ARIES Quick Start

Version: Apr 6, 2022 by Weiqi Lu

> 👉 The ARIES cluster only supports the exclusive mode, i.e., your job will occupy a node with 8 GPUs exclusively. So please use this cluster only when you can make use of 8 GPUs in one job.

ARIES tips

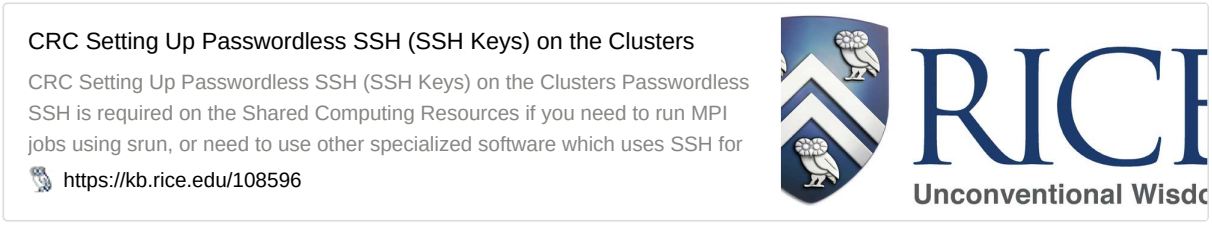
- Only the computing nodes (gn01 - gn19) has `/scratch`, and you can `ssh` to a node **only** when you have active jobs on it.
  - If your output data is small enough, save it in your `/home` directly.
  - If you must save your output data in `/scratch`, copy it out before your job is done.
- Your sub-jobs on one node should not generate the output file of the same name. Use a different name for each output file.

How to use multiple GPUs in one job

1. Use Launcher_GPU

Prerequisites

Set up ssh communication between nodes. The codes needed are as follows.



- Disable ssh host key checking via ssh prompts. Copy the following codes to your `~/.ssh/config`

```
Host *
   StrictHostKeyChecking no
```

- Create your public key

```
$ ssh-keygen -t rsa
```

- Copy your public key to `~/.ssh/authorized_keys`. This will enable mpirun to login from one compute node to another using SSH without a password.

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

TEMPLATE SCRIPTS

Copy the `/opt/apps/examples/OPENMM/multiGPU` to your `/home`. You should get the following files.

```
$ ls /opt/apps/examples/OPENMM/multiGPU
input.pdb   openmm-singularity8.slurm  simulatePdb_3.py  simulatePdb_6.py  simulatePdb.py
jGpu.slurm  simulatePdb_1.py           simulatePdb_4.py  simulatePdb_7.py
omjobs      simulatePdb_2.py           simulatePdb_5.py  simulatePdb_8.py
```

The `jGpu.slurm` is the one to sbatch.

```
#!/bin/bash

#SBATCH --account=commons
#SBATCH --partition=commons
#SBATCH --ntasks=8
#SBATCH --cpus-per-task=6
#SBATCH --threads-per-core=1
#SBATCH --mem-per-cpu=3G
#SBATCH --gres=gpu:8
#SBATCH --time=24:00:00
#SBATCH --export=ALL

module purge
module load  foss/2020b OpenMM Launcher_GPU

export LAUNCHER_WORKDIR=`pwd`
export LAUNCHER_JOB_FILE=omjobs
export LAUNCHER_BIND=1

$LAUNCHER_DIR/paramrun
```

A preview of the `omjobs`.

```
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
```

```
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
python simulatePdb_$LAUNCHER_JID.py > output_$LAUNCHER_JID.log
```

2. Run the sub-jobs directly

Example script

```
#!/bin/bash
#SBATCH --account=commons
#SBATCH --partition=commons
#SBATCH --job-name=wechrom-test
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:30:00
#SBATCH --export=ALL
#SBATCH --gres=gpu:8

# module management
module purge
module load GCC/10.2.0  OpenMPI/4.0.5 OpenMM/7.5.0
pip install -e /home/wl52/wechrom_private

# submit sub-jobs directly
HIP_VISIBLE_DEVICES=0 python sc_350bp_turn0_1.py &
HIP_VISIBLE_DEVICES=1 python sc_350bp_turn0_2.py &
HIP_VISIBLE_DEVICES=2 python sc_350bp_turn0_3.py &
HIP_VISIBLE_DEVICES=3 python sc_350bp_turn0_4.py &
HIP_VISIBLE_DEVICES=4 python sc_350bp_turn0_5.py &
HIP_VISIBLE_DEVICES=5 python sc_350bp_turn0_6.py &
wait
```

NOTES

- If you make use of 8 GPUs in one job, there is a chance your jobs may fail (why?). It's safer to use 6 instead of 8.

3. Run with singularity container (not working now)

Example script

```
#!/bin/bash

# This value can be adjusted to run in a condo if applicable
#SBATCH --account=commons
#SBATCH --partition=commons
# To run SMP applications only request one node, leave this value as is
#SBATCH --nodes=1
# This value should be adusted to match the number of GPUs per node
#SBATCH --gres=gpu:8
# This value can be adjusted if a different maximum wall time is required
#SBATCH --time=00:30:00
#SBATCH --export=ALL
```

```
# purge any modules that might be automatically loaded
module purge

# launch python3 in a container with your input file
ROCR_VISIBLE_DEVICES=0 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
ROCR_VISIBLE_DEVICES=1 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
ROCR_VISIBLE_DEVICES=2 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
ROCR_VISIBLE_DEVICES=3 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
ROCR_VISIBLE_DEVICES=4 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
ROCR_VISIBLE_DEVICES=5 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
ROCR_VISIBLE_DEVICES=6 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
ROCR_VISIBLE_DEVICES=7 singularity exec /opt/apps/examples/OPENMM/openmm_7.6.0.sif python3 simulatePdb.py &
wait
```

Current error:

```
openmm.OpenMMException: There is no registered Platform called "OpenCL"
```