Usage examples for CTBP resources

This page is currently being updated to include examples of submission and configuration commands that can be used on CTBP resources.

It should be noted that this example assumes the use of only one GPU per task and requests an equal amount of memory and CPU resources based on the total resources of each node. The amount of CPU and RAM memory utilized can be increased or decreased based on the user's experience with their system.

Requesting access

To request access to the clusters please use the following form:

https://www.crc.rice.edu/app/rice_signup.php

Slurm configuration

To obtain information about the number of nodes, number of CPUS, memory and number of GPUs in each cluster use the following command:

sinfo -o "%N %c %m %f %G " -p your_partition

NOTS (commons)

This partition includes 16 volta GPU nodes, each equipped with 80 CPUs and 182GB of RAM. In addition, each node includes 2 NVIDIA GPUs.

```
#SBATCH --account=commons
#SBATCH --partition=commons
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=40
#SBATCH --mem=90G
#SBATCH --gres=gpu:1

ml gomkl/2021a OpenMM/7.7.0-CUDA-11.4.2
```

NOTS (ctbp-common)

This partition includes two ampere GPU nodes, each equipped with an AMD EPYC chip featuring 16 CPUs and 512GB of RAM. In addition, each node includes 8 NVIDIA A40 GPUs with 48GB of memory.

```
#SBATCH --account=ctbp-common
#SBATCH --partition=ctbp-common
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --mem=64G
#SBATCH --gres=gpu:1

ml gomkl/2021a OpenMM/7.7.0-CUDA-11.4.2
```

NOTS (ctbp-onuchic)

This partition includes one GPU node, equipped with an AMD EPYC chip featuring 16 CPUs and 512GB of RAM. In addition, each node includes 8 NVIDIA A40 GPUs with 48GB of memory.

```
#SBATCH --account=ctbp-onuchic
#SBATCH --partition=ctbp-onuchic
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --mem=64G
#SBATCH --gres=gpu:1

ml gomkl/2021a OpenMM/7.7.0-CUDA-11.4.2
```

OpenMM on NOTS

You can deploy and run you own version of OpenMM via conda environment. For that, first install the OpenMM inside a conda environment requesting the modules already installed on NOTS. Note that in order to run with Nvidia GPUs, it has to be complicated with **CUDA/<version>**.

Conda environment with OpenMM

```
# Load conda and gpu modules
module load Anaconda3/2022.05 CUDA/11.4.2

# Create the openmm environment
conda create --prefix $HOME/openmm

# Activate the new env.
source /opt/apps/software/Anaconda3/2022.05/bin/activate
conda activate $HOME/openmm

# Then install OpenMM. You can also follow by installing your favorite MD wrapper
conda install -c conda-forge openmm cudatoolkit=11.4.2 h5py openmichrom opensmog
```

This would be an example of a running slurm script.

Slurm running OpenMM via environment

```
#!/bin/bash -l

#SBATCH --account=ctbp-common
#SBATCH --partition=ctbp-common
#SBATCH --job-name=Template-OPENMM
#SBATCH --ntasks=1
#SBATCH --threads-per-core=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2G
#SBATCH --gres=gpu:1
#SBATCH --time=00:05:00
#SBATCH --export=ALL

module purge
module load Anaconda3/2022.05 CUDA/11.4.2
source /opt/apps/software/Anaconda3/2022.05/bin/activate
conda activate $HOME/openmm

python your_script.py
```

ARIES

This partition includes 22 GPU nodes and 2 High Memory CPU nodes:

- **19 x MI50 Nodes (gn01-gn19)**: 1x AMD EPYC 7642 processor (96 CPUs), 512GB RAM, 2TB storage, HDR Infiniband, 8x AMD Radeon Instinct MI50 32GB GPUs.
- **3x MI100 Nodes (gn20-gn22)**: 2x AMD EPYC 7V13 processors (128 CPUs), 512GB RAM, 2TB storage, HDR Infiniband, 8x AMD Radeon Instinct MI100 32GB GPUs
- **2x Large Memory Nodes (hm01-02)**: 2x AMD EPYC 7302 processors (64 CPUs), 4TB RAM, 4TB storage, HDR Infiniband.

