Using the Computer Science Cluster

Alan Crawford
Lars Nesheim

October 31, 2018

Computer Science (CS) High-Performance Computing (HPC) Cluster

- CS cluster has over 5200 nodes.
- Designed to run large scale computing jobs in **batch mode** (non-interactive mode).
  - **Limited graphics**-based interactive computing services.
- Users should set up their code to run in batch mode.
- Users who primarily need graphics-based interactive computing should use Economics Department cluster, ISD services, or desktop computers.
- Users will benefit from learning some basix unix commands.

More information

- CS cluster webpage: `http://hpc.cs.ucl.ac.uk/how_to_login/`.
    - Get username and password from IT support,
- Econ GitHub page: `https://github.com/UCL/ECON-CLUSTER`
    - Contact Fatima or Andrew to be added to the UCL user group.
- If you need help,
    - CS support: `cluster-support@cs.ucl.ac.uk`
    - ECON IT support: `economics.it@ucl.ac.uk`

Obtain account

- Fill out and submit application form (see Fatima or Andrew).
- Wait 5-7 days and go to CS Cluster office: (4.20 Malet Place Engineering Building).
- They will set up two accounts
    1. CS departmental account (for remote access).
    2. CS Cluster account (to use cluster).
- By default, both accounts have same username and password.

Connect to cluster

There are three ways to connect to the cluster:

1. Connect using ThinLinc (graphical interface).
2. Connect using ssh (command line interface).
3. Connect using ftp (data transfer).

Connection option 1: ThinLinc

Connecting using ThinLinc is a two-step process:

1. Connect to a **CSRW** (Computer Science Remote Worker) using your **CS Department Account** username and password.
   - Instructions to download Thinlinc.
2. Then connect to one of the cluster login nodes ("vic" or "wise") using your **CS Cluster Account** username and password.
   1. Open terminal inside CSRW window.
   2. Then type one of the following
      - ssh vic
      - ssh uctpXXX@vic
   3. Enter your cluster password.

Connection option 2: ssh

- For remote access (i.e. when not connected to UCL network), connecting is a two step process:
  1. Connect to “tails”, “storm” or “jet” using your **CS Department Account** username and password.
  2. Connect to the cluster using your **CS Cluster Account** username and password.
- For access from within the UCL network (e.g. a desktop in Drayton House or the Econ HPC), skip step 1.

Option 2 Step 1: ssh to “tails”, “jet” or “storm”

- Open an ssh client (e.g. putty) or a terminal.
- Type
  - ssh -X uctpXXX@tails.cs.ucl.ac.uk
- Enter your CS dept. password.
- Note:
  - The option ‘-X‘ allows graphics to be forwarded from the CS cluster to your computer.
  - The username “uctpXXX” is that given when you are assigned your CS department account.
  - The text after the ‘@‘ is the address of the server.

Option 2 Step 2: ssh to cluster

1. Logon to the CS cluster using ssh.
2. Type
   - ssh -X uctpXXX@vic
3. Enter your CS cluster password.

Connection option 3: sftp (for file transfer)

- To transfer files to cluster storage use sftp instead of ssh.
  1. Open a terminal or an FTP client.
  2. Type
     - sftp uctpXXX@tails.cs.ucl.ac.uk
  3. Enter your CS department password.
- From tails, you can directly access the CS data storage.
- For example, if you have previously requested CS to create a storage directory named “PROJECT_X”, you can access this by typing:
  - cd /slash/economics/research/PROJECT_X
- We discuss requesting CS data storage below.

Typical workflow

1. Logon to cluster.
2. Transfer data and/or files to cluster (using ftp, email, or some version control software like git).
3. Edit files or code.
4. Write a script to submit your job to SGE (Sun Grid Engine).
   - SGE is the scheduler that manages allocation of jobs to nodes on the cluster.
5. Submit your job.
6. Monitor job progress if necessary.
7. Download results to your local computer (using ftp or email).

SGE sessions: two types

- There are two types of SGE sessions
    1. Interactive sessions
        - Start session with “qrsh”.
        - Interact with compute node directly on command line.
        - Good for developing code and testing things.
    2. Non-interactive sessions
        - Start session with “qsub”.
        - Job runs in background.
        - Good for running large scale, long running, or multiple jobs.
        - Cluster is optimised for this type of job.
- SGE jobs are command line only jobs.
- If you need to do GUI based interactive work, it is best to use an alternative cluster.

