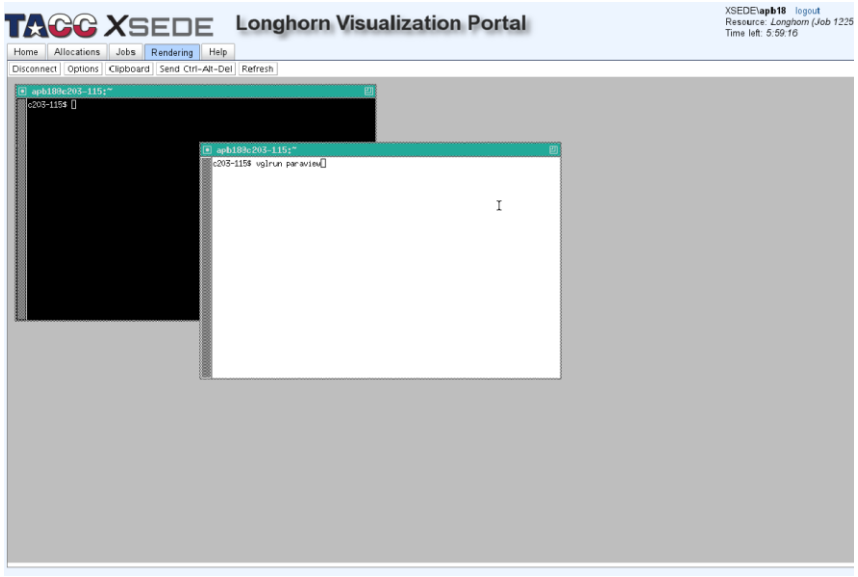
Run ParaView

- You will be brought to the VNC desktop.
- In the white xterm, type `module load python paraview`
- You will receive no visual feedback or return value
- This makes paraview available on the commandline.



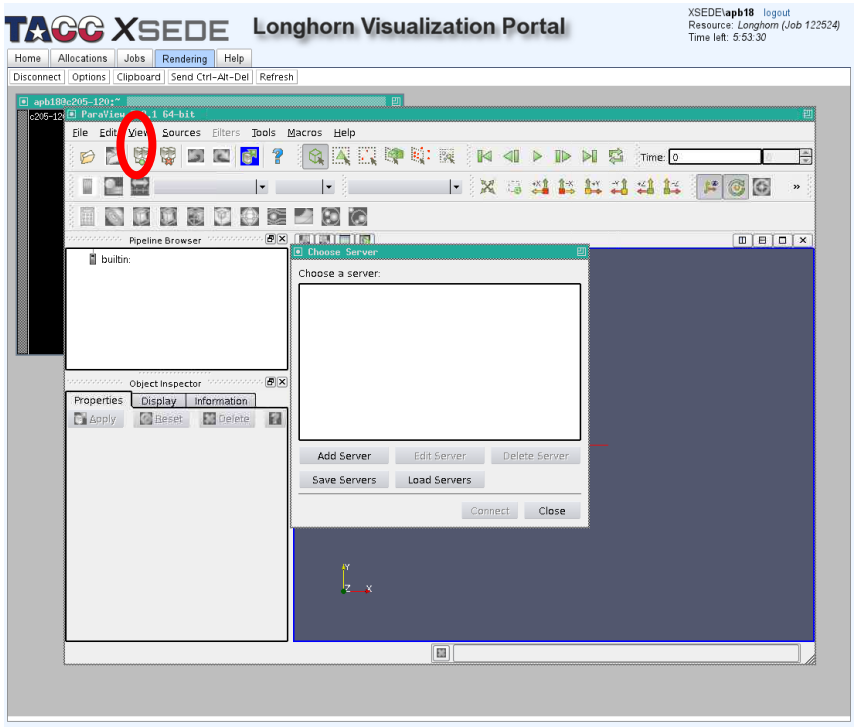
Run ParaView

- Type `vglrun paraview`
- This will start paraview and allow it to use the system GPU for rendering
- By default, it does not start up in parallel mode. That has to be enabled next.



