

Lawrencium 101 : HPC on Lawrencium Supercluster

Sapana Soni

HPCS User Support Team

Outline

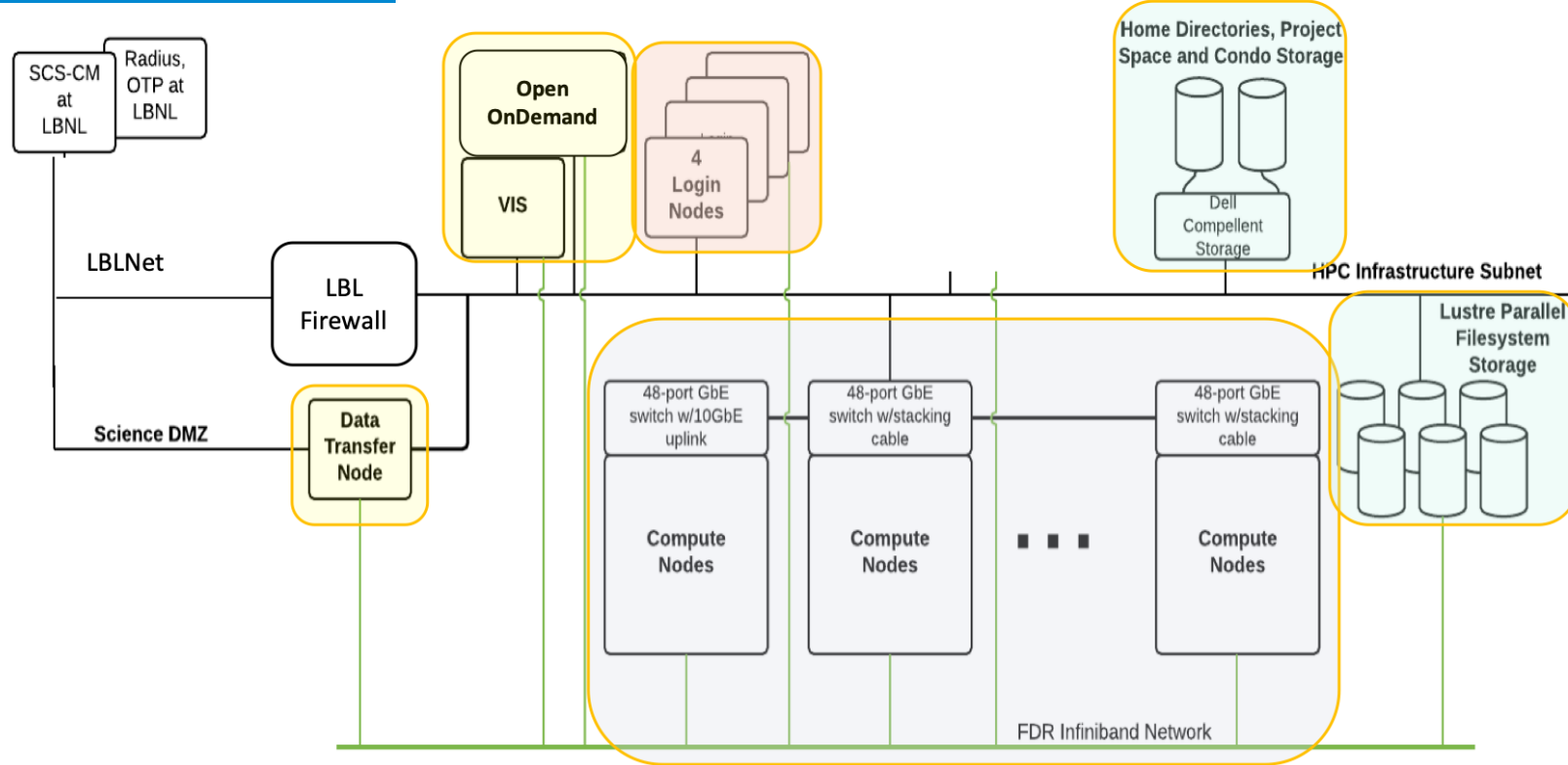
1. [Lawrencium supercluster Overview](#)
2. [Access and login to Lawrencium](#)
3. [Software access and installation](#)
4. [Job submission and monitoring](#)
5. [Data transfer to/from clusters](#)
6. [Open On Demand: a browser based HPCS portal](#)

Lawrencium Cluster Overview

- A LBNL Condo Cluster Computing Program
 - Support researchers in all disciplines at the Lab
 - Significant investment by the IT division
 - Individual PIs buy in compute nodes and storage
 - Computational cycles are shared among all lawrencium users
- Lawrencium Compute Nodes
 - data center is housed in the building 50B
 - Over 2333 compute nodes (~58216 CPU cores)
 - 230 GPU cards
 - 32 partitions, lr3, lr4, lr5,lr6, es1, cm1 ...
 - 1492 user accounts
 - 382 projects/groups

Conceptual Diagram of Lawrencium

[Detailed Information of Lawrencium](#)



