

# Interactive High-performance (HPC) Computing

HPC & Data Science Summer Institute 2024

By: Mary Thomas

<https://github.com/sdsc-complecs/interactive-computing/>

**SDSC**  
SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

# Outline

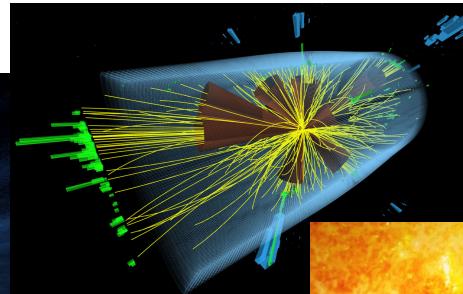
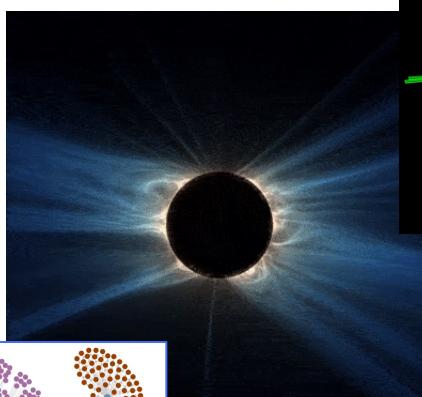
- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

# You can run amazing jobs on supercomputers!

## Typically parallel or very large memory



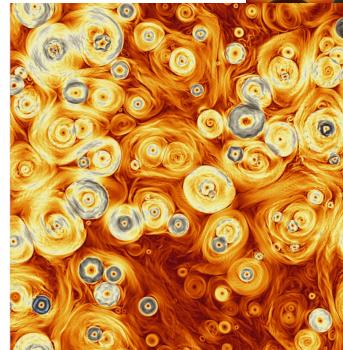
Pelagic fish communities Shapes  
(Jerome Guiet, UCLA)



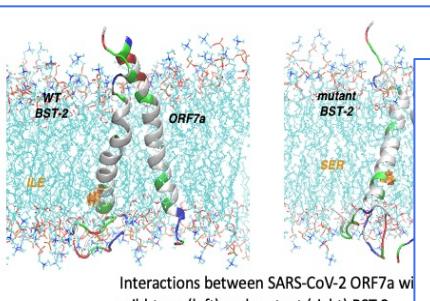
Compact Muon Solenoid (CMS) experiment at the LHC, CERN. Image courtesy of CMS Collaboration; Mc Cauley, Thomas



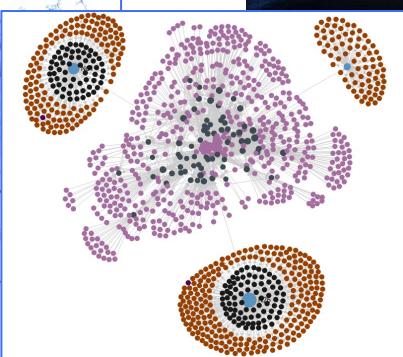
Model of Both Inner and Outer Solar System  
M. S.Clement (Carnegie Institution for Science)



Modeling the sun's corona,  
Alfred Mallet (UC Berkeley)



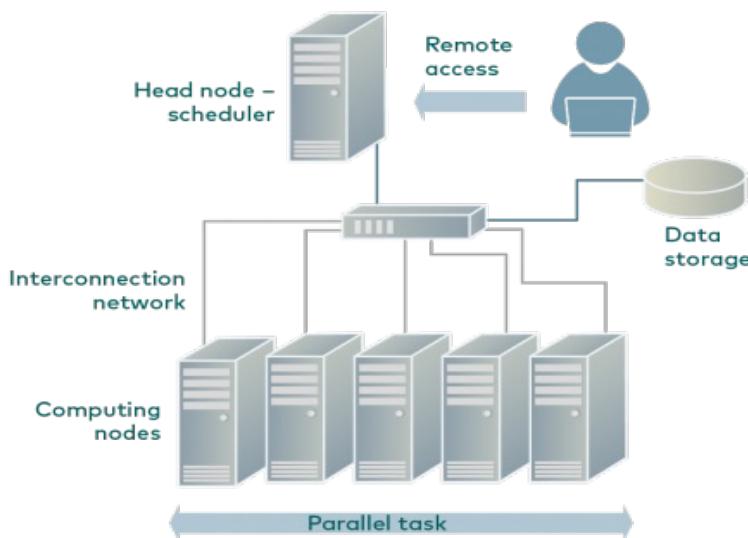
Interactions between SARS-CoV-2 ORF7a with wild-type (left) and mutant (right) BST-2  
Jeff Klauda /U. Maryland



Cooper Downs,  
Predictive Science Inc

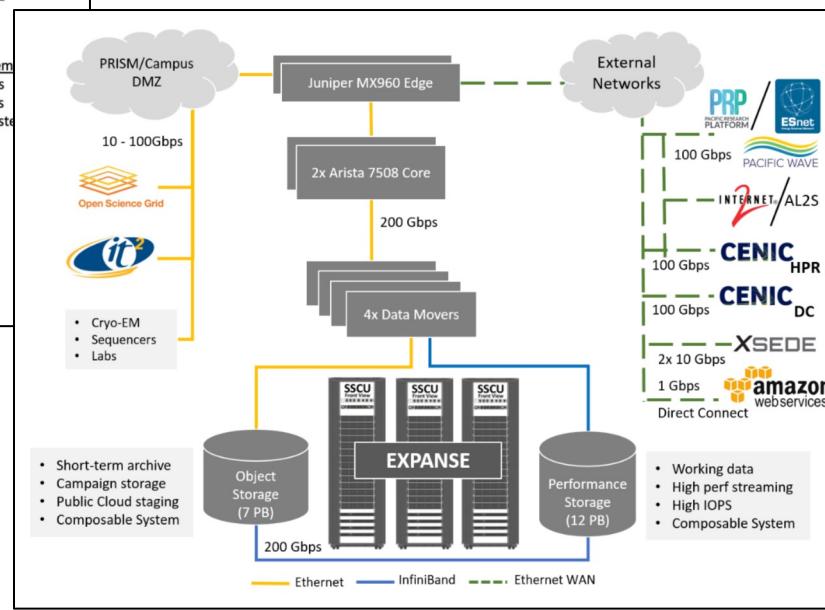
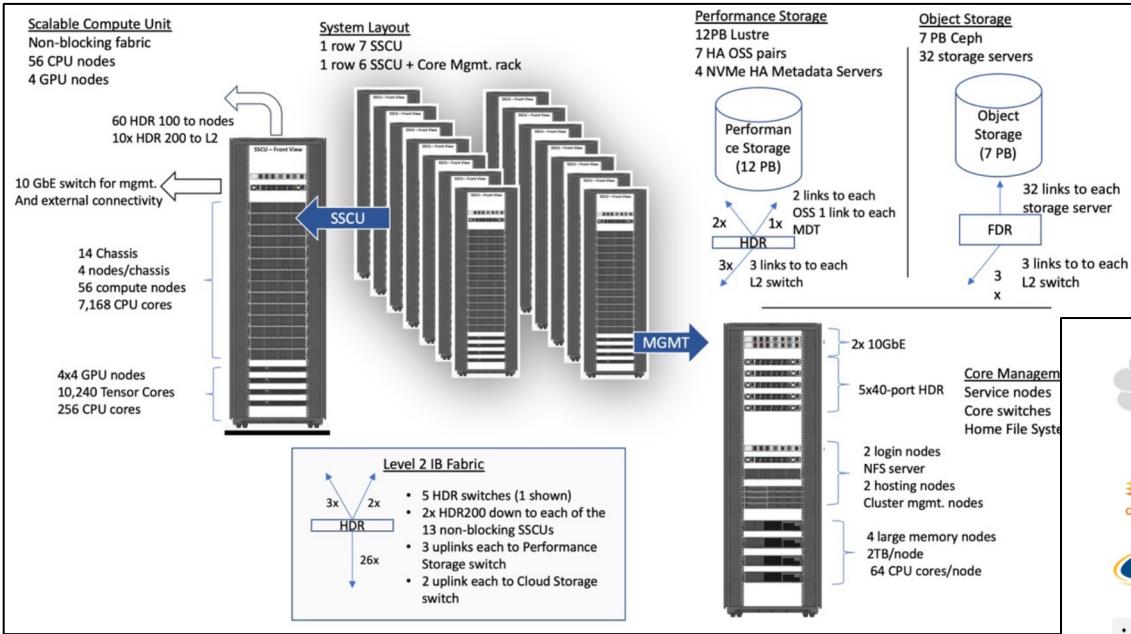
Sample of Internet  
structure from  
CAIDA data (Mark  
Burgess, 12/16/21)

# However, you have to run your jobs on HPC clusters



- Systems powerful but complex
- Jobs can be run from:
  - Command line interface (CLI)
  - The batch queue system
  - Application clients (PyTorch)
- Job scaling:
  - Parallel: 1 core → 1000s of nodes
  - For large jobs, **schedulers** are needed to coordinate tasks

