Using Cryo-EM Software Environment on Bede

Update on April 14, 2021
Contact: ruzhuchen@us.ibm.com

Table of Contents

A. CRYO-EM SOFTWARE PACKAGE

1. Using Slurm

Some of the most frequently use commands are copied below. For detailed user guide with Slurm, please visit https://slurm.schedmd.com/quickstart.html or http://www.schedmd.com/slurmdocs/faq.html.

salloc is used to allocate resources for a job in real time. Typically, this is used to allocate resources and spawn a shell.

srun is used to submit a job for execution or initiate job steps in real time. Usually use with `salloc` or interactively jobs

sbatch is used to submit a job script for later execution. The script will typically contain one or more srun commands to launch parallel tasks.

scancel is used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.

scontrol is the administrative tool used to view and/or modify Slurm state. Note that many `scontrol` commands can only be executed as user root.

sinfo reports the state of partitions and nodes managed by Slurm. It has a wide variety of filtering, sorting, and formatting options.

squeue reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.

sstat is used to get information about the resources utilized by a running job or job step.

sacct is used to report job or job step accounting information about active or completed jobs.

sview is a GUI to get and update state information for jobs, partitions, and nodes managed by Slurm.

Options of the common use (usage: `#SBATCH $option`) are listed below

```
-n, --ntasks=ntasks         number of tasks to run
-N, --nodes=N               number of nodes on which to run (N = min[-max])
-c, --cpus-per-task=ncpus   number of cpus required per task
    --ntasks-per-node=n     number of tasks to invoke on each node
-i, --input=in              file for batch script's standard input
-o, --output=out            file for batch script's standard output
-e, --error=err             file for batch script's standard error
-p, --partition=partition   partition requested
-t, --time=minutes          time limit
-D, --chdir=path            change remote current working directory
-D, --workdir=directory     set working directory for batch script
    --mail-user=user        who to send email notification for job state changes
-w, --nodelist=hosts...     request a specific list of hosts
--mem=M|G                   minimum amount of real memory
--gres=list                 required generic resources (e.g., --gres=gpu:4)
```

When creating Slurm batch script, one of the examples as follow:

```
#!/bin/bash
#SBATCH -o jobout.%J
#SBATCH -e joberr.%J
#SBATCH --mem=512G
#SBATCH --gres=gpu:4
#SBATCH -n 40
#SBATCH -t 10:00:00 #10 hours
```

When submit a Slurm job, simply use "`srun`" command + its options (`srun -help` for more option info). Please make sure you allocate enough memory and GPUs for you job. If running interactive job, start with "`salloc`" command may be easier to handle, for instance,

```
$ salloc -N1 -n20 bash
salloc: Pending job allocation 4241
salloc: job 4241 queued and waiting for resources
$ ssh -X node0052
$ source anaconda/bin.activate
$ conda activate …
```

To run batch jobs, which is preferred for long jobs, please create your run script and the use "sbatch" command to submit job. See simple example in the textbox.

```
$ cat myscript.sh
#!/bin/bash
#SBATCH --mem=512G
#SBATCH -n 20
#SBATCH -t 01:00:00

mpirun -n 20 -bind-to none mympijob.sh

$ sbatch --gres=gpu:4 ./mympijob.sh
```

"`sstat`" or "`sview`" can be used to view your job status after submitting jobs.

2. Using LSF

For more info about using IBM LSF, please visit https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_admin_foundations/run_lsf_jobs.html for details. A typical LSF script "myscript.sh" is shown below. For interativce job, use "bsub -Ip -X … your script or command".

```
$ cat myscript.sh
#BSUB -L /bin/bash
#BSUB -J "Relion"
#BSUB -o "Relion.%J"
#BSUB -n 20
#BSUB -R "span[ptile=20]"
#BSUB -gpu "num=4"
#BSUB -q "normalx"
#BSUB -x
mpirun -n 20 -bind-to none mympijob.sh

$ bsub <mympijob.sh
```

3. Running Relion with Slurm

Before starting your cryo-EM job, please install anaconda environment on your account. Download anaconda installation script from https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-ppc64le.sh using "wget" and install it following default steps.

After installation, you can activate conda environment with "source anaconda3/bin/activate" if not already active (check with "which conda").

