Visualization Applications on HPC

Let’s Visualize

Charles Peterson

👋 Welcome! 💡 Let's Visualize with HPC

In this workshop, we aim to explore the various ways of accessing graphical software and utilizing Graphical User Interfaces (GUI) on Hoffman2, an HPC system. This workshop is designed for researchers, data scientists, and anyone interested in leveraging HPC for visualization tasks.

We will cover:

- 💻 Different ways to visualize software
- 👉 Hands-on examples with simple software
- 📖 Quick introduction to X11, NoMachine, Jupyter, etc.
- 💌 Suggestions and Feedback, Email cpeterson@oarc.ucla.edu

📖 Get the Workshop Files

This presentation and accompanying materials are available on 🔗 UCLA OARC GitHub Repository

You can view the slides in:

- 📄 PDF format - VisualizationHPC.pdf
- 🌐 HTML format: Workshop Slides

Each file provides detailed instructions and examples on the various topics covered in this workshop.

> **Note:** 🛠️ This presentation was built using Quarto and RStudio.

🎨 Let's Visualize

On Hoffman2, typically you would login to a terminal and interact via a Command Line Interface (CLI). However, there are instances where you might want to visualize data or interact with a GUI interface. We will explore various methods to accomplish this, highlighting the benefits and potential challenges of each approach.

We will go over some ways you can do this

X11 Forwarding in HPC

What is X11 Forwarding?

- X11 forwarding lets you run graphical applications installed on a remote server and see the graphics on your local machine.
- It’s a protocol often used in Unix-like operating systems for sending graphical applications over a network.

X11 Forwarding in HPC

- The X11 server runs on your local machine and accepts display updates from the remote server, while passing your inputs to the remote server.
  - X11 forwarding over SSH lets you execute a command on a remote server while creating a secure tunnel for the graphical output.
- Many HPC applications have GUIs that require X11 to display on your local machine.
  - Matlab Desktop, Mathematica Notebooks, gnuplot, etc.

Tip

You'll need to install and setup X11 on your local computer to render the graphics.

X11 Forwarding in Windows 💻

If you have a Windows computer, there are many applications that can enable X11 Forwarding

- Windows Subsystem for Linux: A Windows native compatibility layer for running Linux binaries on Windows.
    - My recommendation ❤️
- MobaXterm: MobaXterm is a free, enhanced terminal for Windows with X11 server, SSH client, network tools, and much more.
    - 😎 It’s a popular choice among HPC users
- Xming: Xming is X Window System Server for Microsoft.
- Cygwin: Cygwin is a free Linux-like environment for Windows. To add Cygwin/X server, select the xinit package from the X11 category.

MobaXterm

- Begin by installing MobaXterm
- Once install, start a local terminal
  - This terminal operates on your **LOCAL** machine
  - To log in to Hoffman2
    - `ssh –Y joebruin@hoffman2.idre.ucla.edu`
    - **The –Y option in SSH enables X11 forwarding**
  - You can also “drag and drop” files to and from your machine using this software

Windows Subsystem for Linux (WSL/WSL2)

- Install
  - Start a Windows Command Prompt in **administrator** mode
  - Execute `wsl --install`
- Operation
  - Open the WSL2 application
  - This is essentially a Linux kernel functioning within your Windows machine
  - Login to Hoffman2 with '-Y'
    - Type `ssh -Y joebruin@hoffman2.idre.ucla.edu`

X11 Forwarding in Mac 🍎

For Mac users, XQuartz is a popular application that supports X windows.

This is a terminal application that facilitates a connection to Hoffman2 with X11 Forwarding enabled.

Once installed, to connect to Hoffman2 and enable X11 Forwarding, run ssh with the -Y option:

```
ssh -Y joebruin@hoffman2.idre.ucla.edu
```

NoMachine

🖥️ Installing NoMachine

NoMachine provides remote desktop functionality on Hoffman2, enabling a Graphical User Interface (GUI) on a remote machine.

- Download NoMachine
  - Use the download link for your operating system (Windows, Mac OSX, or Linux)
- After downloading, install and configure the NoMachine client to connect to Hoffman2:
  - Click on `Add`
  - Set up a `Name` for your connection
  - The `Host` should be `nx.hoffman2.idre.ucla.edu`
  - Set the `Port` to 22 and choose SSH for Protocol

🖥️ Setting Up NoMachine

- After creating this configuration, click on the new icon in the list of machines:
  - You may be prompted to `Verify host identification`
    - If so, click `OK`
  - Enter your Hoffman2 `Username` and `Password`
  - You'll then be asked to `Create a new desktop`
    - Choose to `Create a new Mate virtual desktop`
  - Upon completion, you'll be presented with a desktop view of your Hoffman2 account:
    - Open a terminal to start any GUI applications
      - However, you may want to initiate an interactive job first
        - To do this, type `qrsh -l h_data=10G`

Jupyter/Rstudio

📔 Jupyter

Jupyter can be executed on Hoffman2. This process involves running the `jupyter notebook/lab` command on a Hoffman2 node, forwarding the Jupyter port to your local machine, and accessing Jupyter via your local web browser.