# HPC System Architecture: Expanse @ SDSC



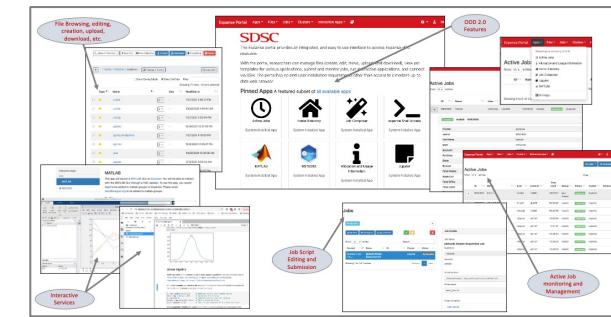
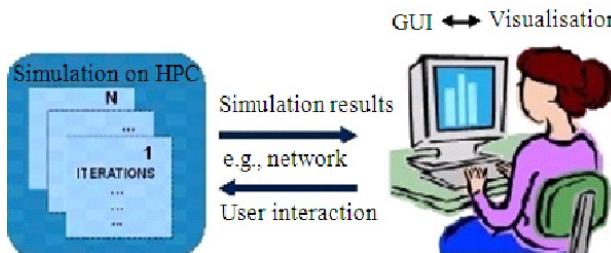
<https://expanse.sdsc.edu>

[https://www.youtube.com/watch?v=uNZyg6X\\_t3s](https://www.youtube.com/watch?v=uNZyg6X_t3s)

<https://access-ci.org>

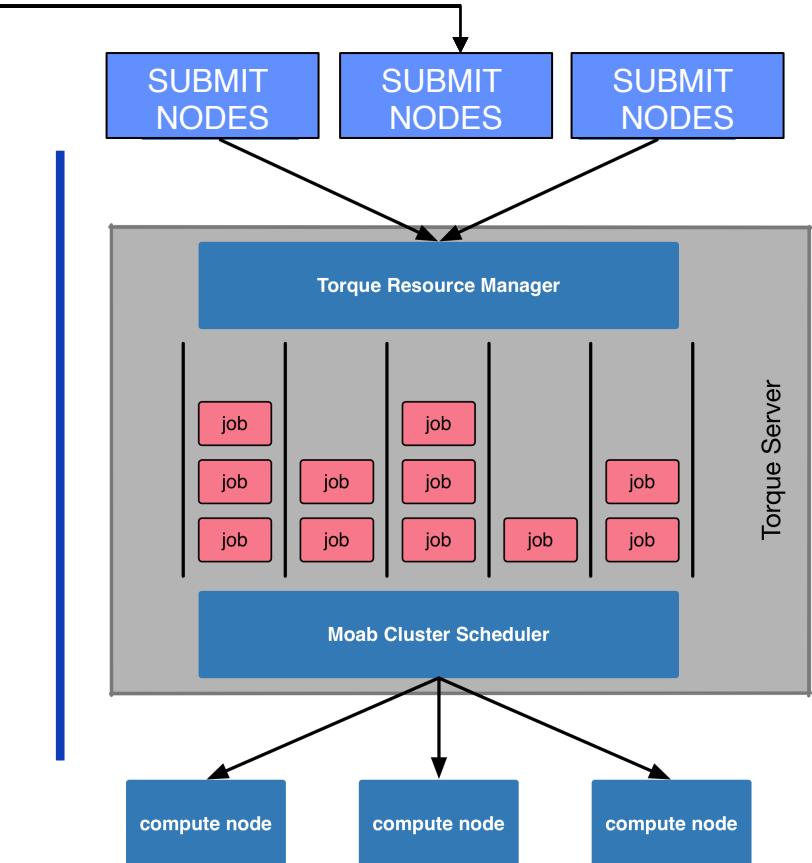
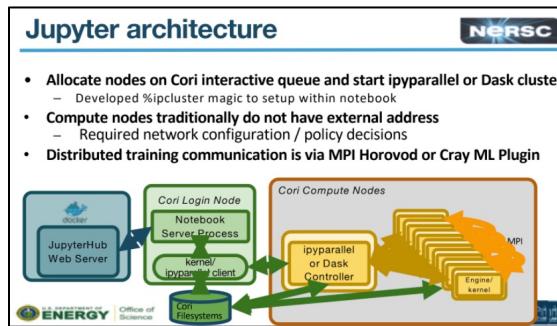
# HPC Jobs can be run as batch jobs (background) or interactively (real time)

Terminal shell – submit batch



<https://portal.expanse.edu>

Interactive Distributed Computing with Jupyter



Src: [https://hpc.dccn.nl/\\_images/torque\\_moab\\_arch.png](https://hpc.dccn.nl/_images/torque_moab_arch.png)

# Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

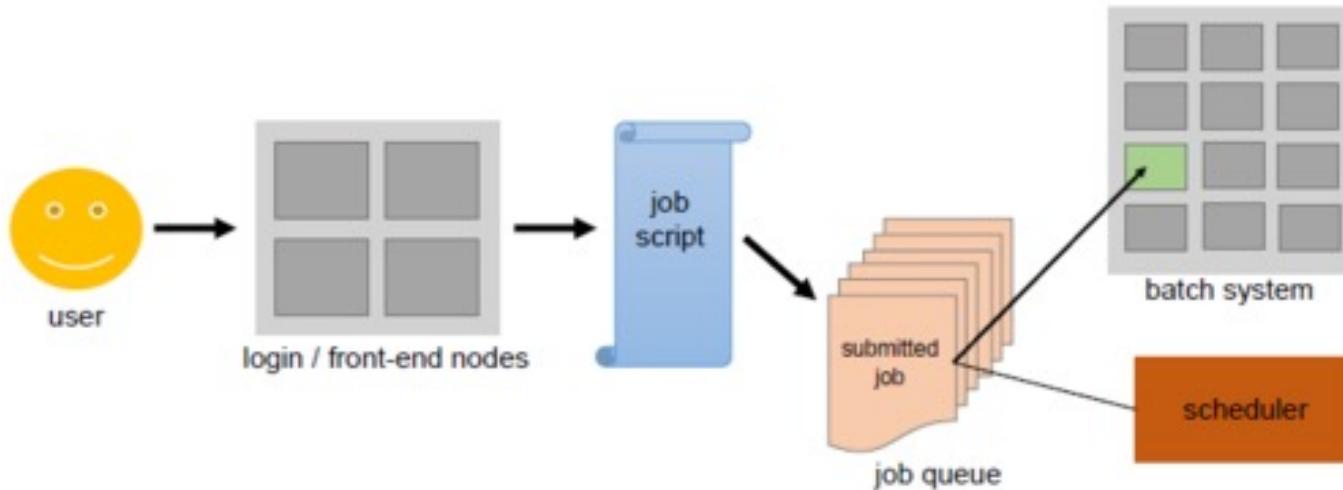
# What is a batch job scheduler & how are they used?

- Any HPC (or HTC) system — usually a cluster of machines — needs a means of **sharing computational resources** fairly between users; without this, there would be anarchy.
- **Batch-queueing systems** — usually abbreviated to simply *batch systems* — are intended to do this.
- All batch systems have at least these features:
  - *a scheduler* for allocating resources (CPUs!) to jobs and for prioritizing jobs;
  - *one or more queues* to which jobs are submitted
  - Job **partitions** (or job queues): queue can be configured for a **particular type of job** (serial or parallel jobs; long or short jobs; or those requiring particularly high memory); members of a group or project).
- You need a basic understanding of batch jobs work in order to run interactive jobs on HPC Clusters (ironic?)

# Batch Scheduler: main goals

- **Minimize time** between job submission and completion:
  - No job should stay in queue for extensive periods of time.
- **Optimize CPU utilization:**
  - Algorithms focus on minimizing CPU idle times.
- **Maximize job throughput:**
  - Manage as many jobs per time unit as possible.
- **Support** running jobs automatically in the background
- **Finalizes jobs:** ensures job data and results are recorded & stored where you want them.