1) Activate cryo-EM conda environment

The cryo-EM software package is located at /projects/bddir04/ibm-lfsapp/CryoEM. This package has all ENV setup to run RELION, CTFfind4, MotionCor2, CrYol and ResMap commands independently or with relion GUI. The Relion v3.1 has default paths to CTFfind, MotionCor2 and ResMap. To activate cryo-EM software environment, issue the following command:

```
conda activate /projects/bddir04/ibm-lfsapp/CryoEM
```

To activate Eman2, use command:

```
conda activate /projects/bddir04/ibm-lfsapp/Eman2
```

2) Running relion GUI

A local installation of an X11 server is required and running relion GUI requires to enable X11 forwarding when login to Bede cluster from your laptop or workstation.

a) On a Mac computer the XQuartz (https://www.xquartz.org) software is a good option.

b) On a Window computer Cygwin/X (http://x.cygwin.com) software provides a free X server.

Here are steps to start relion GUI with Slurm:

Option 1

a) `conda activate /projects/bddir04/ibm-lfsapp/CryoEM`

b) `salloc -N1 -n20 –gres=gpu:4 --mem=512G bash`
`#If it's done, you will see the allocated node name (e.g., node0001)`

c) `ssh -X gpu001 /projects/bddir04/ibm-lfsapp/CryoEM/bin/relion`

(Note: `srun --gres=gpu:4 --pty -x11` … is not working yet. It's a known bug).

Option 2 (recommended)

```
a) salloc -N1 -n20 --x11 --gres=gpu:4 --mem=512G bash
b) srun -n1 --pty -D your-run-path relion
```

Note: i) Flags "`--gres=gpu:4`" and "`--mem`" are important parameters for running relion, since it requires large memory usage.

ii) `--mem` can be more than 512GB, depending the free memory of the node.

iii) Sometimes you may run into "cannot open display" message again after a while, this can be solved to add a config file at $HOME/.ssh/. The config file looks like this:

```
[ruzhu@Bede-login-002 ~]$ cat $HOME/.ssh/config
Host *
        ForwardX11Timeout 172800s #2 days
[ruzhu@Bede-login-002 ~]$ chmod 644 $HOME/.ssh/config
```

3) **Running relion batch job**

Running relion with Slurm batch command is very simple if your script is created with Slurm option. See script below for relion 3D classification job:

```
#!/bin/sh

#SBATCH -o relion_log.%J
#SBATCH -e relion_err.%J
#SBATCH -n 20
#SBATCH -N 1
#SBATCH --mem=512G
#SBATCH --gres=gpu:4
#SBATCH -D /projects/bddir04/##your folder##/relion_benchmark

ulimit -s unlimited
source ~/anaconda3/bin/activate
conda activate /projects/bddir04/ibm-lfsapp/CryoEM
export PATH=/opt/software/apps/mpi/ibm/spectrum_mpi/bin:$PATH

mkdir -p /dev/shm/$USER

(time -p mpirun -bind-to none -n 17 relion_refine_mpi --j 10 --gpu \
   --pool 60 --dont_combine_weights_via_disc --scratch_dir \
   /dev/shm/ruzhu --i Particles/shiny_2sets.star --ref emd_2660.map:mrc \
    --firstiter_cc --ini_high 60 --ctf --ctf_corrected_ref --iter 25 \
    --tau2_fudge 4 --particle_diameter 360 --K 6 --flatten_solvent \
    --zero_mask --oversampling 1 --healpix_order 2 --offset_range 5 \
    --offset_step 2 --sym C1 --norm --scale --random_seed 0 \
    --maxsig 2000 --fast_subsets \
```

Once the script is created, use `sbatch` command (`sbatch your-script`) to submit the job. Alternatively, if you have non-Slurm script, you can use command line options of `sbatch` command to submit the job, e.g.,

```
$ sbatch -n 20 --mem=512G --gres=gpu:4 your-script.sh
```

By the way, if the Slurm is configured with GPU option, we can directly use option "`--gpus=4`", please use "`sbatch --help`" to see more options.

Check you job with "`squeue -u username`" for the job status. If you need to kill the job, use "`scancel your-jobid`".