Interactive SGE sessions

- Use the “qrsh” command and specify **running time** and **memory**.
- For example, at the command line, type:
    - **qrsh -l h_vmem=1.9G,tmem=1.9G,h_rt=8:0:0**
- This command starts the session and requests 1.9 GB memory and 8 hours running time.
    - ‘qrsh‘ is the login command for an interactive session.
    - ‘-l‘ is a flag for resource requests for the interactive session.
- Resource options listed after the ‘-l‘ flag are:
    1. ‘h_vmem=XG,tmem=XG‘ requests X Gb of memory
    2. ‘h_rt= H:M:S‘ requests that the session run for ‘H‘ hours, ‘M‘ minutes, ‘S‘ seconds
- For further command line options for ‘qrsh‘ type: ‘man qrsh‘

User tips

- After starting SGE session, you need to load and open the software you require. See below for details.
- It may take a short while (1 - 5 minutes) to be allocated a node.
- To increase chances for quick allocation, request as little memory as necessary,
- For a small job, 2G is likely to be sufficient. For Matlab, request at least 4G.
- If you need a lot of memory (i.e. $X > 2G$), omit the ‘h_rt‘ option from your ‘qrsh‘ command.
- For example, to request a 14G session type:
    - qrsh -l h_vmem=14G,tmem=14G

Submitting batch jobs

To run a batch job:

- First, write a script (e.g. a text file names ‘job1.sh‘) detailing what resources to request from SGE and what commands/programs to run.
- Then, submit the script using the command ‘qsub‘.
- For example, first create the file ‘job1.sh’ and then type the command
  - qsub job1.sh
- Instructions for writing a script: job script instructions.

Some example command options to include in script

- Request memory and running time:

```
#$ -l h_rt=1:10:35
#$ -l tmem=1.9G,h_vmem=1.9G
```

- Each line containing SGE flags starts with '#$'.

Requesting a parallel environment

- To run a job using more than one worker, you need either to specify a parallel environment or submit multiple jobs using “tasks”.
  - We will discuss using “tasks” later.
- To specify the parallel environment, add the following lines to your shell script:

  ```
  #$ pe [pe_option] [NumWorkers]
  #$ -R y  #
  ```

  - replace [pe_option] option from next slide
  - Replace [NumWorkers] with an integer.
  - Second line reserves resources (needed when using multiple nodes).
- In most cases the best pe_option will be one of ‘orte‘, ‘smp‘, or ‘matlabpe2014b‘

Parallel environment options

- Options for parallel environments are
  - 'smp': single node with multiple workers
  - 'matlabpe2014b': parallel environment specific to Matlab. CS only support b release each year. Matlab 2015b is yet to be supported.
  - 'mpi': Old MPI interface
  - 'mpich': New MPI interface
  - 'orte': Distributed computing across nodes, but tries to cluster processes on nodes
  - 'para': Distributed computing across nodes, no clustering of processes
- Default option is 'orte'.

Examples

- Example 1: Use a single node with 4 workers

```
#$ pe smp 4
```

- Example 2: Use a single node with 16 workers

```
#$ pe smp 16
```

  - This example may “hang” in the queue for a long time; the scheduler will have to wait until a node with 16 cores is free.

- Example 3: 20 workers clustered on a few nodes

```
#$ pe orte 20
#$ -R y
```

Checking job status or deleting jobs

- For more details see Econ wiki.
- You can use the command ‘qstat‘ or ‘qstat-rn‘.
- Note: ‘qstat-rn‘ is only available if you have loaded the module ‘econutils‘ by typing

  ```
  module load econutils
  ```
- To delete a job with id 123456, type

  ```
  qdel 123456
  ```

Software available

- To find full details of software available on cluster go to cluster webpage or contact cluster support.
- The main software packages for economics are most easily used by using a module environment. See module environments.
- To see list of software currently available type

```
module avail
```

Software available

- Current list is

dot	matlab/r2012b	openmpi/gcc/1.10.0
econutils	matlab/r2013b	openmpi/gcc/1.8.1
gcc/5.2.0	matlab/r2014a	openmpi/intel/1.10.0
gcc/6.2.1	matlab/r2014b	openmpi/intel/1.8.1
git/2.8.3	matlab/r2015b	R/3.4.2
intel/composer/2013.1.117	module-info	stata/14
intel/composer/2015.1.133	modules	totalview/8.15.7-6
julia/0.4.5	nag/fll6i25dc	use.own
knitro/10.0.1-z	nag/fll6i26dc	nag/mbl6a23dml
knitro/10.1.2-z	nag/fll6i26df	nag/mbl6a24dnl

Loading modules

- Each software package depends on a large number of files.
- Loading the module file sets “environment” variables on the system so that the software will run correctly.
- To use modules, they must be loaded on every compute node on which you are running.
    - Interactive sessions: first start the interactive session using ‘qrsh‘, then load the modules you need.
    - Non-interactive sessions: include the ‘module load‘ commands in your batch job script or create a ‘.bash_rc‘ file to automatically load them at startup.

