Lawrencium 101 : HPC on Lawrencium Supercluster

Sapana Soni

HPCS User Support Team

Outline

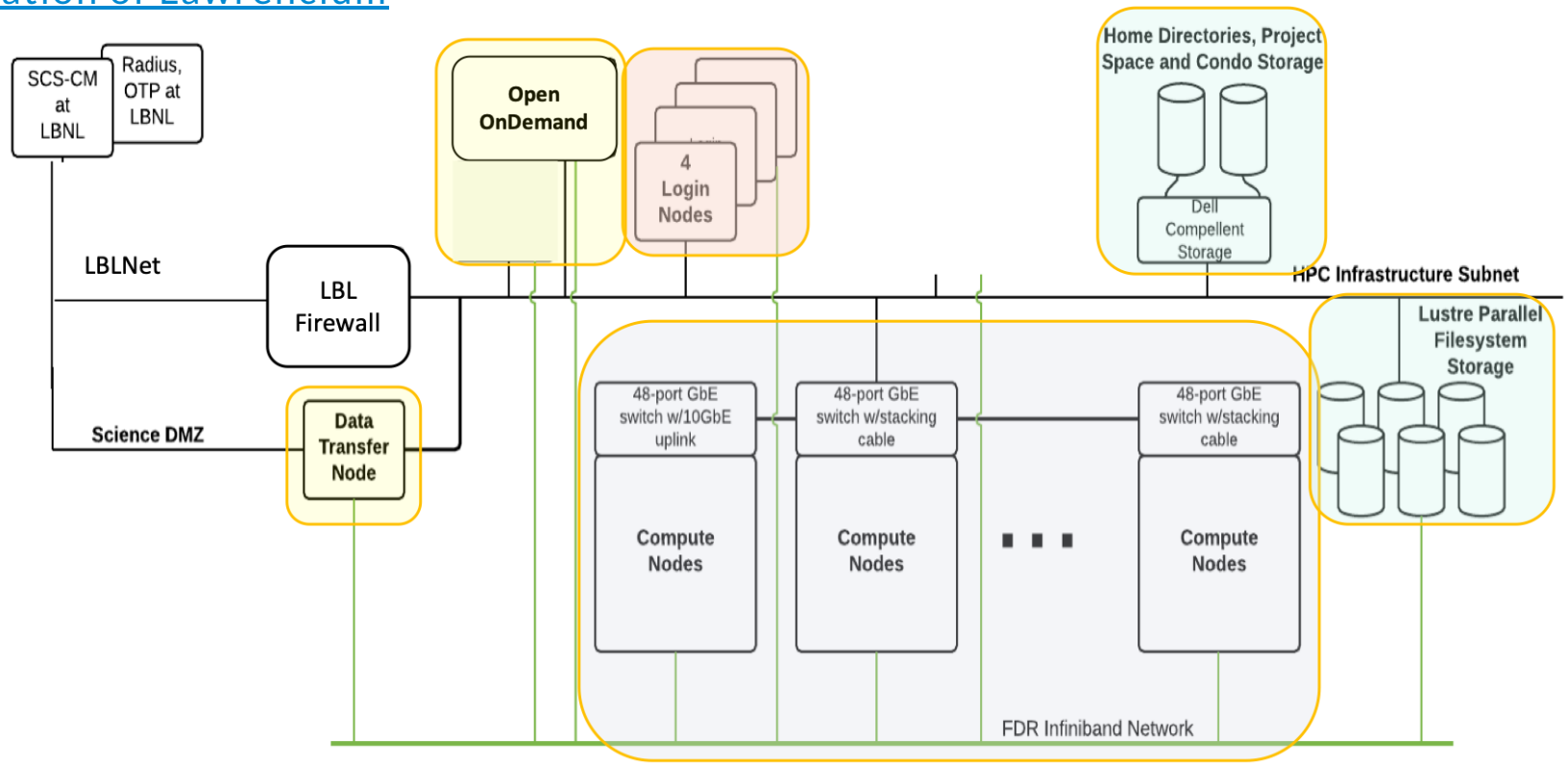
1. Lawrencium supercluster Overview
2. Access and login to Lawrencium
3. Software access and installation
4. Job submission and monitoring
5. Data transfer to/from clusters
6. Open On Demand: a browser based HPCS portal

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 now moved to MyLRC portal
- LBL affiliated PI can request project accout at MyLRC portal
- PIs can grant PCA/condo/recharge project 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.
- Check out documentation.