# Simple “Batch Scheduler” Architecture



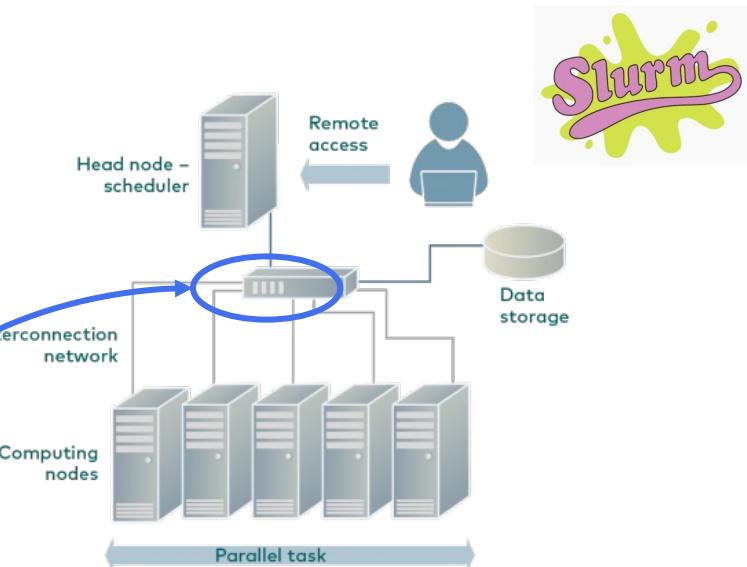
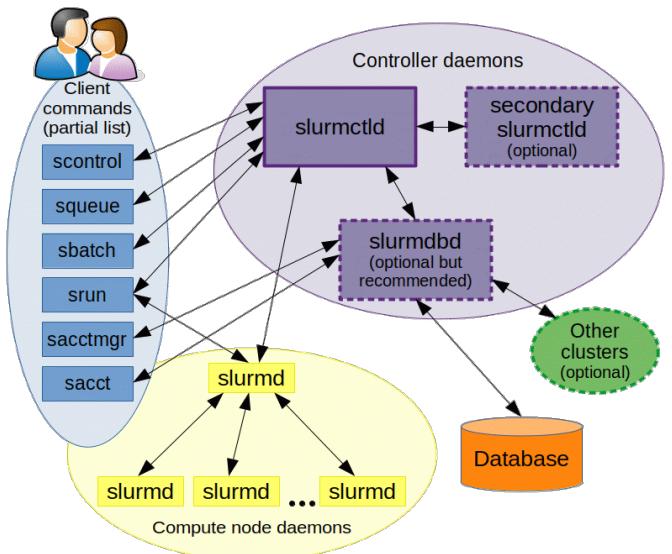
- Batch scheduler: software that implements a *batch system* on a cluster.
- Users do not run calculations interactively -- instead they submit *non-interactive batch jobs* to the *scheduler*.
- All work about the same: some are open source; some cost money; some are very expensive.

[https://hpc-wiki.info/hpc/Scheduling\\_Basics](https://hpc-wiki.info/hpc/Scheduling_Basics)

# Batch jobs: Slurm resource manager

Simple Linux Utility for Resource Management

- Open Source, runs on many systems
- “Glue” for parallel computer to schedule and execute jobs
- Role: Allocate resources within a cluster
  - Nodes (unique IP address)
  - Interconnect/switches
  - Generic resources (e.g. GPUs)
  - Launch and otherwise manage jobs



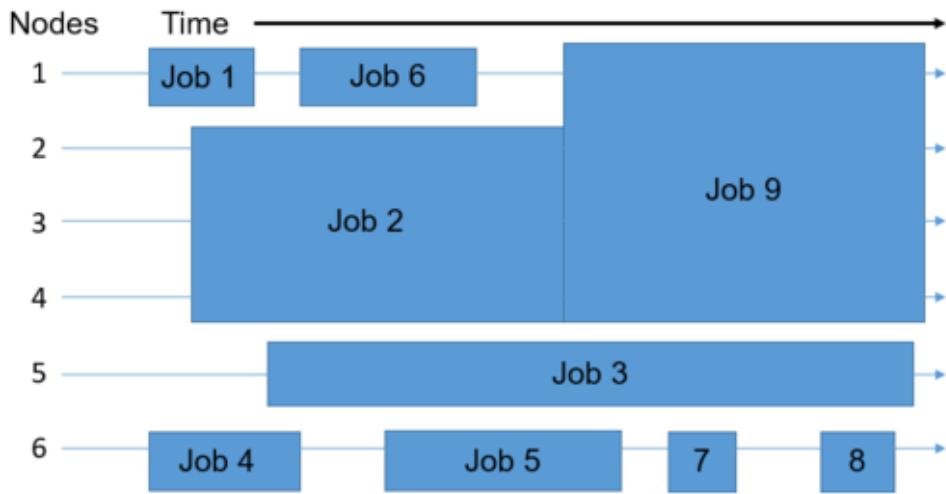
## Functionality:

- Prioritize queue(s) of jobs;
- decide when and where to start jobs;
- terminate job when done;
- Appropriate resources;
- manage accounts for jobs

# How a scheduler schedules jobs

- Simple example:
  - a 6-node system
  - user wants to run 9 jobs.
- Scheduler places the jobs in the queue and then onto the available nodes as they open up.

*Many parameters affect scheduling: number of jobs submitted, required runtime, required number of cores, required main memory, accelerators, libraries, etc.*



**Scheduler needs to play kind of "multidimensional tetris" to fill the cluster's nodes evenly and efficiently.**

[https://hpc-wiki.info/hpc/Scheduling\\_Basics](https://hpc-wiki.info/hpc/Scheduling_Basics)

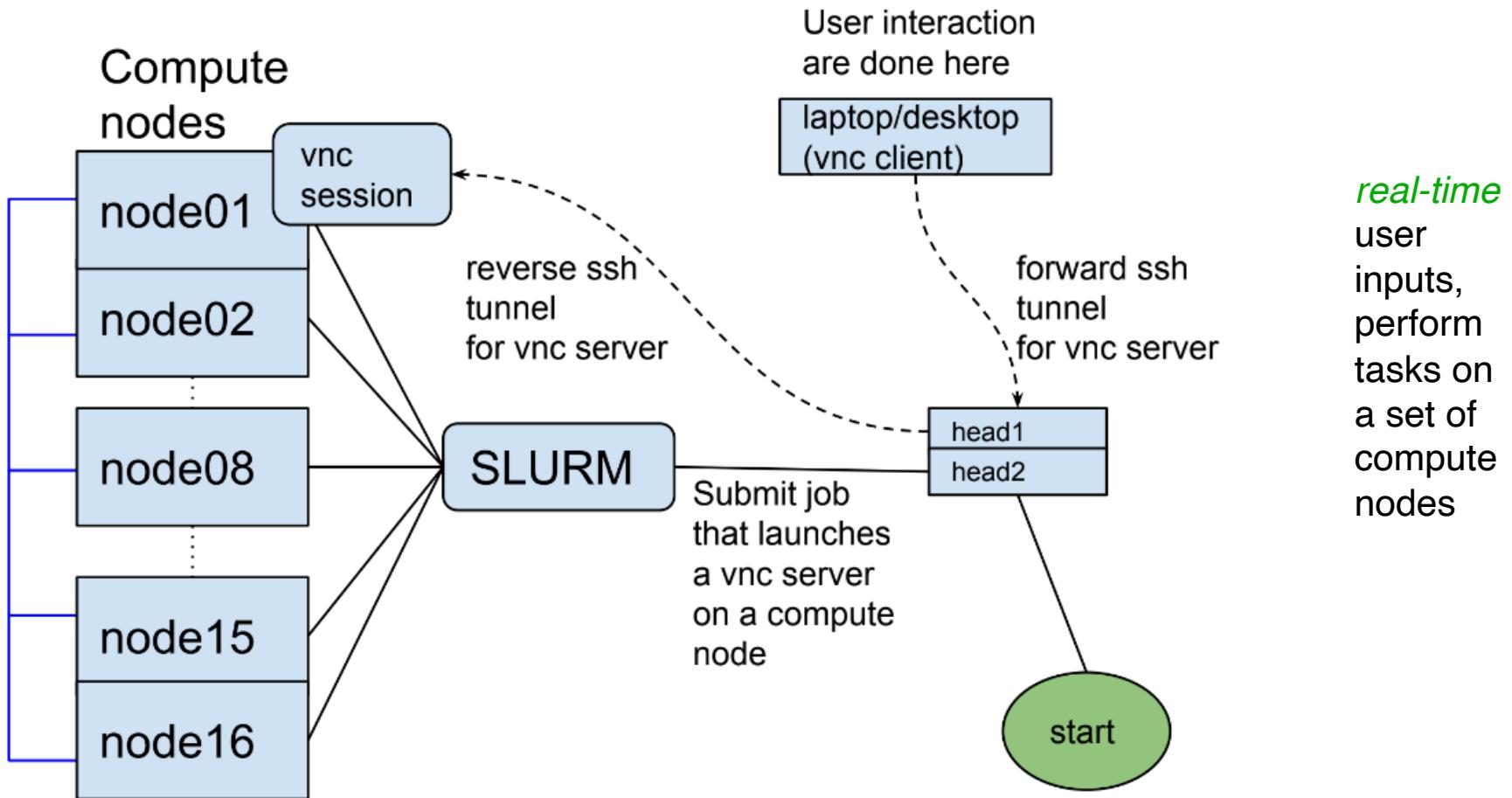
# Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

# What is Interactive HPC Computing

- In **computer science**, **interactive computing** refers to software which accepts input from the user as it runs.
  - **Interactive** software includes commonly used programs, such as word processors or spreadsheet applications.
- **Interactive HPC computing** involves *real-time* user inputs to perform tasks on a set of compute node(s) including:
  - Code development, real-time data exploration, and visualizations.
  - Used when applications have large data sets or are too large to download to local device, software is difficult install, etc.
  - User inputs come via command line interface or application GUI (Jupyter Notebooks, Matlab, R-studio).
  - Actions performed on remote compute nodes as a result of user input or program out.