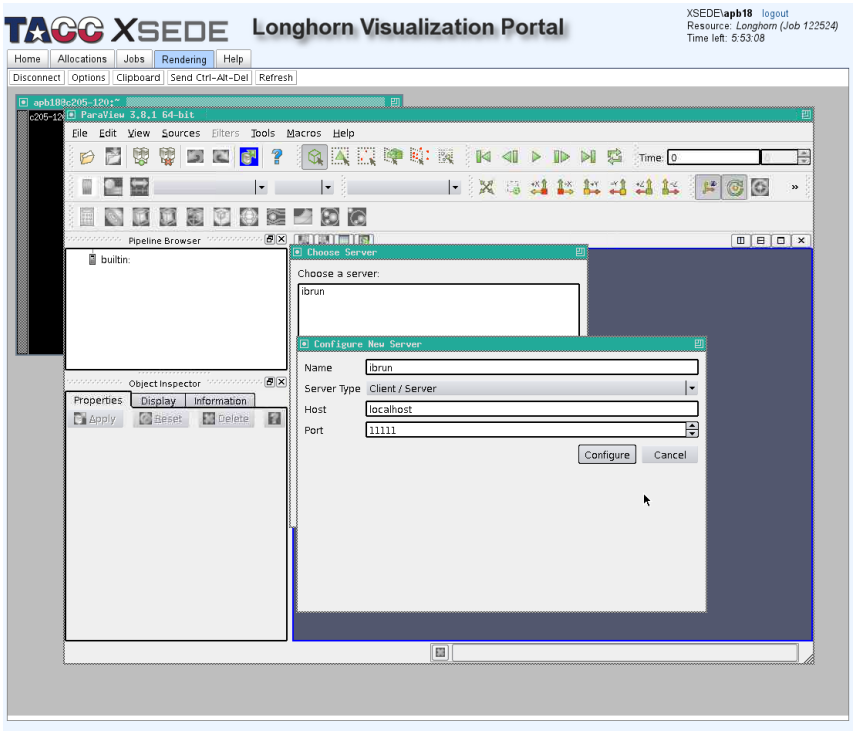
Run ParaView

- Click the “connect” button (or choose file->connect)
- This will bring up a “choose a server” dialog.
- Select “add server”
- Paraview can launch backend parallel processes for us and connect to them, but we need to configure it to do so.



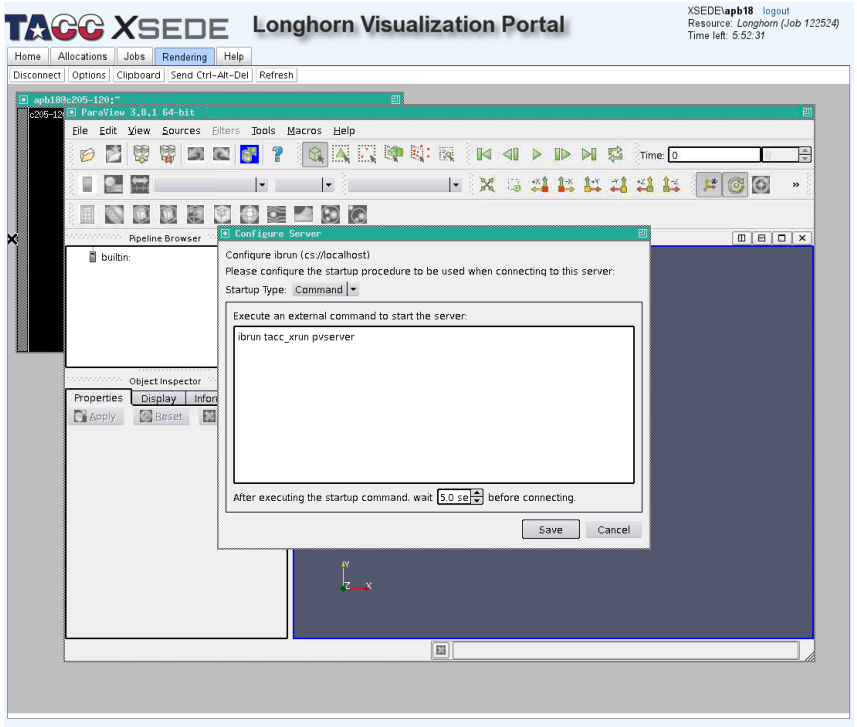
Run ParaView

- Once you click “add”, a “configure new server” dialog will pop up.
- Give it this configuration a name (e.g. “ibrun”) in the “name” field”
- Leave the other fields alone
  - Server Type: Client/Server
  - Host: Localhost
  - Port: 11111
- Click “configure”