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: Characters won't appear on a 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.

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

DO NOT run jobs on login nodes!!

User Spaces

- Home: `/global/home/users/$USER/` 20GB per user, data is backed up, recommended for keeping scripts and final results data
- Global Scratch: `/global/scratch//users/$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)

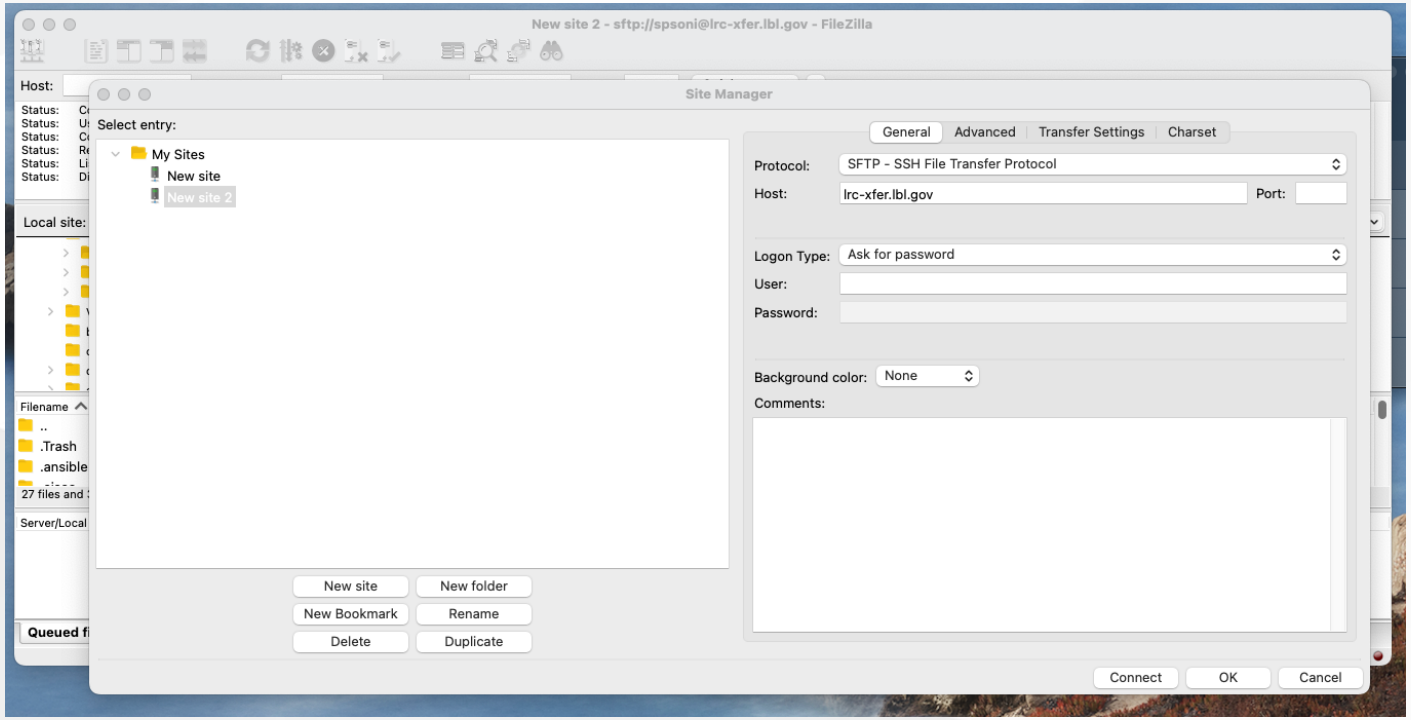
- On Linux: scp/rsync

```
# 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

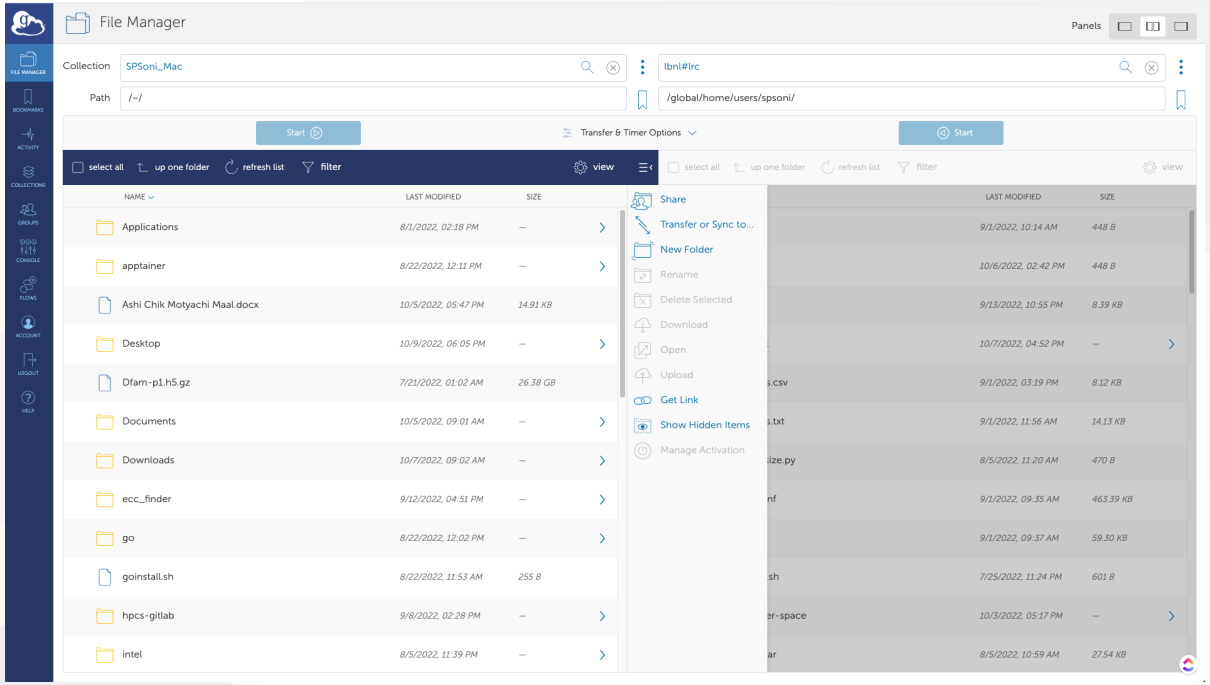
Site Manager (on the top right corner)



Click on connect to access Lawrencium for file transfer.
Enter your username and password(same LRC credentials) in the pop-up window.
Once the coneection is estblished, you are ready to drag and drop files to/from lawrencium.

Data Transfer with Globus

- Globus can be used for fast data transfer and sharing with collaborators. Connect to globus https://globus.lbl.gov
- Globus for Lawrencium
- Berkeley Lab users can use Globus to transfer files in/out of their LBNL Google drive.
- 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 Pesonal on your local machine



Software Module Farm

- Software stack, commonly used compiler, software tools are 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/apptainer, 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/11.3.0                 python/3.9.12
 ...

---- /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 memebers 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`

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

[spsoni@n0000 ~]$ python3 -m site --user-site
/global/home/users/spsoni/.local/lib/python3.9/site-packages
[spsoni@n0000 ~]$ pip install --user ml-python
...
Successfully built ml-python
Installing collected packages: ml-python
Successfully installed ml-python-2.2

[spsoni@n0000 ~]$ export PYTHONPATH=~/.local/lib/python3.9/site-packages/$PYTHONPATH
```

Installing Python Packages

- pip install: `--install-option="--prefix=$HOME/.local" package_name`
- Install from source code: `python setup.py install –-prefix=/home/user/package_dir`
- Create a virtual environmemt: `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 environemnet: `conda create -p /global/scratch/users/spsoni/my_env <package-name>`
  - 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 recomended)
- 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-coding
- 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.

