Running Python Jobs on CBS Research Grid

The Grid supports Python programs in many ways. Python codes can run in interactive mode, batch submission, or via popular development tools such as Jupyter Notebook. This document explains these methods.

Overview

The Grid offers the Anaconda Python 3 distribution (*https://www.anaconda.com/*) which includes many popular data science modules for processing Python jobs.

System Installed Python Compiler

To access Anaconda data science modules and route your job to an appropriate worker node on Grid, you should use Grid's wrapper commands, which are preconfigured Sun Grid Engine commands, to start your Python job.

The following Grid command will invoke Anaconda Python compiler and interactively execute your Python codes:

`anapy3`	– Anaconda Pythyon 3
`ipython3`	– Enhanced Interactive Python

The following commands will invoke popular development environments:

`spyder`	– Interactive Python development tool named Spyder
`notebook`	– Jupyter Notebook
`jpyterlab`	– JupyterLab

All the commands above take Grid options such as `--grid_mem`, `--grid_ncpus`, and `--grid_submit`. These options will allow you to request appropriate resources needed for your job and route the job to a computer node that has resources available.

IMPORTANT: Avoid using the `python` command as it will launch the system default Python, an older version that comes with system OS. It may also cause your job to run on the login node and slow down the performance of the Grid.

Python Work Environment

With the system provisioned Anaconda Python 3, you can install needed packages in your own account without the need of system administrator privileges. This allows you to customize Python to create your own work environment. For example, if you need to install the *wrds* module to use the WRDS API, you must first activate the system Python using the command below:

```
conda activate
```

This will activate the system Anaconda Python environment named `base`. You can then install WRDS API using command:

```
pip install wrds
```

The module will be installed in your home directory and remain available until you remove it (e.g., `pip uninstall wrds)`.

If your need a different distribution other than Anaconda Python, you can install it in your account to build you own work environment: `conda create -n <envName.<packageName>[=<version>]`. To check for information of work environment you have created, use `conda info` or `conda info --envs` commands.

Submitting Python Jobs

The common way to run Python codes is to submit them as scripts using batch mode. To submit the program in batch, your need to include your Python codes in a file. The following is a sample program named `add.py`:

```
#!/apps/anaconda3/bin/python
# This program adds two numbers and display them
num1 = 1.5
num2 = 6.3
sum = num1 + num2
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

The first line of the program, named "shebang," select Python interpreter for executing the codes and must always be included. In this example, the shebang invokes the *Anaconda Python Interpreter*, which is the recommended choice.

You can use the Grid's general job wrapper `grid_run` to submit the job as follows:

```
grid_run --grid_submit=batch ./add.py
```

You can add other Grid options such as `--grid_mem` and `-grid_ncpus`, to request additional resources, should your program require them.

Note that `grid_run` requires the program file to be executable or you will get a `Permission Denied` error. In such case, you can use `chmod` command to change the permission (e.g., `chmod u+x add.py`).

The output and system messages can be viewed in one of the output files generated by the system. The output files will be located in the same folder where you submitted the job and will carry a name similar to `add.py.o6749961`, where the 7-digit numeric number is the job ID.

The basic interactive Python is also supported on the Grid which can be useful if you want to quickly test some commands when you program your codes. You can use the following Grid wrapper command to start an interactive session:

```
anapy3 --grid_mem=10g
```

where `--grid_mem=10g` is an example of a Grid option used to request 10 GB of RAM for the interactive session. To end your interactive Python session, type `quit()`.

Run Program with Python Development Tools

Jupyter Notebook

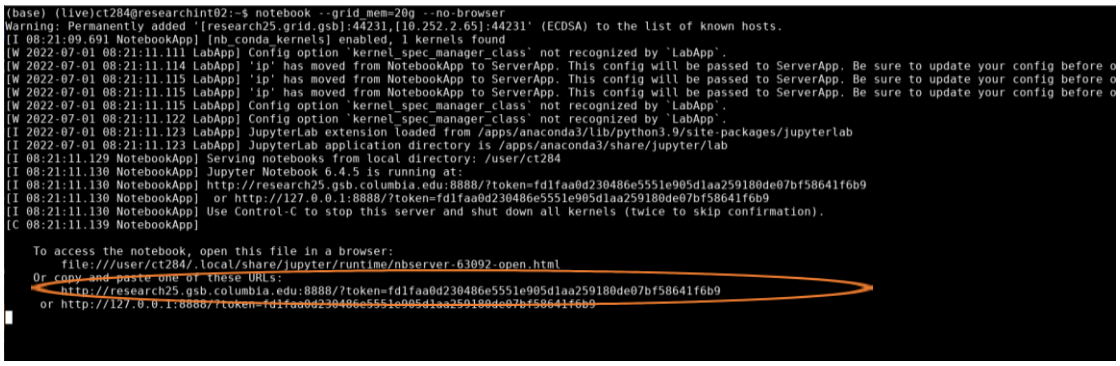
The Grid supports Jupyter Notebook using a server-client model where the compute kernels run on an automatically selected worker node on the Grid, while the frontend runs on the user's local machine web browser.

A two-step procedure is required to launch the Notebook on a worker node and display the interface on a web browser running on your laptop or workstation. First, launch `notebook` from a Grid command line terminal:

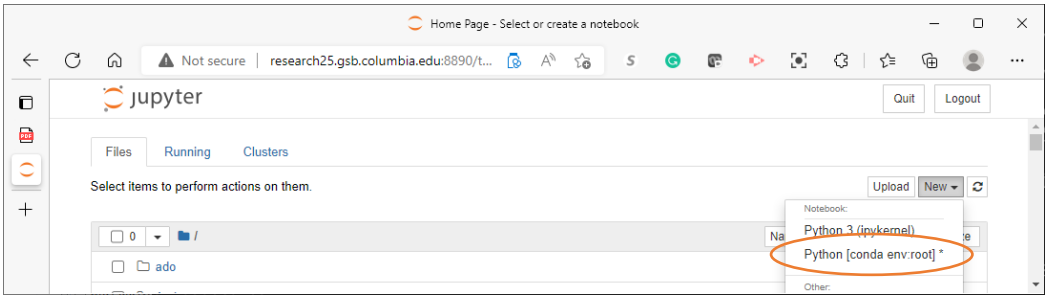
```
notebook --no-browser
```

The `--no-browser` option prevents `notebook` from using Grid's Firefox browser.

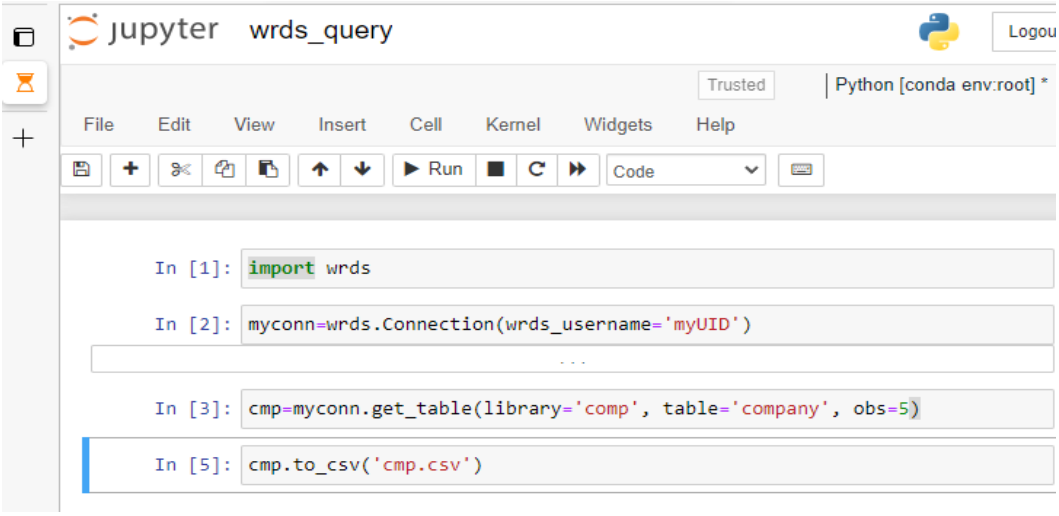
After submitting this command, the remote notebook kernel will display a few messages and wait for you to open your local web-browser:



In the second step, copy the URL containing "gsb.columbia.edu", see highlighted in the screenshot above, and paste it to the address line of your preferred web browser on your workstation. Your familiar `notebook` interface will be displayed:



You can now start a new notebook using Python root environment, see highlighted in above screenshot. Keep the kernel window alive, although you can minimize it, during the entire session. A sample session is displayed below:



Jupyterlab

Jupyterlab works in similar fashion as Jupyter Notebook. The command to start Jupyterlab is:

```
jupyterlab
```

Then you can copy-and-paste the URL containing "gsb.columbia.edu" into a web browser running on your local workstation.

Spyder

Spyder is an integrated development environment similar to PyCharm and VS Code. The command to start Spyder on the Grid is:

```
spyder
```

Because Spyder requires graphic support, you must connect to Grid using the Nomachine client. Refer to the Grid Training Slides (https://wiki.gsb.columbia.edu/research/images/5/53/Intro_to_GRID_2022.pdf) for instructions on how to connect to the Grid in the Virtual Desktop connection.

IMPORTANT: All the development tools presented above (Notebook, JupyterLab, Spyder) accept the usual GRID options for requesting extended system resources (memory and CPUs): `--grid_mem` and `--grid_ncpus`. Simply append them to the main command keyword as in these examples:

```
notebook  --grid_mem=10G  --grid_ncpus=4
jupyterlab --grid_mem=10G
spyder  --grid_mem=15G
```

Research Support

For information how to connect to the Grid and basic commands, see Grid Training Slides:
https://wiki.gsb.columbia.edu/research/images/5/53/Intro_to_GRID_2022.pdf

For technical assistance, please contact Research Support at ResearchSupport@gsb.columbia.edu.