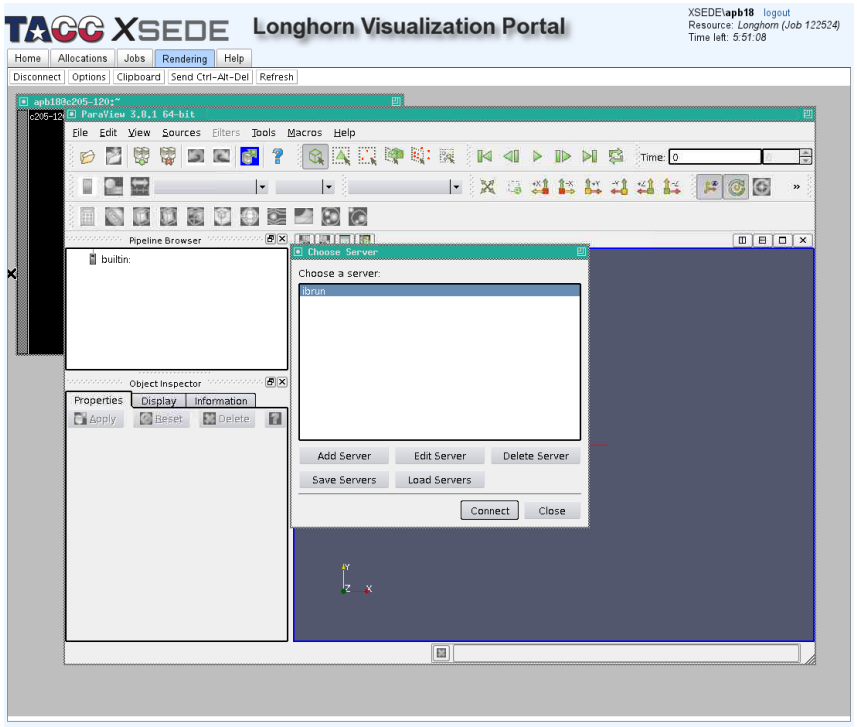
Run ParaView

- A “configure server” dialog will open
- Make sure Startup Type: Command is selected
- In the large text box, type in “`env NO_HOSTSORT=1 ibrun tacc_xrun pvserver`”
  - This is the command that paraview will use to launch parallel backends
- Click “save”



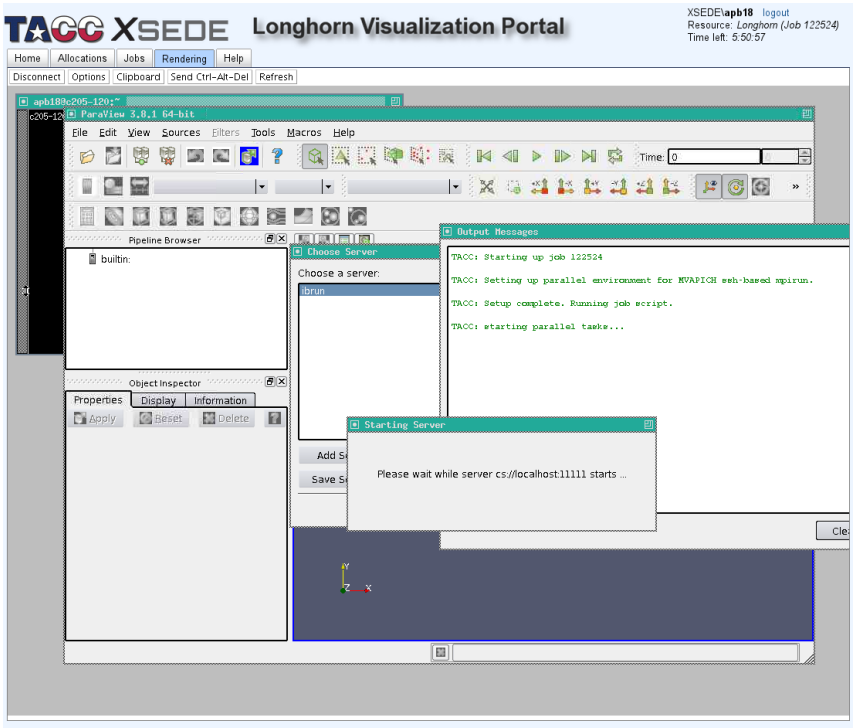
Run ParaView

- Select “Save Servers” to have paraview remember these settings
  - Otherwise, you’ll need to type this information every time you restart paraview!
- It will prompt you for a filename. Just type in something like “servers” and click OK
- Click “connect” to have paraview launch the parallel backends.



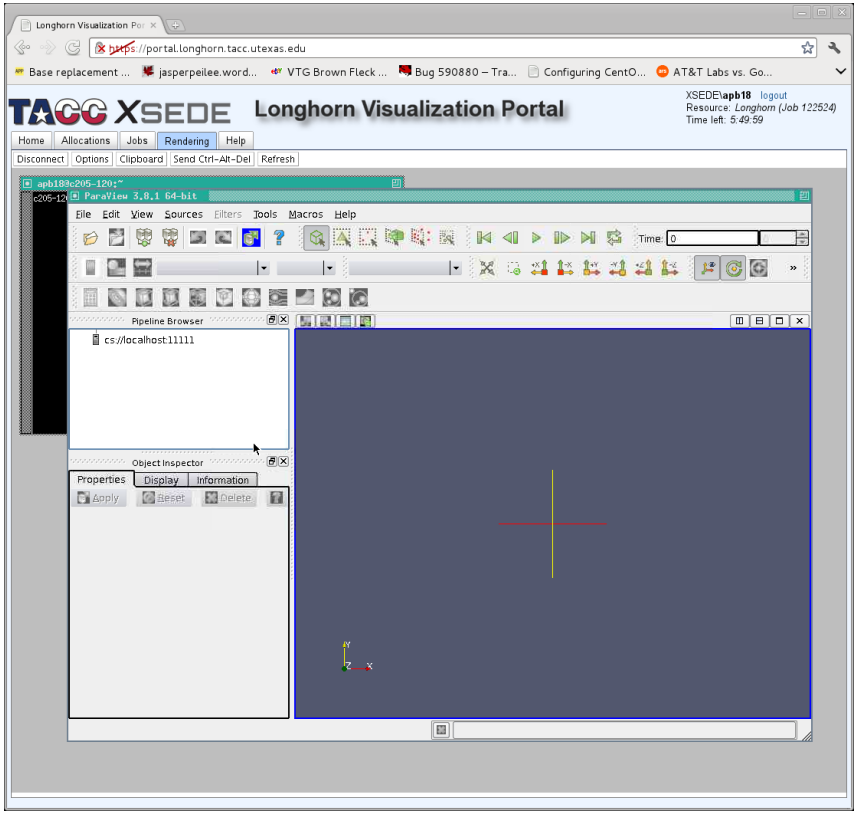
Run ParaView

- You will see several windows pop up that report status.
- Once all backends have been launched, you can close the status dialog window if it's in the way.
- The number and location of backends is automatically determined by your initial settings from submitting the VNC job.



