NAIRR Pilot National Artificial Intelligence Research Resource Pilot

NAIRR Pilot National Artificial Intelligence Research Resource Pilot

How to use an HPC system

Mary Thomas

April 2, 2024, Track 1 - Beginner

AI Workshop Denver, CO April 2-3, 2025

Session: How to Use an HPC System (90 minutes)

This session introduces participants to the fundamentals of using High-Performance Computing (HPC) systems for research and AI applications. Key topics include:

1. **Understanding HPC Architecture:**
   - Overview of research computing resource structures, including nodes, processors, and storage.
2. **Discovering Available Software and Tools:**
   - Using tools like the **ACCESS Recommender** and **ACCESS Software Recommender** to identify suitable software for your projects.
3. **Submitting Jobs to an HPC System:**
   - Step-by-step guidance on using a job scheduler for job submission, along with demonstrations of multiple methods.
   - Reusable example scripts for efficient job setup and execution.
4. **Interactive HPC Access:**
   - Using **Open OnDemand** and platforms like **JupyterHub** for streamlined, user-friendly interaction with HPC resources.

Participants will gain practical skills for navigating HPC systems, leveraging software tools, and optimizing workflows for research and AI projects.

Schedule

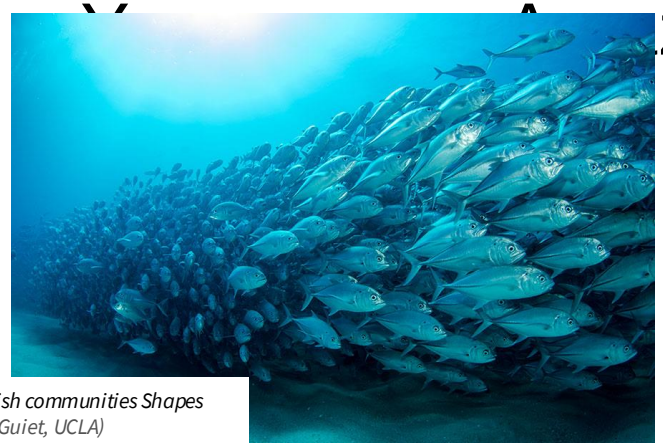
01:15 pm – 02:45 pm: How to use an HPC system, Platte River (90 minutes)

02:45 pm – 03:15 pm: Break, Mezzanine

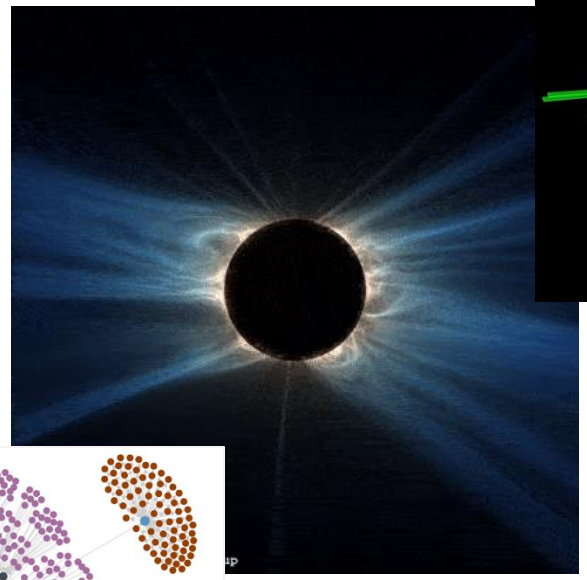
03:15 pm – 04:45 pm: Submitting an AI job, Platte River (90 minutes)

Understanding HPC Architecture

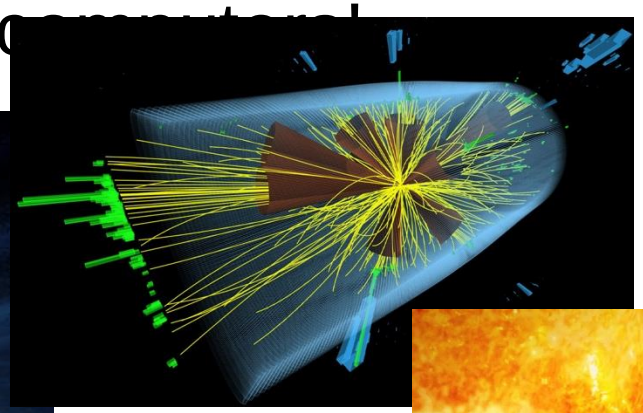
Y[illegible]zing Jobs on Supercomputers



Pelagic fish communities Shapes (Jerome Guiet, UCLA)

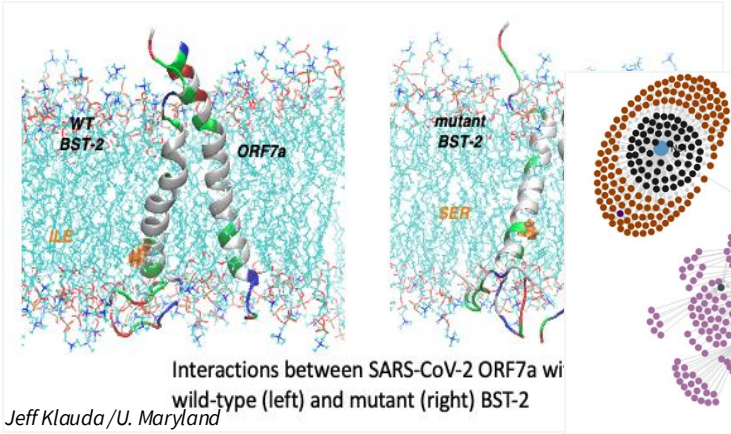


Cooper Downs, Predictive Science Inc

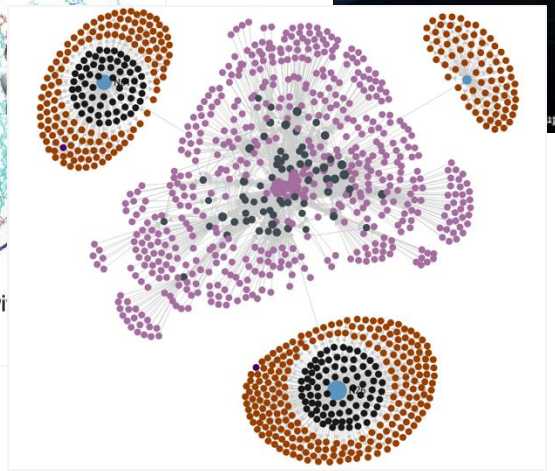


Compact Muon Solenoid (**CMS**)experiment at the LHC, CERN. *Image courtesy of CMS Collaboration; Mc Cauley, Thomas*

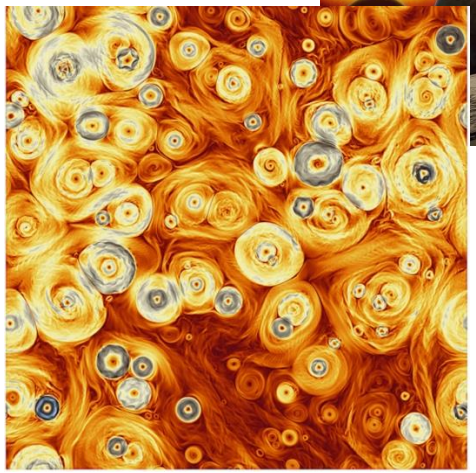
Model of Both Inner and Outer Solar System M. S.Clement (Carnegie Institution for Science)



Jeff Klauda /U. Maryland



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

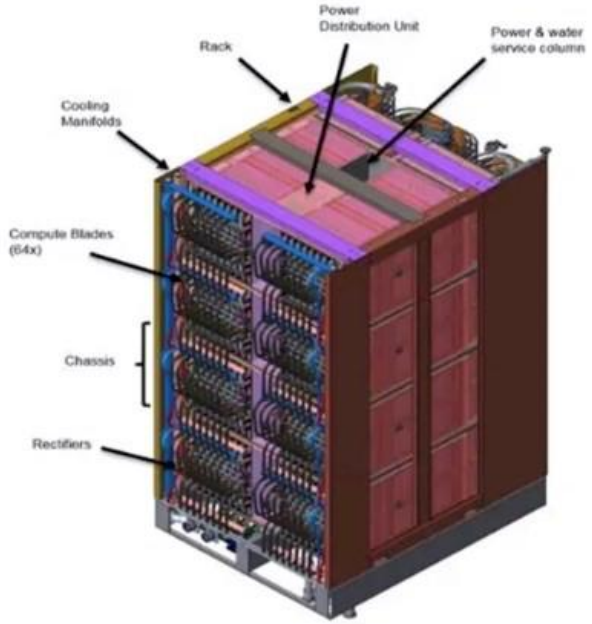


Modeling the sun's corona, Alfred Mallet (UC Berkeley)

HPC System: ORNL FRONTIER

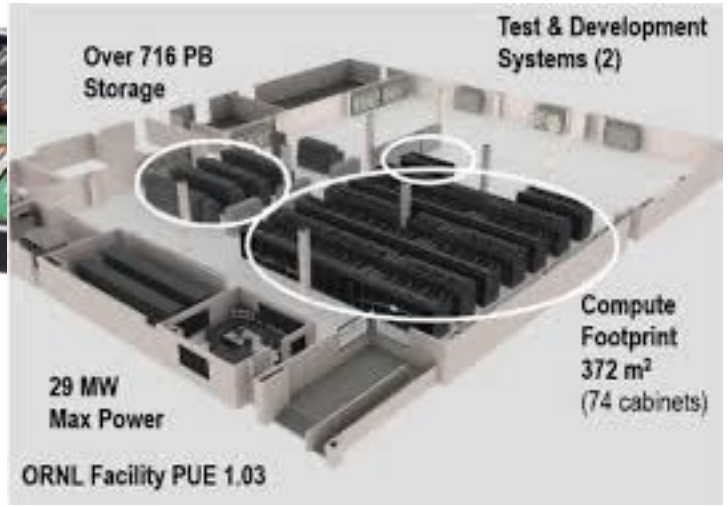
System
- 2 EF Peak DP FLOPS
- 74 compute racks
- 29 MW Power Consumption
- 9,408 nodes
- 9.2 PB memory (4.6 PB HBM, 4.6 PB DDR4)
- Cray Slingshot network with dragonfly topology
- 37 PB Node Local Storage
- 716 PB Center-wide storage
- 4000 ft^2 foot print

Olympus rack
- 128 AMD nodes
- 8,000 lbs
- Supports 400 KW



AMD node
- 1 AMD "Trento" CPU
- 4 AMD MI250X GPUs
- 512 GiB DDR4 memory on CPU
- 512 GiB HBM2e total per node (128 GiB HBM per GPU)
- Coherent memory across the node
- 4 TB NVM
- GPUs & CPU fully connected with AMD Infinity Fabric
- 4 Cassini NICs, 100 GB/s network BW

Compute blade
- 2 AMD nodes

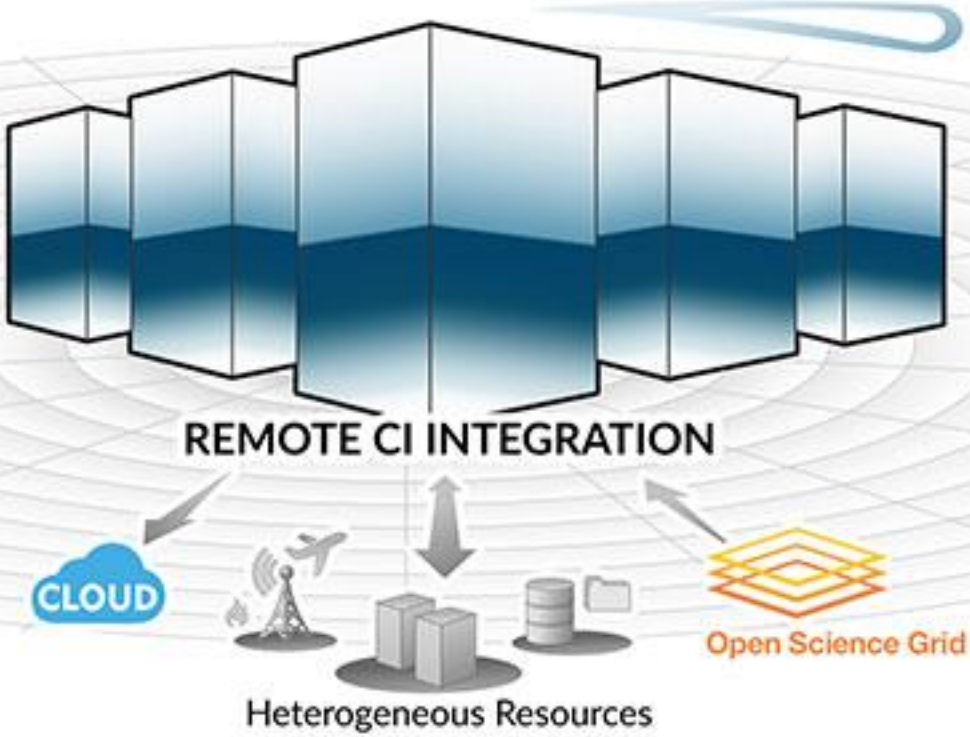


All water cooled, even DIMMS and NICs

HPC Supercomputer: Expanse

HPC RESOURCE
13 Scalable Compute Units
728 Standard Compute Nodes
52 GPU Nodes: 208 GPUs
4 Large Memory Nodes

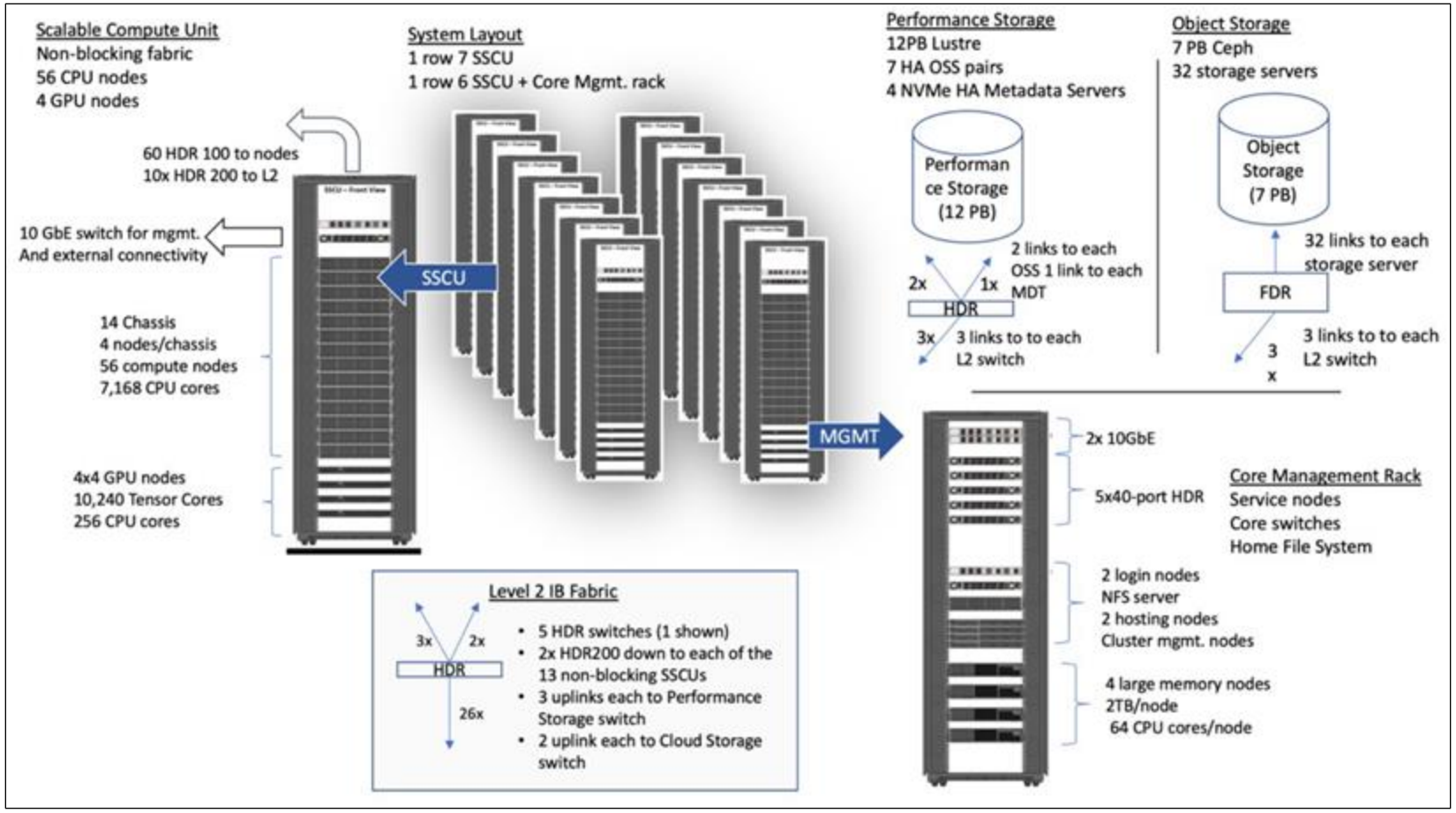
DATA CENTRIC ARCHITECTURE
12PB Perf. Storage: 140GB/s, 200k IOPS
Fast I/O Node-Local NVMe Storage
7PB Ceph Object Storage
High-Performance R&E Networking