4) **Troubleshooting**

a) **Out of memory**: when look into the log file, there is "out of memory" or "not enough memory" error message. Possible reason is the Slurm memory allocation (`--mem`) is not enough. If this is the case, increase the memory whenever starting "salloc" or "sbatch". Second reason is likely due to too much mpi tasks (-np) and/or threads (--j) being used. Recommend combination on single node is -np 17 / --j 10 for dedicated system node and -np 17 / --j 8 for shared node. The 3rd reason is too many jobs (including other users' jobs) are running on the system.

b) **GPU** cannot be allocated: when submitting job to the cluster, if no GPU flag is specified, the job will fail with "no device" found. Please make sure "--gres=gpu:4" is included in your script or command line.
c) **Paths** not found. When showing error message "No such file or directory", please make sure the file or directory exists. If file or directory exists, it may be permission issue. You can change the permission to "0755" in general.
d) **Shared library** not found: error message "`error while loading shared libraries:XXXX: cannot open shared object file: No such file or directory`" is shown. This means either the LD_LIBRARY_PATH is not set or points to the wrong folder. Please make sure the right path is:
```
LD_LIBRARY_PATH=$ LD_LIBRARY_PATH
/opt/software/apps/mpi/ibm/spectrum_mpi/lib:
/projects/bddir04/ibm-lfsapp/CryoEM/lib:/projects/bddir04/ibm-
lfsapp/CryoEM/share/ctffind-
4.1.13/lib:/usr/local/cuda/lib64:/projects/bddir04/ibm-
lfsapp/CryoEM/cuda/lib:/projects/bddir04/ibm-
lfsapp/CryoEM/lib/python3.6/site-packages/tensorflow
```
e) **Display:** error message "`Can't open display: localhost:12.0`". You need to log out and ssh login with "-X" or "-Y" flags. If the display cannot open after sometimes, please login with "`ssh -o ForwardX11Timeout=172800s -X username@login2.bede.dur.ac.uk`"
f) **Your job run extremely slow or hangs:** make sure all the system resource is not overuse, such as memory paging or your user quota is all use up. In shared system, there system resource is fully used by multiple users running jobs at the same time.
g) **Your job stops immediately:** please check your script to make sure the command and parameters are correct. For Relion, the temporary file at /dev/shm/relion_volatile is left on the system by other users.
h) **Known bug of relion v3.1**: `ERROR: cannot execute:  rm -rf /dev/shm/relion_volatile/` At the end of execution, relion try to delete the tmp folder using multiple MPI threads, instead of master thread. This doesn't affect the job. Just ignore it.
i) **Error message "`Bus error`"** is caused by not allocating enough memory to the job. Please increase or set "`--mem`" to 512G and up.

5) Performance tips

a) The optimal combination of MPI tasks (`-np`) and threads (`--j`) is **17** and **10**, respectively if the system has at least 512GB free memory and 4 available GPUs.
b) Set scratch dir to `/dev/shm` which is loading the data into memory. On Bede, each compute node has 1 TB memory, so we can run very large dataset (~500 GB) with scratch dir set to memory folder. If SSD or NVMe drive is available, set the scratch dir to it without performance degradation.
c) Wherever available, always use "`--fast_subsets --maxsig 2000`" parameters to run relion.
d) Whenever possible, please run large job exclusively. Slurm option "`--exclusive=user`", but it may wait in queue for very long time.
e) When running MotionCor2 with relion, there are mpirun options "`-x PAMI_DISABLE_CUDA_HOOK=1 -disable_gpu_hooks`" built in to the relion run command for avoiding runtime error. If run MotionCor2 independently, please have this option added to the script.
f) Use "`ulimit -a`" to verify the stack size and locked memory size. Use "`ulimit -s unlimited`" in the run script to avoid stack error.

g) GPU usage is critical to performance, use "`nvidia-smi`" to monitor the GPU usage. Use "`htop`" or "top" to monitor the CPU usage, especially the thread usage.
h) The `mpirun` option "`-bind-to`" set to "`none`" for relion MPI job, since it's MPI/OpenMP hybrid program.