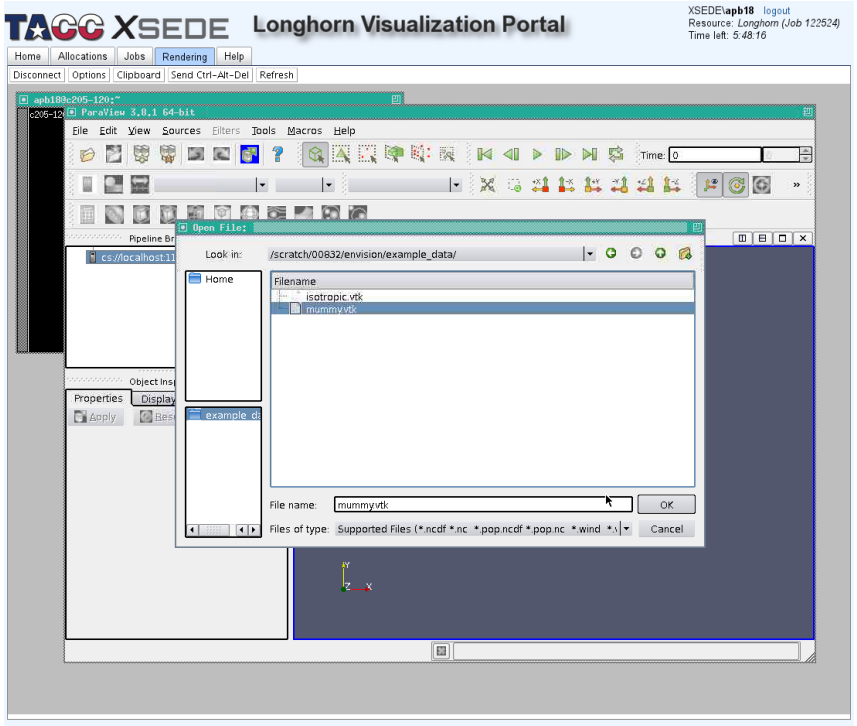
Parallel

- ParaView is now ready to use. It looks no different from “normal”
- The only visible difference: the pipeline browser now starts with “cs://localhost:11111”
- You can proceed as usual, ParaView will automatically use all available resources in parallel.



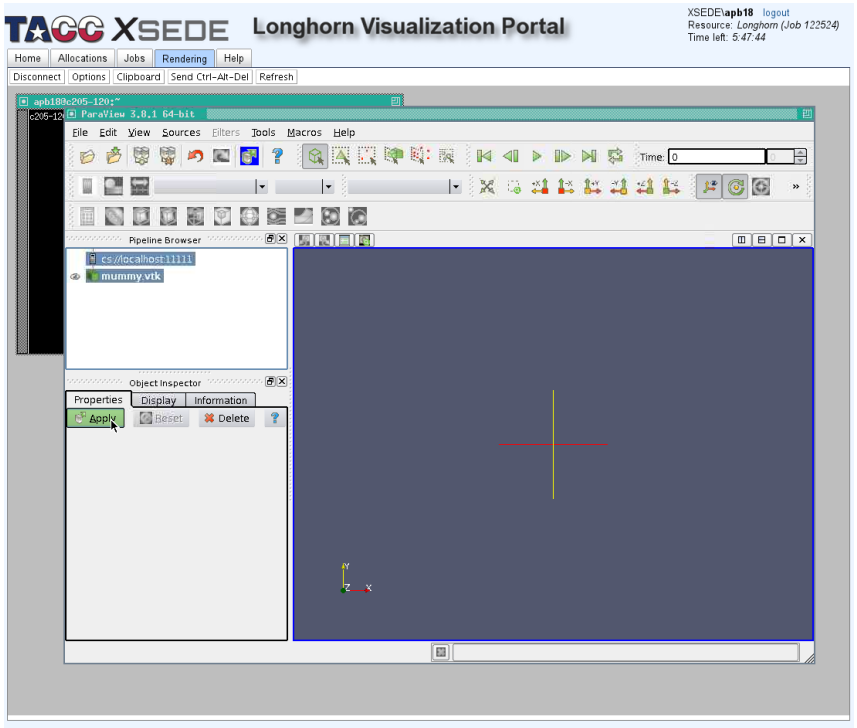
Parallel

- Select File->Open to load the example dataset
- Navigate to `/scratch/00832/envision/example_data/`
- Load the mummy dataset: `mummy.vtk`