LONG-TAIL SCIENCE
Multi-Messenger Astronomy
Genomics
Earth Science
Social Science

INNOVATIVE OPERATIONS
Composable Systems
High-Throughput Computing
Science Gateways
Interactive Computing
Containerized Computing
Cloud Bursting

HPC System Architecture: Expanse @ SDSC



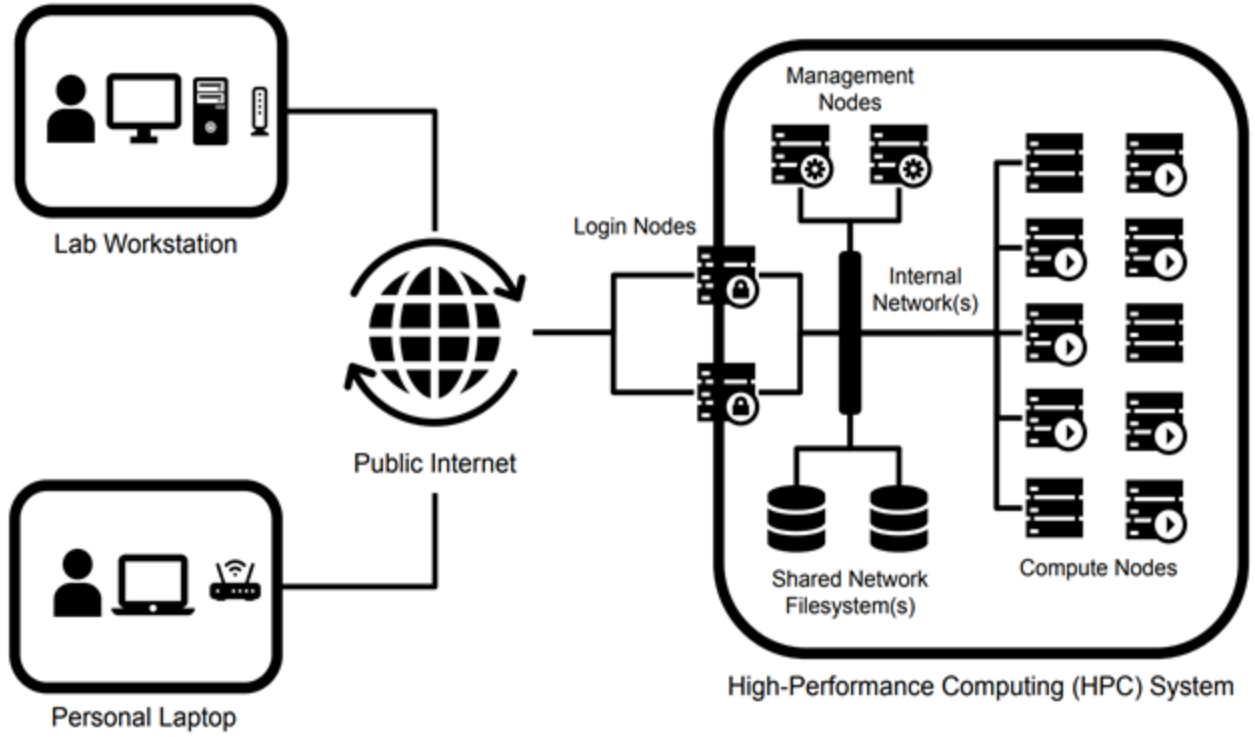
HPC System Architecture: Conceptual Model

Login node(s): Provide remote access to an HPC system; use only for simple tasks such as editing files, limited data transfers to and from the system, and batch job submission

Compute nodes: Run computational workloads: simulations, data analysis and visualization

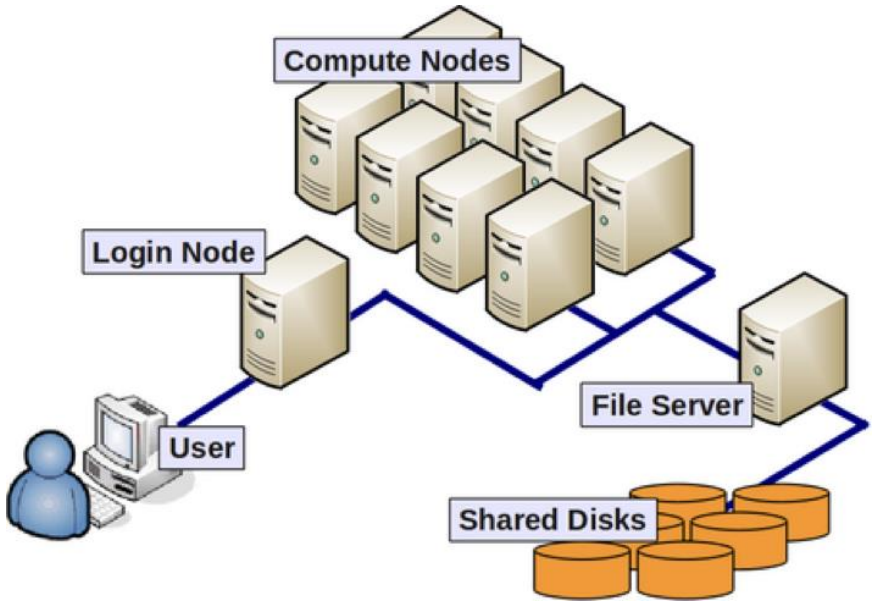
Internal Network(s): Provide high-bandwidth, low-latency communication between computes nodes ; access to shared (parallel) filesystems; system management

Shared Network Filesystem(s): Provide input/output (I/O) access to data storage systems from any compute node



Management node(s): Run core system services such as cluster management software, system monitoring software, *batch job scheduler*, etc

Login and Compute Nodes

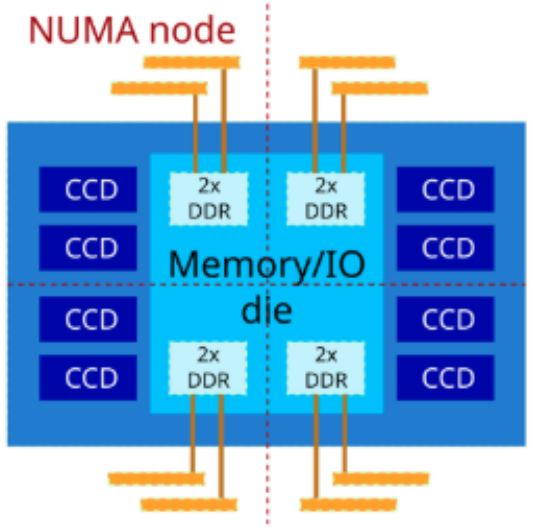


https://www.advancedclustering.com/act_systems/hpc-compute-node/

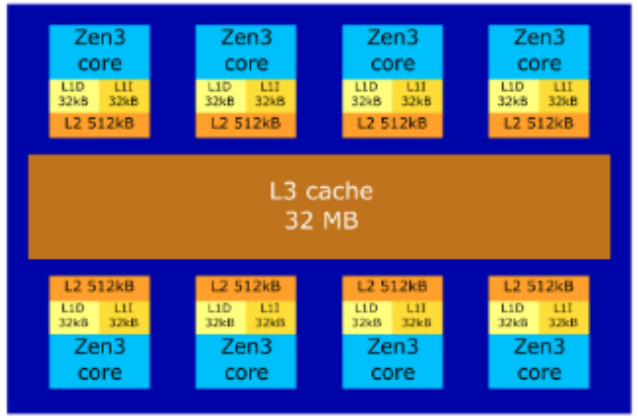
https://hbctraining.github.io/Intro-to-shell-fasrc-flipped/lessons/08_HPC_intro_and_terms.html

Expanse CPU Nodes

- 13+ SDSC Scalable Compute Units (SSCU, racks)
  - 56 CPU nodes
  - 4 GPU nodes (V100+H100)
- 728+ AMD EPYC 7742 AMD Rome Standard Compute Nodes
- 128 cores/node
- 256 GB DRAM/node
- NVME 1TB/node



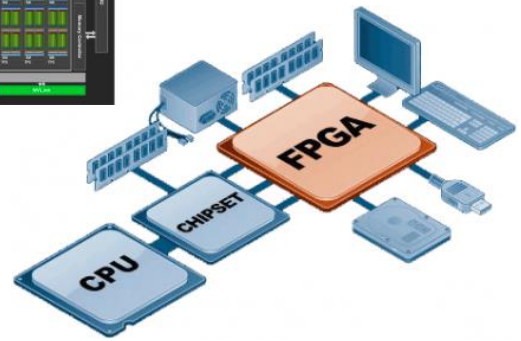
- 8 CCDs or 8 L3 cache regions
- Memory/IO die logically split into 4 NUMA domains with
  - 2 CCDs (16 cores)
  - 2 DDR4 controllers
- Memory/IO die also provides the PCIe links and intersocket links



- Building block: a Core Complex Die (CCD)
- 8 cores
  - Each core has private L1 and L2 caches
  - L3 cache shared
- Instruction set equivalent to Intel Broadwell generation
  - AVX2+FMA, no AVX-512

Compute Accelerators for AI: GPU, FPGAs, QC

- Different accelerators work best for different application spaces
- GPU: Expanse: 52 GPU Nodes; 4 GPUs/node;
  - 384 GB CPU DRAM/node;
  - 80 SMs (Streaming Multiprocessor)
  - NVIDIA Tesla V100 SMX2: 32GB memory/GPU;
  - 21 B transistors; 5120 CUDA cores
  - 640 Tensor cores
- Field Programmable Gate Arrays (FPGA):
  - Three types of modules: I/O blocks; Switch Matrix/ Interconnection Wires; Configurable logic blocks (CLB);
  - Features: customized for application; real-time data processing; parallel processing
  - Emerging AI popularity
- Quantum Computers: in the future
  - https://quantumai.google/



https://www.elprocus.com/fpga-architecture-and-applications/

Expanse File Systems

- Home directories (/home/$USER) – 8 week rolling backup.
  - Login nodes: 100GB limit;
  - Use for source trees, binaries, and small input files.
  - Not good for large scale I/O.
- Lustre filesystems: Good for scalable large block I/O
  - /expanse/lustre/scratch/$USER/temp_project
    - 2.5PB; peak performance: 100GB/s. Good for storing large scale scratch data during a job.
  - /expanse/lustre/projects/ -
    - 2.5PB, peak performance: 100 GB/s. Long term storage.
    - Not good for large # of small files or small block I/O
- Local node “scratch” (SSD) filesystems
  - /scratch local to each native compute node – 210GB on regular compute nodes, 285GB on GPU, large memory nodes, 1.4TB on selected compute nodes.
  - SSD location is good for writing small files and temporary scratch files. Purged at the end of a job.
- Webinar on Data Management and File Systems:
  - https://www.sdsc.edu/event_items/202110_ExpanseWebinar-M.Shantharam.html

Expanse Storage: Global Lustre Filesystem

- Global parallel Lustre filesystems
  - 12 PB Lustre parallel file system
  - 7 PB Ceph Object Store system
  - 140 GB/second performance storage.
- Accessible from all compute and GPU nodes.
  - Lustre Expanse scratch filesystem: limited to 2 millions files/user
    - /expanse/lustre/scratch/$USER/temp_project
  - Lustre NSF projects filesystem: /expanse/lustre/projects/
- Best for scalable large block I/O
- Not good for large # of small files or small block I/O. Please. Don't. Do. This.
  - Contact help@xsede.org for guidance on how to handle this

Expanse Storage: local node/scratch

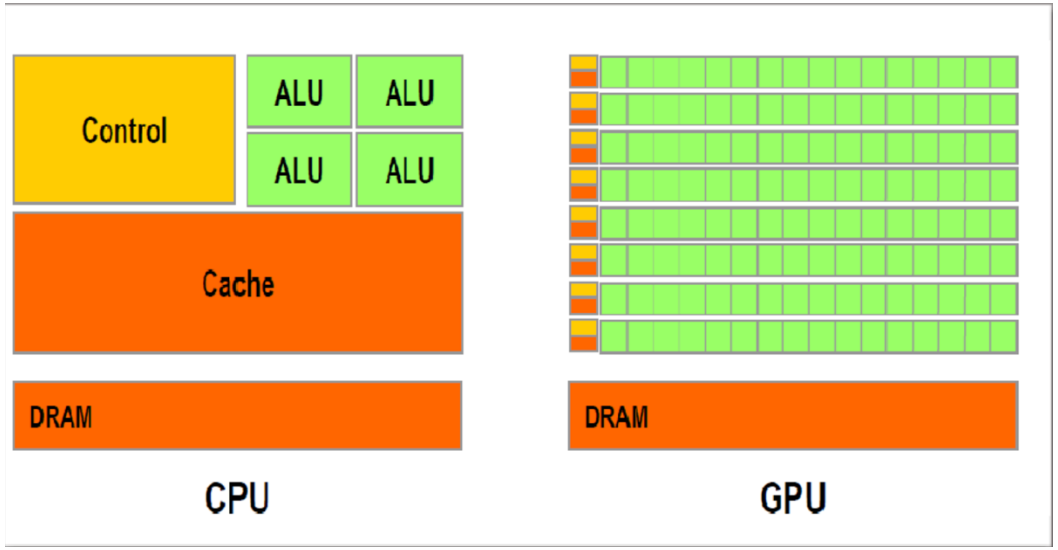
- /scratch is local to each native compute node (SSD)
- Latency to SSDs is several orders of magnitude lower than that for spinning disk (<100 microseconds vs. milliseconds)
- Path changes with Job: /scratch/$USER/job_$SLURM_JOB_ID
- SSD location is good for writing small files and temporary scratch files. Purged at the end of a job.

Partition	Space
compute,shared	1 TB
gpu, gpu-shared	1.6TB
large-shared	3.2 TB

```
[username@exp-7-59 MPI]$ ll /scratch/ username /
total 4
drwx------ 2 username root 4096 Sep 16 01:55 job_5826715
[username @exp-7-59 MPI]$ ll /scratch/ username /job_5826715/
total 0
[username @exp-7-59 MPI]$ !sq
squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5826715 gpu-debug     bash  username   R       0:15      1 exp-7-59
[username@exp-7-59 MPI]$
```

Difference between CPUs and GPUs and why AI likes GPUs

- CPU (compute processor unit) is
  - designed to handle a wide-range of tasks quickly (as measured by CPU clock speed),
  - but are limited in the concurrency of tasks that can be running.
  - Parallelism achieved with multiple cores and/or nodes; data exchanged
- GPU (Graphic Processing Unit) is
  - designed to quickly render high-resolution images and video concurrently;
  - can perform parallel operations on multiple sets of data; but need to get data onto the GPU.
  - Designed for highly parallel computations and were also referred to as throughput processors.
  - GPU-accelerated applications offload these time-consuming routines and functions (also called hotspots
- GPUs dedicate most of their transistors for data processing while CPUs also need to reserve die area for big caches, control units, and so on. CPU processors work on the principle of minimizing latency within each thread while GPUs hide the instruction and memory latencies with computation. Figure 3 shows the difference in computation threads.