Access to Lawrencium Cluster

Getting Project Accounts

- Three types of project accounts can be requested.
 1. **Primary Investigator (PI) Computing Allowance (PCA) account:** free 300K service units (SUs) per year (pc_xxx)
 2. **Condo account:** PIs buy in compute nodes to be added to the general pool, in exchange for their own priority access and share the Lawrencium infrastructure (lr_xxx)
 3. **Recharge account:** pay as you go with minimal recharge rate ~ \$0.01/SU (ac_xxx)
- Check out more details here. [Project Accounts](#)
- The form based request are are now moved to [MyLRC portal](#)
- LBL affiliated PI can request project accout at [MyLRC portal](#)
- PIs can grant PCA/Condo/Recharge projects access to researchers/students and external collaborators.

Access to Lawrencium Cluster

Getting User Accounts

- PIs can sponsor researchers/students and external collaborators for cluster accounts.
- Account requests and approval will be done through [MyLRC portal](#).
 - Account creation request
 - PI approval
 - Account creation
 - Users are notified upon account availability and OTP setup.
- Please check out

Login to Lawrencium Cluster

- Linux: Terminal (command-line) session.
- Mac: Terminal (see Applications -> Utilities -> Terminal).
- Windows: PowerShell, or [PuTTY](#) or [MobaXterm](#).
- One-time passwords (OTPs): set up Google Authenticator app on your smartphone or tablet [Instructions Here](#)
- Login:

```
ssh $USER@lrc-login.lbl.gov  
password:
```

- Password: your 4-digit PIN followed by 6-digit one-time password from your Google Authenticator. Example PIN: 0123 OTP: 456789.

```
password:0123456789
```

Note: No characters will appear on the screen in the password prompt when you enter in the digits.

Login to Lawrencium Cluster

Upon login to Lawrencium; you'll end up on one of the login nodes(n000[0-3].scs00) in your home directory.

```
sponsi@n0000 ~]$ hostname  
n0000.scs00  
[sponsi@n0000 ~]$ pwd  
/global/home/users/sponsi  
[sponsi@n0000 ~]$
```

DO NOT run jobs on login nodes!!

User Spaces

- Home: `/global/home/users/$USER/` 20GB per user, data is backed up, recommended for keeping scripts and final results data
- Global Scratch: `/global/scratch/$USER/`, shared, no backup, high performance Lustre parallel filesystem, recommended for keeping non-persistent data for computation.
- Shared group project space
 - `/global/home/groups-sw/` 200GB backup
 - `/global/home/group/` 400GB no backup
- Condo Storage:
 - e.g. `/clusterfs/etna/` or `/global/scratch/projects/xxx`

Data Transfer

lrc-xfer.lbl.gov: Data Transfer Node (DTN)

- On Linux: scp/rsync

```
# Transfer data from a local machine to Lawrencium
scp file-xxx $USER@lrc-xfer.lbl.gov:/global/home/users/$USER
scp -r dir-xxx $USER@lrc-xfer.lbl.gov:/global/scratch/users/$USER

# Transfer from Lawrencium to a local machine
scp $USER@lrc-xfer.lbl.gov:/global/scratch/users/$USER/file-xxx ~/Desktop

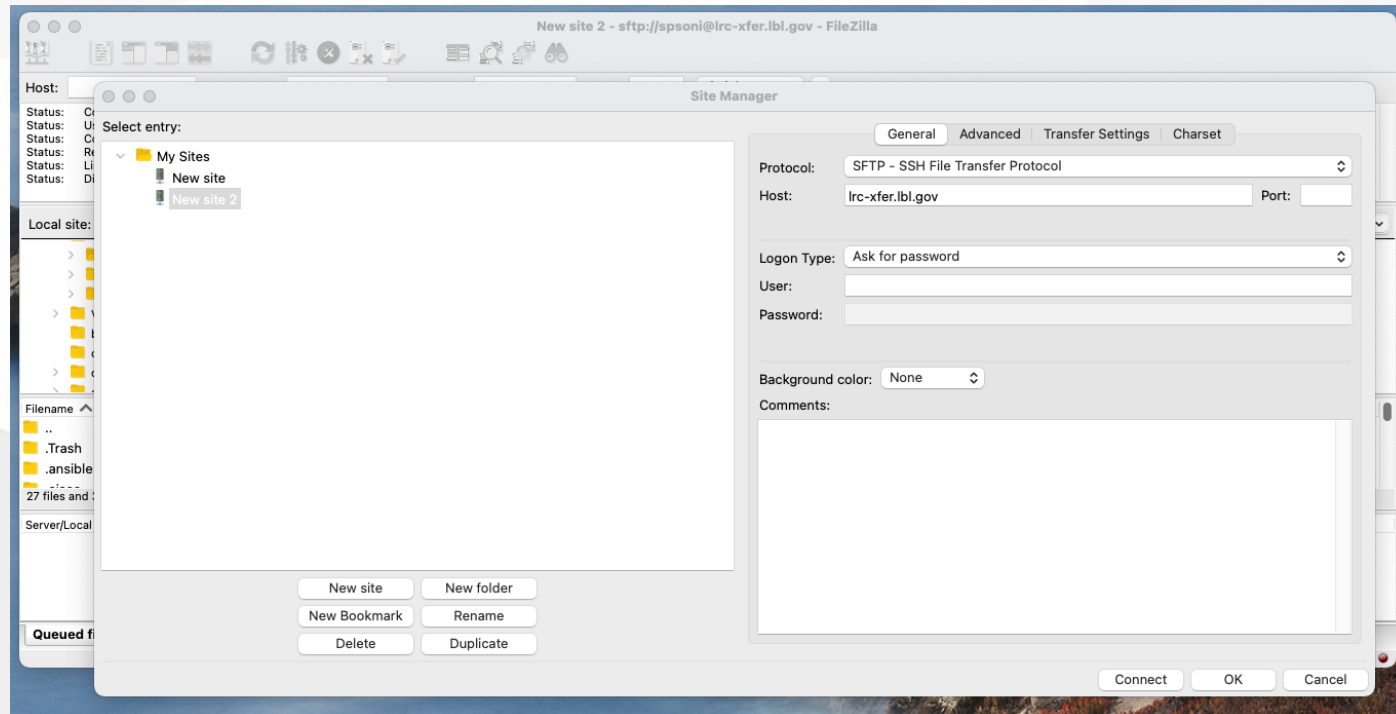
# Transfer from Lawrencium to Another Institute
ssh $USER@lrc-xfer.lbl.gov # DTN
scp -r $USER@lrc-xfer.lbl.gov:/file-on-lawrencium $USER@other-institute:/destination/path/$USER

rsync: a better data transfer or a backup tool
rsync -avpz file-at-local $USER@lrc-xfer.lbl.gov:/global/home/users/$USER
```