Accounts, Partitions, Quality of Service (QOS)

Check slurm association, such as qos, account, partition, the information using following command.

```
sacctmgr show association user=spsoni -p

Cluster|Account|User|Partition|Share|Priority|GrpJobs|GrpTRES|GrpSubmit|GrpWall|GrpTRESMins|MaxJobs|MaxTRES|MaxTRESPerNode|MaxSubmit|MaxWall|MaxTRESMins|QOS|Def QOS|GrpTRESRunMins|
perceus-00|pc_scs|spsoni|ood_inter|1|||||||||||lr_interactive|||
perceus-00|pc_scs|spsoni|lr6|1|||||||||||lr_debug,lr_lowprio,lr_normal|||
perceus-00|ac_test|spsoni|lr5|1|||||||||||lr_debug,lr_lowprio,lr_normal|||
perceus-00|pc_test|spsoni|lr4|1|||||||||||lr_debug,lr_lowprio,lr_normal|||
perceus-00|pc_test|spsoni|lr_bigmem|1|||||||||||lr_debug,lr_lowprio,lr_normal|||
perceus-00|lr_test|spsnoni|lr3|1|||||||||||condo_test|||
perceus-00|scs|spsnoni|es1|1|||||||||||es_debug,es_lowprio,es_normal|||
...
```

This information is required when submitting a job.
For Lawrencium cluster information 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

- Lawrencium 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
```

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`

myjob.sh : Job submission script for serial job

```
#!/bin/bash
#SBATCH --job-name=mytest   # Job name
#SBATCH --partition=lr6     # Partition
#SBATCH --account=pc_test   # Account, replace it with your own account allocation
#SBATCH --qos=lr_normal     # qos
#SBATCH --time=1:00:00      # Wall clock time
#SBATCH --nodes=1           # Node count
#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 >& mypy.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   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

- Specify GPU type : --gres=gpu[*type*]:*count*
  - GRTX2080TI(GPU count = 3 or 4): --gres=gpu:GRTX2080TI:1
  - V100(GPU count = 2): --gres=gpu:V100:1
  - A40(GPU count = 4): --gres=gpu:A40:1
- In above example only one gpu is used but count can be set to total number of GPUs at the max.

```
[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

myjob_gpu.sh: Job Submission Script for GPUs

```
#!/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
#Number of GPUs, this can be in the format of "gpu:[1-4]", or "gpu:V100:[1-2] with the type included

## Commands to run
module load ml/tensorflow/2.5.0-py37
python tf.py >& tf.out
```

Submiting 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, < 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

Submiting MPI Jobs

myjob_mpi.sh: Job submission script

```
#!/bin/bash
#SBATCH --job-name=test_mpi
#SBATCH --account=pc_test
#SBATCH --partition=lr6
#SBATCH --qos=lr_debug
#SBATCH --ntasks=40 # Number of MPI tasks needed for use case (example):
#SBATCH --nodes=2   # Nodes count
##SBATCH --ntasks-per-node=20    ## important with large memory requirement

# Wall clock limit:
#SBATCH --time=00:01:30
## Command(s) to run (example):
module load gcc/11.3.0 openmpi/4.1.4-gcc
srun ./hello >& hello.out
```

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 RNA query sequences against sequence DB
  Serial bash script: **run-blast.sh**

```
[spsoni@n0003 gnu_parallel]$ cat run-blast.sh
#!/bin/bash
module load  bio/blast/2.13.0
module load parallel/20200222
blastn -db database/16S_ribosomal_RNA -query $1 -out $2 -task blastn -dust no -outfmt "7 \
delim=, qacc sacc evalue bitscore qcovus pident" -max_target_seqs 5
```

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

