Running at NERSC

Getting An Account

Note: this should only be necessary for expert developers. Send mail to pcds-ana-l if you have questions.

Apply for an account at this link:

http://www.nersc.gov/users/accounts/user-accounts/get-a-nersc-account/

Use the following information for the various fields:

- Principal Investigator and Repository Name: "Amedeo Perazzo" (this should automatically select the "repo" to be "LCLS")
- Organization: "USA: Stanford Linear Accelerator Center"

Useful NERSC Links

(courtesy of Anton Barty)

- running shifter: https://docs.nersc.gov/development/shifter/how-to-use/
- The ‘module add’ command. Explains why stuff is not there by default, and how to get it easily. https://docs.nersc.gov/environment/#module-faq
- Queues https://docs.nersc.gov/jobs/policy/
- X11 can be slow from afar. I did find NX ("nomachine") instructions: https://docs.nersc.gov/connect/nx/

Batch Job Example

Currently this is only possible for "early access users" who have accounts at NERSC.

- data are available at NERSC in this directory (the equivalent of /reg/d/psdm): /global/project/projectdirs/lcls/d/psdm/. Set environment variable SIT_PSDM_DATA to this location so psana will be able to locate the data
- ssh to cori.nersc.gov (the equivalent of a pslogin node)
- information on the cori batch system ("slurm") is here: https://docs.nersc.gov/jobs/
- there are 32 cores on each cori node
- to get to the equivalent of a "psana" node you should run an "interactive job" as described here: https://docs.nersc.gov/jobs/interactive/

Example slurm batch-job script submitted with "sbatch <scriptname>" ("srun" is the cray-equivalent of "mpirun"). These examples can be found at https://github.com/monarin/psana-nersc.git in "psana1/submit.sh" and "psana1/run_nersc.sh":

```
#!/bin/bash -l
#SBATCH --account=lcls
#SBATCH --job-name=lcls-py2-root
#SBATCH --nodes=1
#SBATCH --constraint=knl
#SBATCH --time=00:15:00
#SBATCH --image=docker:slaclcls/lcls-py2-root:latest
#SBATCH --exclusive
#SBATCH --qos=regular
t_start=`date +%s`
export PMI_MMAP_SYNC_WAIT_TIME=600
srun -n 68 -c 4 shifter ./run_nersc.sh
t_end=`date +%s`
echo PSJobCompleted TotalElapsed $((t_end-t_start)) $t_start $t_end
```

Where run_nersc.sh looks like the usual psana-python command:

```
#!/bin/bash
# activate psana environment
source /img/conda.local/env.sh
source activate psana_base

# set location for experiment db and calib dir
export SIT_DATA=$CONDA_PREFIX/data
export SIT_PSDM_DATA=/global/cscratch1/sd/psdatmgr/data/psdm

# prevent crash when running on one core
export HDF5_USE_FILE_LOCKING=FALSE

python mpiDatasource.py
```

Interactive Example

To run a shorter "interactive" session (very useful for debugging since you don't have to wait for a batch job to start after fixing each typo:

```
monarin@cori02: salloc -C knl -N 1 -t 1:00:00 -q interactive -A lcls --image=docker:slaclcls/lcls-py2-root:
latest
salloc: Pending job allocation 32421205
salloc: job 32421205 queued and waiting for resources
salloc: job 32421205 has been allocated resources
salloc: Granted job allocation 32421205
salloc: Waiting for resource configuration
salloc: Nodes nid02346 are ready for job
monarin@nid02346: srun -n 3 shifter ./run.sh
2
0
1

monarin@nid02346: cat run.sh
#!/bin/bash
source /img/conda.local/env.local
source activate psana_base
python test_mpi.py
```

And another approach that gets you a prompt "inside" the shifter container's conda environment:

```
(login to a cori login node, then execute this command which allocates a node for you to use for 1 hour)
salloc -C knl -N 1 -t 1:00:00 -q interactive -A lcls --image=docker:slaclcls/lcls-py2-root:latest

(once that command completes)
shifter /bin/bash  (get a shell in the shifter image)
source /img/conda.local/env.sh   (setup conda)
source activate psana_base  (activate the appropriate conda environment)
export SIT_PSDM_DATA=/global/cscratch1/sd/psdatmgr/data/psdm

(psana_base) cpo@nid02387:~$ more ~/junk.py
from psana import *
dsource = MPIDataSource('exp=mfx11116:run=602:dir=/global/cscratch1/sd/psdatmgr/data/psdm/MFX/mfx11116/xtc:smd')
det = Detector('Jungfrau1M')
for nevt,evt in enumerate(dsource.events()):
   calib = det.calib(evt)
   if calib is None:
      print 'none'
   else:
      print calib.shape
   if nevt>5: break
(psana_base) cpo@nid02387:~$ python junk.py
(2, 512, 1024)
(2, 512, 1024)
(2, 512, 1024)
(2, 512, 1024)
(2, 512, 1024)
(2, 512, 1024)
(2, 512, 1024)
(psana_base) cpo@nid02387:~$
```