AI Cluster - Slurm

Cluster is up and running now. Anyone with a CS account who wishes to test it out should do so.

Feedback is requested:

#ai-cluster Discord channel or email Phil Kauffman (kauffman@cs dot uchicago dot edu).

Knowledge of how to use Slurm already is preferred at this stage of testing.

The information from the older cluster mostly applies and I suggest you read that documentation: https://howto.cs.uchicago.edu/techstaff:slurm

Infrastructure

Summary of nodes installed on the cluster

Computer/GPU Nodes

- 6x nodes
  - 2x Xeon Gold 6130 CPU @ 2.10GHz (64 threads)
  - 192G RAM
  - 4x Nvidia GeForce RTX2080Ti
- 2x nodes
  - 2x Xeon Gold 6130 CPU @ 2.10GHz (64 threads)
  - 384G RAM
  - 4x Nvidia Quadro RTX 8000
- all:
  - zfs mirror mounted at /local
    - compression to lz4: Usually this has a performance gain as less data is read and written to disk with a small overhead in CPU usage.
    - As of right now there is no mechanism to clean up /local. At some point I'll probably put a find command in cron that deletes files older than 90 days or so.

Storage

- ai-storage1:
  - 41T total storage
  - uplink to cluster network: 2x 25G
  - /home/<username>
    - We intend to set user quotas, however, there are no quotas right now.
  - /net/projects: (Please ignore this for now)
    - Lives on the home directory server.

        - Idea would be to create a dataset with a quota for people to use.
        - Normal LDAP groups that you are used to and available everywhere else would control access to these directories. e.g. jonaslab, sandlab

- ai-storage2:
    - 41T total storage
    - uplink to cluster network: 2x 25G
    - /net/scratch: Create yourself a directory /net/scratch/$USER. Use it for whatever you want.
    - Eventually data will be auto deleted after X amount of time. Maybe 90 days or whatever we determine makes sense.
- ai-storage3:
    - zfs mirror with previous snapshots of 'ai-storage1'.
    - NOT a backup.
    - Not enabled yet.

Login

There are a set of front end nodes that give you access to the Slurm cluster. You will connect through these nodes and need to be on these nodes to submit jobs to the cluster.

```
ssh cnetid@fe.ai.cs.uchicago.edu
```

- Requires a CS account.

File Transfer

You will use the FE nodes to transfer your files onto the cluster storage infrastructure. The network connections on those nodes are 2x 10G each.

Quota

- By default users are given a quota of 20G.

Demo

kauffman3 is my CS test account.

```
$ ssh kauffman3@fe.ai.cs.uchicago.edu
```

I've created a couple scripts that run some of the Slurm commands but with more useful output. cs-sinfo and cs-squeue being the only two right now.

```
kauffman3@fe01:~$ cs-sinfo
```

```
NODELIST    NODES  PARTITION  STATE  CPUS  S:C:T   MEMORY  TMP_DISK WEIGHT
AVAIL_FEATURES                 REASON  GRES
a[001-006]  6      geforce*   idle   64    2:16:2  190000  0         1
'turing,geforce,rtx2080ti,11g'  none    gpu:rtx2080ti:4
a[007-008]  2      quadro     idle   64    2:16:2  383000  0         1
'turing,quadro,rtx8000,48g'     none    gpu:rtx8000:4
```

```
kauffman3@fe01:~$ cs-squeue
JOBID   PARTITION   USER           NAME                      NODELIST
TRES_PER_NSTATE     TIME
```

List the device number of the devices I've requested from Slurm. # These numbers map to /dev/nvidia?

```
kauffman3@fe01:~$ cat ./show_cuda_devices.sh
#!/bin/bash
hostname
echo $CUDA_VISIBLE_DEVICES
```

Give me all four GPUs on systems 1-6

```
kauffman3@fe01:~$ srun -p geforce --gres=gpu:4 -w a[001-006]
./show_cuda_devices.sh
a001
0,1,2,3
a002
0,1,2,3
a006
0,1,2,3
a005
0,1,2,3
a004
0,1,2,3
a003
0,1,2,3
```

give me all GPUs on systems 7-8 # these are the Quadro RTX 8000s

```
kauffman3@fe01:~$ srun -p quadro --gres=gpu:4 -w a[007-008]
./show_cuda_devices.sh
a008
0,1,2,3
a007
0,1,2,3
```

Notes on CUDA_VISIBLE_DEVICES

CUDA_VISIBLE_DEVICES: Displays relative gpu device number available to you.

- This variable should NOT be modified. Ever.
- Relative means that if you requested one gpu it will show up as 0. Even if all other gpus on the server are being used by others.

Fairshare/QOS

By default all usage is tracked and charged to a users default account. A fairshare value is computed and used in prioritizing a job on submission.

Details are being worked out for anyone that donates to the cluster. This will be some sort of tiered system where you get to use a higher priority when you need it. You will need to charge an account on job submission `--account=<name>` and most likely select the priority level you wish to use and that you are allowed to use: `--qos=<level>`

Asked Questions

> Do we have a max job runtime?

Yes. 4 hours. This is done per partition. You are expected to write your code to accommodate for this.

```
PartitionName=geforce Nodes=a[001-006] Default=YES DefMemPerCPU=2900
MaxTime=04:00:00 State=UP Shared
=YES
PartitionName=quadro  Nodes=a[007-008] Default=NO DefMemPerCPU=5900
MaxTime=04:00:00 State=UP Shared=
YES
```

Jupyter Notebook Tips

Batch

The process for a batch job is very similar.

jupyter-notebook.sbatch

```
#!/bin/bash
unset XDG_RUNTIME_DIR
NODEIP=$(hostname -i)
NODEPORT=$(( $RANDOM + 1024))
echo "ssh command: ssh -N -L 8888:$NODEIP:$NODEPORT `whoami`@slurm.ttic.edu"
. ~/myenv/bin/activate
jupyter-notebook --ip=$NODEIP --port=$NODEPORT --no-browser
```

Check the output of your job to find the ssh command to use when accessing your notebook.

Make a new ssh connection to tunnel your traffic. The format will be something like:

```
ssh -N -L 8888:###.###.###.###:#### user@fe01.ai.cs.uchicago.edu.edu
```

This command will appear to hang since we are using the -N option which tells ssh not to run any commands including a shell on the remote machine.

Open your local browser and visit: `http://localhost:8888`

Interactive

1. `srun --pty bash` run an interactive job
2. `unset XDG_RUNTIME_DIR` jupyter tries to use the value of this environment variable to store some files, by defaut it is set to `and that causes errors when trying to run juypter notebook. -` export NODEIP=$(hostname -i) `get the ip address of the node you are using -` export NODEPORT=$(($RANDOM + 1024)) `get a random port above 1024 -` echo $NODEIP:$NODEPORT `echo the env var values to use later -` jupyter-notebook --ip=$NODEIP --port=$NODEPORT --no-browser' start the jupyter notebook
3. Make a new ssh connection with a tunnel to access your notebook
4. `ssh -N -L 8888:$NODEIP:$NODEPORT user@fe01.ai.cs.uchicago.edu` using the values not variables
5. This will make an ssh tunnel on your local machine that fowards traffic sent to `localhost:8888` to `$NODEIP:$NODEPORT` via the ssh tunnel. This command will appear to hang since we are using the -N option which tells ssh not to run any commands including a shell on the remote machine.
6. Open your local browser and visit: `http://localhost:8888`