- On Window
 - [WinSCP](#): SFTP client and FTP client for Microsoft Windows
 - [FileZilla](#): multi-platform program via SFTP

FileZilla

Site Mager (On top right corner)



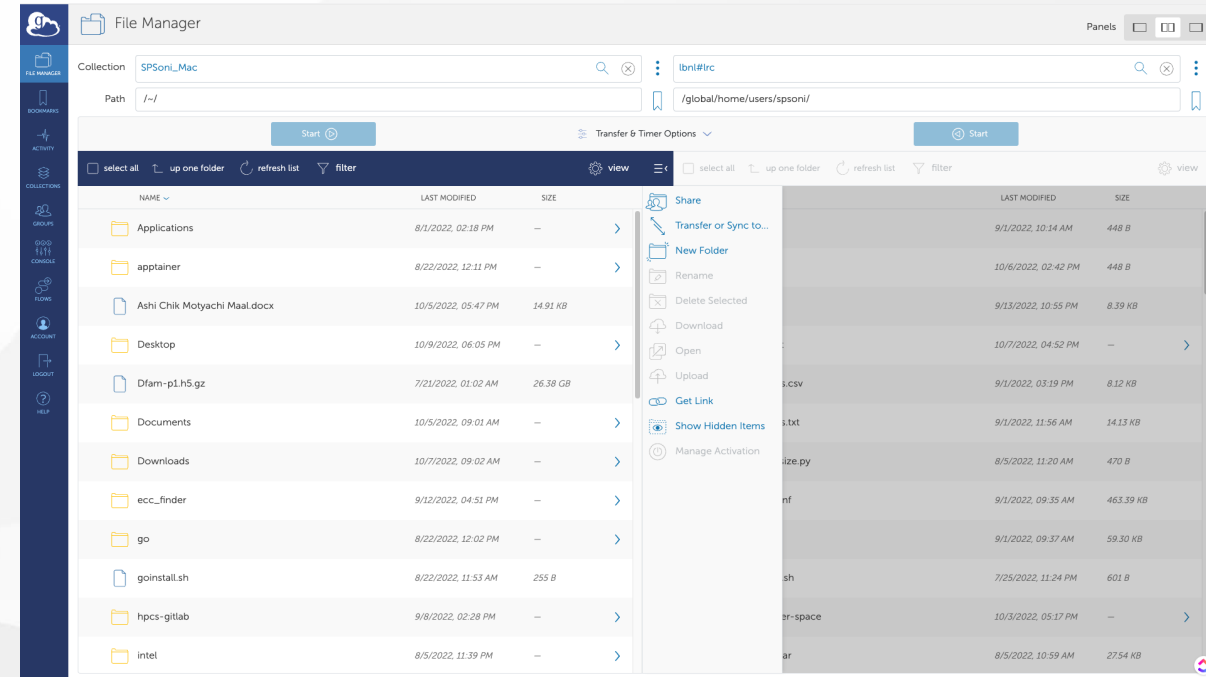
Click on connect to connect Lawrenceium for file transfer.

Enter your username and password(same LRC credentials) in the pop-up window.

Once the connection is established you are ready to drag and drop files to/from Lawrenceium.