# Real-time User Interactions



# Interactive HPC Computing- Motivation

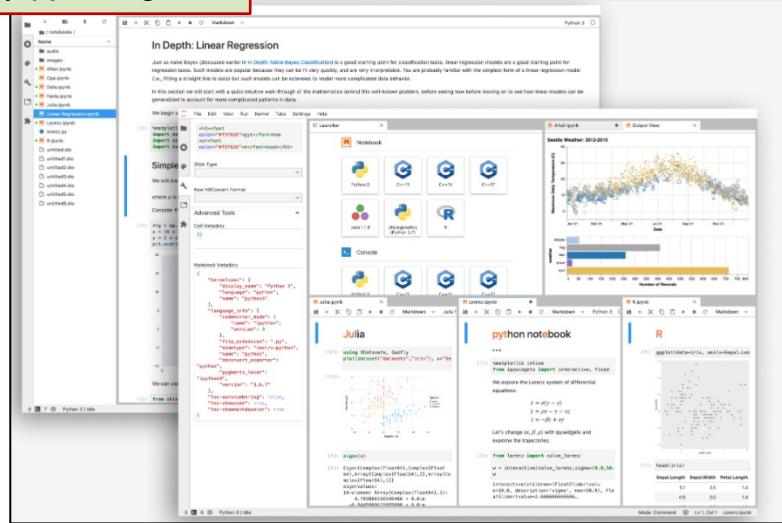
- **Need more memory:** Your jobs no longer fit onto the CPU/system you have been using:
  - That is why Expanse has 768 nodes (128 cores per node), with 256 GB DDR memory on each node
  - My MacBook Pro: 1 node, 8 cores, 16 GB DDR
- **Too much data:** your application needs more room:
  - Expanse has 1TB NVME/node, and 12 PB file system.
  - My MBP: 500 GB, no NVME
- **Your network is too slow:**
  - Expanse has connections to ~ 150GB/sec (or faster) networks
  - My MBP: 300 Mbps download; 11 Mbps upload
  - Bioinfo lab, running analysis on PC: FastQ dataset ~ 500 MB: would need 360+ seconds to upload 1 run.

# Interactive HPC Methods & Applications

- Interact with data after job is done:
  - Unix: query file info, location, output, grep, awk, sed
  - Cat the file contents from batch job or raw data
  - Data browser
- Plot results:
  - From within the code/model using libraries
  - Command-line driven graphing utility : Gnuplot
- Data visualization apps
  - NetCDF, HPF, TAU, ParaView
  - Notebooks, others
- Data Analysis Platforms: Matlab, R

# Interactive HPC Computing

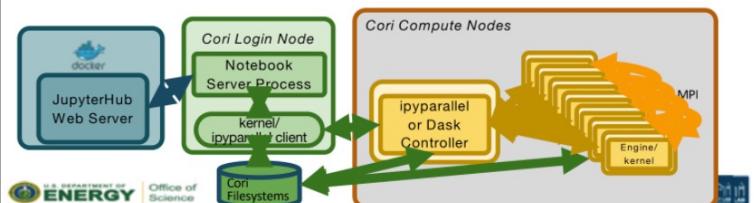
<https://jupyter.org/>



Interactive Distributed Computing with Jupyter (NERSC)

## Jupyter architecture

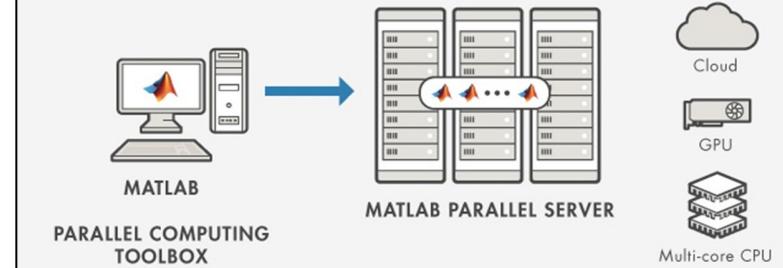
- Allocate nodes on Cori interactive queue and start ipyparallel or Dask cluster
  - Developed %ipcluster magic to setup within notebook
- Compute nodes traditionally do not have external address
  - Required network configuration / policy decisions
- Distributed training communication is via MPI Horovod or Cray ML Plugin



<https://drive.google.com/file/d/1-OFjrk1q3L1d3uakr2xkozrPn2c2VZpZ/view>

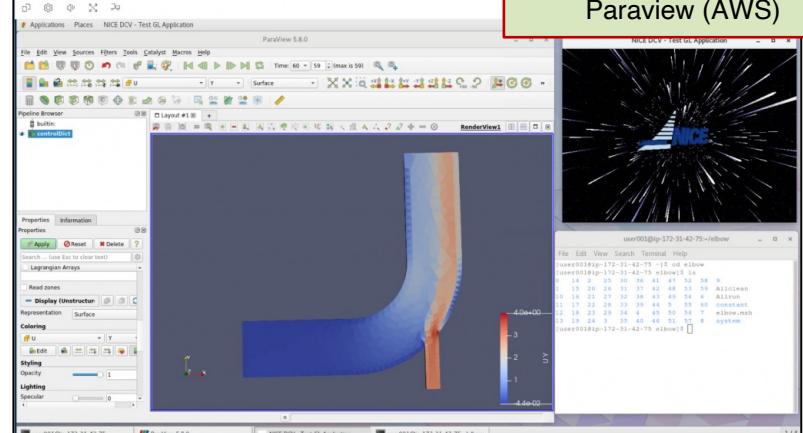
## Parallel Matlab (AWS)

```
>> parpool(parcluster('HPC1'),100);
>> parfor i=1:3000
>> c(i,:) = eig(rand(1000));
>> end
```



<https://azuremarketplace.microsoft.com/en-us/marketplace/apps/mathworks-inc.matlab-parallel-server-listing?tab=Overview>

## Paraview (AWS)



<https://aws.amazon.com/blogs/compute/how-to-run-3d-interactive-applications-with-nice-dcv-in-aws-batch/>

# Expanse User Portal

<https://portal.expanse.sdsc.edu>

- Provides Web based access to interactive applications including Jupyter Notebooks & Jupyter Lab, Matlab, Rstudio.
- Access using your ACCESS portal account

The image shows a composite screenshot of the Expanse User Portal interface across four panels:

- (1) Log onto the Portal:** Shows the main dashboard with a red banner for "Open OnDemand / Jupyter Session".
- (2) Fill out form inputs:** Shows the "Jupyter Session" configuration form with fields for Account (sdsc54), Partition (gpu), Number of cores (1), Memory required per node (GB) (9), and GPUs (optional) (0).
- (3) Copy the URL and paste into Web browser:** Shows the URL <https://dipping-ether-pureblood.expanse-user-content.sdsc.edu/token=e5cb892765d875d33875ae>.
- (4) Monitor status window: may take a long time:** Shows the "SDSC Expanse" status window with a "Job State: Proxied" message and a timeline with four green dots labeled "In Queue", "Running", "Mapped", and "Proxied". A note says "Job has not yet started".
- (5) Access your Jupyter Service:** Shows a Jupyter Notebook interface with a plot of a bell curve and a section titled "Linear algebra" with code demonstrating matrix operations.

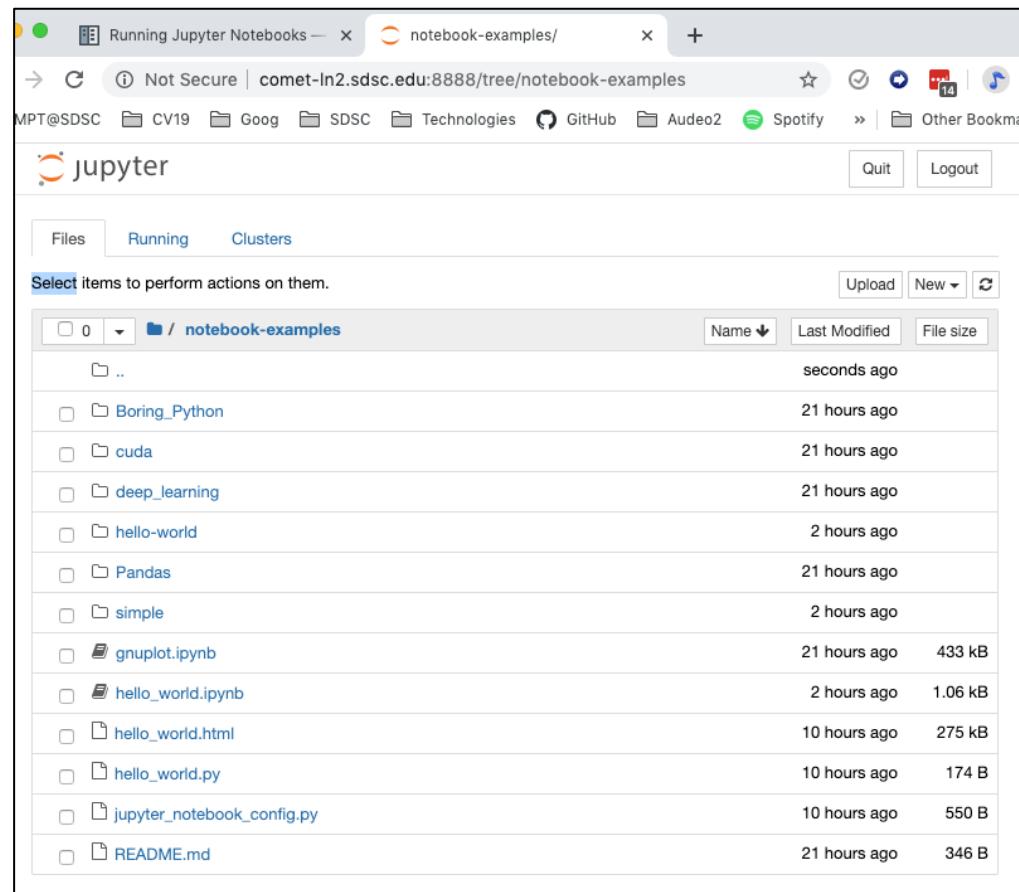
# Interactive Services have a Key Vulnerability: They Provide Access to HPC File Systems, often over HTTP