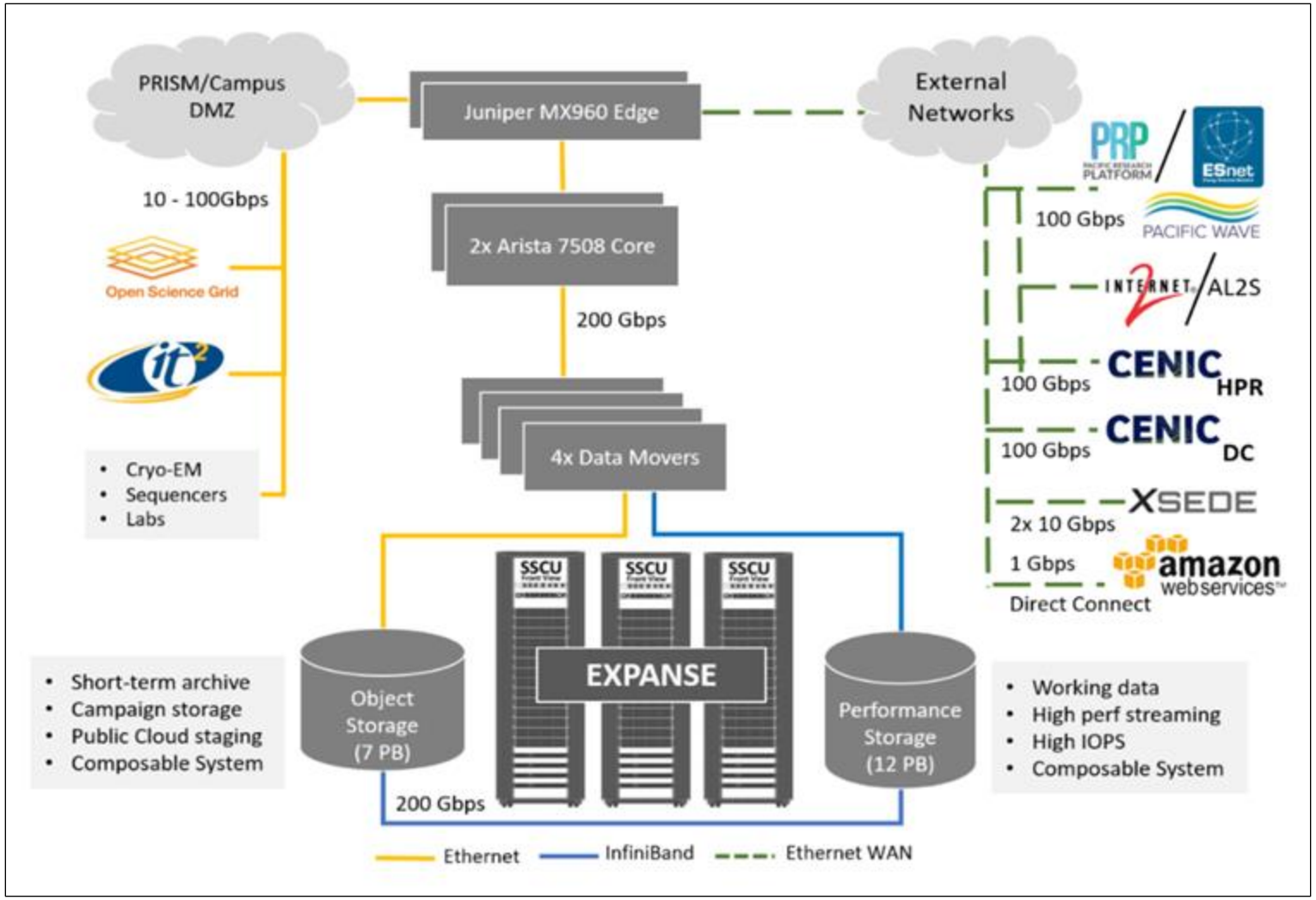


https://www.heavy.ai/technical-glossary/cpu-vs-gpu

https://developer.nvidia.com/blog/cuda-refresher-reviewing-the-origins-of-gpu-computing/

Expanse Network

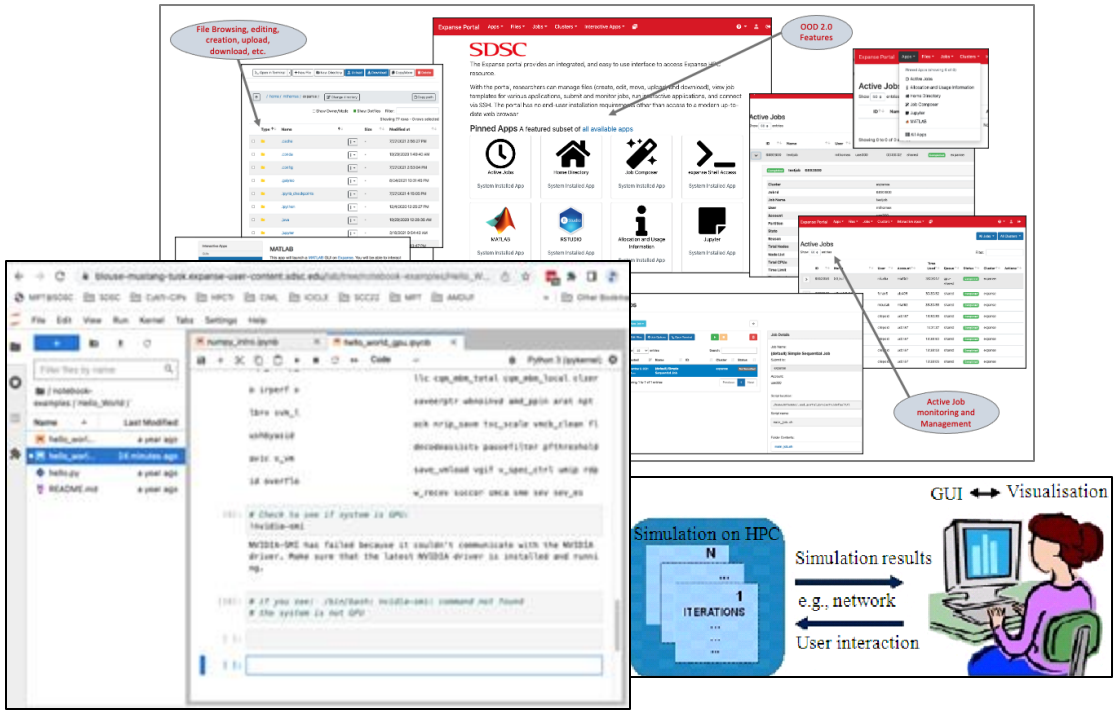
- Infiniband:
  - 200 Gbps internal
  - 100 Gbps external
- 256 GB/node (DRAM)
- Multiple storage devices & types
  - Local node (1TB)
  - Scratch file system
  - Luster (PB)
  - Ceph Object
- High-speed interconnection networks
  - Local, remote
- Running in parallel
- Optimized software

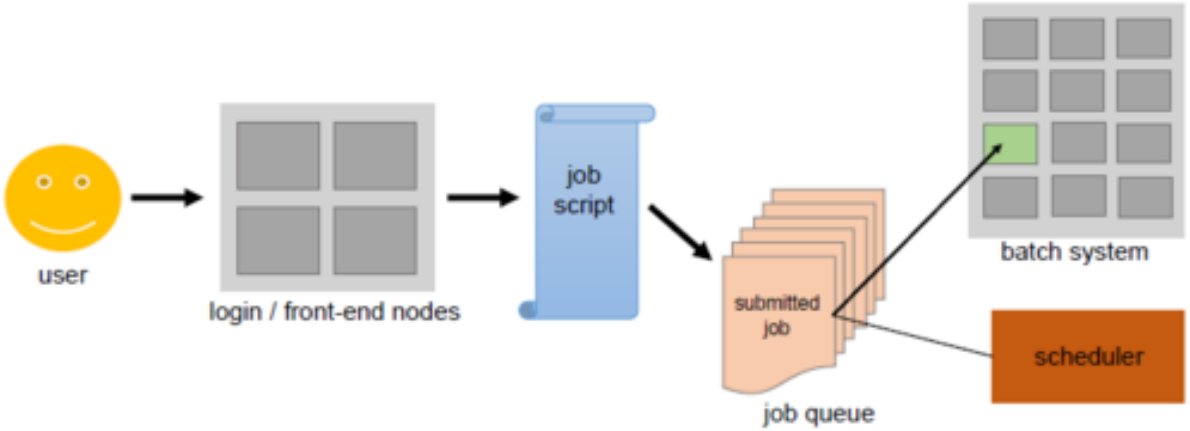


Jobs Can be Run Interactively or as Batch Jobs (Background)

Interactive Jobs:

- User has direct access one or more HPC nodes (or cores).
- Only user's job and user are allowed on the node.
- Parallel jobs and applications allowed to run on node.





Batch Jobs:

- Batch queue system runs on HPC system.
- Users do not run calculations interactively -- instead they submit non-interactive batch jobs to the scheduler.
- Scheduler manages when and where job will run, tracks jobs, and collects results (based on job).

Expanse access via several mechanisms

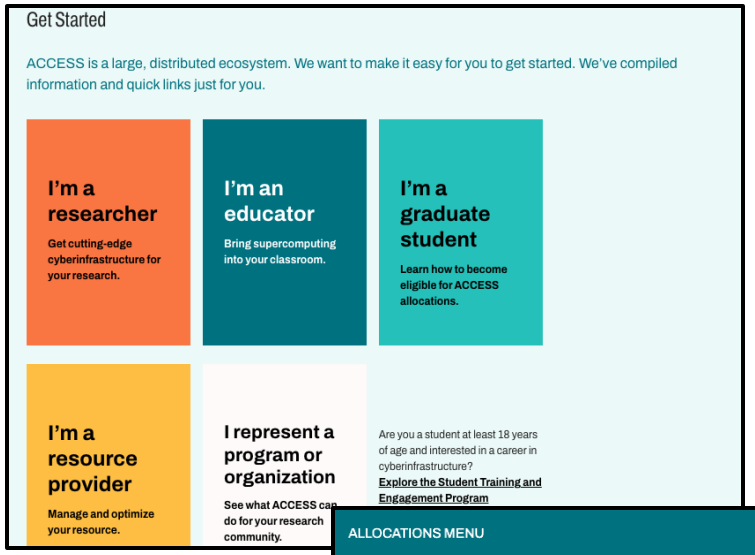
- **Command line** using an ACCESS-wide password or ssh-keys with TOTP: login.expanse.sdsc.edu
- **Web-based** access via the Expanse User Portal: portal.expanse.sdsc.edu
- Customized services
- Note: CPU and GPU resources are allocated separately, but the login nodes are the same.

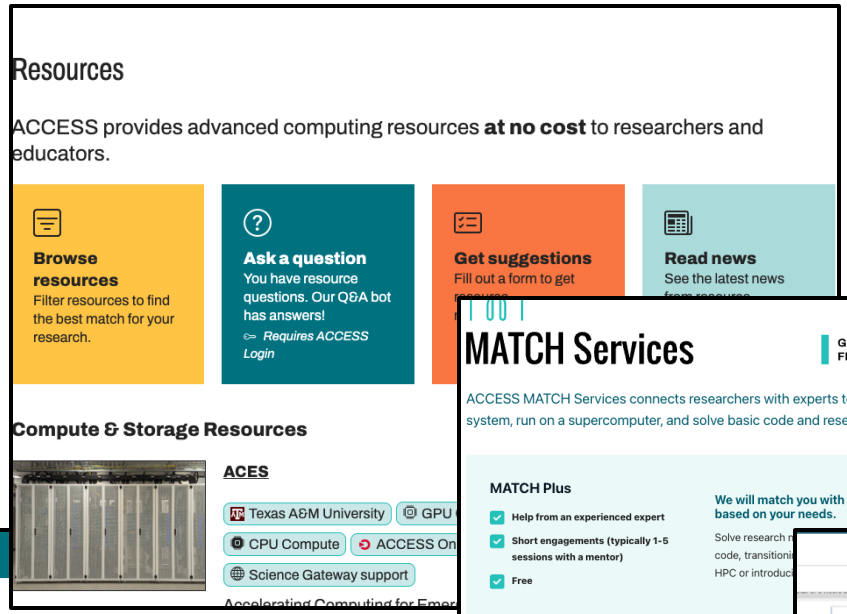
Hands On: Logging onto *Expanse*: login.expanse.sdsc.edu

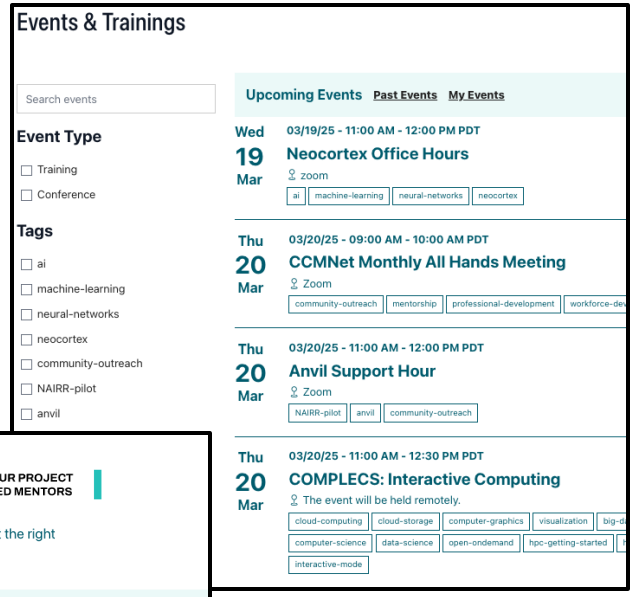
- *Expanse* supports access via:
  - the command line using an ACCESS-wide password or ssh-keys with TOTP
  - Web-based access via the **Expanse User Portal**. While CPU and GPU resources are allocated separately, the login nodes are the same.
- Log onto *Expanse*:
  - ssh <your_username>@login.expanse.sdsc.edu
  - ssh -l <your_username> login.expanse.sdsc.edu
  - To set up 2FA authorization with TOTP, see the *Expanse* instructions here:
    - https://www.sdsc.edu/systems/expanse/user_guide.html#narrow-wysiwyg-2
- Browse around the Expanse system:
  - What is in /cm directories?
  - For basic HPC Linux skills, see: https://hpc-training.sdsc.edu/basic_skills/

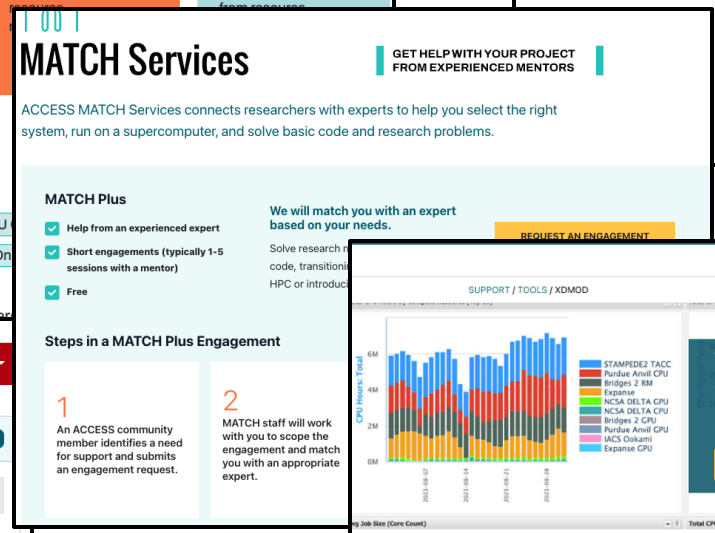
Discovering Available Software and Tools

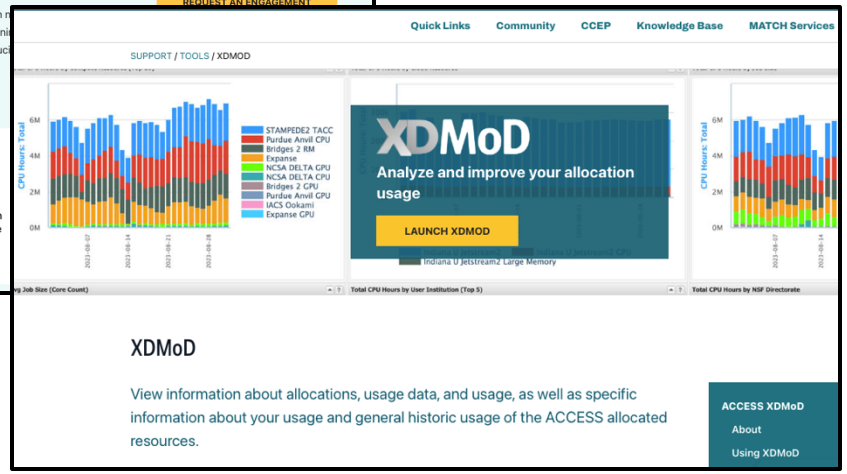
ACCESS Portal: rich suite of services

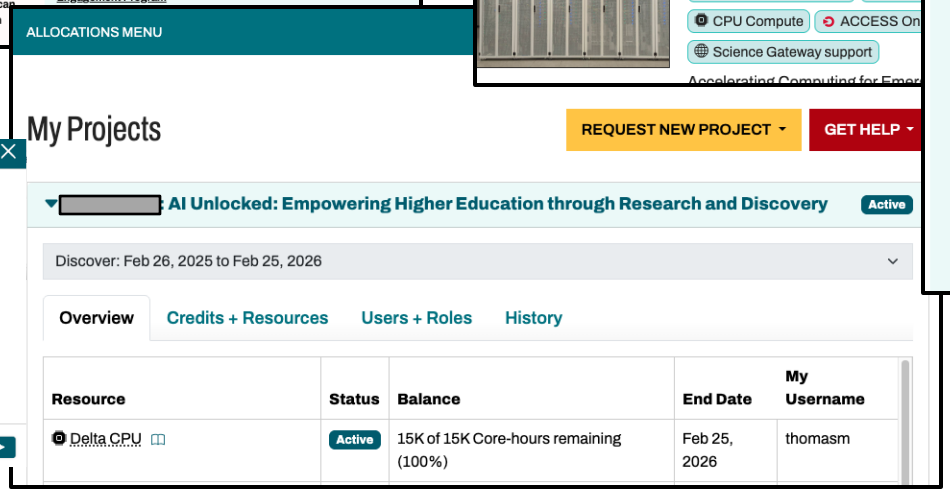


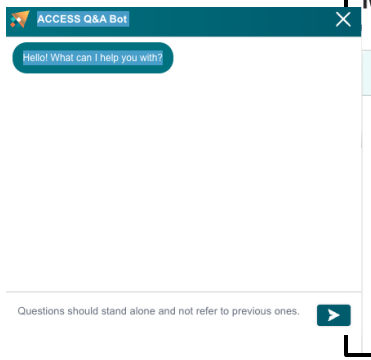












NAIRR Pilot National Artificial Intelligence Research Resource Pilot

Software Documentation Service

Disclaimer: This tool is a work in progress. Fields marked with a sparkles emoji (✨) have largely been generated by AI. Additional information will continue to be added in future updates. Users are encouraged to respond via the 'Report Issue' and 'Provide Feedback' buttons to provide missing information, report errors, or suggest corrections.