To submit a job to GPU queue, it is necessary to launch 8 processes in parallel, each with a similar runtime to minimize waiting time. This ensures that all of the GPUs are used efficiently.

```
#SBATCH --account=commons
#SBATCH --partition=commons
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --export=ALL
#SBATCH --gres=gpu:8

module load foss/2020b OpenMM
```

*Alternative if running 8 jobs in parallel

```
#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 load foss/2020b OpenMM
```

Checking usage

In order to determine if your process is running correctly, in each cluster you can connect directly to each compute server while you are running the file with ssh. Then use the command top to check the CPU and memory usage, rocm-smi to check the GPU usage for AMD/RADEON GPUs and nvidia-smi to check the GPU usage for NVIDIA GPUs,

Remote access to the clusters

To access the servers from outside of Rice it is recommended to connect to them through the gw.crc.rice.edu server. Here we will show how to create a passwordless ssh tunnel that will allow you to securely connect to a remote machine without having to enter a password everytime you connect.

Generate the keys

First generate a pair of public and private keys on your local machine. Open a terminal and enter the following command:

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

Save the key in the default key and leave the passphrase empty. This will generate a pair of public and private keys, with the default file names id_rsa and id_rsa.pub. Don't share or expose your private key.

Copy the keys to the gateway server

Copy the public key to the remote machine (The gw.crc.rice.edu server). Enter the following command in your terminal, replacing user with the correct rice user id:

```
ssh-copy-id user@gw.crc.rice.edu
```

This will copy your public key, id_rsa.pub, to the remote machine and add it to the authorized_keys file on the remote machine.

Test the connection. Enter the following command in your terminal to connect to the remote machine:

```
ssh user@gw.crc.rice.edu
```

You should be able to connect to the remote machine without being prompted for a password. Exit to your local machine using Ctrl+D

Create a ssh config file

To make it easier to connect to the remote machine in the future, you can create or edit your ssh config file in ~/.ssh/config. This file allows you to specify connection settings and aliases for different remote machines. To create an ssh config file, open the ~/.ssh/config file in a text editor and enter the following information, replacing user_id with your username on the remote machine:

```
Host crc
    User user_id
    HostName gw.crc.rice.edu
    IdentityFile ~/.ssh/id_rsa

Host aries
    User user_id
    HostName aries.rice.edu
    ProxyJump crc
    Port 22
    IdentityFile ~/.ssh/id_rsa

Host nots
    User user_id
    HostName nots.rice.edu
    ProxyJump crc
    Port 22
    IdentityFile ~/.ssh/id_rsa
```

Test the connection from you local machine to the remote machine using the alias. The gateway will be accessible without a password. You should be able to connect to the gateway enter the following command in your terminal:

```
ssh crc
```

Exit to your local machine with Ctrl+D

Copy the keys to the compute servers

To add the keys to the compute servers add the keys from your local machine to ~/.ssh/authorized_keys in the compute machine. For that in your local machine get the public key by executing the following command:

```
cat ~/.ssh/id_rsa.pub
```

Connect from your local machine to the compute servers using the settings and alias specified in the ssh config file with the following command:

```
ssh nots
```

You will be prompted for a password. Once you have entered it, you can edit or create the ~/.ssh/authorized_keys file on the compute server using a text editor like vi. Make sure to create the folder .ssh first if it doesn't exist:

```
mkdir .ssh
vi ~/.ssh/authorized_keys
```

Add the contents of your local machine's ~/.ssh/id_rsa.pub file to a new line in the authorized_keys file. Save the file exit the text editor (:wq) and then exit to your local machine with Ctrl+D.

To test the connection. Enter the following command in your terminal to connect to the remote machine:

```
ssh nots
```

You should now be able to connect to the compute server without being prompted for a password.

Repeat these steps for each additional compute server you want to connect to.

More Information

	File	Modified
	PDF File ARIES_Quick_Start_wl52_20220406.pdf	Dec 19, 2022 by Carlos Bueno Basurco