## SDSC Interactive Services Policy:

- Portals, JupyterHub, and other services cannot be mounted directly to disk (must be on VM or external)
  - Many use root in vulnerable ways
  - If a user launches Jupyter Lab or Notebooks, the jobs will be killed.
- Applications cannot run on login nodes

## SDSC recommendation:

- Use secure connections: when you choose insecure connections your account is vulnerable to hacking
- Use portal.expanse.sdsc.edu



# Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

# When would you need to use an Interactive node?

- The application is interactive (Matlab, Notebooks)
- Need to stage large amounts of data
- AI applications (Pytorch, Tensorflow)
- Need to compile a very large, complex application that may take a long time.
  - Java code, 1 million lines, > 30 minutes on DELL 5511 laptop

# Accessing Interactive Compute Nodes on Expanse

- Connect to HPC system (e.g. Expanse) via terminal using SSH → secure connections
- Use the *srun* command to obtain nodes for ‘live,’ command line interactive access:

CPU	<code>srun --partition=debug --pty --account=&lt;&lt;project&gt;&gt; --nodes=1 --ntasks-per-node=4 --mem=8G -t 00:30:00 --wait=0 --export=ALL /bin/bash</code>
GPU	<code>srun --partition=gpu-debug --pty --account=&lt;&lt;project&gt;&gt; --ntasks-per-node=10 --nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash</code>

(Tested 04/17/2024)

# Using An Interactive CPU node

```
[mthomas@login01 calc-prime]$ srun --partition=gpu-debug --pty --account=use300 --ntasks-per-node=10 --nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash
```

```
srun: job 24457429 has been allocated resources
[mthomas@exp-9-55 calc-prime]$ module purge
[mthomas@exp-9-55 calc-prime]$ module load slurm
[mthomas@exp-9-55 calc-prime]$ module load cpu
[mthomas@exp-9-55 calc-prime]$ module load gcc/10.2.0
[mthomas@exp-9-55 calc-prime]$ module load openmpi/4.1.1
[mthomas@exp-9-55 calc-prime]$ mpirun -n 64 ./mpi_prime
06 August 2023 11:10:26 PM
PRIME_MPI    n_hi= 5000000      C/MPI version
An MPI example program to count the number of primes: #
processes is 64
      N          Pi        Time
      1            0       0.013258
      2            1       0.001058
      4            2       0.000101
      8            4       0.000101
      [SNIP]
     131072      12251     0.110848
     262144      23000     0.410792
     524288      43390     1.527210
    1048576      82025     5.733612
    2097152     155611    21.725862
PRIME_MPI - Master process: Normal end of execution.
06 August 2023 11:12:26 PM
```

Request an interactive node  
for 30 minutes

- Exit interactive session when your work is done or you will be charged more CPU time.
- Beware of oversubscribing your job: don't ask for more cores than you have requested.
- Intel compiler allows this, but your performance will be degraded.

# Using Interactive GPU nodes

```
[snip]
```

```
Last login: Fri Feb 18 12:58:32 2022 from 76.176.117.51
```

```
[username@login02 ~]$
```

```
[username@login02 ~]$ srun --partition=gpu-debug --pty --account=use300 --ntasks-per-node=10 --nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash
```

```
srun: job 9794018 queued and waiting for resources
```

```
srun: job 9794018 has been allocated resources
```

```
[mthomas@exp-14-57 ~]$
```

```
[mthomas@exp-14-57 ~]$ nvidia-smi
```

```
Fri Feb 18 13:04:19 2022
```

```
+-----+-----+-----+
| NVIDIA-SMI 460.32.03     Driver Version: 460.32.03     CUDA Version: 11.2 |
+-----+-----+-----+
| GPU  Name      Persistence-M | Bus-Id      Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp     Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|                               |             |            MIG M. |
+-----+-----+-----+
| 0  Tesla V100-SXM2... On   | 00000000:86:00.0 Off  |           0 |
| N/A   34C     P0    41W / 300W |        0MiB / 32510MiB | 0%      Default |
|                               |                           N/A |
+-----+-----+-----+
```

```
+-----+
| Processes:
| GPU  GI  CI          PID  Type  Process name          GPU Memory |
| ID   ID              ID   ID   |                    Usage |
+-----+
| No running processes found |
+-----+
```

```
[username@login02 ~]$ exit
```

Request an interactive node  
for 30 minutes

Verify you are on a GPU node

Exit when tasks are done

# Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

# Visualizing Data Methods

- Viewing Data: unix file ops (grep, awk, cat),
  - Note: gnuplot, NetCDF – apps don't work well on Expanse: can be very slow
- Programming & Visualization Platforms:
  - Matlab, R, Jupyter Notebooks, Paraview
  - Many require X11 servers on the local host; which is slow and unstable (especially MacOC)
- Jupyter notebooks have many packages that can be used to visualize data.
- Gateways & Portals:
  - simplify access to interactive apps

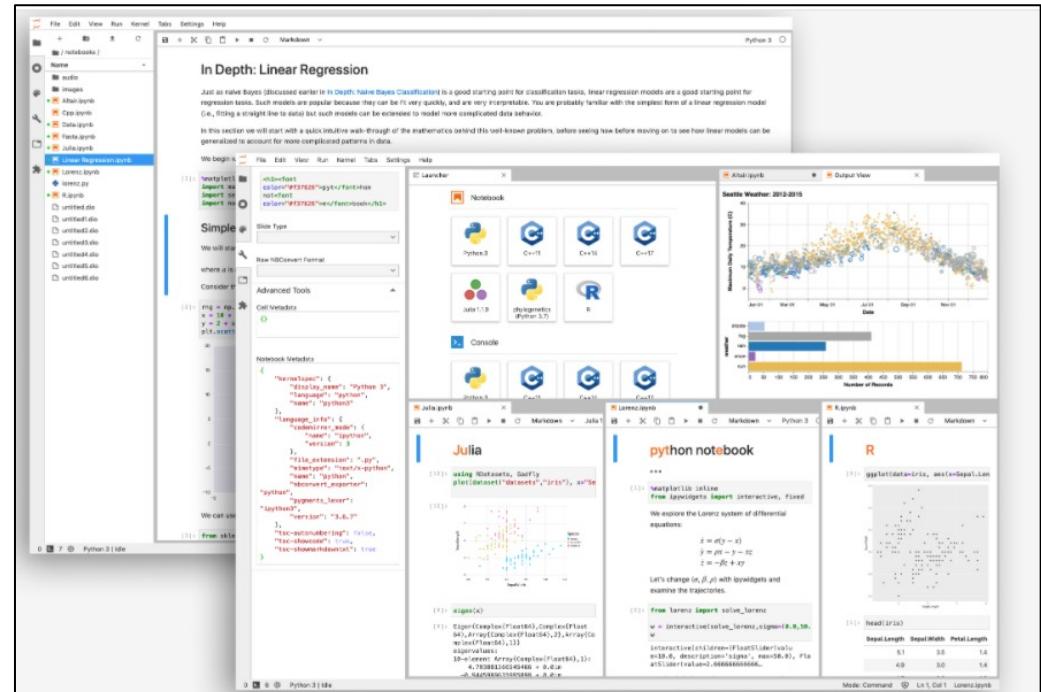
# Jupyter Notebooks

## What is Jupyter?

*Jupyter is a free, open-source, **interactive** web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document. (J. Perkel, <https://www.nature.com/articles/d41586-018-07196-1>)*

## Common Jupyter Services:

- Jupyter Notebooks (single user)
- JupyterLab: advanced version of notebook
- JupyterHub: multiuser Jupyter service



# Accessing and Running Secure Notebooks on SDSC HPC Systems

- Install notebook application:
  - Locally: install Anaconda on your laptop
  - Remotely:
    - Install Anaconda/conda on the remote machine (**default is HTTP**) – not recommended
- Running remotely:
  - Connect over HTTP (default, insecure)
  - Connect over HTTP + SSH tunneling (secure, but inconvenient)
  - **Connect over HTTPS + using the *Satellite Reverse Proxy Service (SRPS)* and *galyleo client* (secure, convenient)**
- You can launch Jupyter services on SDSC:
  - Launch securely (HTTPS) using SRPS/*galyleo* -- **recommended**
  - CPU and GPUs
  - Interactive nodes: command line *or* Slurm batch script
- **Treat the Notebook URL like a Password!**

# SDSC Satellite Reverse Proxy Service

- SRPS: prototype system that allows users to launch secure standard Jupyter Notebooks on any Expanse compute node using a reverse proxy server.
  - Notebooks will be hosted on the internal cluster network as an HTTP service using standard Jupyter commands.
  - Service available to the user outside of the cluster firewall over HTTPS connection between the external users web browser and the reverse proxy server.
- Goal: minimize software changes for users, improve security of user notebooks running on SDSC systems.
- SRPS can run on any HPC system capable of supporting Apache on internal network.