Report Issue | Provide Feedback

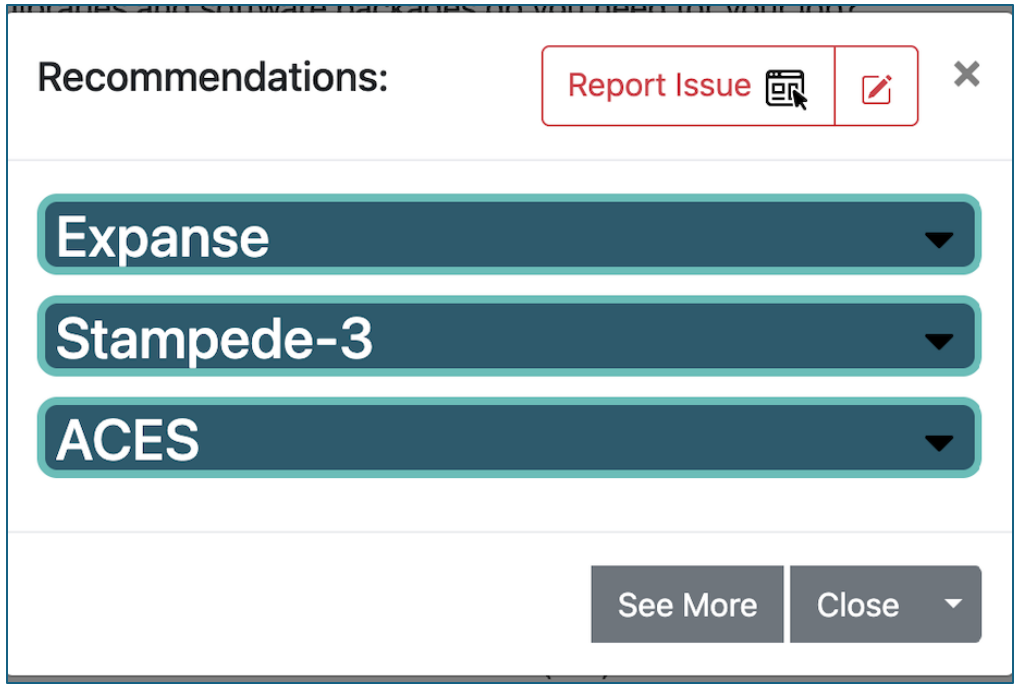
25 entries per page | Show/Hide Columns | Restore All | Show Search Tools | Hover your mouse to the edge of the table to scroll | Search:

Software	RP Name	Software Description	✨AI Description
7z	Ookami	7-Zip is a file archiver with a high compression ratio. Description Source: https://www.7-zip.org/	7-Zip is a file archiver with a high compression ratio. It s several archive formats and can be used to compress decompress files efficiently.
abacas	Anvil	Abacas is a tool for algorithm based automatic contiguation of assembled sequences.	ABACAS is a tool for rapid bacterial genome contig asse takes contigs produced by an assemblrer and scaffolds generate a pseudo-chromosome. It can handle multiple and uses reference genomes for scaffolding.
abaqus	Expanse	Abaqus is a software suite for finite element analysis and computer-aided engineering, primarily used in engineering and design fields for simulating the behavior of structures and materials under various conditions. It offers advanced capabilities for modeling, analysis, and visualization, making it a powerful tool for solving complex engineering challenges.	Abaqus is a software suite used for finite element analy computer-aided engineering simulations for a wide rar industrial applications. It provides powerful simulation ca to analyze the behavior of materials and structures unde conditions.

https://access-sds.ccs.uky.edu:8080/

ACCESS Resource Advisor

- https://access-ara.ccs.uky.edu:8080/
- Tool designed to help researchers (in particular new users) find appropriate infrastructure for their research.
- The term “supercomputer” refers to any substantial computation system or computing resource such as a computing cluster, cloud platform, large server system or high-performance computer.



Environment Modules – discover software on systems

- Expanse uses *Lmod*, a Lua based module system.
  - https://lmod.readthedocs.io/en/latest/010_user.html
- What modules let you do:
  - Dynamic modification of your shell environment
  - Choose between different versions of the same software or different combinations of related codes.
  - Setup custom environments by loading available modules into the shell environment including needed compilers and libraries and the batch scheduler.
- You can only see the software associated with the module, even though there are many other choices
- Use the command "module spider" option to see if a particular package exists and can be loaded, run command
  - module spider <package>
  - module keywords <term>
- For additional details, and to identify module dependencies modules, use the command: module spider <application_name>
- The module paths are different for the CPU and GPU nodes. Users can enable the paths by loading the CPU or GPU modules

Module Command Examples

```
[username @login02 ~]$ module reset
[username@login02 ~]$ module list
Currently Loaded Modules:
  1) cpu/0.15.4   2) slurm/expanse/20.02.3   3) intel/19.1.1.217
username@login02 ~]$ module avail
---------------------- /cm/shared/apps/spack/cpu/lmod/linux-centos8-x86_64/intel/19.1.1.217 ---------
   bamtools/2.5.1      grace/5.1.25              libpng/1.6.37             openmpi/4.0.4 (D)
   bedtools2/2.27.1    gsl/2.5                   libtirpc/1.2.6            papi/6.0.0.1
     [SNIP]
   eigen/3.3.7         jasper/2.0.16             openmpi/3.1.6
-------------------------- /cm/shared/apps/spack/cpu/lmod/linux-centos8-x86_64/Core ----------------
   abaqus/2018                 emboss/6.6.0          gmp/6.1.2           parallel/20200822
   anaconda3/2020.11           freesurfer/7.1.1      go/1.15.1           pciutils/3.7.0
  [SNIP]
   cmake/3.18.2                gcc/9.2.0             mpfr/4.0.2          zstd/1.4.5
   curl/7.72.0                 gcc/10.2.0       (D)  nbo/7.0-openblas
   doxygen/1.8.17              gh/1.13.1             openjdk/11.0.2
------------------------------------------- /cm/local/modulefiles ------------------------------
   boost/1.71.0    cmjob    lua/5.3.5    shared    singularitypro/3.5    slurm/expanse/20.02.3 (L)
------------------------------------ /cm/shared/apps/xsede/modulefiles -----------------------
   cue-login-env    xdinfo/1.5-1    xdusage/2.1-1
---------------------------------------- /usr/share/modulefiles ---------------------------
   DefaultModules    cpu/0.15.4 (L)    gct/6.2    globus/6.0    gpu/0.15.4    nostack/0.15.4
---------------------------------------- /cm/shared/modulefiles ---------------------------
   AMDuProf/3.4.475    default-environment    sdsc/1.0

  Where:
   L:  Module is loaded      D:  Default Module
```

List Current environment

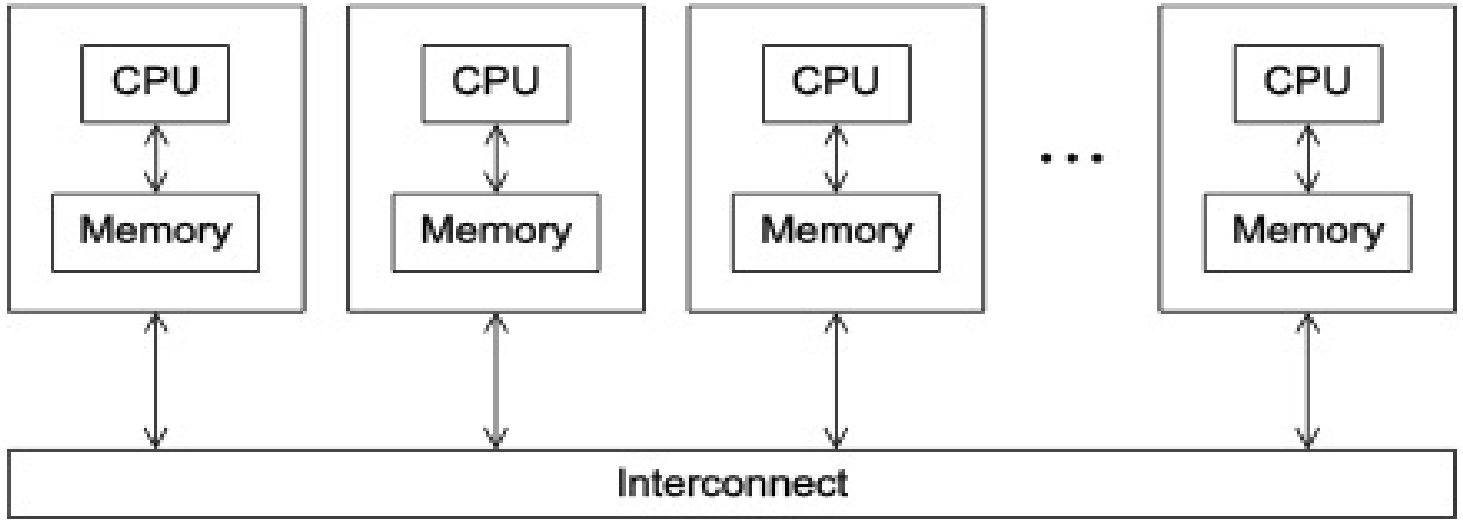
Show available modules

Hands On: Discovering Available Software and Tools

- Log onto ACCESS site
  - Tour site / links
  - Check out what modules you have access to,
    - module list
- HPC system user guides
  - User guides: e.g. expanse.sdsc.edu
- Search for event information
  - SDSC Training Catalog
  - SDSC GitHubs, etc.
  -
- https://github.com/sdsc-hpc-training-org/Expanse-Notebooks/blob/main/Tensorflow_Simple_Training/SimpleTraining.ipynb
- Expanse 101

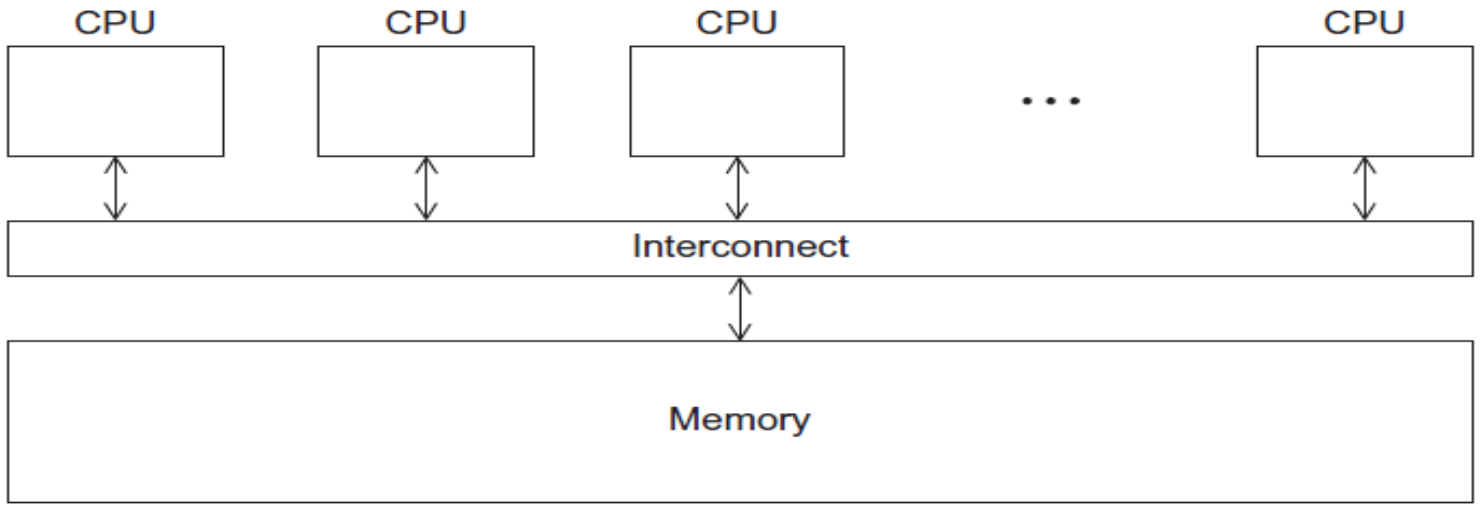
Submitting Jobs to an HPC System

Parallel Models: Distributed Memory



- Programs run asynchronously, pass messages for communication and coordination between resources.
- Examples include: SOA-based systems, massively multiplayer online games, peer-to-peer apps.
- Different types of implementations for the message passing mechanism: HTTP, RPC-like connectors, message queues
- HPC historically uses the Message Passing Interface (MPI)

Parallel Models: Shared Memory



- CPUs all share same localized memory (SHMEM): Coordination and communication between tasks via interprocessor communication (IPC) or virtual memory mappings.
- May use: uniform/non-uniform memory access (UMA or NUMA); cache-only memory architecture (COMA).
- Most common HPC API's for using SHMEM:
  - Portable Operating System Interface (POSIX); Open Multi-Processing (OpenMP) designed for parallel computing – best for multi-core computing.

