14 September 2020

ML Benchmarks on Discovery Cluster

Research Computing

Northeastern University

Table of contents

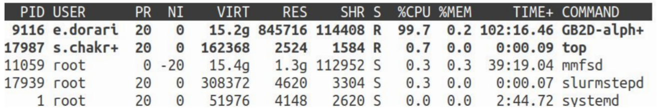
01

Configuration

Ensuring exclusive GPU usage

Non Exclusive Request for NVIDIA Tesla P100 GPU

$ srun --partition gpu --pty --gres=gpu:p100 /bin/bash

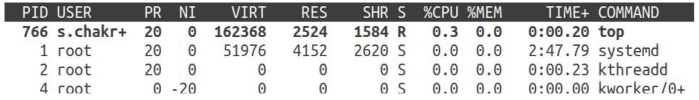


```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9116 e.dorari  20   0   15.2g 845716 114408 R  99.7  0.2 102:16.46 GB2D-alph+
17987 s.chakr+  20   0  162368   2524   1584 R   0.7  0.0   0:00.09 top
11059 root       0 -20   15.4g   1.3g 112952 S   0.3  0.3  39:19.04 mmfsd
17939 root      20   0  308372   4620   3304 S   0.3  0.0   0:00.07 slurmstepd
    1 root      20   0   51976   4148   2620 S   0.0  0.0   2:44.72 systemd
```

Discovery user **e.dorari** can be seen hogging **99.7%** of the available E5-2680v4@2.40GHz CPU which would heavily skew results

Exclusive Request for NVIDIA V100 SXM2 GPU

$ srun --partition gpu --pty --gres=gpu:p100 --exclusive /bin/bash

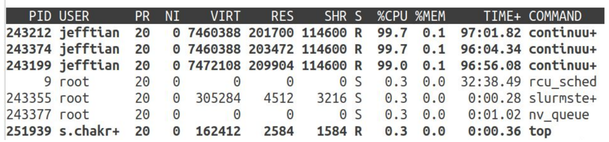


```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  766 s.chakr+  20   0  162368   2524   1584 R   0.3  0.0   0:00.20 top
    1 root      20   0   51976   4152   2620 S   0.0  0.0   2:47.79 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.23 kthreadd
    4 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0+
```

My user **s.chakravarty** is now free to utilize all the compute

Non Exclusive Request for NVIDIA V100 SXM2 GPU

$ srun --partition gpu --pty --gres=gpu:v100-sxm2 /bin/bash

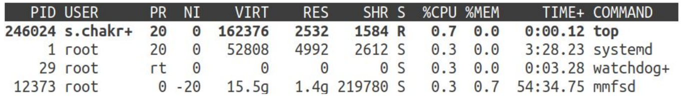


```
   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
243212 jefftian  20   0 7460388 201700 114600 R  99.7  0.1  97:01.82 continu+
243374 jefftian  20   0 7460388 203472 114600 R  99.7  0.1  96:04.34 continu+
243199 jefftian  20   0 7472108 209904 114600 R  99.0  0.1  96:56.08 continu+
     9 root      20   0       0      0      0 S   0.3  0.0  32:38.49 rcu_sched
243355 root      20   0  305284   4512   3216 S   0.3  0.0   0:00.28 slurmste+
243377 root      20   0       0      0      0 S   0.3  0.0   0:01.02 nv_queue
251939 s.chakr+  20   0  162412   2584   1584 R   0.3  0.0   0:00.36 top
```

Discovery user **jeff.tian** can be seen hogging **99.7%** of the available Intel Gold 6132@2.60Ghz CPU which would heavily skew results

Exclusive Request for NVIDIA V100 SXM2 GPU

$ srun --partition gpu --pty --gres=gpu:v100-sxm2 --exclusive /bin/bash



```
   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
246024 s.chakr+  20   0  162376   2532   1584 R   0.7  0.0   0:00.12 top
     1 root      20   0   52808   4992   2612 S   0.3  0.0   3:28.23 systemd
    29 root      rt   0       0      0      0 S   0.3  0.0   0:03.28 watchdog+
 12373 root       0 -20   15.5g   1.4g 219780 S   0.3  0.7  54:34.75 mmfsd
```

My user **s.chakravarty** is now free to utilize all the compute

Creating a miniconda environment

Clean Conda Environment Steps

Download Miniconda 2 from the internet

```
$ wget https://repo.anaconda.com/miniconda/Miniconda2-latest-Linux-x86_64.sh
```

Change the permissions of the installation script

```
$ chmod +x Miniconda2-latest-Linux-x86_64.sh
```

Run the installation script to install Miniconda 2

```
$ ./Miniconda2-latest-Linux-x86_64.sh
```

Agree to license agreement >> yes

Directory to install >> /work/rc/s.chakravarty

```
cd /work/rc/s.chakravarty/bin
```

Activate your base miniconda environment

```
$ source activate
```

Update all your conda packages

```
conda update conda
Proceed? >> yes
```

Configuring Conda Environments

rapids.ai **cuML**

Steps

```
$ conda create --name cuml_env --no-default-packages
$ conda activate cuml_env
(cuml_env) $ conda install -c rapidsai-nightly -c nvidia -c conda-forge -c defaults
rapids=0.16 python=3.8 cudatoolkit=11.0
```

Intel® **DAAL4py**

Steps

```
$ conda create --name daal4py_env --no-default-packages
$ conda activate daal4py_env
(daal4py_env) $ conda install -c intel daal4py
```

H_2O.ai **H2O4GPU**

Steps

```
$ conda create --name h2o4gpu_env --no-default-packages
$ conda activate pydaal_env
(h2o4gpu_env ) $ conda create -n h2o4gpuenv -c h2oai -c conda-forge -c rapidsai
h2o4gpu-cuda10
```

Configuring Conda Environments

rapids.ai **cuML**

Sample Code

```
Python 3.8.5 | packaged by conda-forge |
>>> from cuml.cluster import KMeans
>>> import cudf, numpy as np, pandas as pd
>>> def np2cudf(df):
      ... df = pd.DataFrame({'fea%d'%i:df[:,i] for i in range(df.shape[1])})
     ... pdf = cudf.DataFrame()
     ... for c,column in enumerate(df):
         ... pdf[str(c)] = df[column]
     ... return pdf
>>> kmeans_float = KMeans(n_clusters=2).fit(np2cudf(np.array([[1.,1.], [1.,4.], [1.,0.]])))
>>> print(kmeans_float.cluster_centers_)
array([[1. , 0.5], [1. , 4. ]])
```

Intel® **DAAL4py**

Sample Code

```
$ python3
Python 3.7.7 :: Intel(R) Corporation
>>> from daal4py import kmeans_init
>>> import numpy as np
>>> X = np.array([[1.,1.], [1.,4.], [1.,0.]])
>>> kmi = kmeans_init(10, method="plusPlusDense")
>>> result = kmi.compute(X)
>>> print(result.centroids)
array([[1. , 0.5], [1. , 4. ]])
```

H_2O.ai **H2O4GPU**

Sample Code

```
$ python3
Python 3.6.11 | packaged by conda-forge |
Type "help", "copyright", "credits" or "license" for more information
>>> import h2o4gpu
>>> import numpy as np
>>> X = np.array([[1.,1.], [1.,4.], [1.,0.]])
>>> model = h2o4gpu.KMeans(n_clusters=2,random_state=1234).fit(X)
>>> model.cluster_centers_
array([[1. , 0.5], [1. , 4. ]])
```