UConn Biotechnology and Bioservices Center

Introduction to Scientific Computing

Bioinformatics Facility

Co-Heads : J.P. Gogarten, Paul Lewis
Facility Scientist: Jill Wegrzyn
Hardware/Software Manager: Jeff Lary

Biotechnology and Bioservices Center

Introduction to Scientific Computing

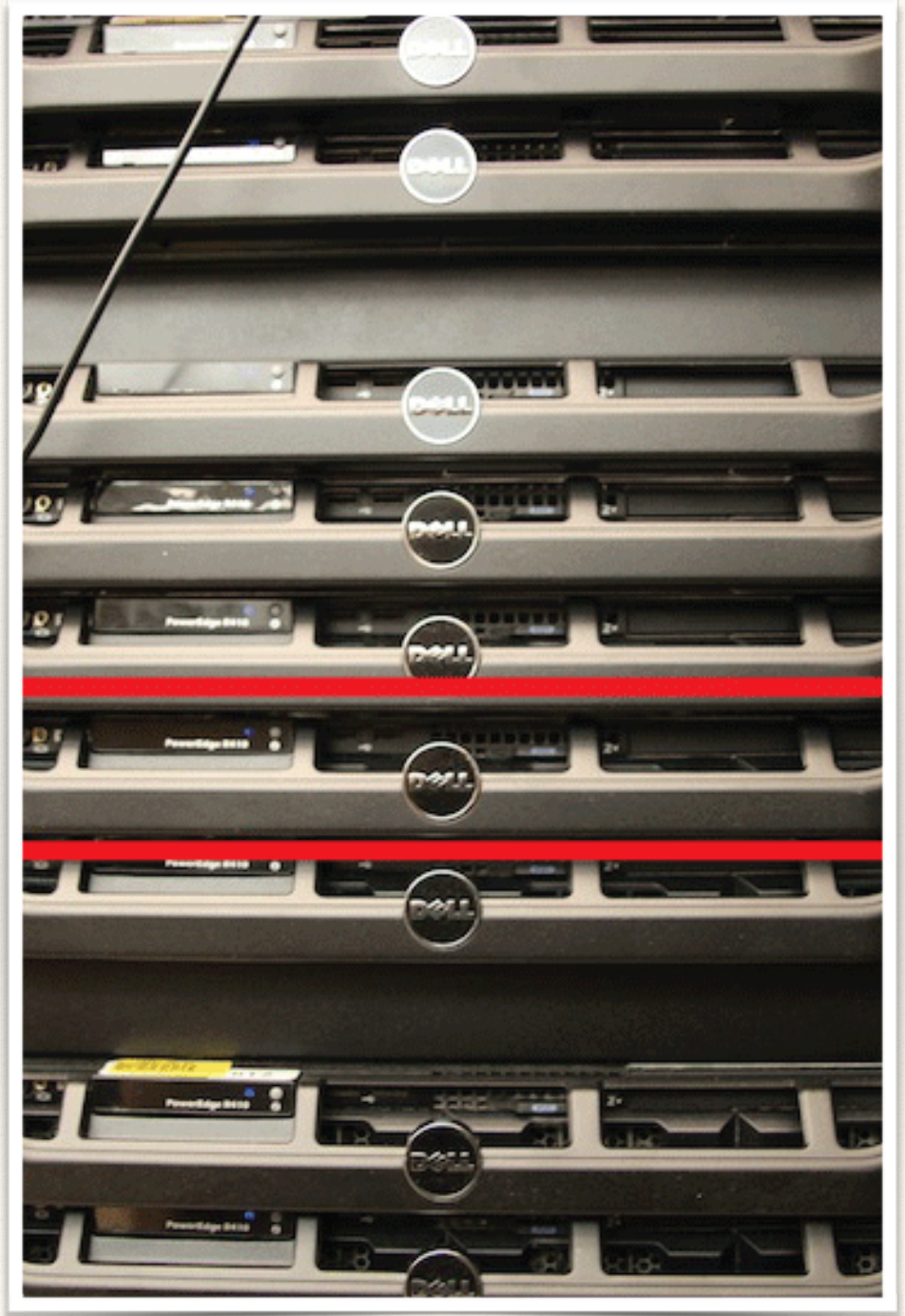
Applications for Client Side

Connecting to the Cluster

Basic UNIX Commands

Running Interactive Jobs

Submitting Scripts



Install Locally

- **SSH Client (Connect to the system)**
- File Transfer (sFTP and SCP)
- Plain Text Editor (Built-in text editors introduce platform-specific end of line characters)
- **Windows**
  - SSH Client
    - SSH
    - Putty (http://www.putty.org/)
  - sFTP client - Cyberduck (http://cyberduck.io)
  - Text Editor - Jedit (http://www.jedit.org/)
- **Mac**
  - SSH Client
    - Built-in Terminal or iTerm2 (http://www.iterm2.com/#/section/home)
  - sFTP Client - Cyberduck (http://cyberduck.io/)
  - Text Editor - Text Wrangler (http://www.barebones.com/products/textwrangler/download.html)