Data Transfer with Globus

- Globus can be used for fast data transfer and sharing with collaborators: Click for [Instructions](#)
- Berkeley Lab users can use Globus to transfer files in/out of their LBNL Google drive. Click for [Instructions](#)
- Possible endpoints include: lbnl#lrc, ucb#brc, your laptop/desktop, NERSC.
- Transfer data to/from your laptop (endpoint setup)
 - Create an endpoint on your machine using Globus Connect Personal [globus-connect-personal](#)
 - Run Globus Connect Personal on your local machine



Software Module Farm

- Software stack, commonly used compiler, software tools provided to all users
- Installed and maintained on a centralized storage device and mounted as read-only NFS file system
 - Compilers: e.g. intel, gcc, MPI compilers, Python
 - Tools: e.g. matlab, singularity, cuda
 - Applications: e.g. machine learning, QChem, MD, cp2k
 - Libraries: e.g. fftw, lapack

```
[spsoni@n0003 spsoni]$ module avail
----- /global/software/sl-7.x86_64/modfiles/langs -----
clang/11                gcc/7.5.0                perl/5.36.0
clang/3.9.1             gcc/9.2.0                python/2.7
...

---- /global/software/sl-7.x86_64/modfiles/tools ----
cmake/3.7.2  gnuplot/5.0.5  octave/4.2.0  matlab/r2017b(default)  ...

---- /global/software/sl-7.x86_64/modfiles/apps ----
bio/blast/2.6.0  math/octave/current  ml/tensorflow/2.5.0-py37  ...
...
```

Environment Modules

- Manages users' software environment by dynamically setting up \$PATH, \$LD_LIBRARY_PATH...
- Avoid clashes between incompatible software versions

```
module purge: clear user's work environment
module available: check available software packages
module load packagename/version: load a package
module list: check currently loaded software
```

- Modules are arranged in a hierarchical fashion, some of the modules become available only after the parent module(s) are loaded
- e.g., MKL, FFT, and HDF5/NetCDF software is nested within the gcc module
- Example: load an OpenMPI package

```
module available openmpi mkl
module load intel/2016.4.072
module av openmpi
module load mkl/2016.4.072 openmpi/3.0.1-intel
```

Environment Modules

- Want to learn more about the Environment Modules? [Click here](#)
- Users are allowed to install software in their home or group space. All group members can access packages installed in group space.
- Users don't have admin rights, but most software can be installed at custom path using `--prefix=/path/to/your/dir`

Installing Python Packages

- Python modules: abundantly available but cannot be installed in the default location without admin rights.
- `pip install --user package_name`
- `export PYTHONPATH`

```
[sponsi@n0000 ~]$ module load python/3.9.12

[sponsi@n0000 ~]$ python3 -m site --user-site
/global/home/users/wfeinstein/.local/lib/python3.7/site-packages

[sponsi@n0000 ~]$ pip install --user ml-python
...
Successfully built ml-python
Installing collected packages: ml-python
Successfully installed ml-python-2.2

[sponsi@n0000 ~]$ export PYTHONPATH=~/.local/lib/python3.7/site-packages:$PYTHONPATH
```


Installing Python Packages

- pip install: `--install-option="--prefix=$HOME/.local" package_name`
- Install from source code: `python setup.py install --home=/home/user/package_dir`
- Create a virtual environment: `python -m venv my_env`
 - Activate environment: `source my_env/bin/activate`
 - Install packages into it: `python -m pip install <package-name>`
 - Deactivate environment: `deactivate`
- Conda environment: `conda create -p /global/scratch/users/spsoni/my_env numpy=1.21`
 - Activate environment: `source activate /global/scratch/users/spsoni/my_env`
 - Deactivate environment: `conda deactivate`

SLURM: Resource Manager & Job Scheduler

SLURM is the resource manager and job scheduler for managing all the jobs on the cluster.

Why is this necessary?

- Prevent users' jobs running on the same nodes.
- Allow everyone to fairly share Lawrencium resources.

Basic workflow:

- login to Lawrencium; you'll end up on one of the login nodes in your home directory
- cd to the directory from which you want to submit the job (scratch recommended)
- submit the job using sbatch or an interactive job using srun (discussed later)
- SLURM assign compute node(s) to your jobs
- your jobs will run on a compute node, not the login node

Slurm-Related Environment Variables

- Slurm provides global variables
 - Can be used in your job submission scripts to adapt the resources being requested in order to avoid hard-code
 - Examples of Slurm variables
 - `SLURM_SUBMIT_DIR` : The path of the job submission directory.
 - `SLURM_NTASKS` : The number of tasks. Same as `-n`, `-ntasks`.
 - `SLURM_CPUS_PER_TASK`: Number of CPUs per task.
 - `SLURM_CPUS_ON_NODE`: Number of CPUs on the allocated node.
 - `SLURM_NODELIST`: Contains the definition (list) of the nodes that is assigned to the job.
 - `SLURM_NNODES`: Total number of nodes in the job's resource allocation.
- Recently, slurm on lawrencium is updated to version 22.05.3

Accounts, Partitions, Quality of Service (QOS)

Check slurm association, such as qos, account, partition, the information is required when submitting a job

```
sacctmgr show association user=sponsi -p
```

Lawrencium Cluster Info Click [Here](#)

Job Submission: Interactive Job

Interactive job submission is typically used for code debugging, testing, monitoring.

