NewGridFAQ

Contents

How to Install and Connect to the Columbia AnyConnect VPN client?

See instructions on how to install CISCO AnyConnect VPN at the CUIT website: AnyConnect_VPN (https://www.cuit.columbia.edu/install-vpn).

How to connect to the Grid using NoMachine?

Nomachine client supports virtual desktops and graphic applications. It provides applications persistency. You can disconnect or reconnect without disrupting the progress of the jobs. Follow the steps below to download and install Nomchine client and configure virtual desktop or command line terminal connections. Download and install Nomachine Enterprise Client

1. Download from Nomachine_Download (https://www.nomachine.com/download-enterprise)
2. Create a new connection template:
   - Give it a friendly name
   - Host: research.gsb.columbia.edu
   - Protocol: NX (Port: 4000)
   - Authentication: Password

Establish Virtual Desktop Connection

1. Doubleclick the connection icon just created in step 2 and fill in the authentication information
2. Doubleclick on “Create a new virtual desktop”
3. A virtual desktop running on one of the three servers will appear
   - CTRL+ALT+0 and select the DISPLAY config:
   - Select “Resize remote display”
   - You can now extend your desktop’s effective area by resizing the window
   - To terminate the session, click your name on the top right corner and select “Log Out…”. Confirm on the popup window.
   - To disconnect the session but leave it active for further reconnection, close the desktop window from the (X) corner buttons.
   - Mac users need to install XQuartz (www.xquartz.org) and restart!

Establish Command Line Terminal Connection

1. Doubleclick the connection icon just created in step 2 and fill in the authentication information
2. Doubleclick on “Create a new custom session”
   - Select “Run the following command”…
   - Enter “/usr/bin/x-terminal-emulator” in the field
   - Check “Save this setting in the connection file”
3. A command line terminal will appear in a floating window
   - To terminate the session, type “exit” or CTRL-d, or close the terminal window from the corner buttons.
   - To disconnect the session but leave it active for further reconnection, close the Nomachine control window from the (x) corner buttons.
   - Mac users need to install XQuartz (www.xquartz.org) and restart!

How to connect to the Grid via SSH client?

SSH clients Support only command line terminals (no virtual desktops). Although SSH can support graphical applications via X11 tunnel, we recommend users who need GUI support use Nomachine client.

- Mac platforms can use the native “Terminal” application:
  Connect via “ssh UNI@research.gsb.columbia.edu”
- Windows platforms should use PuTTY PuTTY.pdf (http://wiki.gsb.columbia.edu/research/images/7/76/PuTTY.pdf)
  Host: research.gsb.columbia.edu
  Port: 22 (default)

A VPN connection is required before using SSH clients off campus.

How to transfer data to or from Grid?

The GRID uses a dedicated data transfer portal for data transfer.

- Hostname: researchfiles.gsb.columbia.edu
- (Protocol: SFTP ; Port: 22

Use your favorite data transfer (sftp or SCP) client:

- Filezilla (https://filezilla-project.org/)
- Cyberduck (https://cyberduck.io/)
- WinSCP - A secure file transfer over the internet (Protocol: SCP). See instructions here: Install_WinSCP (https://www.cuit.columbia.edu/content/winscp).
- For high-volume data transfers use GLOBUS (globus.org)

A VPN connection is required before establishing a connection off campus.

How to submit batch jobs?

See instructions in the article Running Batch Jobs on CBS Research Grid (https://wiki.gsb.columbia.edu/research/images/e/e9/Running_Batch_Jobs_on_CBS_Research_Grid.pdf)

How to check the status and delete batch jobs after being submitted?

The system utility qstat can be used to check the status of the jobs after they are submitted. Its usages are listed below:

```
qstat              - list the status of all the jobs owned by the caller – YOUR jobs
qstat -j <jobID>   - list the detailed status of the job with the provided jobID, a 7-digit number
qstat -u           - list all YOUR jobs
qstat -u  “*”      - list all jobs - everyone
```

System utility qstat can be used to check the status of the job. The “state” column in the output displays the status of the job:

```
r = running
E = error
q = queued
w = waiting
t = transferring
```

qdel can be used to kill a job. You can do the following:

```
qdel <jobID>       - delete the job identified by job ID
qdel -u $(whoami)  - delete all jobs owned by the user. Use with great caution!!
```

What options are available for submitting jobs to the Grid?