Methods for Running Jobs on Expanse

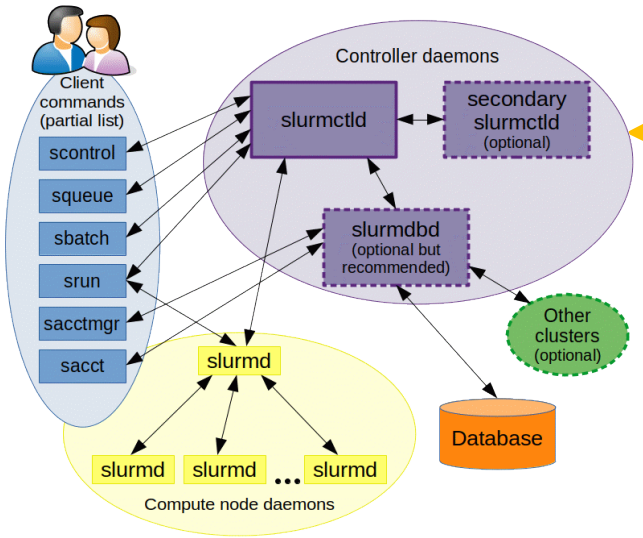
- Expanse uses the **Simple Linux Utility for Resource Management (SLURM)** batch environment.
- Any HPC (or HTC) system — usually a cluster of machines — needs a means of sharing computational resources fairly between users; without, there would be anarchy.
- Batch-queueing systems — usually abbreviated 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 prioritising jobs;
  - one or more queues to which jobs are submitted
  - note: each queue might be configured for a particular type of job, for example, serial or parallel jobs, long or short jobs,or those requiring particularly high memory. These are called partitions (or job queues)
- Batch Jobs: Submit batch scripts to Slurm from the login nodes:
  - Partition (queue)
  - Time limit for the run (maximum of 48 hours)
  - Number of nodes, tasks per node; Memory requirements (if any)
  - Job name, output file location; Email info, configuration
- When you run in the batch mode, you submit jobs to be run on the compute nodes using the sbatch command as described below.
- *Remember that computationally intensive jobs should be run only on the compute nodes and not the login nodes.*
- Expanse places limits on the number of jobs queued and running on a per group (allocation) and partition basis.
- Please note that submitting a large number of jobs (especially very short ones) can impact the overall scheduler response for all users.
- Interactive Jobs: Use the srun command to obtain nodes for 'live,' command line interactive access:
  - CPU:
    - srun --partition=debug --account=XYZ123 --pty --nodes=1 --ntasks-per-node=128 --mem=248 -t 00:30:00 --wait=0 --export=ALL /bin/bash
  - GPU:
    - srun --pty --account=XYZ123 --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-debug --gpus=1 -t 00:10:00 /bin/bash

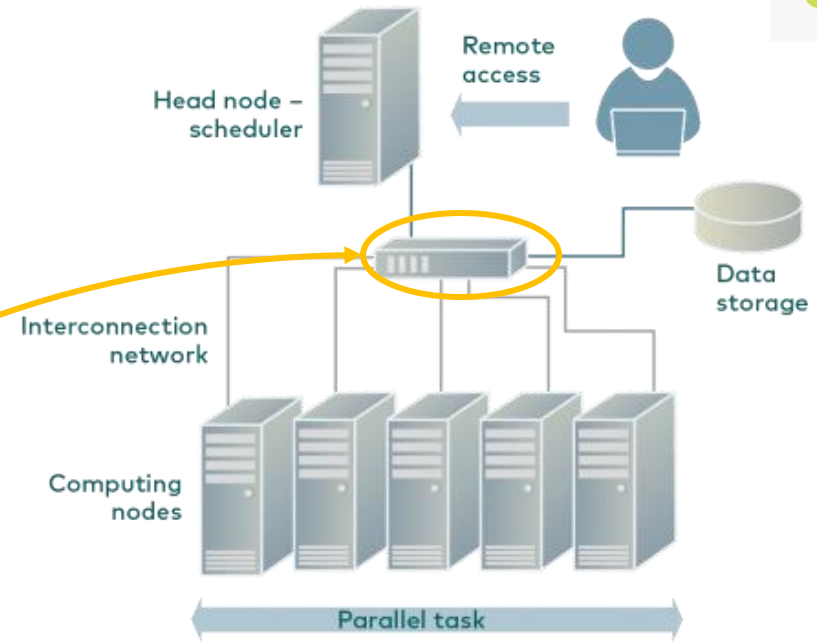
What is a Scheduler & How are They Used?

- Any HPC (or HTC) system — usually a cluster of machines — needs a means of sharing computational resources fairly between users; without, 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 prioritising jobs;
  - one or more queues to which jobs are submitted
  - note: each queue might be configured for a particular type of job, for example, serial or parallel jobs, long or short jobs, or those requiring particularly high memory. These are called partitions (or job queues)

Batch Jobs: Slurm Resource Manager

- 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

Batch Scheduler Main Goals

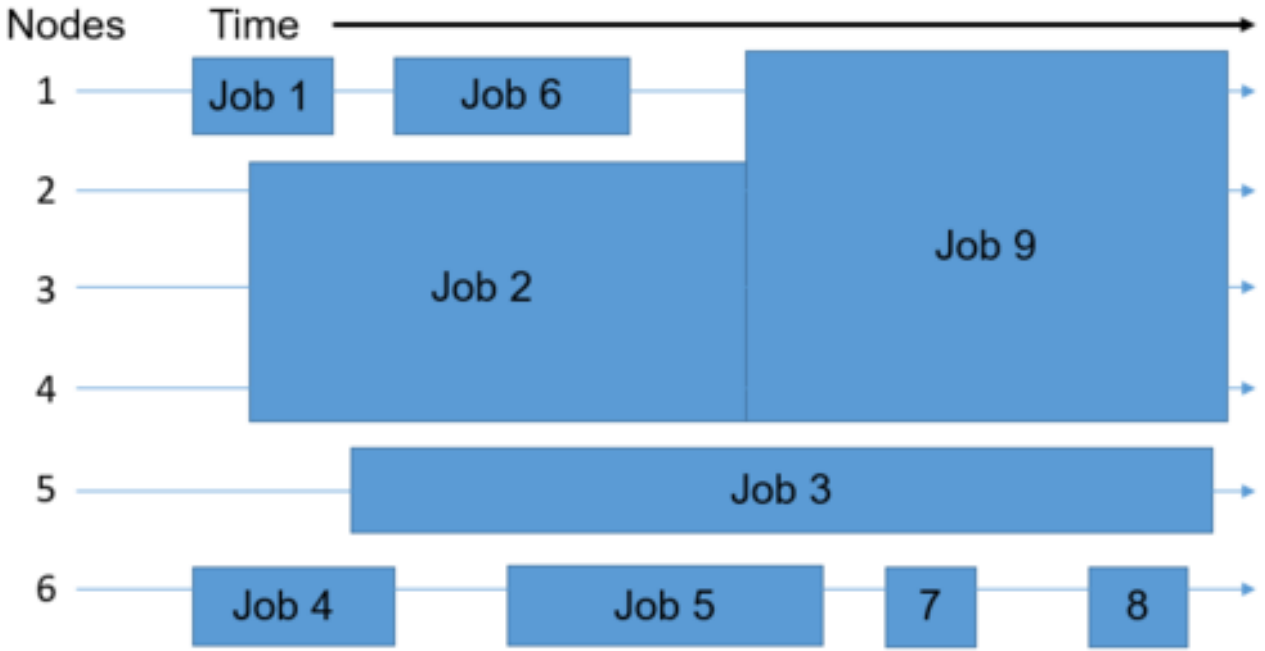
- Minimize time between job submission and completion:
  - No job should stay in queue for extensive periods of time.
- Optimize CPU utilization:
  - Algorithms focus on minimizing CPU idle times.
- Maximize job throughput:
  - Manage as many jobs per time unit as possible.
- Support running jobs automatically in the background

Common Batch Scheduler Terms

- batch processing comes from the idea of batch production (production of a "batch" of multiple items at once, one stage at a time)
- Batch job scheduler: computer application for controlling unattended background execution of jobs.
- batch scheduling: execution of non-interactive jobs
- batch processing: method of running software programs called jobs in batches automatically.
- Typically, users required to submit the jobs, no other interaction by the user is required to process the batch.
- Batches may automatically be run at scheduled times as well as being run contingent on the availability of computer resources
- The data structure of jobs to run is known as the job queue.
- Synonyms include: batch system, distributed resource management system (DRMS), distributed resource manager (DRM), workload automation (WLA).

- How a Scheduler Schedules Jobs
  - a 6 node system
  - yser 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

Batch Scripts -- Used to Launch Jobs

- Batch Jobs: Submit batch scripts from the login nodes to a batch service:
  - Expanse uses the Simple Linux Utility for Resource Management (SLURM) batch environment.
- Can be used to run serial jobs (1 core), multi-threaded (OpenMP), multi-core jobs, and parallel (multi-node, MPI) jobs
- Parameters you can set:
  - Partition (queue)
  - Time limit for the run (maximum of 48 hours)
  - Number of nodes, tasks per node; Memory requirements (if
  - You can define the job name, output file location; email info configuration

https://www.sdsc.edu/support/user_guides/expanse.html#running

Batch Scripts Contents

```
[uswername@login02 calc-prime]$ cat mpi-prime-slurm.sb
#!/bin/bash
#SBATCH --job-name="mpi_prime"
#SBATCH --output="mpi_prime.%j.%N.out"
####SBATCH --partition=compute
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:10:00
#SBATCH -A use300

## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## echo job name and id:
echo "SLURM_JOB_NAME:  $SLURM_JOB_NAME"
echo "SLURM_JOB_ID:    $SLURM_JOB_ID"
d=`date`
echo "DATE:  $d"
```

Batch Script 4 Main Parts:

- “Shebang” !
  - #!/bin/bash command tells the shell to run the script using the bash
- Resource Request:
  - options preceded with "#SBATCH" before any executable commands in the script
- Dependencies
  - Loads software that the project depends on to execute
- Job Steps:
  - Specify the list of tasks to be carried out.

Batch Script Output: ENV_Info

```
[uswername@login02 env_info ]$ cat env-slurm.sb
#!/bin/bash
#SBATCH --job-name=" env_info "
#SBATCH --output="mpi_prime.%j.%N.out"
####SBATCH --partition=compute
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:10:00
#SBATCH -A use300

## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## echo job name and id:
echo "SLURM_JOB_NAME:  $SLURM_JOB_NAME"
echo "SLURM_JOB_ID:    $SLURM_JOB_ID"
d=`date`
echo "DATE:  $d"
e=`env`
echo "env= $e"
```

```
[mthomas@login02 env_info ]$ sbatch env-slurm.sb
Submitted batch job 14126259
[mthomas@login02 env_info]$ !sq
squeue -u mthomas
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
14126259     debug env_info   mthomas  R       0:04      1 exp-9-55
[mthomas@login01 env_info ]$ cat env_info.14126259.exp-4-35.out
SLURM_JOB_NAME:  env_info
SLURM_JOB_ID:    14126259
hostname=  exp-4-35
date=  Sun Jun 26 22:05:15 PDT 2022
whoami=  mthomas
pwd=  /home/mthomas/hpctr-examples/env_info
Currently Loaded Modules:       1) slurm/expanse/21.08.8   2) cpu/0.15.4
---------------------------------
env= LD_LIBRARY_PATH=/cm/shared/apps/slurm/current/lib64/slurm:
[SNIP]
/cm/shared/apps/slurm/current/lib64
SLURM_SUBMIT_DIR=/home/mthomas/hpctr-examples/env_info
HISTCONTROL=ignoredups
DISPLAY=localhost:16.0
HOSTNAME=exp-4-35
[SNIP]
```

Example Code: https://github.com/sdsc-hpc-training-org/hpctr-examples

Basic Job management

- squeue - View information about jobs in scheduling queue
- A few common commands:

-A, --account=<account_list>	Filter by accounts (comma-separated list)
j, --jobs=<job_id_list>	Filter by job IDs (comma-separated list)
-p, --partition=<partition_list>	Filter by partitions (comma-separated list)
-u, --user=<user_list>	Filter by users (comma-separated list)

SLURM Environment Variables

Internal ENV variables that exist when job is submitted:

INPUT ENVIRONMENT VARS

- Upon startup, sbatch will read and handle the options set in the following environment variables.
- SBATCH_JOB_NAME
  - Same as -J, --job-name
- SBATCH_ACCOUNT
  - Same as -A, --account
- SBATCH_TIMELIMIT
  - Same as -t, --time
- More…

OUTPUT ENVIRONMENT VARS

- The Slurm controller will set the following variables in the environment of the batch script.
- SLURM_EXPORT_ENV
  - Same as --export.
- SLURM_JOB_ID
  - The ID of the job allocation.
- SLURM_JOB_NAME
  - Name of the job.
- More…

Slurm Partitions on Expanse

Partition Name	Max Walltime	Max Nodes/Job	Max Running Jobs	Max Running + Queued Jobs	Charge Factor	Notes
compute	48 hrs	32	32	64	1	Used for exclusive access to regular compute nodes; *limit applies per group*
shared	48 hrs	1	4096	4096	1	Single-node jobs using fewer than 128 cores
gpu	48 hrs	4	4	8 (32 Tres GPU)	1	Used for exclusive access to the GPU nodes
gpu-shared	48 hrs	1	24	24 (24 Tres GPU)	1	Single-node job using fewer than 4 GPUs
large-shared	48 hrs	1	1	4	1	Single-node jobs using large memory up to 2 TB (minimum memory required 256G)
debug	30 min	2	1	2	1	Priority access to shared nodes set aside for testing of jobs with short walltime and limited resources
gpu-debug	30 min	2	1	2	1	Priority access to gpu-shared nodes set aside for testing of jobs with short walltime and limited resources; *max two gpus per job*
preempt	7 days	32		128	.8	Non-refundable discounted jobs to run on free nodes that can be pre-empted by jobs submitted to any other queue
gpu-preempt	7 days	1		24 (24 Tres GPU)	.8	Non-refundable discounted jobs to run on unallocated nodes that can be pre-empted by higher priority queues

https://www.sdsc.edu/support/user_guides/expanse.html#running

Slurm Commands

- sacct
- sacctmgr
- salloc
- sattach
- sbatch
- sbcast
- scancel
- scontrol
- scrontab
- sdiag
- sh5util
- sinfo
- sprio
- squeue
- sreport
- srun
- sshare
- sstat
- strigger
- sview

Common Slurm Command Examples

- Submit jobs using the sbatch command:
  - $ sbatch mycode-slurm.sb
  - Submitted batch job 8718049
- Check job status using the squeue command:
  - $ squeue -u $USER
  - JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
  - 8718049 compute mycode user PD 0:00 1 (Priority)
- Once the job is running, monitor its state:
  - $ squeue -u $USER
  - JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
  - 8718049 debug mycode user R 0:02 1 expanse-14-01
- Cancel a running job:
  - $ scancel 8718049

https://slurm.schedmd.com/sbatch.html

SLURM “srun” Command

- Used to launch a parallel job on cluster managed by Slurm.
- If necessary, srun will first create a resource allocation in which to run the parallel job.
- Common arguments used on Expanse:
  - --mpi=<mpi_type>Identify the type of MPI to be used. Use ‘pmi2’
  - -n, --ntasks=<number>Specify the number of tasks to run.
  - -cpu-bind: bind tasks to CPUs
- what is the difference between mpirun and SLURM srun?
  - srun is optimized for Expanse (via the PMI interface) and more efficiently allocates, organizes, and starts up the MPI processes.

srun --mpi=pmi2 -n 24 --cpu-bind=rank ./mpi_prime Y

https://slurm.schedmd.com/srun.html

HPC Systems Have Multiple File Systems (Expanse)

- Home directories (/home/$USER) – 8 week rolling backup.
  - Login nodes: 100GB limit;
  - Use for source trees, binaries, and small input files.
  - Not good for large scale I/O.
- Lustre filesystems: Good for scalable large block I/O
  - /expanse/lustre/scratch/$USER/temp_project
    - 2.5PB; peak performance: 100GB/s. Good for storing large scale scratch data during a job.
  - /expanse/lustre/projects/ -
    - 2.5PB, peak performance: 100 GB/s. Long term storage.
    - Not good for large # of small files or small block I/O
- Local node “scratch” (SSD) filesystems
  - /scratch local to each native compute node – 210GB on regular compute nodes, 285GB on GPU, large memory nodes, 1.4TB on selected compute nodes.
  - SSD location is good for writing small files and temporary scratch files. Purged at the end of a job.
- Webinar on Data Management and File Systems:
  - https://www.sdsc.edu/event_items/202110_ExpanseWebinar-M.Shantharam.html

Hands On: Logging on and checking account info

CIS250186: AI Unlocked: Empowering Higher Education through Research and Discovery Active

Discover: Feb 26, 2025 to Feb 25, 2026

Overview | Credits + Resources | Users + Roles | History

Resource	Status	Balance	End Date	My Username
Delta CPU	Active	15K of 15K Core-hours remaining (100%)	Feb 25, 2026	thomasm
NCSA Delta GPU	Active	5K of 5K GPU Hours remaining (100%)	Feb 25, 2026	thomasm
NCSA DeltaAI	Active	5K of 5K GPU Hours remaining (100%)	Feb 25, 2026	thomasm
SDSC Expanse CPU	Active	15K of 15K Core-hours remaining (100%)	Feb 25, 2026	mthomas
SDSC Expanse GPU	Active	8K of 8K GPU Hours remaining (100%)	Feb 25, 2026	ux415294

```
[mthomas@login02 ~]$ expanse-client user
Resource  expanse

┌──┬──────────┬─────────┬──────────┬──────────────┬──────┬───────────┬─────────────────┐
│  | NAME     | STATE | PROJECT | TG PROJECT    | USED | AVAILABLE | USED BY PROJECT |
├──┼──────────┼─────────┼──────────┼──────────────┼──────┼───────────┼─────────────────┤
| ..... [SNIP].....
| 3 | mthomas | allow | abc123  | TG-abcd      |    0 |     40000 |           15091 |
| 4 | mthomas | allow | ukl119  | TG-CIS250186 |    0 |     15000 |               0 |
| 5 | mthomas | allow | abc123  |              | 2908 |   5050000 |         4427508 |
└──┴──────────┴─────────┴──────────┴──────────────┴──────┴───────────┴─────────────────┘
[mthomas@login02 ~]$
```