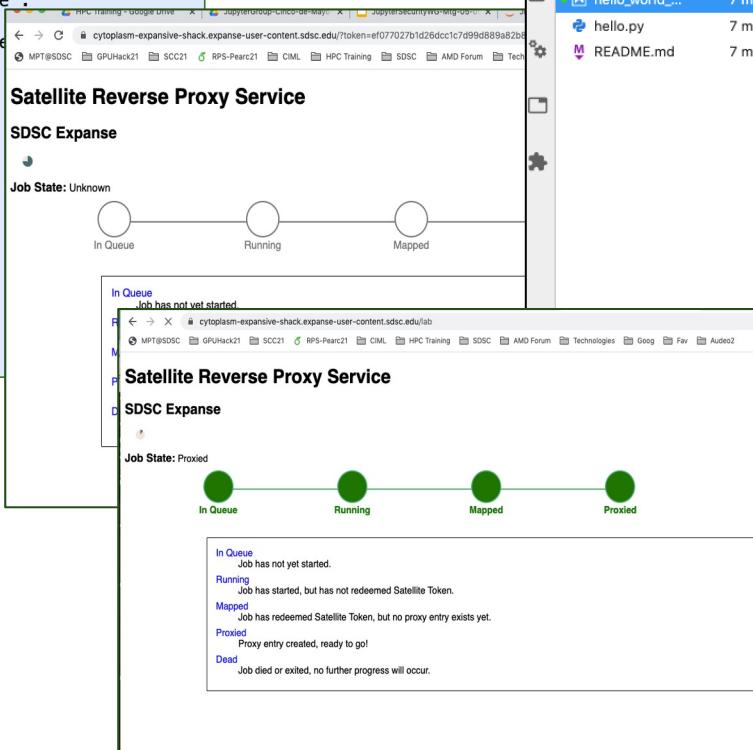
# galyleo

- 2nd generation shell utility developed to orchestrate a user's interaction with both Satellite and Slurm to start a Jupyter session within a batch job.
- Developed while reviewing start-jupyter (prototype client) codebase to sort out how best to support Expanse (OOD) Portal and HPC User Services Group long-term; integrated into an existing SSH tunneling orchestration utility to use Satellite proxy service instead
- **Key features** in design:
  - HTTPS URL
  - Supports containers (Singularity on Expanse) → **GPU environment**
  - No need to install conda environment or update packages
  - Increases flexibility for users to configure software environment; but also try to makes it simpler for them to do this themselves
  - Batch job script is generated completely on-the-fly.
  - Command-line argument driven.
  - Quiet mode for OOD portal

<https://github.com/mkandes/galyleo>

# Satellite-Galileo System

```
[username@login01 ~]$ export  
PATH="/cm/shared/apps/sdsc/galileo:$PATH"  
  
[username@login01 ~]$ galileo.sh --help  
USAGE: galileo.sh launch [command-line  
option] {value}  
command-line option      : value  
-A | --account          :  
-R | --reservation       :  
-P | --partition         :  
-Q | --gpus              :  
-N | --nodes              :  
-n | --ntasks_per_node   :  
-C | --cpus_per_task     :  
-M | --memory_per_node   :  
-m | --memory_per_cpu    :  
-G | --gres               :  
-t | --time_limit         :  
-j | --jupyter             :  
-d | --notebook_dir        :  
-r | --reverse_proxy       :  
-D | --dns_domain         :  
-s | --sif                 :  
-B | --bind                :  
-nv | --nv                  :  
-e | --env_modules         :  
--conda-env              :  
-Q | --quiet               :
```



carload-spray-koala.expanse-user-content.sdsc.edu/lab

MPT@SDSC SDSC HPCTr SRPS CML AMDUF ICICLE MPT Fav COVID-19 Vaccine... Other Books

File Edit View Run Kernel Tabs Settings Help

numpy\_intro.ipynb hello\_world\_gpu.ipynb hello\_world\_cpu.ipynb boring\_python\_chap.ipynb

Hello World

FileName: hello\_world\_cpu.ipynb

CPU Version

No package dependencies

```
[ 8]: print('Hello world!!!!')
Hello world!!!!
```

```
[ 9]: # Import hello module
import hello

# Define a local function
def world2(name):
    print(name)
```

```
[10]: # Call function
world2("mary")
mary
```

```
[11]: hello.greeting("good times")
Greetings, good times
```

```
[12]: hello.world("World.")
Hello, World.
```

# Satellite Client: galyleo

Key features in design:

- User calls galyleo.sh launch script, which requests token from Satellite, passes token to batch job script and submits the job to Slurm; token redeemed from batch job once it runs
- Increase flexibility for users to configure software environment; but also try to make it simpler for them to do themselves
- Batch job script is generated completely on-the-fly.
- Command-line argument driven.
- Quiet mode for OOD portal

```
[username@login01 ~] export PATH="/cm/shared/apps/sdsc/galyleo:${PATH}"  
[username@login01 ~]$ galyleo.sh --help  
USAGE: galyleo.sh launch [command-line option] {value}  
command-line option      : value  
  -A | --account          :  
  -R | --reservation       :  
  -p | --partition         :  
  -q | --qos               :  
  -N | --nodes              :  
  -n | --ntasks-per-node   :  
  -c | --cpus-per-task     :  
  -M | --memory-per-node   : GB  
  -m | --memory-per-cpu    : GB  
  -G | --gpus               :  
  --gres                  :  
  -t | --time-limit         :  
  -j | --jupyter             :  
  -d | --notebook-dir        :  
  -r | --reverse-proxy       :  
  -D | --dns-domain         :  
  -s | --sif                 :  
  -B | --bind                :  
  --nv                      :  
  -e | --env-modules         :  
  --conda-env               :  
  -Q | --quiet                : 1
```

<https://github.com/mkandes/galyleo>

# Launching CPU notebooks using galileo

**Step 1:**  
Login and setup user environment

**Step 2:**  
Run command to launch secure notebook

**Step 3:**  
Copy URL; paste into browser on local system

**Step 4:**  
Monitor Slurm queue;  
wait for job to start

```
[username@login01 ~]$ which galileo.sh  
/usr/bin/which: no galileo.sh in (/cm/shared/apps/spack/cpu/opt/spack/linux-  
centos8-zen/gcc-8.3.1  
[SNIP]  
home/username/.local/bin:/home/username/bin)  
[username@login01 ~]$ export PATH="/cm/shared/apps/sdsc/galileo:${PATH}"  
[username@login01 ~]$ which galileo  
/cm/shared/apps/sdsc/galileo/galileo  
[username@login01 ~]$ galileo launch --account abc123 --partition shared --cpus 2  
--memory 4 --time-limit 00:30:00 --env-modules cpu/0.17.3b,anaconda3/2021.05  
[snip]  
Submitted Jupyter launch script to Slurm. Your SLURM_JOB_ID is 9773665  
[snip]  
Your Jupyter notebook session will begin once compute resources are allocated to  
your job by the scheduler.  
https://carload-spray-koala.expanse-user-content.sdsc.edu/lab  
  
[username@login01 ~]$ squeue -u username  
JOBID PARTITION NAME USER ST TIME NODES  
NODELIST(REASON)  
9773665 gpu-debug galileo- username PD 0:00 1 (None)  
[username@login01 ~]$ squeue -u username  
JOBID PARTITION NAME USER ST TIME NODES  
NODELIST(REASON)  
9773665 gpu-debug galileo- username R 0:20 1 exp-7-59
```

# Satellite Server Status/Pending Page

- Load notebook URL in browser; wait for it to launch
- Monitor pending page
- Run the “squeue” command on the HPC system to check job status
- If the job queue is busy, it may take a while to launch the notebook
- **Treat Jupyter Notebook URL as a password!**

### Satellite Reverse Proxy Service

SDSC Expanse

Job State: Mapped

In Queue  
Job has not yet started.

Running  
Job has started, but has not redeemed Satellite Token.

Mapped  
Job has redeemed Satellite Token, but no proxy entry exists yet.

Proxied  
Proxy entry created, ready to go!