All Grid options start with --grid. Below is a list of commonly used options: Resource Requests:

```
  --grid_ncpus=n – Number of CPU cores (n => 1 – 500, default n=1)
  --grid_mem=nG – Amount of memory (n=> 4 – 1000GB, default 4GB)
  --grid_long – Unlimited execution time (Max 100 days. default 20 days)
```

Execution mode:

```
  --grid_submit=batch|interactive – (default: interactive)
```

Notification:

```
  --grid_email=“myemail@columbia.edu” – Email notification at the completion
```

Array jobs:

```
  --grid_array=<start>-<end>[:<step>][/maxConc]
```

It is important to request the appropriate number of CPUs and size of RAM for your job. For instance, applications that support multi-threading internally by default (e.g., Matlab and SAS) can run faster with more CPUs. Programs/applications that do not support multi-threading won't benefit from more CPUs. Programs processing large datasets and performing sophisticated statistic computations require a larger size of RAM as insufficient memory size can cause the program to fail or run very slowly. However, requesting excessive RAM, more than the program requires can cause the program to stay in waiting status longer as the system tries to find a worker node with the requested size. There is no simple way to accurately forecast the amount of resources needed for a particular program. Our suggestion is that you start with a conservative but educated guess, and progressively increase your requests by a small incremental until the jobs run efficiently and successfully.

How to run Python jobs on the Grid?

See instructions in the article Running Python Jobs on CBS Research Grid (https://wiki.gsb.columbia.edu/research/images/f/f6/Running_Python_Jobs_on_CBS_Research_Grid.pdf)

How to read zipped files without unzipping them?

See instructions in the article Read Compressed Files without Decompressing the Files on Disk (https://wiki.gsb.columbia.edu/research/images/c/c4/Reading_Compressed_File_in_Stata_and_SAS.pdf)

How to Install and Run Web Testing Tool Selenium on the Grid?

Selenium is a Python tool that supports various testing approaches, such as functional testing, regression testing, and load testing. The Selenium WebDriver API allows users to interact with web elements on a page, simulate user inputs such as clicks, typing, and scrolling, and extract data from web pages.

To install Selenium module, run these two commands:

conda activate pip install selenium

The Grid uses Firefox as the default web browser and Google Chrome is not supported because it does not support Grid's system configuration. The following script demonstrates how to run Selenium with Firefox:

from selenium import webdriver from selenium.webdriver.firefox.options import Options options = Options() options.headless = True driver = webdriver.Firefox(options=options) driver.get("http://google.com/") print ("Headless Firefox Initialized") driver.quit()

How to Install and Run TensorFlow with GPU Server on Research Grid?

TensorFlow is an open-source machine learning framework developed by Google. It is designed to facilitate the creation of neural networks and perform various numerical computations. It can run on both CPU (Central Processing Unit) and GPU. However, utilizing a GPU can significantly speed up the training and inference processes for deep learning models, as GPUs are optimized for

parallel computations commonly found in neural networks. To run the program on GPU server, the Python TensorFlow library must be installed. The procedure below demonstrates how to accomplish this.

1. Log to the GPU server:

```
    grid_run  --grid_gpu
```

2. Issue the following commands one at a time:

```
    conda create -n tf python=3.9
    conda deactivate
    conda activate tf
    conda install -c conda-forge cudatoolkit=11.8.0
    pip install nvidia-cudnn-cu11==8.6.0.163
    CUDNN_PATH=$(dirname $(python -c "import nvidia.cudnn;print(nvidia.cudnn.__file__)"))
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/:$CUDNN_PATH/lib
    mkdir -p $CONDA_PREFIX/etc/conda/activate.d
    echo 'CUDNN_PATH=$(dirname $(python -c "import nvidia.cudnn;print(nvidia.cudnn.__file__)"))' >>
$CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
    echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/:$CUDNN_PATH/lib' >>
$CONDA_PREFIX/etc/conda/activate.d/env_vars.sh
    pip install tensorflow==2.12.*
```

3. Verify installation:

```
    a. python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
    b. python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

4. Run Python Scripts and Submit the job to GPU:

To run Python scripts using TensorFlow, add the following statement as the first line of the script,

```
    #!/user/grid_user_id/.conda/envs/tf/bin/python
```

and issue the following command to submit the job:

```
    grid_run  --grid_submit=batch --grid_gpu --grid_mem=<xxx>G   ./yourPyScript.py
```

How to Install and Run Pytorch with GPU Server on Research Grid

PyTorch is an open-source machine learning framework primarily used for developing and training deep learning models. PyTorch can be used on both CPU (Central Processing Unit) and GPU devices. When running PyTorch on a system without a GPU, it automatically falls back to CPU execution. To install and run Pytorch with GPU server on the Grid, follow the steps below.

1. Log to the GPU server:

```
    grid_run  --grid_gpu
```

2. Issue the following command one at a time:

```
conda create -n pytorch
conda deactivate
conda activate pytorch
conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
```

3. Verify installation in Python:

```
>>> import torch
>>> x = torch.rand(5, 3)
>>> print(x)
tensor([[0.7268, 0.5792, 0.2185],
       [0.4668, 0.1664, 0.3163],
       [0.1219, 0.9713, 0.9201],
       [0.7946, 0.7810, 0.5992],
       [0.1060, 0.4422, 0.6397]])
>>> torch.cuda.is_available()
True
>>>
```

How to Search and Download Captial IQ Conference Call Transcripts?

See instructions in the article Search and Download Capital IQ Conference Call Transcript (https://wiki.gsb.columbia.edu/research/images/5/5a/Search_and_Download_Capital-IQ_Conference_Call_Transcripts.pdf)

How to Search and Download SEC Filings from WRDS Server?

See instructions in the article Search and Download SEC Filings on WRDS Server (https://wiki.gsb.columbia.edu/research/images/c/cc/Search_SEC_Filings_from_WRDS_Server.pdf)

How to submit array jobs and pass the task ID to Python scripts?

Grid option “--grid_array” allows users to submit array jobs:

```
grid_run --grid_submit=batch --grid_array=1-10 ./myprog.py
```

This will submit “myprog.py” 10 times. This is equivalent to a bash script:

```
for x in {1..10}
do
   grid_run --grid_submit=batch ./myprog.py
done
```

For batch jobs, the system typically generates 4 types of log files:

```
*.peXXXXXXXXXX
*.poXXXXXXXXXX
*.eXXXXXXXXXXX
*.oXXXXXXXXXXX
```

Note *.pe and *.po files are not currently used so ignore them. The *.pe files capture any errors/warnings messages, and the *.po files capture output sent to the Standard I/O. For array jobs, the system will add an N at the end of the files so it carries the format "*.oXXXXXXX.N" where N corresponds to the Task-ID which in this case is 1-10. "-–grid_quiet" option can be used to suspend these files.

As for all batch jobs, a directive statement, shebang, should be included as the first line of script. In this case, the first line of myprog.py should include the following: "#!/apps/anaconda3/bin/python". The script also must be executable. Use "chmod +x ./myprog.py" to change the mode if necessary.

The system environment variables related to array jobs can be captured in Python as a variable. The following variables can be used:

```
SGE_TASK_ID - This is the task ID of array jobs (e.g. 1 – 10).
SGE_TASK_FIRST - The first iteration.
SGE_TASK_LAST - The last iteration.
```

In Python script, the following statements can be used to capture these variables:

```
import os
i = int(os.getenv('SGE_TASK_ID'))
s=sstart+i*0.01
```

For more information on how to run batch jobs, see Run Batch Job on Grid (https://wiki.gsb.columbia.edu/research/images/e/e9/Running_Batch_Jobs_on_CBS_Research_Grid.pdf). For more information on how to run Python jobs, see Run Python on Grid (https://wiki.gsb.columbia.edu/research/images/f/f6/Running_Python_Jobs_on_CBS_Research_Grid.pdf).

How to submit MATLAB job with parameter input from the command line?

To submit MATLAB job with parameter input on the grid, the following can be used:

```
matlab --grid_submit=batch -batch "export myvar=123; /user/bc2021/myprog"
```

Where myvar is any name user can give in the script and the keyword "export" is need in Linux environment to make myvar a global variable. Note the use of quotes to protect the semicolon from being interpreted by the running shell. "myprog" refers to the myprog.m located in the user home folder.

For more information, please see Related Aricle on MATHWORKS website (https://www.mathworks.com/matlabcentral/answers/97204-how-can-i-pass-input-parameters-when-running-matlab-in-batch-mode-in-windows).