Hands On: Submitting a Job to the SLURM Batch Queue

```
[mthomas@login01 ENV_INFO]$ cat env-slurm.sb
#!/bin/bash
#SBATCH --job-namea="envinfo"
#SBATCH --output="envinfo.%j.%N.out"
#SBATCH --partition= compute
#SBATCH –nodes =1
#SBATCH –tasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:01:00

## Environment
module purge
module load slurm
module load cpu

##  perform some basic unix commands
echo "------------------------------"
echo "hostname= " `hostname`
echo "date= " `date`
echo "whoami= " `whoami`
echo "pwd= " `pwd`
echo "module list= " `module list`
echo "------------------------------"
echo "env= " `env`
echo "------------------------------"
```

```
[mthomas@login01 ENV_INFO]$ cat envinfo.108867.exp-6-56.out
--------------------------------
hostname=  exp-6-56
date=  Wed Oct 7 23:45:43 PDT 2020
whoami=  mthomas
pwd=  /home/mthomas/DEMO/ENV_INFO
Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/1.0
--------------------------------
env=  SLURM_MEM_PER_CPU=1024
LD_LIBRARY_PATH=/cm/shared/apps/slurm/current/lib64/slurm:/cm/shared/ap
ps/slurm/current/lib64 LS_COLORS=rs=0

[SNIP]

 MODULESHOME=/usr/share/lmod/lmod LMOD_SETTARG_FULL_SUPPORT=no
HISTSIZE=5000 LMOD_PKG=/usr/share/lmod/lmod
LMOD_CMD=/usr/share/lmod/lmod/libexec/lmod SLURM_LOCALID=0
LESSOPEN=| |/usr/bin/lesspipe.sh %s LMOD_FULL_SETTARG_SUPPORT=no
LMOD_DIR=/usr/share/lmod/lmod/libexec BASH_FUNC_module%%=() { eval
$($LMOD_CMD bash "$@") && eval $(${LMOD_SETTARG_CMD:-:} -s sh) }
BASH_FUNC_ml%%=() { eval $($LMOD_DIR/ml_cmd "$@") } _=/usr/bin/env
--------------------------------
```

MPI Hello World: Batch Script

- To run the job, use the sbatch script submission command.
- Monitor the job until it is finished using the squeue command.
- Be sure to compile your code: see instructions in the github repo

```
[username@login02 MPI]$ cat hellompi-slurm.sb
#!/bin/bash
#SBATCH --job-name="hellompi"
#SBATCH --output="hellompi-gnu.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=128
#SBATCH --export=ALL
#SBATCH –A abc123
#SBATCH -t 00:10:00

#This job runs with 2 nodes,
128 cores per node for a total of 256 cores.

## Environment
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## Use srun to run the job

srun --mpi=pmi2 -n 256 --cpu-bind=rank ./hello_mpi

[username@login02 MPI]$
```

```
[username@login02 MPI]$ sbatch hellompi-slurm-gnu.sb;
squeue -u username
Submitted batch job 108910
             JOBID PARTITION     NAME     USER
ST       TIME  NODES NODELIST(REASON)
            108910   compute hellompi  username
PD       0:00      2 (None)
[username@login02 MPI]$ cat hellompi-gnu.108910.exp-12-
54.out
 node         4 : Hello world!
 node         5 : Hello world!
 node         7 : Hello world!
 node         0 : Hello world!
 node         2 : Hello world!
 node         3 : Hello world!
 node         9 : Hello world!
 node        10 : Hello world!
 [SNIP]
 node       249 : Hello world!
 node       186 : Hello world!
 node       220 : Hello world!
 node       203 : Hello world!
 node       135 : Hello world!
```

Hands On: Submitting a GPU Job

```
/*
* Copyright 1993-2010 NVIDIA Corporation. All rights reserved. *
* NVIDIA Corporation and its licensors retain all intellectu
property and
*
* Updated by Mary Thomas, April 2023, for simple cuda compile example
*/
#include <stdio.h>
__global__ void kernel( void ) { }
int main( void ) { kernel<<<1,1>>>();
printf( "Hello,  SDSC HPC Training World!\n" ); return 0;
}
```

- To run the job, use the sbatch script submission command on the login node.
- Monitor the job until it is finished using the squeue command.
- Be sure to compile your code: see instructions in the github repo README file.

```
[username@login02 OpenACC]$ cat openacc-gpu-shared.sb
#!/bin/bash
#SBATCH --job-name="OpenACC"
#SBATCH --output="OpenACC.%j.%N.out"
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --gpus=1
#SBATCH –A abc123
#SBATCH -t 01:00:00

#Environment
module purge
module load slurm
module load gpu
module load pgi

#Run the job
./laplace2d.openacc.exe

[username@login02 OpenACC]$ sbatch openacc-gpu-shared.sb
[username@login02 OpenACC]$ sbatch openacc-gpu-shared.sb ; squeue -u username
Submitted batch job 108915
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            108915 gpu-share  OpenACC  username PD       0:00      1 (None)
```

```
[username@login02 OpenACC]$ cat OpenACC.108915.exp-7-57.out
main()
Jacobi relaxation Calculation: 4096 x 4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 1.084470 s
[username@login02 OpenACC]$
```

Interactive HPC Access

Defining 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.

Interactive HPC Methods & Applications

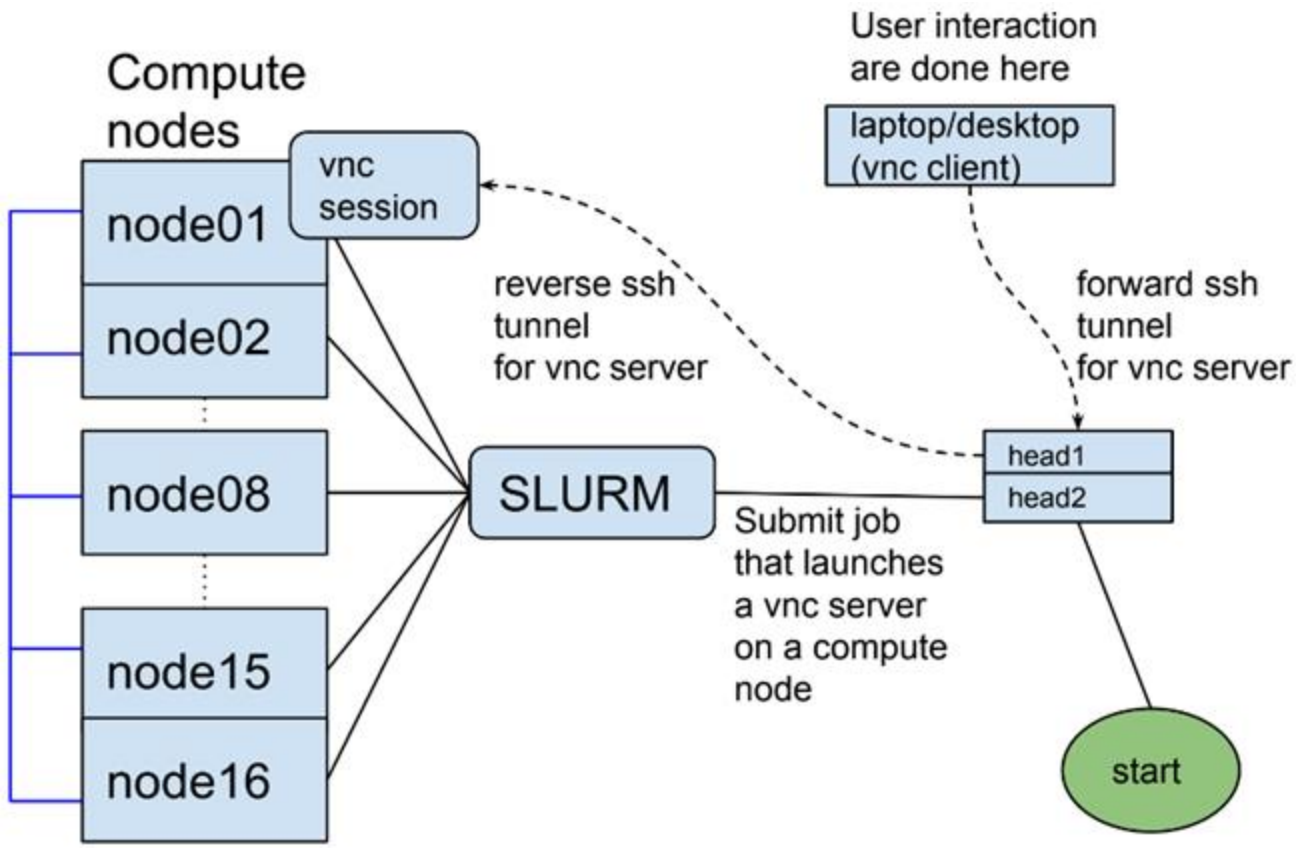
- Interact with data after job is done:
  - Unix: query file info, location, output, grep, awk, sed
  - Cat the file contents from batch job or raw data
  - NetCDF data browser
- Plot results:
  - From within the code/model using libraries
  - Command-line driven graphing utility : Gnuplot
- Data visualization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

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.

Real-time User Interactions

- *Real-time:* user inputs and performs tasks on a set of compute nodes
- Examples: Juptyer notebooks; ParaView visualization tools



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	srun --partition=debug --pty --account=abc123 --nodes=1 --ntasks-per-node=4 --mem=8G -t 00:30:00 --wait=0 --export=ALL /bin/bash
GPU	srun --partition=gpu-debug --pty --account=abc123 --ntasks-per-node=10 --nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash

- (Tested March/2025)

Using An Interactive CPU node

```
[mthomas@login01 calc-prime]$ srun --partition=debug  --pty --account=abc123 --nodes=1
--ntasks-per-node=16  --mem=8G -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 16 ./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

- 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.
- Exit interactive session when your work is done or you will be charged more CPU time.

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                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[username @exp-14-57 ~]$ exit
```

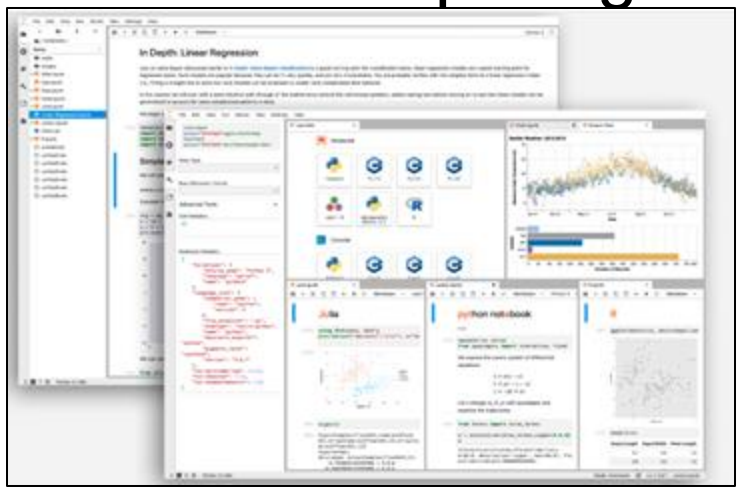
Request an interactive node for 30 minutes

Verify you are on a GPU node

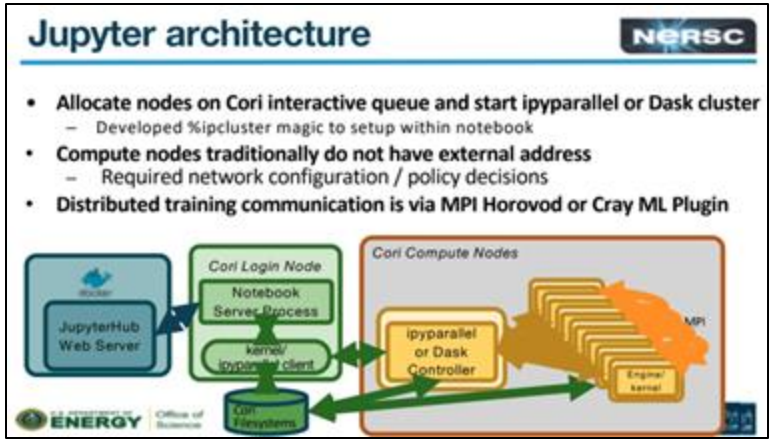
Exit when tasks are done

Interactive HPC Computing

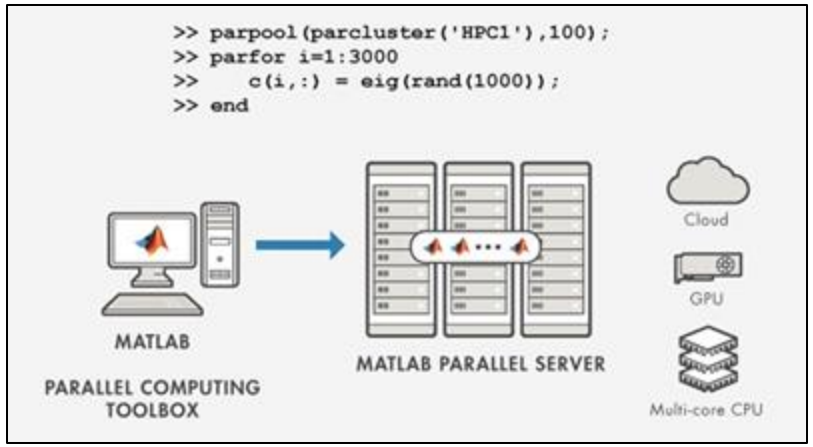
https://jupyter.org/



Interactive Distributed Computing with Jupyter (NERSC)

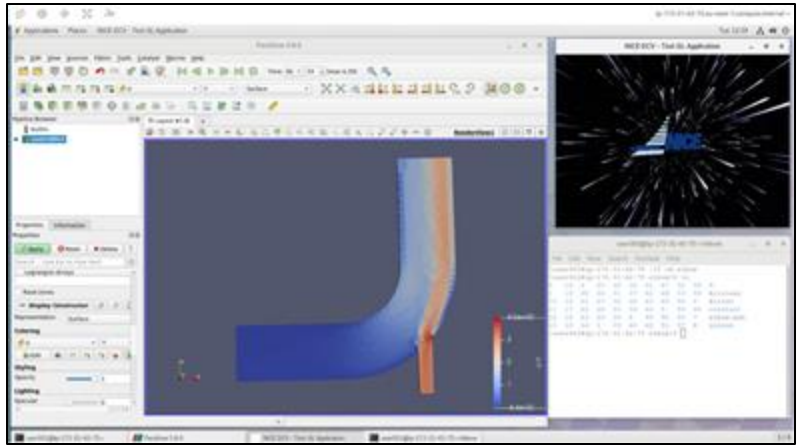


https://drive.google.com/file/d/1-OFjrk1q3L1d3uakr2xkozrPn2c2VZpZ/view



Parallel Matlab (hosted on AWS)

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



Paraview (running on AWS)