- `srun`: add your resource request to the queue.
- When the allocation starts, a new bash session will start up on one of the granted nodes
- `srun --account=ac_xxx --nodes=1 --partition=lr5 --qos=lr_normal --time=1:0:0 --pty bash`
- `srun -A ac_xxx -N 1 -p lr5 -q lr_normal -t 1:0:0 --pty bash`

```
[spsoni@n0003 ~]$ srun --account=scs --nodes=1 --partition=lr6 --time=1:0:0 --qos=lr_normal --pty bash
srun: Granted job allocation 28755918
srun: Waiting for resource configuration
srun: Nodes n0101.lr6 are ready for job
[spsoni@n0101 ~]$ squeue -u spsoni
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
28755918	lr6	bash	spsoni	R	0:14	1	n0101.lr6

Once you are on the assigned compute node, start application/commands directly

- `salloc`: similarly to `srun --pty bash`.
- a new bash session will start up on the compute node, `-pty` gives you a pseudo terminal

Node Features

Compute nodes may have different hardware within a SLURM partition, e.g. LR6 - lr6_sky: Intel Skylak, lr6_cas: Intel Cascade Lake, lr6_cas,lr6_m192: lr6_cas + 192GB RAM, lr6_sky,lr6_m192: lr6_sky + 192GB RAM

- Lawrence nodes features can be found [here](#).
- wait time is longer when a specific type of node is requested
- Slurm flag: --constrain

```
[spsoni@n0000 ~]$ srun --account=scs --nodes=1 --partition=lr6 --time=1:0:0 --qos=lr_normal --constrain=lr6_sky --pty bash
[spsoni@n0081 ~]$ free -h
              total        used        free      shared  buff/cache   available
Mem:           93G         2.2G         83G         3.1G         7.4G         87G
Swap:          8.0G           0B         8.0G
[spsoni@n0081 ~]$ exit
exit
[spsoni@n0000 ~]$ srun --account=scs --nodes=1 --partition=lr6 --time=1:0:0 --qos=lr_normal --constrain=lr6_sky,lr6_m192 --pty bash
[spsoni@n0023 ~]$ free -h
              total        used        free      shared  buff/cache   available
Mem:          187G         2.6G        172G         1.7G         12G        182G
Swap:          8.0G         1.5G         6.5G
```

Memory Specification

- Most Lawrencium partitions are exclusive: a compute node allows only one user
- Some condo accounts or partitions, such as ES1 (GPUs), each compute node can be shared by multiple users
- Slurm flag: `--mem` (MB) is required when using a shared partition:
- e.g. a compute node with 96GB RAM, 40 core node: 2300 RAM/core
 - `--ntask=1 --mem=2300` (request one core)
 - `--ntask=2 --mem=4600` (request 2 cores)
- LR6 partition `lr_bigmem`: two large memory nodes (1.5TB)
- Slurm flag: `--partition=lr_bigmem`

Submit a Batch Job

- Get help with the complete command options `sbatch --help`
- sbatch: submit a job to the batch queue system `sbatch myjob.sh`

```
#!/bin/bash
# Job name:
#SBATCH --job-name=mytest
# Partition:
#SBATCH --partition=lr6
# Account:
#SBATCH --account=pc_test
# qos:
#SBATCH --qos=lr_normal
# Wall clock time:
#SBATCH --time=1:00:00
# Node count
#SBATCH --nodes=1
#SBATCH --constrain=lr6_cas
#SBATCH --mail-user=xxx@lbl.gov
##SBATCH --mail-type=BEGIN/END/FAIL
#SBATCH --mail-type=ALL
# cd to your work directory
cd /your/dir
## Commands to run
module load python/3.9.12
python my.py >& mpy.out
```


Submit Jobs to ES1 GPU Partition

- `--gres=gpu:type:GPU#`
- `--ntasks=CPU_CORE#`
- ratio CPU_CORE#:GPU# = 2:1

```
[spsoni@n0000 ~]$ srun -A scs -N 1 -p es1 --gres=gpu:1 --ntasks=2 -q es_normal -t 0:30:0 --pty bash
```

```
[spsoni@n0022 ~]$ nvidia-smi
```

```
Mon Oct 10 16:17:49 2022
```

```
+-----+
| NVIDIA-SMI 460.84          Driver Version: 460.84          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...    Off      | 00000000:62:00:0 Off |                    0 |
| N/A   44C    P0      52W / 300W | 0MiB / 16160MiB |      0%    Default  |
|                                           N/A               |
+-----+-----+
|   1   Tesla V100-SXM2...    Off      | 00000000:89:00:0 Off |                    0 |
| N/A   43C    P0      54W / 300W | 0MiB / 16160MiB |      0%    Default  |
|                                           N/A               |
+-----+-----+
+-----+
| Processes:                                                       GPU Memory |
|  GPU   GI    CI          PID    Type   Process name                  Usage      |
|  ID     ID                                 |             |
+-----+-----+
| No running processes found                                     |
+-----+
```

- Specify GPU type
 - GRTX2080TI: --gres=gpu:GRTX2080TI:1
 - V100: --gres=gpu:V100:1
 - A40: --gres=gpu:A40:1

