Scaling Up Work with Batch (Non-interactive) Jobs

Working with code using GUI tools such as NoMachine is usually a default choice for many users. However, as the datasets grow larger, the task grows more complex, or the number of necessary repetitions increases, this approach is no longer scalable. Running batch (or background) jobs, where one does not interact with the program that is running, allows the user to scale one's work. On the HBSGrid cluster, for example, one can run hundreds of scripts at once, analyzing numerous data files simultaneously, or performing other parallelizable or automatable jobs.

This transition to background-only, non-GUI work may seem daunting: how does the program know what files to work on or write out? How do I monitor its progress? How do I even start the program, let alone hundreds of them? This session will demystify the process of transitioning to running batch jobs, give you several approaches to make this transition, and highlight a few useful tools.

Narrative

Setup: You a have a folder with a number of data files (5? 20?) and a script file. Or you have a parameter sweep that will run over 100s of combinations of values.

Code: script file that take input file with numbers, will output running sum after 10 values and also print out status line of how many #s have been seen so far.

Questions for attendees

- What do you like about using the GUI for running your script file?
- What do you (or would you) find limiting??

Objectives:

- Know how to launch batch jobs from the terminal
- Know how to set up inputs and outputs
- Monitor the progress of your job
- Scaling up to larger #s of files

Steps:

- Take baby steps: run your script code from the terminal
    - switch from GUI mode to launching jobs in the terminal - interactive
    - run something in batch
    - bjobs & options
- Know how to set up inputs and outputs
    - abstract code to accept any input file and test change
    - change code to make unique output file and test change
    - introduce error & use bhist & options to look at details and past info
- Better visibility with progress of jobs, job information, and job control
    - get email notifications on job starts/ends (-B -N)
    - making separate output and error logs (-o -e)

    - making unique job output files with %J
- Scaling up work
    - Cluster etiquette on scaling
        - bundle work: don't overwhelm scheduler
        - test small, run big
        - ensure resource requests are appropriate
    - bash for loops
    - job arrays
    - workflow tools
- For further investigation
    - using job arrays to handle
    - using #BSUB directives in job submissions scripts