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

Accessing and Running Secure Notebooks on SDSC HPC Systems

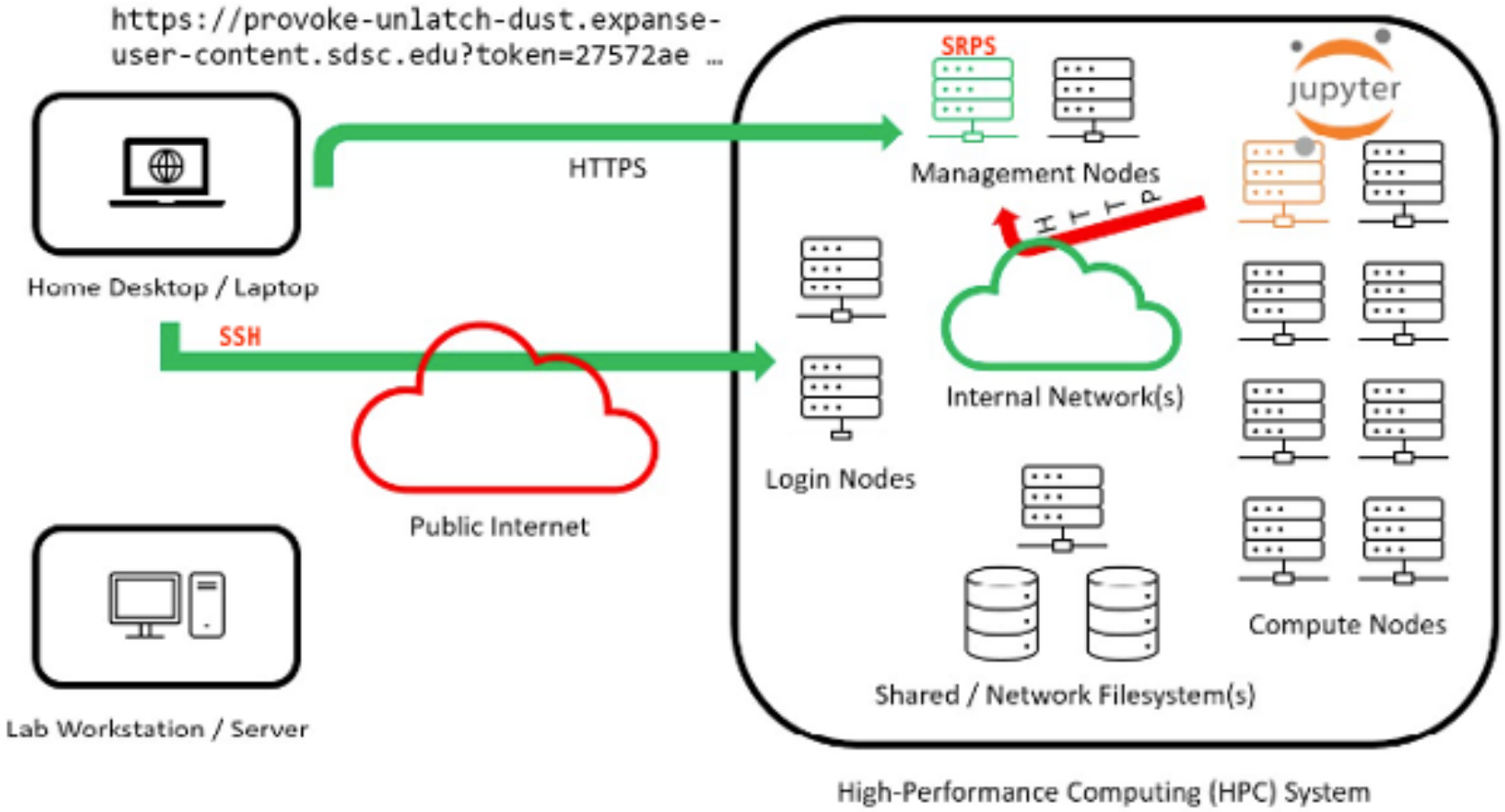
- Install notebook application:
  - Locally: install Anaconda on your laptop
  - Remotely:
    - Install Anaconda/conda on the remote machine (default is HTTP) – not recommended
- Running remotely:
  - Connect over HTTP (default, insecure)
  - Connect over HTTP + SSH tunneling (secure, but inconvenient)
  - 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!