4. Running CrYolo with Slurm

1) Activate `CryoEM` software environment

```
source anaconda3/bin/activate
conda activate /projects/bddir04/ibm-lfsapp/CryoEM
```

2) Running GUI

```
salloc -N1 -n20 --x11 --gres=gpu:4 --mem=512G bash
srun -n1 --pty -D your-run-path cryolo_gui.py 2>/dev/null
```

5. Running Eman2 with Slurm

1) Activate `CryoEM` software environment

```
a. source anaconda3/bin/activate
b. conda activate /projects/bddir04/ibm-lfsapp/Eman2
```

2) Checking Eman2

```
Checking the version with e2version.py and display with
e2display.py before running your test.
```

3) Running GUI

```
a. salloc -N1 -n20 --x11 --gres=gpu:4 --mem=512G bash
b. srun -n1 --pty -D your-run-path e2***.py 2>/dev/null
```

6. Install Watson Machine Learning (WML) CE

WML CE has all AI software tools such pyTorch, tensorflow, caffe, SnapML and so on. You can install individual tools or the whole package. To install locally,

a) Add WML conda channel to your conda environment

```
conda config --add channels
https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda
conda config --add channels conda-forge
conda info
```

b) Create a new conda environment

```
conda create –name or -p your-path-to/wmlce python=3.6 cython
```

c) Use "conda install" to install packages

```
conda activate your-path-to/wmlce
conda install pytorch=1.3.1
conda install tensorflow-gpu=2.1.0
# whole package
conda install powerai=1.7.0
```

d) Check your installation

```
python -c "import pytorch"
or
python -c "import tensorflow"
```

WML CE sets up all dependencies for your program to run or develop. It's GPU support with CUDA driver up to 10.2. See reference at https://mit-Bede.github.io/Bede-ai-frameworks.html#ibm-watson-machine-learning-community-edition-wmlce

B. DATA TRANSFER

1. External data transfer to Bede (Optional)

If Aspera is setup onb the cluster, we can use it for faster data transfer. Install Aspera client or connect on your local machine

a) **Mac, Windows or Linux-x86**: please download and install Aspera connect from https://downloads.asperasoft.com/en/downloads/8?list

   Mac or Windows: please following the software installation step to complete the installation. For Linux as non-root user, untar the gz file and run the install script file. After the installation is complete, it will be located at your home directory .aspera/connect

b) PowerPC (ppc64le) system: please download and install Aspera desktop client from https://downloads.asperasoft.com/en/downloadsArchive/2

   To install desktop client on Power system after download deb or rpm file:
   Ubuntu:
   ```
   sudo apt-get install ibm-aspera-desktopclient-3.9.6.176567-linux-ppc64le.deb
   ```
   RHEL|CentOS:
   ```
   sudo yum install ibm-aspera-desktopclient-3.9.6.176567-linux-ppc64le.rpm
   ```

2. Internal Data Copy

Copying your data from one location to another is faster if use “msrsync” command. By the way, the command is available when you activate the CryoEM environment.

```
$ which msrsync
/projects/bddir04/ibm-lfsapp/CryoEM/bin/msrsync
$ msrsync --help
usage: msrsync [options] [--rsync "rsync-options-string"] SRCDIR [SRCDIR2...]
DESTDIR
   or: msrsync --selftest

msrsync options:
    -p, --processes ...   number of rsync processes to use [1]
    -f, --files ...       limit buckets to <files> files number [1000]
    -s, --size ...        limit partitions to BYTES size (1024 suffixes: K, M, G,
T, P, E, Z, Y) [1G]
    -b, --buckets ...     where to put the buckets files (default: auto temporary
directory)
    -k, --keep            do not remove buckets directory at the end
    -j, --show            show bucket directory
    -P, --progress        show progress
    --stats               show additional stats
    -d, --dry-run         do not run rsync processes
    -v, --version         print version

rsync options:
    -r, --rsync ...       MUST be last option. rsync options as a quoted string ["-
aS --numeric-ids"]. The "--from0 --files-from=... --quiet --verbose
                            --stats --log-file=..." options will ALWAYS be added,
no matter what. Be aware that this will affect all rsync
                            *from/filter files if you want to use them. See
rsync(1) manpage for details.

self-test options:
    -t, --selftest        run the integrated unit and functional tests
    -e, --bench           run benchmarks
    -g, --benchshm        run benchmarks in /dev/shm or the directory in $SHM
environment variable
```