Examples

- MATLAB

```
module load gcc/5.2.0 nag/mbl6a24dnl matlab/r2015b
module load gcc/5.2.0 nag/mbl6a24dnl matlab/r2014b
module load gcc/5.2.0 nag/mbl6a24dnl matlab/r2014a
module load gcc/5.2.0 nag/mbl6a24dnl matlab/r2013b
module load nag/mbl6a23dml matlab/r2012b
```

- Stata

```
module load stata
```

- R

```
module load gcc/6.2.1 R/3.4.2
```

Examples

- Fortran. A module for Fortran has not been setup. Instead proceed as follows:
  1. Set INTEL_LICENSE_FILE.
  2. Run 2 Intel scripts that set Unix environment variables.
  3. Compile and link program.
  4. Run and/or debug your program.
  5. Steps 1 and 2 can be completed by following instructions on the next slide.
- Julia

```
module load git/2.8.3 julia
```

- Note: Julia users should speak to cluster support about getting setup on the CS cluster. Amongst other things, there are file limit, version control, library paths, and non-trivial batch-mode issues to address before job processing can begin.
- KNITRO

```
module load knitro/10.1.2-z
```

Set up Unix environment to use Intel Fortran 2018.

- The simplest way to do this is to create a file named ".bashrc" in your home directory with the following commands

```
# Sample .bashrc file
# Run script to start modules program
#     (do not forget the . at the beginning of the line)
. /share/apps/econ/Modules/modules.sh

# Export intel_license_file and run scripts to set Intel environment variables
export INTEL_LICENSE_FILE=28518@lic-intel.ucl.ac.uk
export INTEL_DIR="/share/apps/intel_parallel_studio_xe_2018/parallel_studio_xe_2018.3.051"
source ${INTEL_DIR}/bin/psxevars.sh
source ${INTEL_DIR}/compilers_and_libraries_2018/linux/mpi/intel64/bin/mpivars.sh
```

- Now, these commands will be automatically run every time you login.
- Then the fortran compiler command is "ifort".

Compiling fortran

- To compile:

```
ifort test.f90 -o test.exe.    # compile serial fortran pr
mpiifort test.f90 -o test_mpi.exe.  # compile MPI program
```

- Add flags for debugging or for optimization (type “man ifort” for help).
- Link to NAG libaries or other libraries if desired.

Running software

- Interactive sessions (command line versions)
  - Type the relevant command: ‘matlab -nodisplay‘, ‘stata-mp‘, ‘R‘, ‘ifort‘.
- Non-interactive sessions
  - Add the relevant command to your job script: ‘matlab -nodisplay‘, ‘stata-mp‘, ‘R‘, ‘ifort‘.
  - See examples below.
- Interactive GUI sessions
  - First logon to ‘jake‘ or ‘elwood‘ using ssh with the ‘-X‘ option for x-forwarding of graphics

    ```
    ssh -X uctpXXX@jake
    ```
  - Load the relevant modules.
  - Start your program using ‘matlab‘, ‘xstata-mp‘, ‘R‘, ‘ifort‘
  - Note: There are only two servers with x-forwarding enabled. You do not need to use SGE to access these servers. They can only be used for small scale temporary jobs.

Disk storage

- All accounts have a limited amount of 'local space' for storing working files. This space is limited (a few gigabytes), is not backed up, and should not be used to store important outputs or large datasets.
- Large scale storage (project store) with back ups is available on the CS SAN (storage area network).
- Storage can be made accessible to individuals or to groups.
- To request storage for a project named 'PROJECT_X', fill in the online storage request form.
- Space will then be allocated to you in the location '/SAN/economics/PROJECT_X' where 'PROJECT_X' is the name of your project.

Transferring Files to and from the CS HPC

- Use sftp.
- Use email.
- The SAN can be accessed directly from a CSRW or from ‘tails‘ at the location ‘/slash/economics/research/PROJECT_X‘

Cluster rules/best practice

- Never run jobs on head nodes (wise, vic)
- Store important data and/or large data on /SAN/economics/...
- Avoid high frequency reading/writing of large datafiles to/from disk
- If you need support, send email to cluster support, go visit them in person.
- Update wiki page with tips for colleagues about how to solve common problems.
- See cluster etiquette.

Memory management

- Most nodes have between 2 and 8GB per core. The physical machines vary from 4GB to 1TB.
- If you want to run a job requiring more than 47.2GB please contact request@cs.ucl.ac.uk.
- For the best job throughput please request less than 1.9G. That is:
- There are only a few machines with more than 64GB of memory. If you request more than 8G you will probably need to add a resource reservation to your job, unless the cluster is very empty.
- Although machines have 8, 16, 24, 48, 96, 128, 256 and 1TB of memory, in reality this translates into requestable memory of 7.9, 15.7, 23.5, 62.9, 47.2, 94.4GB. i.e. if you request 16G you will not be able to run on any of the 16G machines and your job will queue for longer.
- See memory management for more details.

Use SGE ‘task arrays‘ to run parallel jobs

- Add a line like the following to your script

  ```
  #$ -t 1-100
  ```

- Creates an array of 100 identical tasks each with an id: SGE_TASK_ID between 1 and 100.
- For example, use this to run 100 bootstrap replications or 100 Monte Carlo replications.
  - Highly efficient, highly robust way to run multiple jobs on cluster.
- Use SGE_TASK_ID in your program to control inputs/outputs
  - e.g. set random number seed to SGE_TASK_ID and save results with name based on task id.
- Task arrays can also be used to solve much more complex parallelization tasks.