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

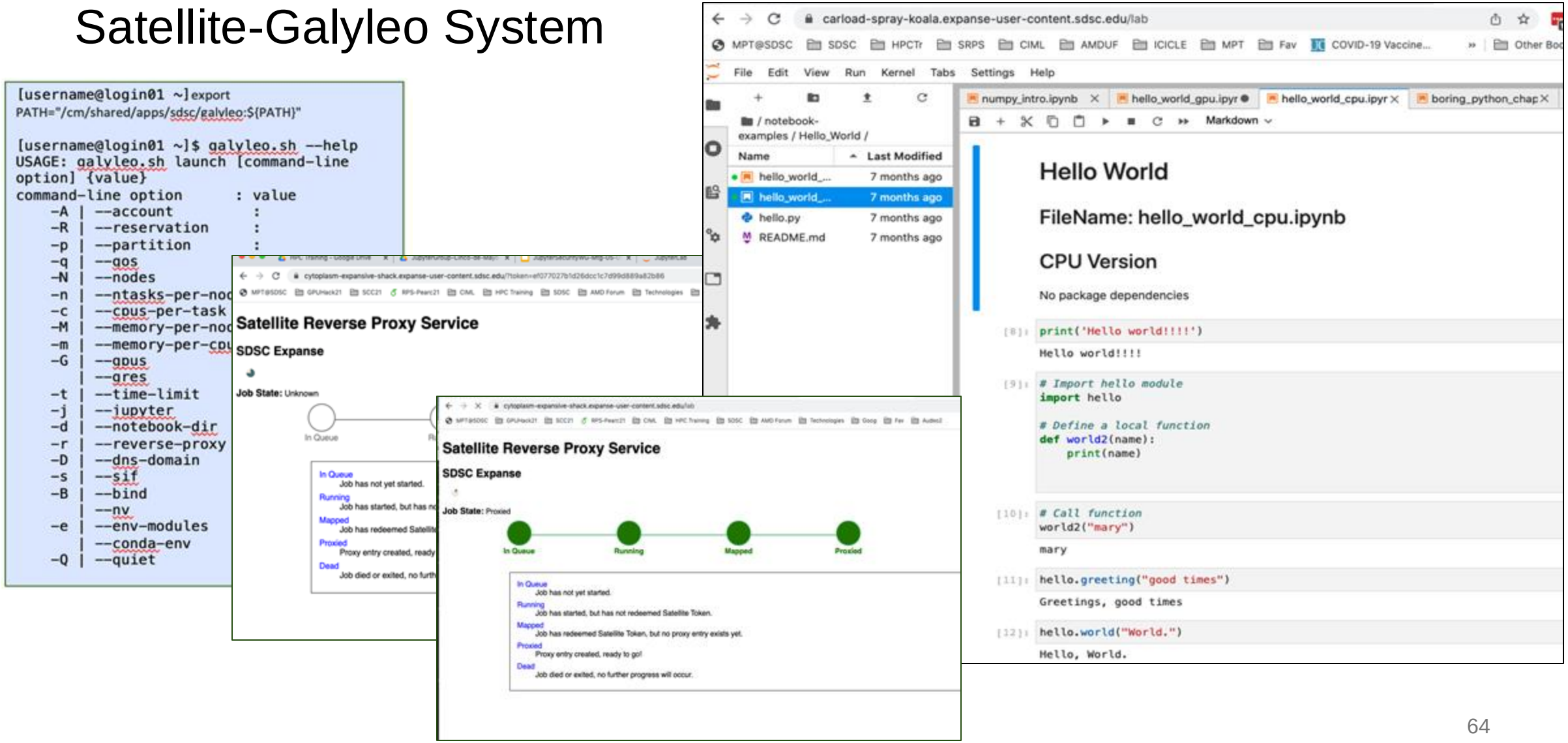
https://github.com/sdsc/galyleo

Running Notebooks Remotely & Securely Using SRPS



Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

Satellite-Galyleo System



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/sdsc/galyleo

Launching CPU notebooks using galyleo

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 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 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 galyleo-  username PD       0:00      1 (None)
[username@login01 ~]$ squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9773665 gpu-debug galyleo-  username  R       0:20      1 exp-7-59
```

Expanse User Portal

Secure (HTTPS) environment

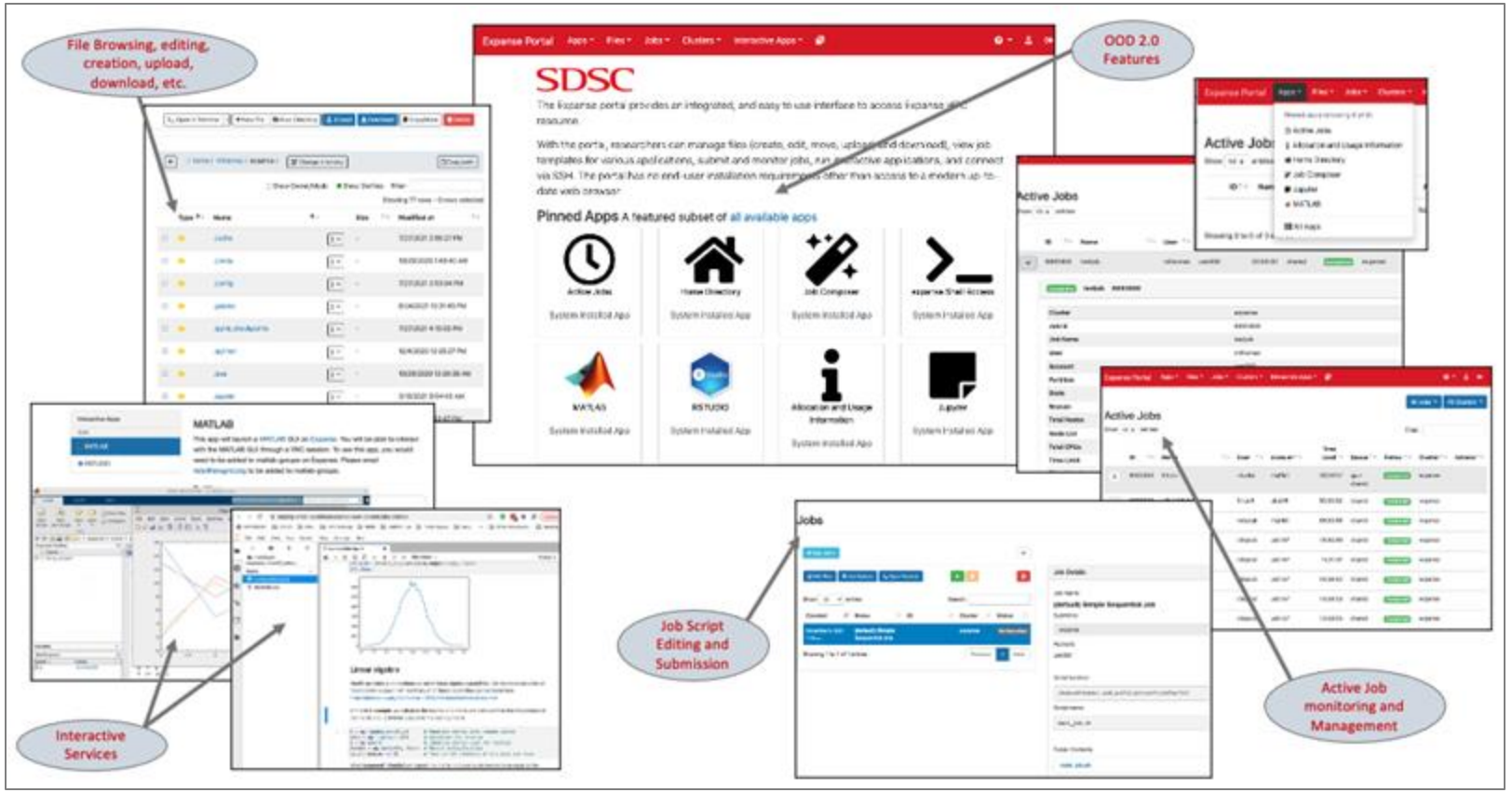
- Authenticate using ACCESS credentials
- Manages software dependencies

Secure services including:

- Data & File Management
- Job Builder
- Batch Job Submission
- Job Monitoring

Interactive applications:

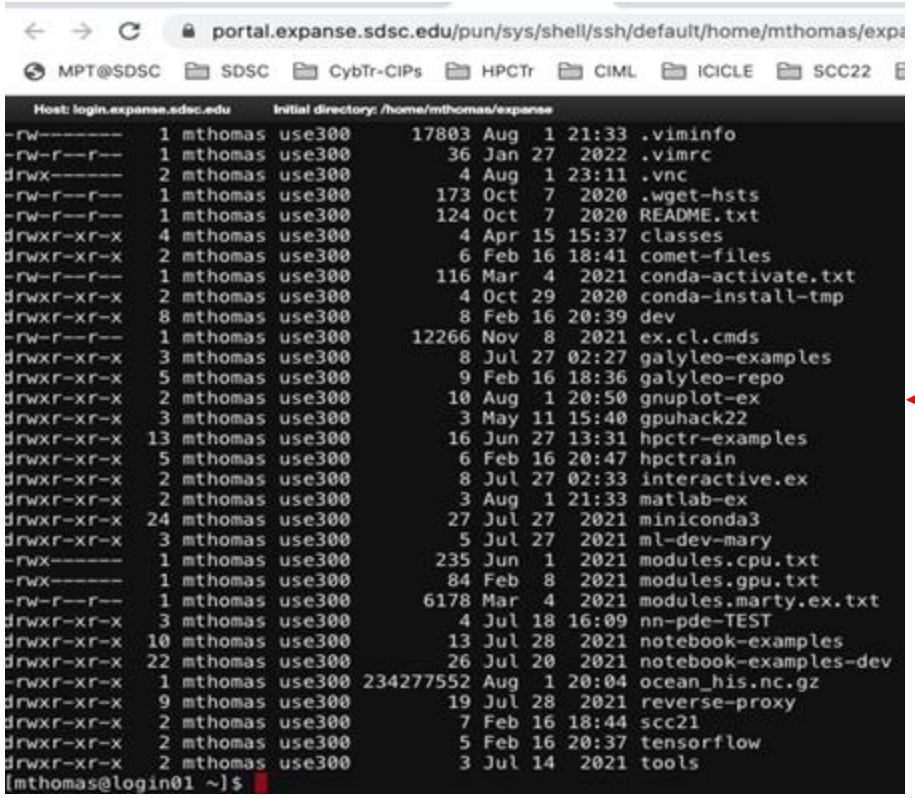
- Jupyter Notebooks
- Jupyter Lab
- Matlab
- Rstudio

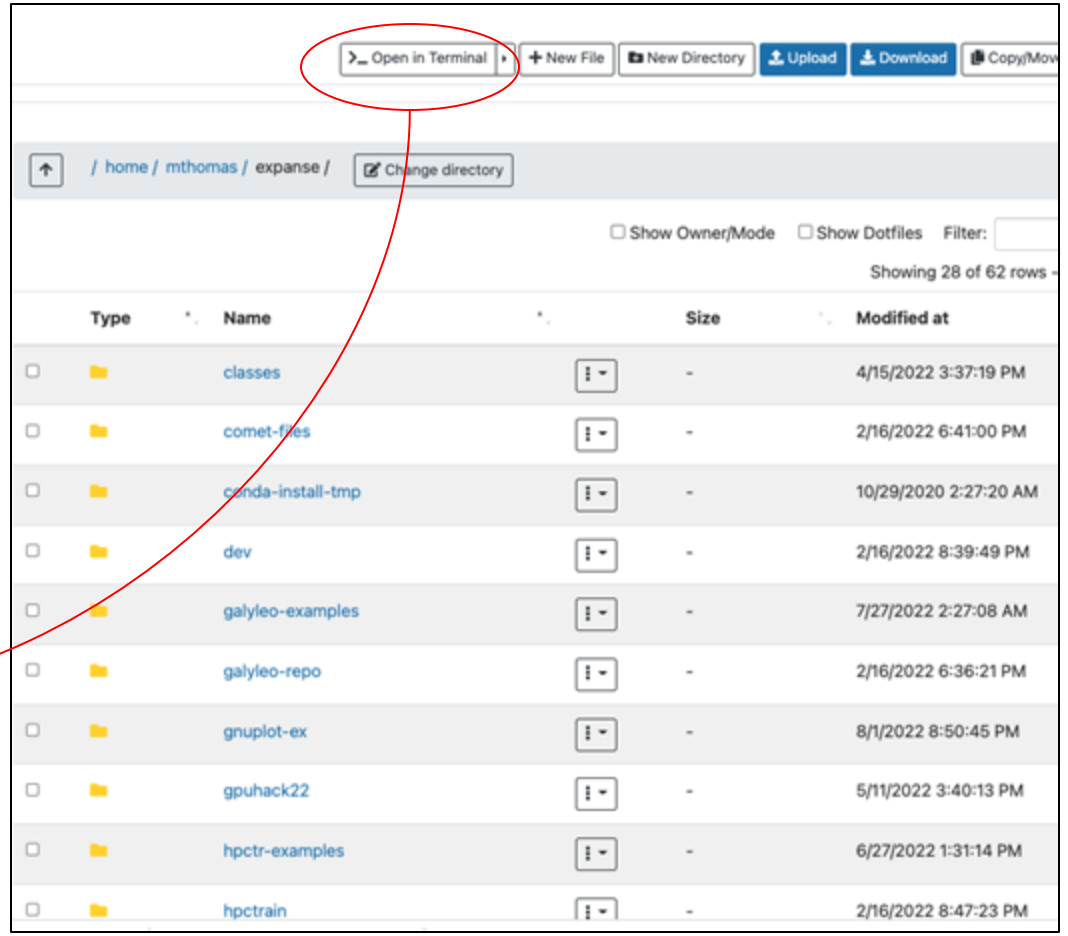


https://portal.expanse.sdsc.edu;

Expanse Portal: File Management

- Can view folders and files on Expanse





Secure Services: Job Builder

Secure services including:

- Job Builder
- Hands-on:
  - Modify and run hello-AI-Unlocked
  - Build a simple job: using Calc Prime Number

Secure Services: Batch Job Submission, Monitoring, Management

- Batch Job Submission
- Job Monitoring
- Job Management
- TODO:
  - Capture images for each stage
  - HANDS ON:

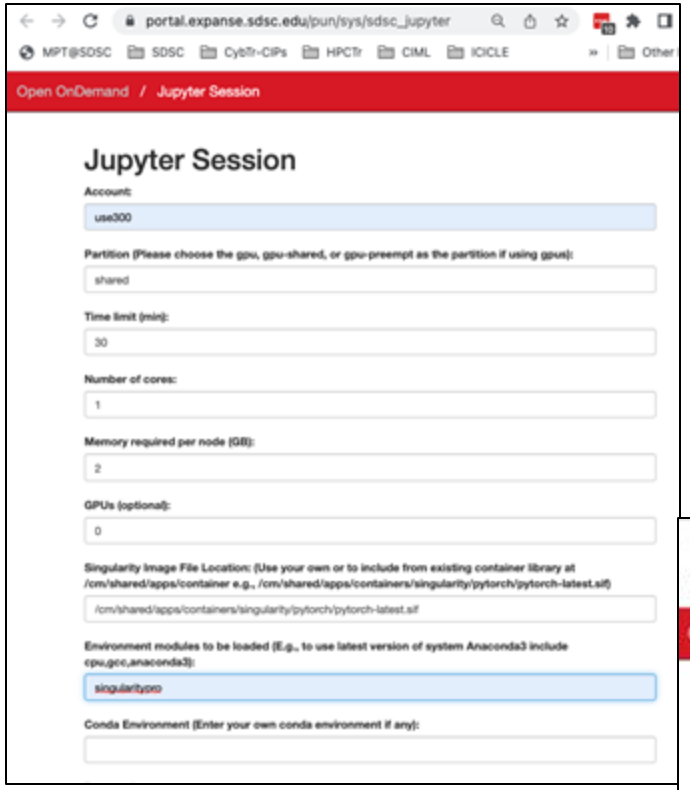
Secure Services: Interactive applications

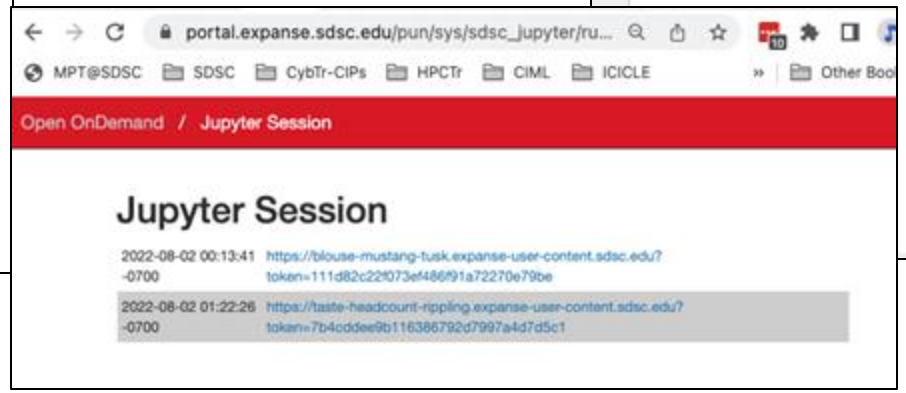
Secure services including:

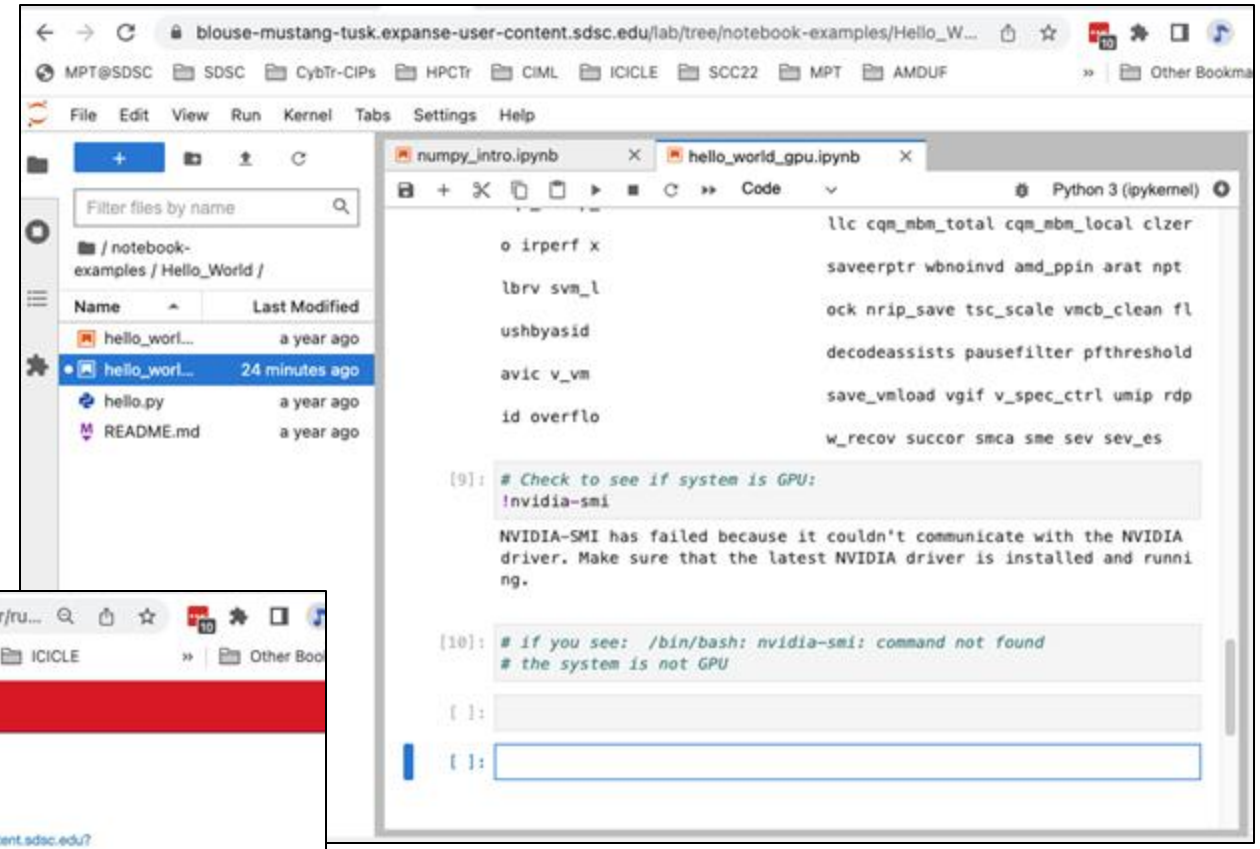
- Interactive applications including: Jupyter Notebooks & Jupyter Lab, Matlab, Rstudio.

Expanse Portal: Launching Jupyter Notebooks or Jupyter Lab

Step 1: Fill out Jupyter Launch form





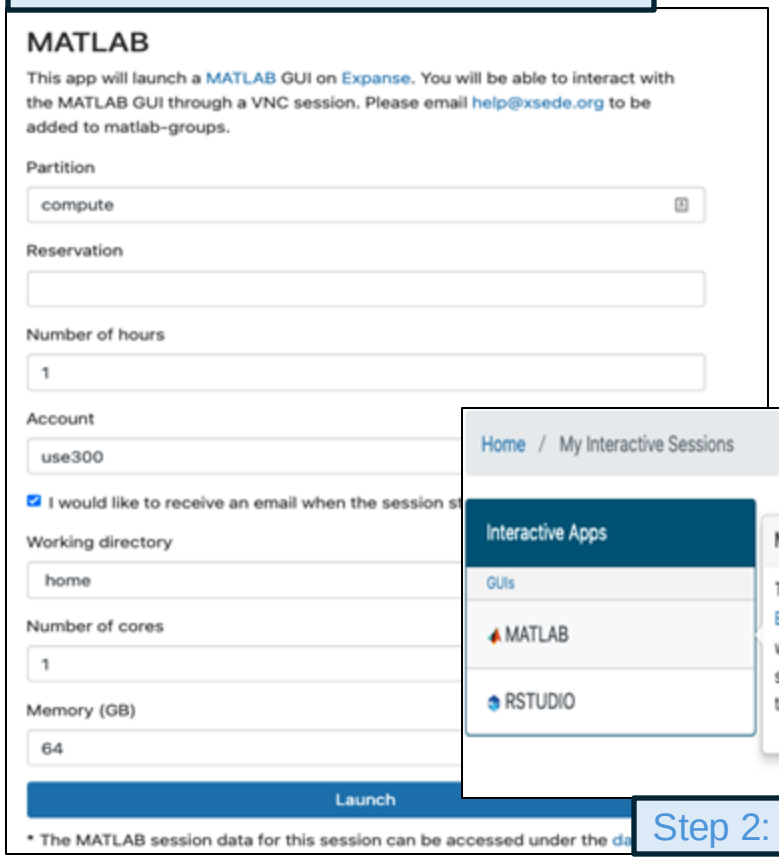


Expanse Portal: Input data example for Jupyter Notebook

ELEMENT	VALUE
Account:	ukl119 (only works for this NAIRR workshop)
Partition: (choose gpu, gpu-shared, or gpu-preempt as the partition if using gpus)	debug. Or. shared
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 OR /cm/shared/apps/containers/singularity/tensorflow/tensorflow-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-packK:	
Reservation:	
QoS:	
Working directory:	HOME
Type:	JupyterLab

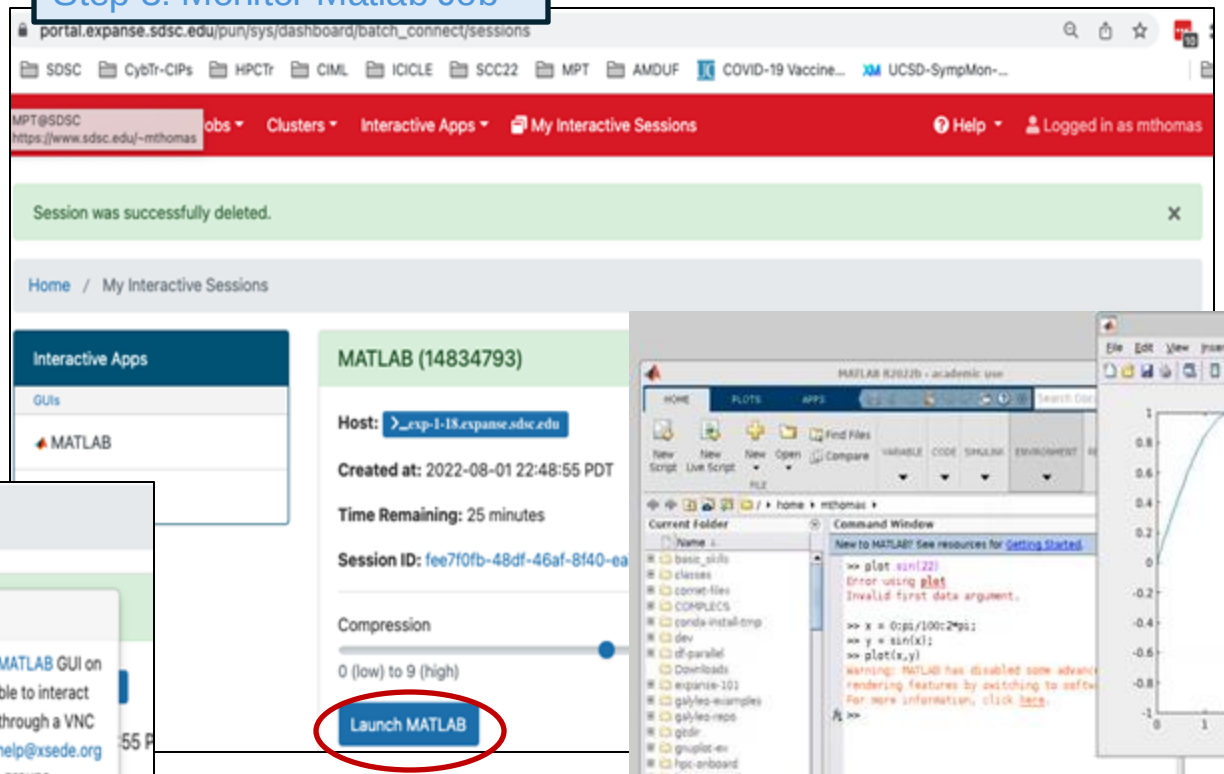
Expanse Portal: Running Matlab

Step 1: Fill out Matlab Launch form

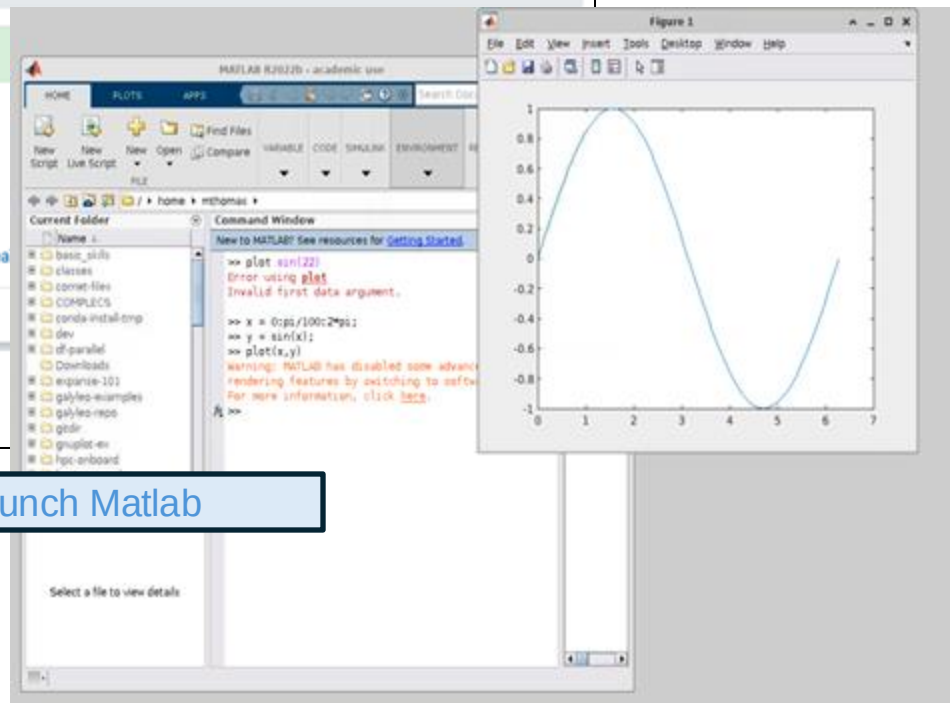


Step 2: Launch Matlab

Step 3: Monitor Matlab Job



Step 4: Launch Matlab



Hands On: Interactive Computing

- Log onto portal
  - Open terminal window/shell
  - Clone hpctrain examples
- Browse files via portal
- Job Builder
  - Pick example and submit
  - Build a new Job
- Launch Jupyter lab
  - Run examples from AI Unlocked GitHub examples
- Launch Jupyter lab

Thank You!

Q&A

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

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

Resources

- AI Unlocked GitHub Repo, including this presentation:
  - https://github.com/sdsc-complecs/interactive-computing/
- SDSC Training Resources
  - HPC/CI On-Demand Training Catalog: https://www.sdsc.edu/education/on-demand-learning/index.html
  - HPC Example Code: https://github.com/sdsc-hpc-training-org/hpctr-examples
  - Hands-on HPC/CI Training: https://hpc-training.sdsc.edu/
  - Running notebooks
    - Using galyleo: https://github.com/sdsc/galyleo
    - Expanse Notebooks Collection: https://github.com/sdsc-hpc-training-org/Expanse-Notebooks
- Expanse :
  - Project page: expanse.sdsc.edu
  - User Guide: https://expanse.sdsc.edu/support/user_guides/expanse.html
- Problems? Contact consult@sdsc.edu