```
[spsoni@n0000 ~]$ srun -A scs -N 1 -p es1 --gres=gpu:V100:2 --ntasks=4 -q es_normal -t 0:30:0 --pty bash
```

```
[spsoni@n0016 ~]$ nvidia-smi -L
```

```
GPU 0: Tesla V100-SXM2-16GB (UUID: GPU-7979861e-e0ad-000f-95fb-371e34593991)
```

```
GPU 1: Tesla V100-SXM2-16GB (UUID: GPU-50d24ac9-9eea-f96b-cc8b-db849f9c9427)
```

```
[spsoni@n0016 ~]$ echo $CUDA_VISIBLE_DEVICES
```

```
0,1
```

Submit A GPU Batch Job

Job Submission Script Example

```
#!/bin/bash -l

#SBATCH --job-name=mytest
#SBATCH --partition=es1          ## es1 GPU partition
#SBATCH --account=pc_test
#SBATCH --qos=es_normal         ## qos of es1
#SBATCH --time=1:00:00
#SBATCH --nodes=1
#SBATCH --gres=gpu:V100:2       ## GPUs
#SBATCH --ntasks=4              ## CPU cores
#
cd /your/dir

## Commands to run
module load python/3.7
python my.py >& mpy.out
```

Submitting MPI Jobs

When using multiple nodes, you need to carefully specify the resources. The key flags to use in your job script are:

- `--nodes` (or `-N`): number of nodes
- `--ntasks-per-node`: number of tasks (i.e., processes) to run on each node, especially useful when your job uses large memory, $< \text{Max Core\#}$ on a node
- `--ntasks` (or `-n`): total number of tasks and let the scheduler determine how many nodes and tasks per node are needed.
- `--cpus-per-task` : number of cpus to be used for each task

Submitting MPI Jobs

Job submission script

```
#!/bin/bash -l

#SBATCH --job-name=myMPI
#SBATCH --partition=lr6
#SBATCH --account=scs
#SBATCH --qos=lr_normal
#SBATCH --time=2:00:00
#SBATCH --nodes=2                ## Nodes count
##SBATCH --ntasks=40            ## Number of total MPI tasks to launch (example):
##SBATCH --ntasks-per-node=20   ## important with large memory requirement

cd /your/dir

## Commands to run
module load gcc/11.3.0 openmpi/4.1.4-gcc
mpirun -np 40 ./my_mpi_exe      ## Launch your MPI application
```

Submit Serial Tasks in Parallel (GNU Parallel)

GNU Parallel is a shell tool for executing jobs in parallel on one or multiple computers.

- A job can be a single core serial task, multi-core or MPI application.
- A job can also be a command that reads from a pipe.
- Typical input:
 - bash script for a single task
 - a list of tasks with parameters
- Example Using GNU Parallel

Bioinformatics tool *blastp* to compare 200 target protein sequences against sequence DB
Serial bash script: [run-blast.sh](#)

```
#!/bin/bash
module load bio/blast/2.6.0
blastp -query $1 -db ../blast/db/img_v400_PROT.00 -out $2 -outfmt 7 -max_target_seqs 10 -num_threads 1
```

task.lst: each line provides one parameter to one task:

```
[spsoni@n0002 ]$ cat task.lst
../blast/data/protein1.faa
../blast/data/protein2.faa
...
../blast/data/protein200.faa
```

Instead submit single core-jobs 200 times, which potentially need 200 nodes, GNU parallel sends single-core jobs in parallel using all the cores available, e.g. 2 compute nodes $32 \times 2 = 64$ parallel tasks. Once a CPU core becomes available, another job will be sent to this resource.

```
module load parallel/20200222
JOBS_PER_NODE=32
parallel --jobs $JOBS_PER_NODE --slf hostfile --wd $WDIR --joblog task.log --resume --progress \
-a task.lst sh run-blast.sh {} output/{/.}.blst
```

Detailed information of how to submit serial tasks in parallel with [GNU Parallel](#)

Monitoring Jobs

- **sinfo**: check node status of a partition (idle, allocated, drain, down)

```
[spsoni@n0000 ~]$ sinfo -r -p lr5
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
lr5        up    infinite    3  drain* n0004.lr5,n0032.lr5,n0169.lr5
lr5        up    infinite   14   down n0048.lr5,n0050.lr5
lr5        up    infinite   58  alloc n0000.lr5,n0001.lr5,n0002.lr5,n0003.lr5,n0006.lr5,n0009.lr5
lr5        up    infinite  115   idle n0005.lr5,n0007.lr5,n0008.lr5
...
```

- **squeue**: check job status in the batch queuing system (R or PD)

