SLAC NATIONAL ACCELERATOR LABORATORY

Introduction to SDF and SLURM

Wei Yang

On behalf of the Scientific Computing Services

SLAC ML School Tutorial, Dec, 6-10, 2021

Outline

Intro of the SDF environment

Interactive (SSH or Jupyter) access to SDF

SLURM 101

- Simple job submission with GPU resources
- Best practices

Introduction to the SDF environment

SDF components:

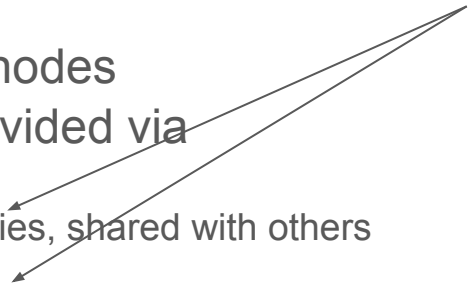
- **Interactive nodes**
  - SSH login nodes
  - Jupyter via web portal (https://sdf.slac.stanford.edu) with documentation
- **SLURM batch systems**
  - HPC cluster and GPU cluster
- Lustre shared file system
  - /sdf
  - There are also a few older GPFS file systems. No AFS, no NFS
- Data transfer nodes
  - sdf-dtn.slac.stanford.edu
  - Globus service: slac#sdf

SDF nodes use SLAC Windows username and password !

Interactive access 1

SSH login nodes:

- ssh username@sdf-login.slac.stanford.edu
- Accessing from anywhere
- Also SLURM job submission nodes
- Software tools are usually provided via
  - Operating systems
  - Modules tools by user communities, shared with others
  - Yourself or your own group



This is a very broad topic.

- SDF provides space and help
- Users compile, test, install and maintain those software tools

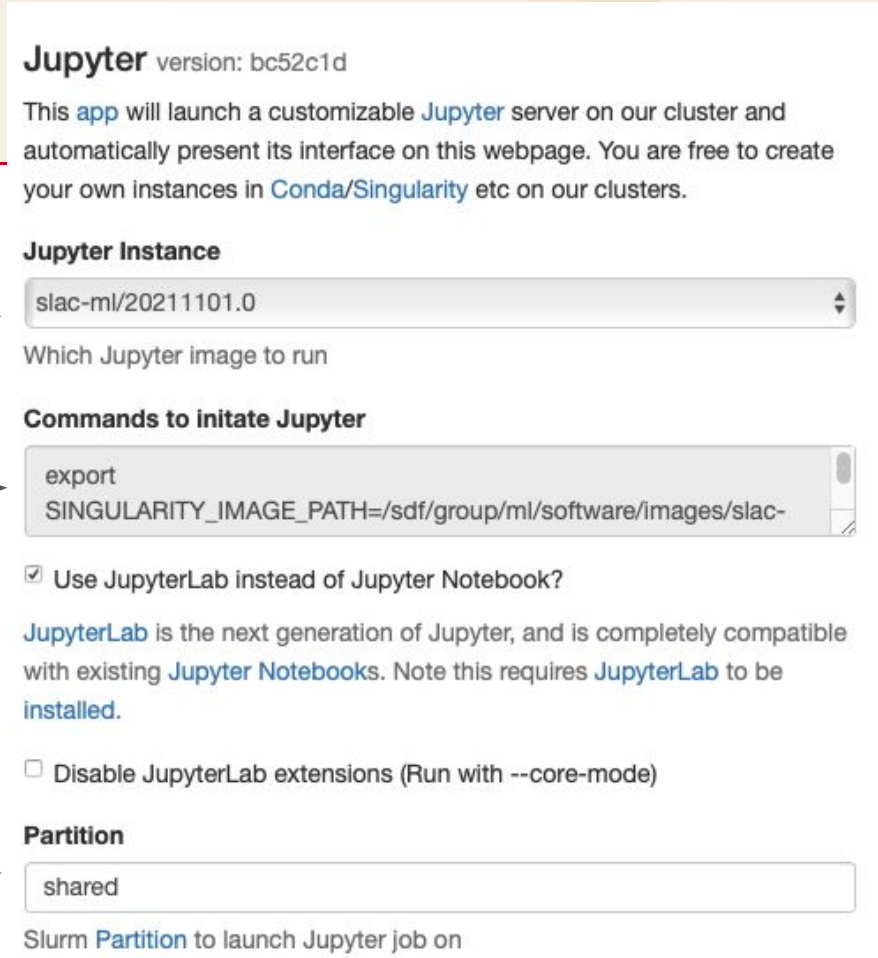
OR:

- https://sdf.slac.stanford.edu and choose “Shell”

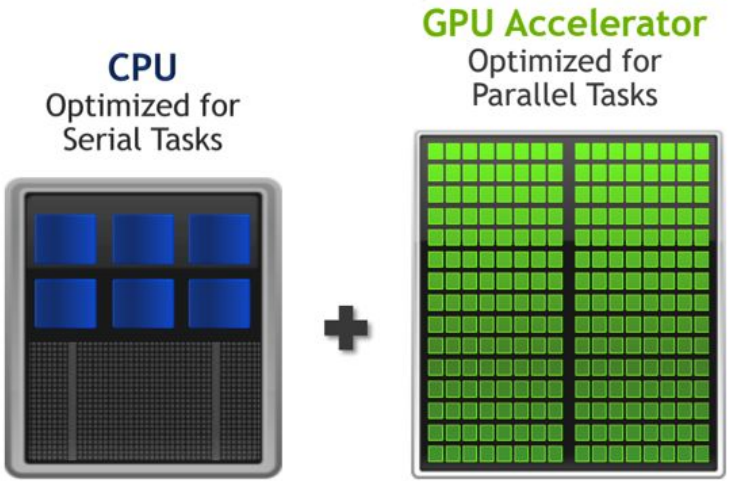
Interactive access 2

Jupyter via web portal

- There are many prebuild Jupyter envs.
- Or you can run your own Jupyter
- Jupyter instances are run as SLURM jobs
  - One can run Jupyter on login nodes
  - But running Jupyter via web portal/SLURM provide a lot more resource



GPU computing



General-Purpose GPU (GPGPU) computing refers to architectures that use GPUs as co-processors, in addition to CPUs.

- CPU has a few fast cores
- GPU has a lot of slower cores
  - Optimized for parallelling identical computing tasks.
    - Such as matrix operations.
    - ML has lots of matrix operations
  - GPU uses non-X86 instruction set
    - Need to compile GPU tasks and load to GPU as “kernels”
- GPU and CPU usually do not share memory
  - Moving data in and out of the GPUs is time consuming.

SLURM 101 for users: submit a job

ssh sdf-login

$ sbatch mygpujob.sh
Submitted batch job `1299143`

A job is usually a shell script

A job script includes several SLURM directives, starting with #SBATCH

- Partition to run (e.g. batch “queue”)
- Run time limit
- CPU, GPU, memory resources
- ...

The job payload. These are the commands you normally run in terminal

This example is tailored to typical GPU jobs

- Running on a single host
- Using a subset of GPUs on the host

```
$ cat mygpujob.sh
#!/bin/sh

#SBATCH --partition=ml
#SBATCH --time=10:00
#SBATCH --job-name=test
#SBATCH --output=out-%j.txt --error=out-%j.txt
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1 --mem-per-cpu=4g
#SBATCH --gpus=1

hostname
nvidia-smi
```

Output will be saved at out-**1299143**.txt

Ask for one GPU of any kind

SLURM 101 for users: checking your jobs

```
$ squeue -j 1299143
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1299143    shared     test    yangw PD       0:00      1 (Resources)
$ squeue -j 1299143
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
$ sacct -j 1299143
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
1299143            test     shared     shared          2  COMPLETED      0:0
1299143.bat+      batch                shared          2  COMPLETED      0:0
1299143.ext+     extern                shared          2  COMPLETED      0:0
$ cat out-1299143.txt
tur003
Wed Dec  1 21:24:00 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.19.01    Driver Version: 465.19.01    CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| 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  NVIDIA GeForce ...  On   | 00000000:1B:00.0 Off |                  N/A |
| 47%   65C    P5    26W / 250W |      1MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

- Job is pending for resource
- Job has completed
- More inform about the job
  - Not very useful except for MPI jobs.

The output file

- The job ran on node tur003
- Only one GPU is visible
  - Because the job only asked for 1 GPU

SLURM 101 for users: select GPU type

What if I want to use a specific kind of GPUs?

- First, check the available GPU types and their names:

```
$ sacctmgr list tres Type=gres format=Type,Name%25,ID
    Type                      Name     ID
-------- ------------------------- ------
    gres                       gpu   1001
    gres   gpu:geforce_gtx_1080_ti   1002
    gres   gpu:geforce_rtx_2080_ti   1003
    gres                  gpu:v100   1004
    gres                  gpu:a100   1005
```

- Then in job script

```
##SBATCH --gpu=1          ← comment this out with two "#"s.
#SBATCH --gres=gpu:geforce_rtx_2080_ti:1  ← Ask for one GPU of type geforce_rtx_2080_ti
```

SLURM 101 for users: a few more things 1

- Use a different partition (aka. “batch queue” in other batch systems)
  - At SLAC, we simplify the concepts of SLURM partition and “account”. They have a 1-to-1 matching.
  - Will not go into the detail of SLURM “account” for now.
    - But want you to aware there is such a thing.
    - Ask your group leader when SLURM account become an issue
  - All of you should be using partition “ml” in these exercises.
- Kill my job

  $ scancel -j 1299143
- Details about my running job

  $ scontrol show job 1299143 or sstat -j 1299143
- My job since 2021-10-15

  $ sacct --starttime 2021-10-15

Check manual page on login nodes:

- man sstat
- man scancel
- man sacct
- ...

SLURM 101 for users: a few more things 2

1. Your job inherit most but not all of your submitting environment
   - Current working directory, unix environment variables are inherited
   - Shell aliases, functions are not
2. Your job is limited to the resources you asked for
   - Via cgroup, SLURM will limit you to the CPU cores exclusively assigned to you.
   - Can not see GPUs than those exclusively assigned to you.
   - Going overboard on memory allocation? You job will be killed
3. Specify a run time limit to help SLURM scheduling jobs.
   - Default is 30 minutes.

Congrats, but be cautious

- Now you can access a lot more computing resources via SLURM
- But you can also do a lot more damage, especially on Lustre file system
  - Lustre is one of the two most commonly used shared posix file systems in the HPC world.
    - Shared means it is available on all nodes (interactive and batch).
    - It also means it is a networked file system, with lots of metadata exchange over the network (Infiniband on HPC cluster, ethernet on GPU cluster)
  - **Lustre is good at delivery high Bytes/s, it is not good at delivery high Files/s**
  - Where are high Bytes/s demand coming from: read/write large data files
  - Where are high File/s demand coming from: loading your program: .so, .py, config files, etc.
    - The Python ecosystem is particularly unfriendly in terms of Files/s
    - This is one of the reasons why you feel “The disk is so slow” - Often, the disk is not slow, but we have to learn how to live within the limitation of spinning disks

Suggested practices.

- Reduce Files/s
  - When possible, create Conda (or other software) environments in Singularity container
    - Lustre will see a Singularity .sif image as a single file. Accessing files inside the image counts toward Bytes/s, not Files/s
  - Working with your group to create a central software repository.
- Staggering your many jobs starting times
  - High Files/s usually happens at the beginning of a job. Avoid all jobs starting at the same time.
- Each GPU node has large solid state based scratch space
  - Accessible via unix environment $LSCRATCH
  - Keep in mind this is a temporary storage for this job only. It is not persistent storage.
    - Create your own directory under $LSCRATCH
    - Copy your data to $LSCRATCH
    - Clear your data and directory after the job is completed.
- Estimate the resource you need for repeated jobs
  - # of CPU cores, # of GPUs, RAMs, Run time
  - A good estimation help SLURM to schedule jobs efficiently

Document and SDF support

SDF document is available at the main web portal: https://sdf.slac.stanford.edu

SLURM Quick Start User Guide https://slurm.schedmd.com/quickstart.html

SDF staffs provides system level support

- Not application level support
- Mailing lists (these are backed by a ticketing system to keep track of request)
  - unix-admin@slac.stanford.edu
  - s3df-help@slac.stanford.edu
- Slack channel: https://slac.slack.com/app_redirect?channel=comp-sdf
- Use mailing lists (preferred) or Slack channel (short questions), avoid asking a specific staff member -- He/She may be unavailable.