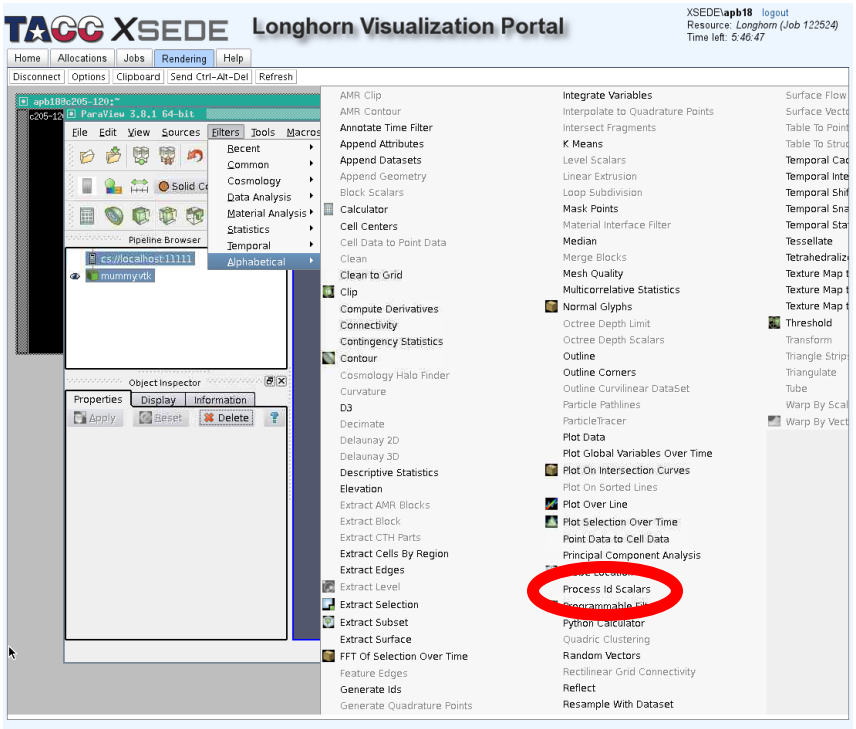
Parallel

- Once loaded, click “apply” in the “properties” tab of the object inspector
- We will now add a filter which augments this data by adding another variable representing backend process ID for each point. This will indicate which backend is processing which portion of the data.



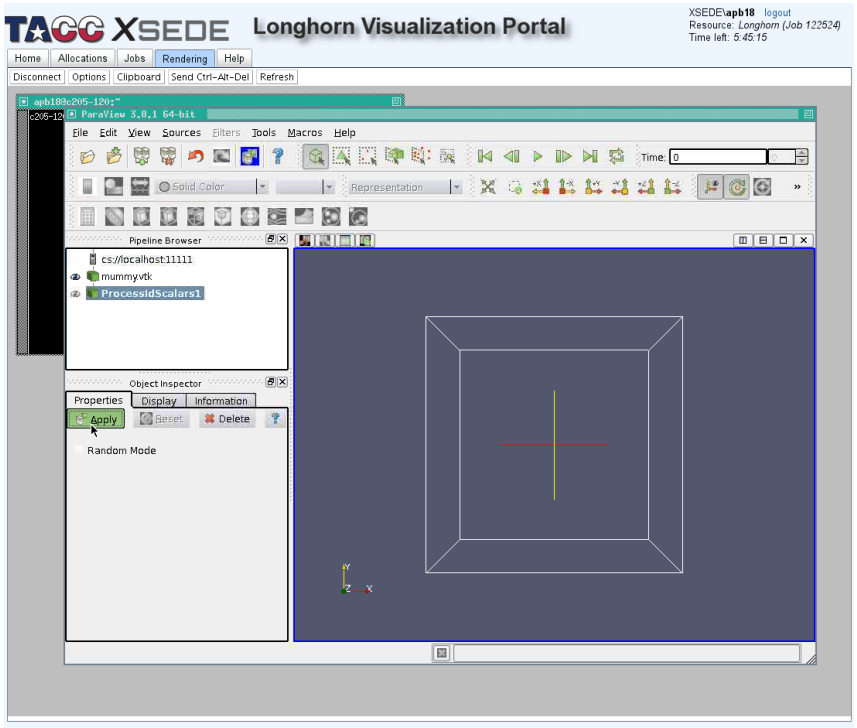
Parallel

- In the “Filters” menu, go down to “Alphabetical”
- This will open a large menu showing all available filters. Navigate to the “Process Id Scalars” filter and choose it.