```
[spsoni@n0003 gnu_parallel]$ cat task.lst
data/query1.fa
data/query2.fa
.
.
 data/query200.fa
```

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*2=64 parallel tasks. Once a CPU core becomes available, another job will be sent to this resource.

myjob_gnuparallel.sh: Slurm script for gnu-parallel job submission on lawrencium

```
#!/bin/bash
#SBATCH --job-name=test_gnupar
#SBATCH --account=pc_test
#SBATCH --partition=lr6
#SBATCH --qos=lr_normal
#SBATCH --nodes=2
#SBATCH --time=2:00:00

## Command(s) to run (example):
module load bio/blast/2.13.0
module load parallel/20200222

export WORKDIR=/global/scratch/users/spsoni/LRC101/gnu_parallel
cd $WORKDIR

echo $SLURM_JOB_NODELIST |sed s/\,/\\n/g |awk -v cores=$SLURM_CPUS_ON_NODE '{print cores"/"$1}' > hostfile
mkdir -p output logs
export JOBS_PER_NODE=32
parallel --jobs $JOBS_PER_NODE --slf hostfile --wd $WORKDIR --joblog task.log  --progress -a task.lst sh run-blast.sh {} output/{/.}.blst
```

Host file will be generated at run time by the above script.
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 resouce utilization of an active job from a login node

```
[spsoni@n0000 ~]$ wwall -j 28757187
--------------------------------------------------------------------------------
Total CPU utilization: 0%
          Total Nodes: 1
               Living: 1                            Warewulf
          Unavailable: 0                       Cluster Statistics
             Disabled: 0                   http://warewulf.lbl.gov/
                Error: 0
                 Dead: 0
--------------------------------------------------------------------------------
 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

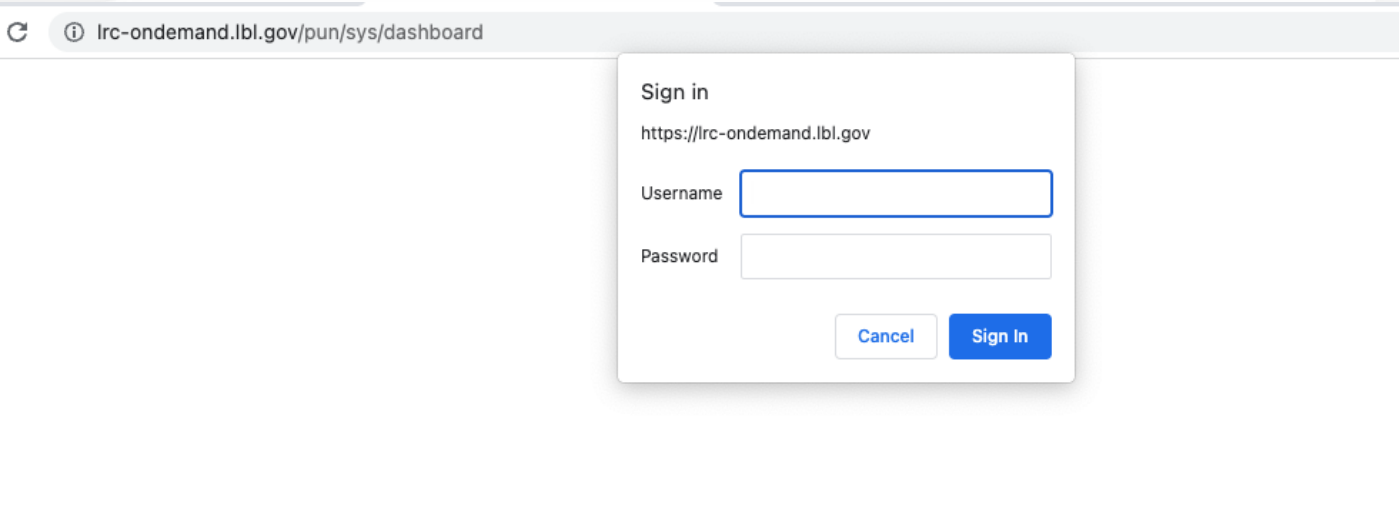
More Information of [Slurm Usage]