Dead  
Job died or exited, no further progress will occur.

```
[mthomas@login02 ~]$ [mthomas@login02 ~]$ squeue -u mthomas
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST(REASON)
      9774239    shared galileo-  mthomas R      3:49      1 exp-1-13
      9774274    debug galileo-  mthomas R      0:12      1 exp-9-55
[mthomas@login02 ~]$
```

# Launching GPU notebooks using galyleo

- GPU Notebooks run better when using containers. SDSC maintains several containers on Expanse
- See: /cm/shared/apps/containers/singularity

Step 1:  
Login and setup user environment

```
[username@login01 ~]$ which galyleo.sh  
/usr/bin/which: no galyleo.sh in (/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-  
zen/gcc-8.3.1  
[SNIP]  
home/username/.local/bin:/home/username/bin)  
[username@login01 ~]$ export PATH="/cm/shared/apps/sdsc/galyleo:${PATH}"  
[username@login01 ~]$ which galyleo  
/cm/shared/apps/sdsc/galyleo/galyleo  
[username@login01 ~]$ galyleo launch --account use300 --partition gpu-debug --cpus 1  
--memory 93 --gpus 1 --time-limit 00:5:00 --env-modules singularitypro --sif  
/cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif --bind  
/expanse,/scratch --nv  
[snip]
```

Step 2:  
Run command to  
launch secure  
notebook

```
Submitted Jupyter launch script to Slurm. Your SLURM_JOB_ID is 9773912  
[snip]  
Your Jupyter notebook session will begin once compute resources are allocated to  
your job by the scheduler.
```

Step 3:  
Copy URL; paste into  
browser on local  
system

```
https://grief-fantastic-given.expanse-user-content.sdsc.edu?token=5097acb6f32ab82dd51b4524c267d2fd
```

Step 4:  
Monitor Slurm queue;  
wait for job to start

	JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REAON)
	9773912	gpu-debug	galyleo-	username	PD	0:00	1	(None)
	9773912	gpu-debug	galyleo-	username	R	0:20	1	exp-7-59

# Verify notebook launched on GPU device

MPT@SDSC

File Edit View Run Kernel Tabs Settings Help

numpy\_intro.ipynb X hello\_world\_gpu.ipynb

+

Filter files by name

/ notebook-examples / Hello\_World /

Name	Last Modified
hello_wor...	7 months ago
hello_wor...	28 minutes ago
hello.py	7 months ago
README.md	7 months ago

[9]: # Check to see if system is GPU:  
!nvidia-smi

Fri Feb 18 00:15:50 2022

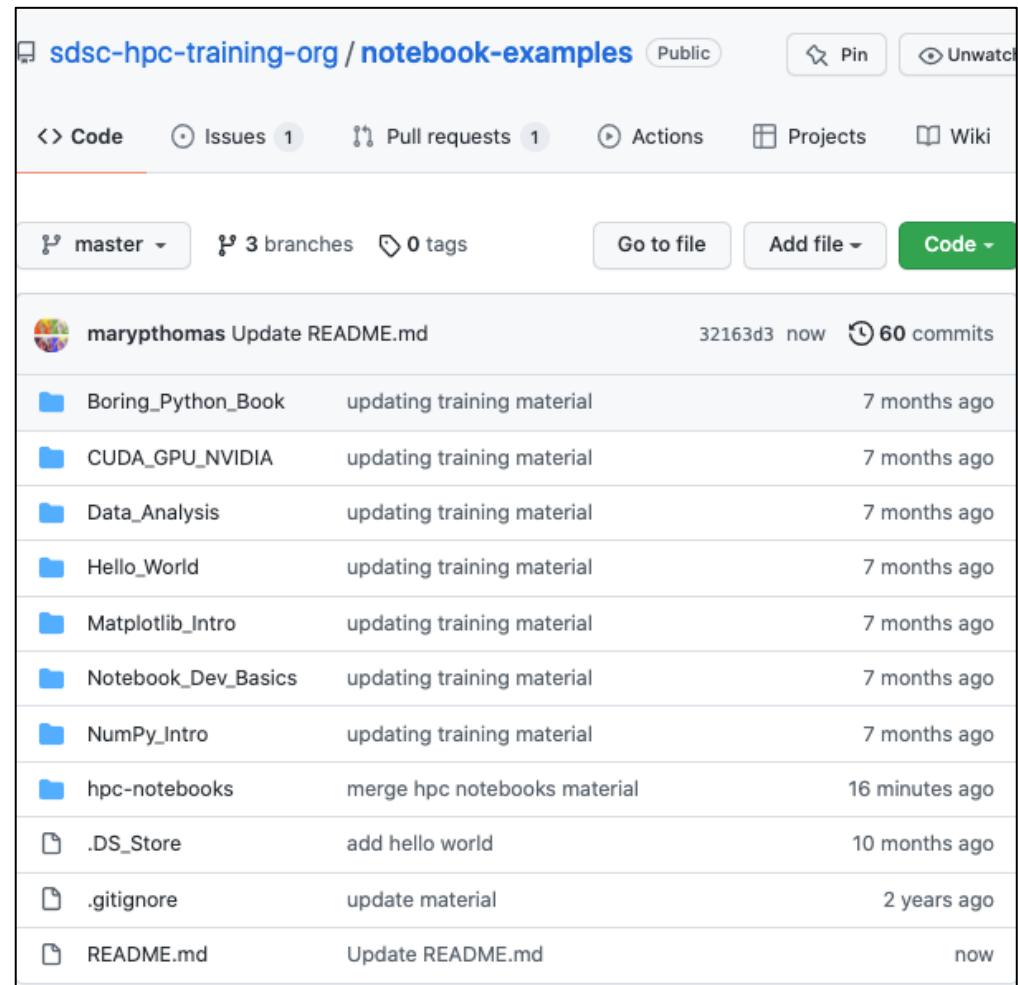
NVIDIA-SMI 460.32.03		Driver Version: 460.32.03	CUDA Version: 11.2			
GPU Name	Persistence-M	Bus Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.
0	Tesla V100-SXM2...	On	00000000:18:00.0	Off	0%	Default
N/A	37C	P0	68W / 300W	0MiB / 32510MiB		N/A

[10]: # if you see: /bin/bash: nvidia-smi: command not found  
# the system is not GPU

# Notebook Examples

<https://github.com/sdsc-hpc-training-org/notebook-examples>

- Collection of tested, working notebooks tested on Expanse and other SDSC HPC systems
- Includes range of materials from “hello world” to Spark ML notebooks.
- Note: collection changes often, based on testing and contributions



The screenshot shows a GitHub repository page for 'sdsc-hpc-training-org / notebook-examples'. The repository is public, as indicated by the 'Public' button. The 'Code' tab is selected. The master branch is current, with 3 branches and 0 tags. There are 60 commits from 'marypthomas' updating README.md. Other commits include 'Boring\_Python\_Book', 'CUDA\_GPU\_NVIDIA', 'Data\_Analysis', 'Hello\_World', 'Matplotlib\_Intro', 'Notebook\_Dev\_Basics', 'NumPy\_Intro', and 'hpc-notebooks'. A merge commit for 'hpc-notebooks' was made 16 minutes ago. A file named '.DS\_Store' was added 10 months ago, and '.gitignore' was updated 2 years ago. The 'README.md' file was last updated 'now'.

File	Description	Time
marypthomas Update README.md	updating training material	32163d3 now 60 commits
Boring_Python_Book	updating training material	7 months ago
CUDA_GPU_NVIDIA	updating training material	7 months ago
Data_Analysis	updating training material	7 months ago
Hello_World	updating training material	7 months ago
Matplotlib_Intro	updating training material	7 months ago
Notebook_Dev_Basics	updating training material	7 months ago
NumPy_Intro	updating training material	7 months ago
hpc-notebooks	merge hpc notebooks material	16 minutes ago
.DS_Store	add hello world	10 months ago
.gitignore	update material	2 years ago
README.md	Update README.md	now

# Expanse User Portal

**SIMPLIFY!  
EXPANSE PORTAL**

The portal provides an integrated, and easy to use interface to access Expanse HPC resources.

With the portal, researchers can manage files (create, edit, move, upload, and download), view job templates for various applications, submit and monitor jobs, run interactive applications, and connect via SSH. The portal has no end-user installation requirements other than access to a modern up-to-date web browser.

**Pinned Apps A featured subset of all available apps**

- Action Jobs
- Home Directory
- Job Composer
- Expanse Shell Access
- MATLAB
- RSTUDIO
- Allocation and Usage Information
- Jupyter

**Interactive Services**

This app will launch a MATLAB GUI on Expanse. You will be able to interact with the MATLAB GUI through a VNC session. To use this app, you would need to be added to matlab-groups on Expanse. Please email [help@expanse.org](mailto:help@expanse.org) to be added to matlab-groups.

**Job Script Editing and Submission**

**Active Job monitoring and Management**

**OOD 2.0 Features**

- <https://portal.expanse.sdsc.edu>; authenticate using ACCESS credentials
- Securely hosts batch job submission & monitoring, and interactive applications
- Simplifies launching supported interactive applications → manages software dependencies

# Expanse Portal: File Management

The image shows the Expanse Portal interface. At the top, there is a navigation bar with buttons for 'Open in Terminal', 'New File', 'New Directory', 'Upload', 'Download', and 'Copy/Mov'. A red circle highlights the 'Open in Terminal' button. Below the navigation bar is a breadcrumb trail: '/ home / mthomas / expanse /' and a 'Change directory' button. There are also checkboxes for 'Show Owner/Mode' and 'Show Dotfiles'.