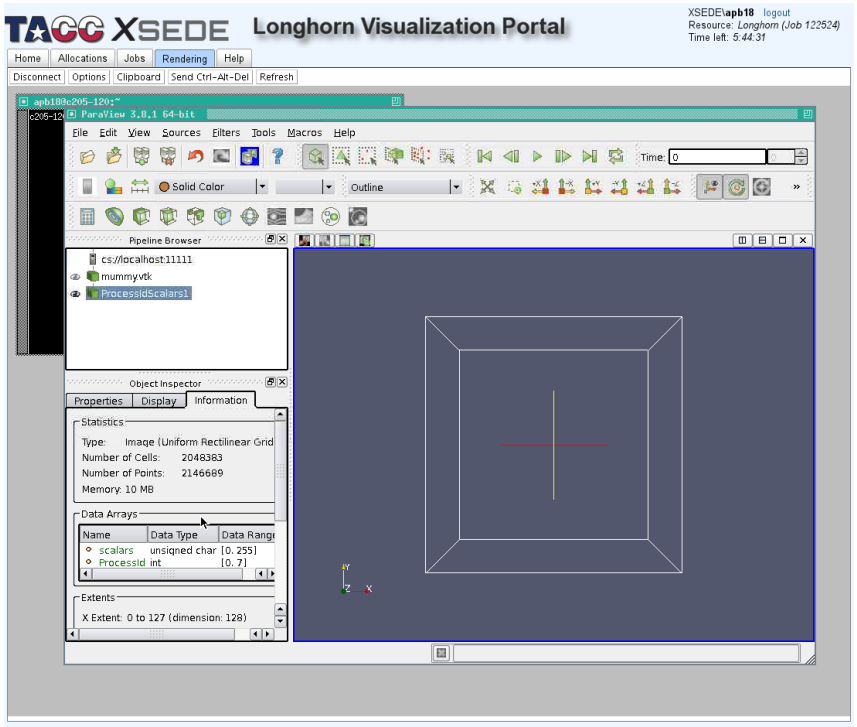
Parallel

- Click “apply” in the properties tab of the object inspector for the ProcessID Scalars filter.



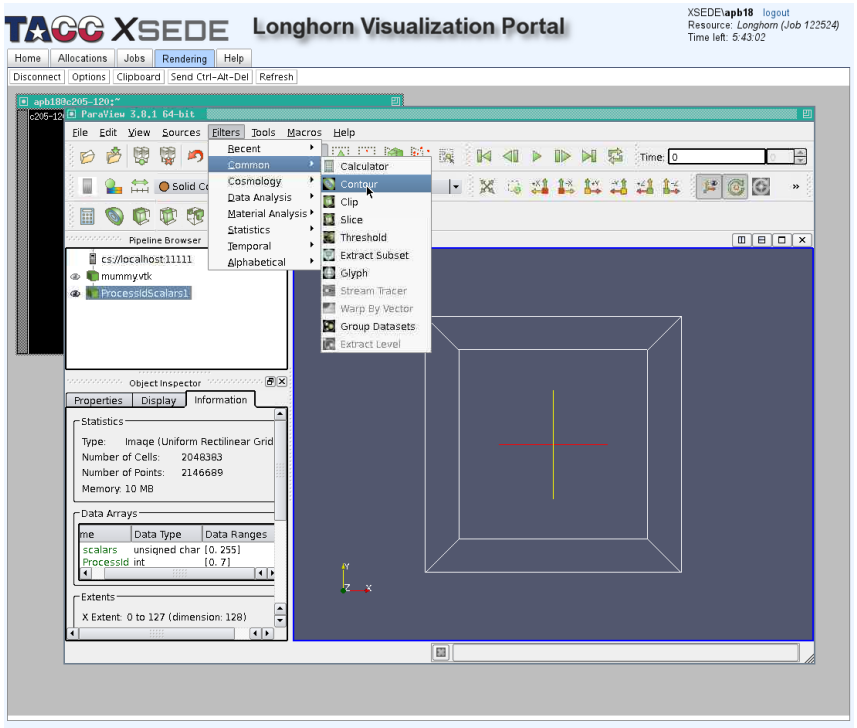
Parallel

- Go to the “information” tab of the object inspector for the ProcessID Scalars filter. Scroll down until you can see the “Data Arrays” section.
- Observe that there is a new array named “ProcessId” containing integers ranging from 0-7.
  - These values map to our eight backend servers.



