Job Scripts

September 24, 2012

Note: Do not attempt to copy/paste out of this document. Commands will likely not work!

Learning by doing:

1. Connect to HPC Center: `ssh user@submit.hpc.ufl.edu`
    a. Type your password and hit return (no characters display while you type).
2. Lets work on a test node, so ssh to a test node (01-06), eg: `ssh test03`
3. Go to your scratch directory: `cd /scratch/hpc/$USER`
4. Copy the example files to your scratch directory:
   `cp –r /project/bio/training/2012-09-24/wordcloud/ .`
5. All of the scripts have <your email> rather than your actual email. We need to change that. There is a script that will do this for you, but we need to edit it to have your email address. Edit the script, where it says "magitz\@ufl.edu" change that to your email. (If you skip this step, I'll know because I'll get a ton of email about your jobs!)
    a. `nano modify_email.sh`
    b. On the 2nd line, change the email address (leave the \ before the @).
    c. Save the changes
    d. Now run the script: `./modify_email.sh`
6. Let's look at one of the scripts: `more cloud.pbs`
    a. This is basically the same script we used last week. It runs a Perl script that counts the word frequencies in a text file and outputs an HTML file with words ordered and scaled by frequency.
    b. This week, we're complicating things a bit in that we have a bunch of input files to process. Each file is the text of the first research article published in PNAS each decade from 1950 to 2010. Has word usage in science changed over time? OK, this might not be the best experiment, but the idea is that you have a bunch of tasks to complete and we're going to explore some ways to do this.
    c. The cloud.pbs script will run one file through the process.
    d. Let's look at some ways to run lots of files through.
7. **Writing individual job scripts for each task**
    a. One way to process each of our tasks is to write one script per task.
    b. There's an example of this in `one_by_one/`
    c. `cd one_by_one`
    d. `ls`
    e. Notice a job script for each file. You can look at these if you want and submit one or two: `qsub cloud.1950.pbs`
    f. This works fine and helps keep track of things, but can get tedious to write if there's a bunch of tasks to process.
    g. Note that it is possible to write scripts that write the submission scripts, and even scripts to submit them for you. It is important to think about how long each task will run for and how many tasks you have. This method works well for a few jobs, but with hundreds, or thousands of jobs, especially if the tasks are short, the scheduling overhead, how much time the scheduler spends tracking your job, setting it up, and cleaning up after it, becomes high relative to the work being done. **It is inefficient to run lots of short jobs.**

8. **Using a variable in the submission script**
    a. Another way to tackle this is to use a variable in the submission script.
    b. There's an example of this in `as_variable/`
    c. `cd ../as_variable`
    d. `ls`
    e. `more cloud_variable.pbs`
    f. Let's notice a few things here. First notice the warning:
    ```
    ################################################
    # This will only run one task, so DO NOT submit
    # lots of these at once if the tasks are short.
    ################################################
    ```
    g. As this says, it will only run one task at a time. While there are ways you could use this to submit hundreds of these at a time, DON'T! Especially if the tasks are short. Remember that lots of short jobs cause problems.
    h. This would be a good way to submit jobs if you have a few longer tasks to run.
    i. Notice also that it shows how to submit the jobs. And lastly notice where the $FILE variable is used in the line that runs the script.
    j. Go ahead and submit one or two jobs with this:
    ```
    qsub cloud_variable.pbs -v FILE=../Decades_of_PNAS/PNAS_1970.txt
    ```

9. **Running lots of tasks in one submission script**
    a. So far, our submission scripts have only done one thing. There is no reason to limit jobs to a single task. All of our tasks could easily be accomplished in one script. Especially since these tasks are so short, this would be much preferred as we avoid paying the scheduling overhead for each task.
    b. There are several examples of ways to do this in `many_short_tasks/`
    c. `cd ../many_short_tasks`
    d. Let's look first at `cloud_explicit_list.pbs`:
    ```
    more cloud_explicit_list.pbs
    ```
    e. Notice that this has one line for each file. This is only a bit better than writing all the individual files, but could be generated by a script or some other way. This method works well for running lots of short tasks, all in one job.
    f. Let's look at the next example: `more cloud_defined_list.pbs`
    g. This script defines a variable, $LIST, as all of the files in the directory `../Decades_of_PNAS/` that end in "`.txt`". The script loops over each of those, running each one after the other.
    h. Notice that this looks a lot like the `cloud_variable.pbs` script we looked at above, but in this case, it runs all the files in one job, rather than in individual jobs. This is MUCH preferred for short jobs since we don't need to incur the scheduling overhead for each job.
    i. Lastly the third way, `cloud_list.pbs`, keeps the flexibility of the `cloud_variable.pbs` script, while at the same time combining all of the tasks into a single job. You pass a list of files to run as the variable in the qsub command. To run:
    ```
    qsub cloud_list.pbs -v LIST=../Decades_of_PNAS/*.txt
    ```