However, for a more streamlined experience, we have created a script, `h2jupynb`, which automatically creates a Jupyter session on a Hoffman2 compute node.

```
wget https://raw.githubusercontent.com/rdauria/jupyter-notebook/main/h2jupynb
chmod u+x h2jupynb
```

Setting up Jupyter Session

Ensure Python is installed on your local machine before executing the script:

```
python h2jupynb -u joebruin -t 5 -m 10 -a "intel-gold\*"
```

The above command will initiate the script with your Hoffman2 username (replace “joebruin” with your username), a duration of 5 hours (-t 5), memory request of 10 GB (-m 10), and specify the architecture as “intel-gold”.

```
Usage:

h2jupynb [-u <Hoffman2 user name>] [-v <python-version>]
         [-t <time, integer number of hours>] [-m <memory, integer number of GB per core>]
         [-e <parallel environment: 1 for shared, 2 for distributed>] [-s <number of slots>]
         [-o <run on group owned nodes: yes/no>] [-x <run on an exclusively reserved node: yes/
         [-a <CPU-type>] [-d <path to directory from which the jupyter NB/lab will start>]
         [-g <run on a gpu node: yes/no>] [-c <gpu-card-type>] [-l <cuda-version>]
         [-p <port number>]
```

Utilizing Anaconda Environment in Jupyter 🐍

To use your Anaconda environment created on Hoffman2 with the `h2jupynb` script, you'll need to install ipykernel in your Anaconda environment.

Let's say your Anaconda environment is named `myconda`. Follow the steps below on Hoffman2:

Request an interactive session with 10GB of memory for 1 hour:

```
qrsh -l h_data=10G,h_rt=1:00:00
```

Load Anaconda and activate your environment:

```
module load anaconda3
conda activate myconda
```

Install ipykernel and add it to your environment:

```
conda install ipykernel
python -m ipykernel install --user --name=myconda --display-name "myconda"
```

🐍 Running Jupyter with Anaconda

Run the h2jupynb script on your local machine using the -v anaconda3 option:

```
python h2jupynb -u joebruin -v anaconda3
```

Now, `myconda` should appear in the list of available kernels in Jupyter.

For more information on using Jupyter on Hoffman2, check out our Hoffman2 documentation

📊 Utilizing RStudio on Hoffman2

You can conveniently interact with Hoffman2 using RStudio Server, allowing you to utilize R in a familiar, intuitive environment.

Find detailed instructions for using RStudio on Hoffman2 on our GitHub page

We've also conducted a dedicated workshop on using Rstudio on Hoffman2. Feel free to explore this resource for additional insights and usage tips.

Paraview

Paraview

Paraview is an open-source application that supports interactive, scientific visualization.

For using Paraview, you will first need to install ParaView on your local machine.

We will run Paraview on a compute node as a server process. Then use our local install of Paraview to connect to Hoffman2.

CPU Mode

Initiate an interactive job on a compute node and load the ParaView module.

```
qrsh -l h_data=10G
echo $HOSTNAME #Take note of the hostname of the compute node you landed on
module load apptainer
apptainer exec $H2_CONTAINER_LOC/h2-paraview_5.10.0-cpu.sif pvserver --server-port=11111
```

Leave this terminal open.

Open another local terminal for port forwarding:

```
ssh -L 11111:nXXXX:11111 joebruin@hoffman2.idre.ucla.edu
```

Replace `nXXXX` with the hostname of the compute node and `joebruin` with your Hoffman2 username.

- Then you will start up ParaView on your local machine
  - This MUST be version 5.10.0 of paraview (though 5.10.1 may work).
- Click on `Connect`
  - Create a Server with `localhost:11111`

GPU Mode (EGL rendering)

To utilize the GPU compute nodes, follow the steps below:

```
qrsh -l h_data=5G,gpu,V100
apptainer exec --nv $H2_CONTAINER_LOC/h2-paraview_5.10.0-gpu.sif pvserver --server-p
```

Follow the same port forwarding and connection steps as described above.

👏 Thanks for Joining! ❤️

Questions? Comments?

- cpeterson@oarc.ucla.edu
- Look at for more Hoffman2 workshops at https://idre.ucla.edu/calendar