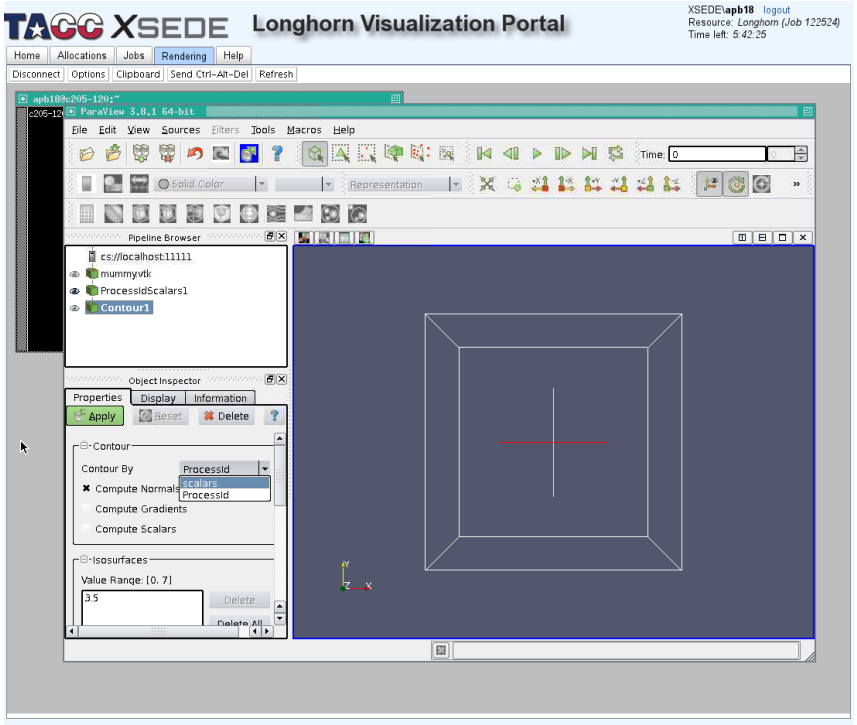
Parallel

- Add and apply a contour filter via Filters -> Common -> Contour
- We will use this to produce an isosurface of the mummy skull where the value is 128.
- We will color this surface by ProcessId
- Be sure to click “Apply” after selecting the Contour filter.



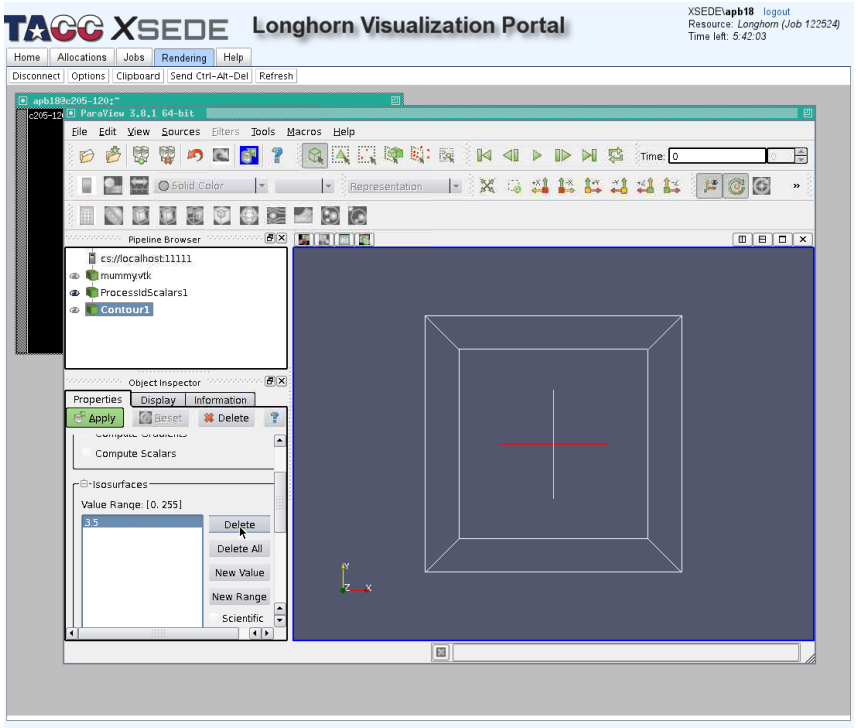
Parallel

- In the “properties” tab of the object inspector for the contour filter, find the “contour section”
- The value will likely be “ProcessId” change it to “scalars”
  - We want to use the “scalars” data to produce the surface.



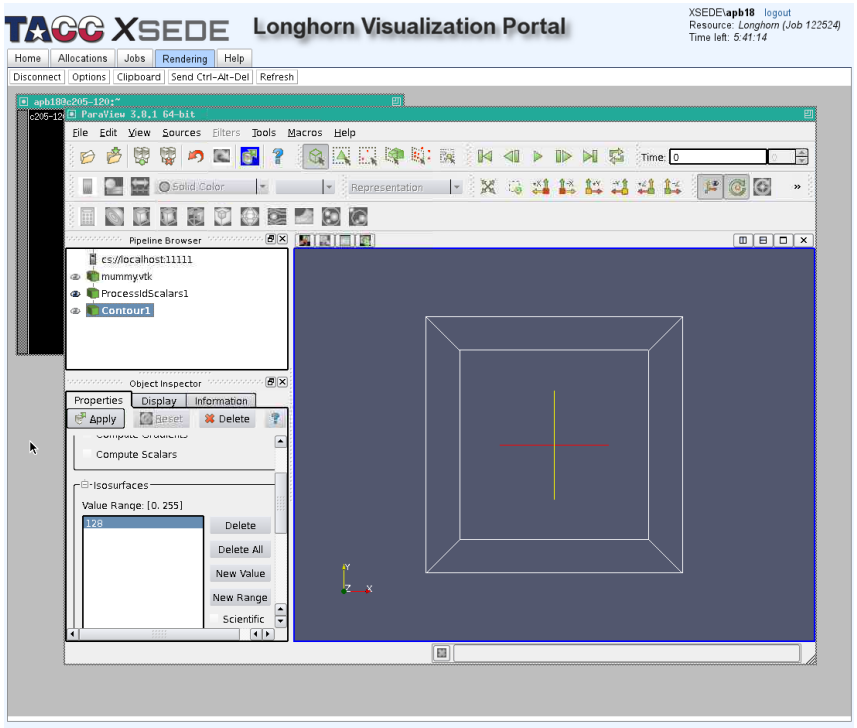
Parallel

- Still in the “properties” tab, scroll down to the “isosurfaces” section. Delete the existing value (probably) 3.5, and add a new value of 128.