Connecting to the Server

- **17 node Dell Linux cluster running Redhat EL5 (UNIX)**
- Each compute node is equipped with 2 x Quad-core 2.53 GHz Intel Xeon processors and 32 GB of memory

Over a hundred software packages installed:

- Programming Languages: C, C++, Fortran90, Java, Perl, Python
- Statistical Packages: R
- Sequence comparison: NCBI BLAST (custom and standard databases)
- Phylogenetics, Sequence Alignment, Sequence Assembly, Metagenomics, Transcriptomics, Proteomics
- Complete List of Software: http://bioinformatics.uconn.edu/software/

- Need More Compute Power?
  - BECAT (HORNET): http://becat.uconn.edu/hpc/
  - Amazon Cloud (EC2): http://aws.amazon.com/ec2/
  - Cyberinfrastructure Initiatives (TACC): http://www.iplantcollaborative.org/about/cyberinfrastructure-overview

Learn More!

http://bioinformatics.uconn.edu

Shell and Unix

Shell is…

- Interpreter that turns text that you type (at the command line) into actions
- User Interface: takes commands from user
- Customization through shell-specific start-up files
  - Inherits global parameters first (parent file)
  - **~/.bashrc** (or ~/.cshrc or ~/.tcshrc)
- Two main flavors of Unix shells
  - Bourne (or Standard Shell): sh, ksh, **bash**, zsh ($)
  - C shell : csh, tcsh (%, >)

Navigating Unix

- *Directories* are analogous to Windows *folders*
  - Delimited by / rather than \ as on a PC
- When you first log in, current directory is called your *home* directory
- Directory where you are located at any given time is your *working* directory
  - pwd means *print working directory*
- To create a subdirectory, use mkdir (*make directory*)
  - mkdir sub1, where sub1 is the subdirectory name
- Questions on how to use any Unix command?
  - man *command*

Navigating Unix

- ls lists all files and directories in your current directory
- Shorthand directory names:
  - ~ home directory
  - . current directory
  - .. one level above current directory
- cd dirname (*change directory*) moves you to dirname
  - cd sub1
  - pwd
- cp *from to* will copy a file
  - *from* is the file name you're copying from
  - *to* is either a file name or a directory
    - if it's a file name, the copy will be given that name
    - if it's a directory, the file will retain the old name and be placed in the specified directory

Navigating Unix

- rmdir *dirname* # Removes empty directory
- rm *filename* # Removes file name
- rm -r *dirname* # Removes directory including its content
- mv *from to* # Renames directories or files
- mv *from to* # Moves file/directory as specified in path
- history # shows all commands you have used recently
  - Up and down keys to scroll through commands at prompt
- more *file* # views text, use space bar to browse, hit 'q' to exit
- less *file* # a more versatile text viewer than 'more', 'q' exits, 'G' end of text, 'g' beginning, '/' find forward, '?' find backwards
- cat *file* # concatenates files and prints content to standard output
- grep *pattern file* # provides lines in 'file' where pattern 'appears',

Unix Redirects

By default, UNIX commands read from standard input (STDIN) and send their output to standard out (STDOUT).

You can redirect them by using the following commands:

- ls > *file* # prints ls output into specified file
- *command* >> *file* # appends output of one command to file
- grep *pattern file* | wc # Pipes (|) output of 'grep' into 'wc'

File Permissions

ls -al # shows something like this for each file/dir: drwxrwxrwx

- d: directory
- rwx: read write execute
  - first triplet: user permissions (u)
  - second triplet: group permissions (g)
  - third triplet: world permissions (o)

chmod ugo-rwx *file* *(remove all permissions from all three groups)*

- '+' causes the permissions selected to be added
- '-' causes them to be removed
- '=' causes them to be the only permissions that the file has

chmod ug+rx *file*

Editors

Compress files, compare, sort, search, calculate, and more:

- http://bioinformatics.uconn.edu/unix-basics/
- Count the number of unique lines:
  - cat file.txt | sort | uniq | wc -l
- Find the number of lines shared by 2 files:
  - sort file1 file2 | uniq -d

Create and edit files on the server:

- Emacs, Vi (alias vim), nano
  - http://bioinformatics.uconn.edu/vim-guide/

Cluster Etiquette

- **NEVER** run anything on the head node of a cluster (default login node).
- Keep track of what you are running
- There are no official limits on the number of jobs you can run on the cluster but refrain from using all the nodes a the same time.
- Each nodes have their own hard drive (scratch drive). It is advised when possible to run and write your output files on this drive (/scratch), then copy the file back to your home directory when done.