```
squeue -u $USER
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      28757187      lr6    bash  spsoni   R        0:09        1 n0215.lr6
      28757723      es1    bash  spsoni   R        0:16        1 n0002.es1
      28759191      lr6    bash  spsoni  PD        0:00       120 (QOSMaxNodePerJobLimit)
```


- **sacct**: check job information or history

```
[spsoni@n0002 ~]$ sacct -j 28757723
```

JobID	JobName	Partition	Account	AllocCPUS	State	ExitCode
28757723	bash	es1	scs	2	RUNNING	0:0

```
[spsoni@n0002 ~]$ sacct -X -o 'jobid,user,partition,nodelist,stat'
```

JobID	User	Partition	NodeList	State
28755594	spsoni	lr5	n0192.lr5	COMPLETED
28755597	spsoni	lr6	n0101.lr6	COMPLETED
28755598	spsoni	lr5	n0192.lr5	COMPLETED
28755604	spsoni	csd_lr6_s+	n0144.lr6	COMPLETED
28755693	spsoni	lr6	n0101.lr6	CANCELLED+
....				
28757187	spsoni	lr6	n0215.lr6	COMPLETED
28757386	spsoni	es1	n0019.es1	FAILED
28757389	spsoni	es1	n0002.es1	TIMEOUT
28757723	spsoni	es1	n0002.es1	RUNNING

- `wwall -j <JOB_ID>` : check resource utilization of an active job from a login node