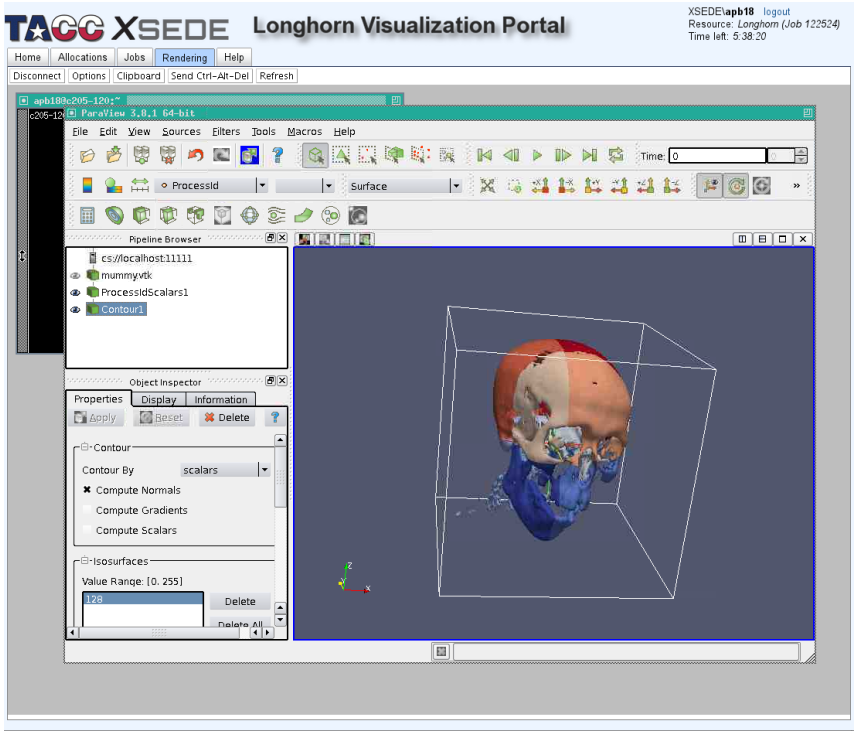
Parallel

- When finished, you should have a single value of “128” selected.
- Click “Apply”.
- Go to the “Display” tab and in the “color” section, verify that “ProcessId” is selected in the “Color by” selector.



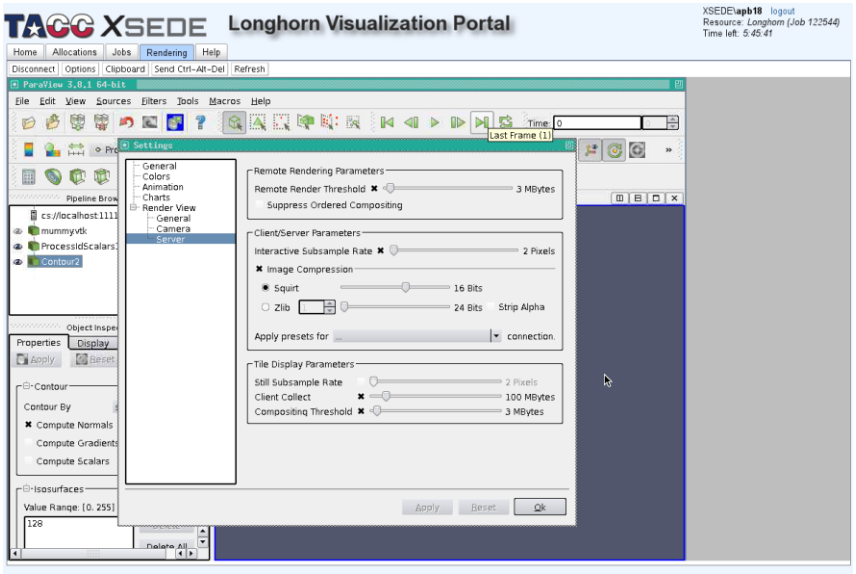
Parallel

- You should now see a rendering where the contour surface is determined by the original data, but colored by process ID.
- Since we have 2 nodes at 4 processes per node, we should see 8 distinct colors. This is showing data parallelism.



Parallel

- ParaView will automatically determine if backends perform data processing only (sending triangles for client to render), or perform rendering as well (send the pixels to directly display).
- This can be tweaked via Edit->Settings->Render View->Server



Experiment!

- Try applying the tetrahedralize filter, viewing by volume, looking at memory usage, tweaking remote rendering threshold in settings
- Load the other dataset in the examples directory (isotropic.vtk), try plotting streamlines and coloring by process Ids
- Try running VisIt in parallel mode.