The main area displays a list of files and directories:

Type	Name	Size	Modified at
Folder	classes	-	4/15/2022 3:37:19 PM
Folder	comet-files	-	2/16/2022 6:41:00 PM
Folder	conda-install-tmp	-	10/29/2020 2:27:20 AM
Folder	dev	-	2/16/2022 8:39:49 PM
Folder	galileo-examples	-	7/27/2022 2:27:08 AM
Folder	galileo-repo	-	2/16/2022 6:36:21 PM
Folder	gnuplot-ex	-	8/1/2022 8:50:45 PM
Folder	gpuhack22	-	5/11/2022 3:40:13 PM
Folder	hpctr-examples	-	6/27/2022 1:31:14 PM
Folder	hptrain	-	2/16/2022 8:47:23 PM

A red arrow points from the 'Open in Terminal' button in the navigation bar to a terminal window on the right side of the screen. The terminal window shows a session on host 'login.expanse.sdsc.edu' with the initial directory '/home/mthomas/expanse'. The session lists numerous files and their details, such as owner, size, and modification date. The terminal prompt is '[mthomas@login01 ~]\$'.

```
Host: login.expanse.sdsc.edu Initial directory: /home/mthomas/expanse
-rw----- 1 mthomas use300 17803 Aug 1 21:33 .viminfo
-rw-r--r-- 1 mthomas use300 36 Jan 27 2022 .vimrc
drwxr----- 2 mthomas use300 4 Aug 1 23:11 .vnc
-rw-r--r-- 1 mthomas use300 173 Oct 7 2020 .wget-hsts
-rw-r--r-- 1 mthomas use300 124 Oct 7 2020 README.txt
drwxr-xr-x 4 mthomas use300 4 Apr 15 15:37 classes
drwxr-xr-x 2 mthomas use300 6 Feb 16 18:41 comet-files
-rw-r--r-- 1 mthomas use300 116 Mar 4 2021 conda-activate.txt
drwxr-xr-x 2 mthomas use300 4 Oct 29 2020 conda-install-tmp
drwxr-xr-x 8 mthomas use300 8 Feb 16 20:39 dev
-rw-r--r-- 1 mthomas use300 12266 Nov 8 2021 ex.cl.cmds
drwxr-xr-x 3 mthomas use300 8 Jul 27 02:27 galileo-examples
drwxr-xr-x 5 mthomas use300 9 Feb 16 18:36 galileo-repo
drwxr-xr-x 2 mthomas use300 10 Aug 1 20:50 gnuplot-ex
drwxr-xr-x 3 mthomas use300 3 May 11 15:40 gpuhack22
drwxr-xr-x 13 mthomas use300 16 Jun 27 13:31 hpctr-examples
drwxr-xr-x 5 mthomas use300 6 Feb 16 20:47 hpctrain
drwxr-xr-x 2 mthomas use300 8 Jul 27 02:33 interactive.ex
drwxr-xr-x 2 mthomas use300 3 Aug 1 21:33 matlab-ex
drwxr-xr-x 24 mthomas use300 27 Jul 27 2021 miniconda3
drwxr-xr-x 3 mthomas use300 5 Jul 27 2021 ml-dev-mary
-rwx----- 1 mthomas use300 235 Jun 1 2021 modules.cpu.txt
-rwx----- 1 mthomas use300 84 Feb 8 2021 modules.gpu.txt
-rw-r--r-- 1 mthomas use300 6178 Mar 4 2021 modules.marty.ex.txt
drwxr-xr-x 3 mthomas use300 4 Jul 18 16:09 nn-pde-TEST
drwxr-xr-x 10 mthomas use300 13 Jul 28 2021 notebook-examples
drwxr-xr-x 22 mthomas use300 26 Jul 20 2021 notebook-examples-dev
-rwxr-xr-x 1 mthomas use300 234277552 Aug 1 20:04 ocean_his.nc.gz
drwxr-xr-x 9 mthomas use300 19 Jul 28 2021 reverse-proxy
drwxr-xr-x 2 mthomas use300 7 Feb 16 18:44 scc21
drwxr-xr-x 2 mthomas use300 5 Feb 16 20:37 tensorflow
drwxr-xr-x 2 mthomas use300 3 Jul 14 2021 tools
[mthomas@login01 ~]$
```

# Expanse Portal: Running Matlab

The image shows a multi-step process for launching a Matlab session on the Expanse Portal.

**Step 1: Interactive Apps Selection**

The user selects "MATLAB" from the "Interactive Apps" list. A tooltip provides instructions: "This app will launch a MATLAB GUI on Expanse. You will be able to interact with the MATLAB GUI through a VNC session. Please email help@xsede.org to be added to matlab-groups." The "Session ID" is listed as "fee7f0fb-48df-46af-814".

**Step 2: Session Configuration**

The user fills out the following configuration fields:

- Partition:** compute
- Reservation:** (empty)
- Number of hours:** 1
- Account:** use300
- Checkboxes:** "I would like to receive an email when the session starts" (checked) and "Working directory" (set to "home")
- Number of cores:** 1
- Memory (GB):** 64

**Step 3: Session Launch**

The user clicks the "Launch" button. A success message appears: "Session was successfully deleted."

**Step 4: Session Overview**

The user views the session details:

- Host:** >\_exp-1-18-expance.sdsc.edu
- Created at:** 2022-08-01 22:48:55 PDT
- Time Remaining:** (not shown)
- Session ID:** 14834793
- Nodes:** 1 node | **Cores:** 128 cores | **Status:** Running

**Step 5: Matlab Session View**

The user launches the Matlab session. The Matlab interface shows:

- Current Folder:** /home/mthomas
- Command Window:** (partial history)

```
>> plot sin(22)
Error using plot
Invalid first data argument.

>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
Warning: MATLAB has disabled some advanced rendering features by switching to software rendering. For more information, click here.
>>
```
- Figure 1:** A plot of the sine function from 0 to 2π.

# Expanse Portal: Launching Notebooks

The screenshot illustrates the Expanse Portal's Jupyter Session feature. It consists of three main components:

- Left Panel (Top):** A "Jupyter Session" configuration form. Fields include:
  - Account: use300
  - Partition (Please choose the gpu, gpu-shared, or gpu-preempt as the partition if using gpus): shared
  - Time limit (min): 30
  - Number of cores: 1
  - Memory required per node (GB): 2
  - GPUs (optional): 0
- Middle Panel:** A file browser window showing files in the directory /notebook-examples/Hello\_World/. The files listed are:
  - hello\_world.ipynb (selected)
  - hello.py
  - README.md
- Right Panel:** A Jupyter Notebook interface with two tabs open:
  - `numpy_intro.ipynb`: Contains code related to numpy, such as `irperf x`, `lrv svm_l`, `ushbyasid`, `avic v_vm`, and `id overflo`.
  - `hello_world_gpu.ipynb`: Contains code related to GPU usage, including a command to check for NVIDIA-SMI availability: `[9]: # Check to see if system is GPU:  
!nvidia-smi`. The output indicates that NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver.

# Expanse Portal: Input data example for Jupyter Notebook

**Account:** use300

**Partition:** (Please choose the gpu, gpu-shared, or gpu-preempt as the partition if using gpus): debug

**Time limit (min):** 30

**Number of cores:** 1

**Memory required per node (GB):** 2

**GPUs (optional):** 0

**Singularity Image File Location:** (Use your own or to include from existing container library at /cm/shared/apps/container e.g.,

/cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif)

/cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif

**Environment modules to be loaded** (E.g., to use latest version of system Anaconda3 include cpu,gcc,anaconda3): singularitypro

**Conda Environment** (Enter your own conda environment if any):

**Conda Init** (Provide path to conda initialization scripts):

**Conda Yaml** (Upload a yaml file to build the conda environment at runtime) No file chosen:

**Turn on use of mamba for speeding up conda-yml installs:**

**Enable use of new caching mechanism that will store and reuse conda-yml created environments using conda-pack !????**

**Reservation:**

**QoS:**

**Working directory:** HOME

**Type:** JupyterLab

# Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

# Thank You!

## Q&A

If you have problems, please contact **consult@sdsc.edu**

<https://github.com/sdsc-complecs/interactive-computing/>

# Resources

- GitHub Repo for this presentation:
  - <https://github.com/sdsc-complecs/interactive-computing/>
- SDSC Training Resources
  - [https://www.sdsc.edu/education\\_and\\_training/training](https://www.sdsc.edu/education_and_training/training)
  - <https://github.com/sdsc-hpc-training-org/notebook-examples>
  - <https://github.com/mkandes/galyleo>
- Expanse :
  - Landing page: [expanse.sdsc.edu](https://expanse.sdsc.edu)
  - User Guide: [https://expanse.sdsc.edu/support/user\\_guides/expanse.html](https://expanse.sdsc.edu/support/user_guides/expanse.html)
  - Training: [https://www.sdsc.edu/education\\_and\\_training/training\\_hpc.html](https://www.sdsc.edu/education_and_training/training_hpc.html)
- Problems? Contact [consult@sdsc.edu](mailto:consult@sdsc.edu)