qhost

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
 -------------------------------------------------------------------------------
 global                 -               -     -       -       -       -       -
 bbcsrv3                linux-x64       8  0.18   35.4G    2.0G    2.0G     0.0
 compute-1-0            linux-x64       8  1.00   31.5G  857.7M    2.0G     0.0
 compute-1-1            linux-x64       8  0.03   31.5G 1017.8M    2.0G     0.0
 compute-1-10           linux-x64       8  2.33   31.5G    3.8G    2.0G     0.0
 compute-1-11           linux-x64       8  0.05   31.5G  989.8M    2.0G     0.0
 compute-1-12           linux-x64       8  0.00   31.5G 1018.1M    2.0G     0.0
 compute-1-13           linux-x64       8  0.00   31.5G  903.9M    2.0G     0.0
 compute-1-14           linux-x64       8  1.00   31.5G    8.4G    2.0G     0.0
 compute-1-15           linux-x64       8  2.14   31.5G    1.0G    2.0G     0.0
 compute-1-16           linux-x64       8  0.10   31.5G 1017.8M    2.0G     0.0
 compute-1-2            linux-x64       8  0.00   31.5G    1.0G    2.0G     0.0
 compute-1-3            linux-x64       8  0.00   31.5G 1001.1M    2.0G     0.0
 compute-1-4            linux-x64       8  0.00   31.5G  720.6M    2.0G     0.0
 compute-1-5            linux-x64       8  0.13   31.5G  770.4M    2.0G     0.0
 compute-1-6            linux-x64       8  2.09   31.5G  876.3M    2.0G     0.0
 compute-1-7            linux-x64       8  0.14   31.5G  572.3M    2.0G     0.0
 compute-1-8            linux-x64       8  2.01   31.5G  619.3M    2.0G    17.4M
 compute-1-9            linux-x64       8  0.14   31.5G  790.8M    2.0G     0.0
```

Interacting with SGE

Users submit an interactive (qrsh/qlogin) or a batch job (qsub) to the Sun Grid Engine (SGE).

- For an interactive job: (qrsh/qlogin) - no hard limit on job time
  - If there are resources immediately available, job gets started
  - Otherwise the user is informed about the lack of resources and job gets abandoned.
- For a batch job: (qsub)
  - If there are resources immediately available the job gets started
  - Otherwise the job is kept in a queue until resources to execute it becomes available.
- Jobs are always passed onto the available executing hosts
- Records of each jobs progress through the system are kept and reported when requested.
- qrsh -l hipriority=TRUE = special queue for jobs < 6 hrs

Interacting with SGE

A submitted job will either be:

1. **Waiting in the queue**
2. **Executing**
3. **Completed and left the SGE scheduling system**

In order to monitor the progress of your job while in **states (1) and (2)** use the qstat command that will inform you if the job is still waiting or started executing.

While executing **(state 2):**

use qstat –j job_number to monitor the jobs status including time and memory consumption.

Better still use qstat –j job_number | grep mem that will give time and memory consumed information.

Finished executing **(state 3)**:

qacct is the only command that may be able to tell you about the past jobs by referring to a database of past usage.

qacct -j job_number

Job Status

qstat command will list all the jobs in the system that are either waiting to be run or running. This can be a very long list !

qstat –f full listing (even longer)

qstat –u *username* (specific to user)

qstat –f –u *username* (detailed information)

Status of the job is indicated by letters in qstat listings as:

qw	waiting	**t**	**t**ransfering
r	**r**unning	**s,S**	**s**uspended
R	**r**estarted	**T**	**t**hreshold

Job Removal

qdel command will remove from the queue the specified jobs that are waiting to be run or kill jobs that are already running:

Individual job:

qdel 15112

List of jobs:

qdel 15154 15923 15012

All jobs running or queueing under a given username:

qdel –u *username*

QSub

qsub for single and multiple node jobs:

qsub scriptname.sh

Create script with text editor or via vi or emacs on the server (single node):

```
#!/bin/bash
#$ -S /bin/bash
cd $HOME/username
#$ -cwd            # tells GE to execute the job from  the  current  working  directory
perl do_blast.pl
```

Qsub

Create script for multiple nodes:

```
#!/bin/bash
 # ---------------------------
 #$ -S /bin/bash
 # job name
 #$ -N blastp_job
 #
 # output file
 #$ -o $HOME/blastp_$JOB_ID.out
 # error file
 #$ -e $HOME/blastp_$JOB_ID.err
 #
 # pe request
 #$ -pe smp 6
 #
 cd $HOME/my-sequence/
 #$ -cwd
 blastp -query my-fasta.fa -db $HOME/data/my-db -num_alignments 1 -num_descriptions 1 -num_threads 6 -out my-
results
```

Bioinformatics Facility

- Cluster is available for FREE (both hardware and technical assistance) to affiliates of UConn.
- Feel free to contact us for assistance with your project, software requests, custom code, etc.
- If you do use the resources or plan to use the resources of the center, talk to us!
- Contact:
  - bioinformatics@uconn.edu