```
[spsoni@n0000 ~]$ wwall -j 28757187
```

```
-----
Total CPU utilization: 0%
```

```
    Total Nodes: 1
```

```
        Living: 1
```

```
    Unavailable: 0
```

```
        Disabled: 0
```

```
        Error: 0
```

```
        Dead: 0
```

```
Warewulf
```

```
Cluster Statistics
```

```
http://warewulf.lbl.gov/
```

```
-----
Node      Cluster      CPU      Memory (MB)      Swap (MB)      Current
Name      Name      [util/num] [% used/total] [% used/total] Status
n0215.lr6      0%   (40) % 3473/192058 % 1655/8191  READY
```

- `scancel <jobID>` : scancels a job

More Information of [Slurm Usage](#)

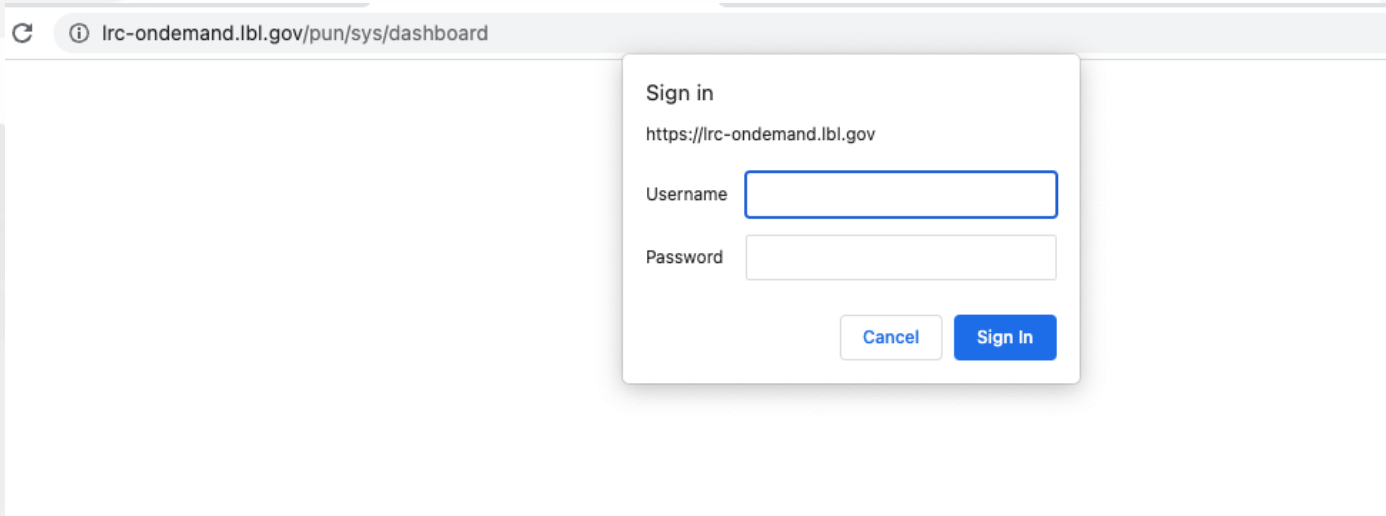
Open OnDemand

- OpenOnDemand is a web platform that provides an easy access to the cluster's HPC resources and services.
- Designed and developed by Ohio Supercomputer Center.
- Intuitive and easy access to computing resources, alternative and convenient way to traditional command line access
- Allow access to Lawrencium compute resources
 - File browser: file editing, data transfer
 - Shell command line access - terminal
 - Job monitoring
- Interactive applications: Jupyter Server, RStudio Server, MATLAB, Desktop
- Sever: <https://lrc-ondemand.lbl.gov/>
 - Intel Xeon Gold processor with 32 cores, 96 GB RAM

Accessing OOD on Lawrencium

1. Web link to connect : <https://lrc-ondemand.lbl.gov/>

Note: Use Chrome or Firefox to browse this page. Safari has known [authentication issues](#).

A screenshot of a web browser window showing the login page for the LRC On-Demand dashboard. The browser's address bar displays "lrc-ondemand.lbl.gov/pun/sys/dashboard". A "Sign in" dialog box is centered on the screen. The dialog box has a title "Sign in" and a URL "https://lrc-ondemand.lbl.gov". It contains two input fields: "Username" and "Password". Below the input fields are two buttons: "Cancel" and "Sign In".

Sign in

https://lrc-ondemand.lbl.gov

Username

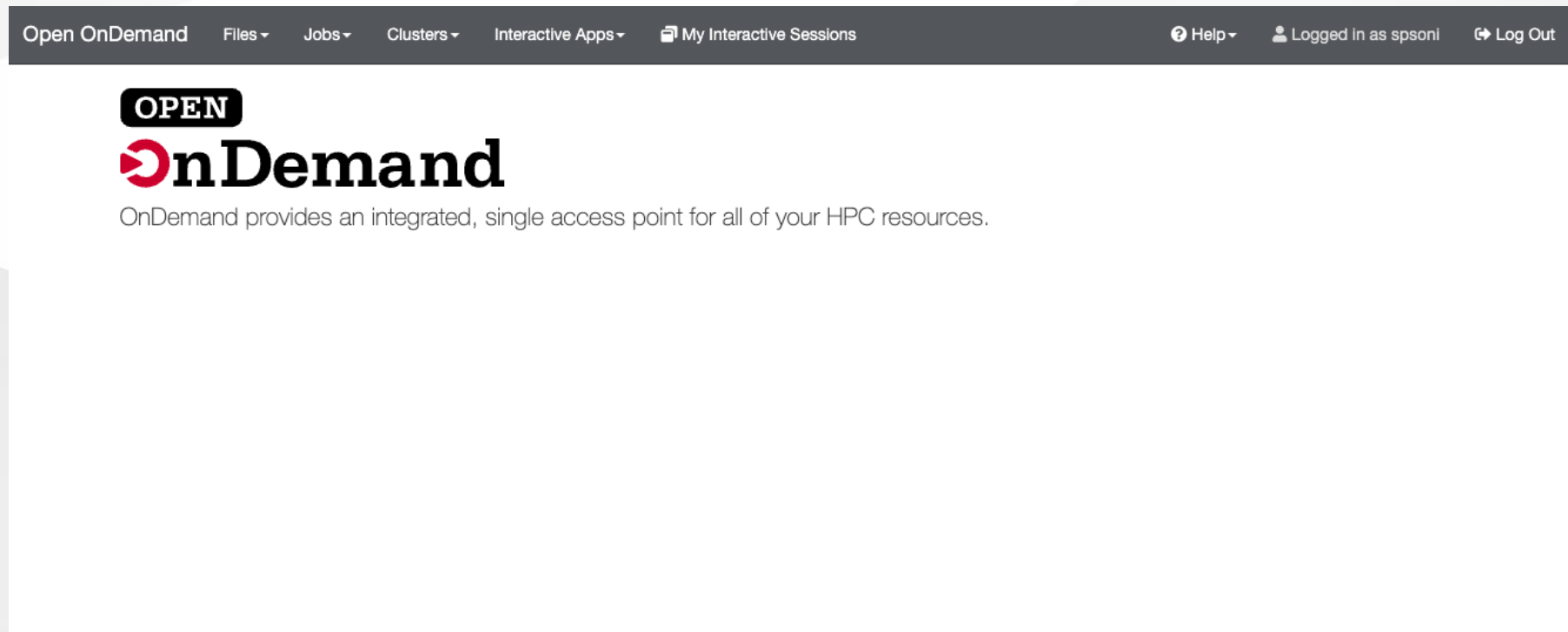
Password

Cancel Sign In

2. Use your LRC username and PIN+one-time password (OTP)
- same credentials you use to login Lawrencium cluster

OOD Dashboard on Lawrencium

On successful authentication you will see a OOD dashboard.



Lets do quick demo!

Detailed training materials can be found on [github](#).

Getting help

- Virtual office hours:
 - Time: 10.30 am to noon every Wednesday
 - Online [request](#)
- Send us tickets at hpcshelp@lbl.gov
- More information about LBNL Supercluster and scientific computing services can be found [here](#).
- Looking for more trainings? Upcoming training are regularly announced [here](#). Other than LBNL HPC trainings you can also access [DLab](#) courses.

Your feedback is important to us for improving HPC services and training.
Please fill out [training survey](#)