Open OnDemand

- OpenOnDemand is a web platform that provides an easy access to the cluster's HPC resourses and services.
- Designed and developed by Ohio Supercomputer Center.
- Intuitive and easy access to computing resourses, 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

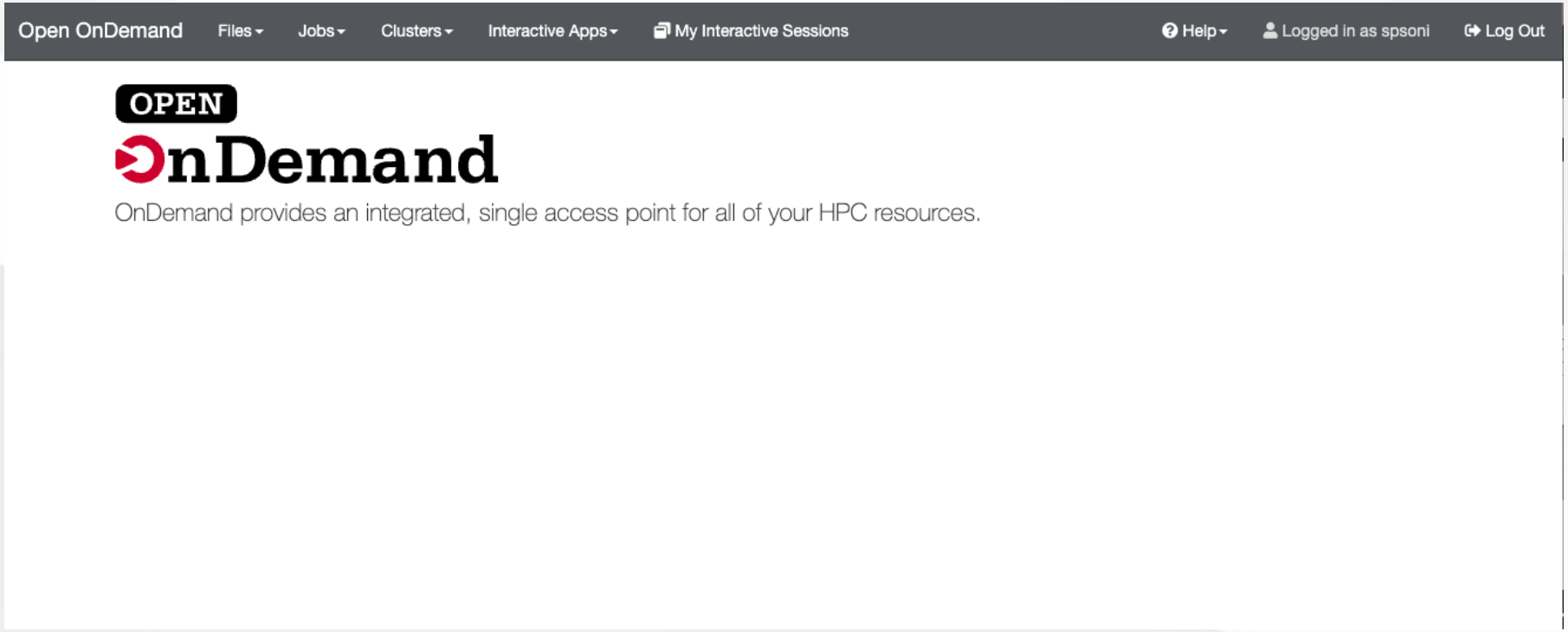
1. Web link to connect : https://lrc-ondemand.lbl.gov/
   Note: Use Chrome or Firefox to browse this page. Safari has known authentication issues.



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 scientic 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