CBC Data Therapy

High-Performance Computing Basics

Xanadu Cluster
Development of models begins at small scale.

Working on your laptop is convenient, simple.

Actual analysis, however is slow.

“Scaling up” typically means a small server or fast multi-core desktop.

Speed exists, but for very large models, not significant.

Single machines don’t scale up forever.
For the larger problems/models, a different approach is required
High-performance computing (HPC)

High-Performance computing involves mainly distinct computer processors working together on the same problem/calculation.

Large problem/calculations are divided into smaller parts and distributed among the many computers.

HPC is a cluster of quasi-independent computers which are coordinated by a central scheduler.
Typical HPC cluster

- **Login**
- **Scheduler**
- **File Server**
- **Computes**

**External connection**

- **submit-int**
- **submit-ext**

**Directory Paths**

- `/$HOME`
- `/linuxshare`

**Submission Commands**

- `submit-int`
- `submit-ext`

**Software**

- **SGE:** Sun Grid Engine
- **PBS:** Portable Batch System
- **SLURM:** Simple Linux Utility for Resource Management

**Network Connection**

- High-speed network (10GigE / Infiniband)
Performance comes at a price: **Complexity**

- Applications must be written specifically to take advantage of distributed computing
- Debugging becomes more of a challenge

Applications must be written specifically to take advantage of distributed computing.

- Explicitly split your problem into smaller “chunks”
- “Message passing” between processes
In Summary

• Why HPC
  – A huge number of computational and memory requirements
  – Cannot be afforded by a PC efficiently
  – Speeds and feeds are the keywords

• Who uses High-Performance Computing
  – Research institutes, universities and government labs
    • Weather and climate research, bioscience, energy, military etc.
  – Engineering design: more or less every product we use
    • Automotive, aerospace, oil and gas explorations, digital media, financial simulation
    • Mechanical simulation, package designs, silicon manufacturing etc.

• Similar concepts
  – Parallel computing: computing on parallel computers
  – Super computing: computing on world 500 fastest supercomputers
Parallel Computing on a Large Number of Servers is More Efficient than using Specialized Systems
HPC at UCONN

BBC (Storrs): SGE  - Research and Teaching
HPC1 (UCH) : PBS  - Advanced Research
Xanadu (UCH): SLURM  - Advanced research
Connecting to Xanadu

Mac: Terminal: ssh username@xanadu-submit-ext.cam.uchc.edu

Windows: Putty

Open Putty it will open window 1.
1. Provide host name e.g. username@xanadu-submit-ext.cam.uchc.edu
2. Expand SSH tab and select X11 (shown in window 2)
3. Enable X11 forwarding by selecting it. (window 2)
4. Scroll up the left panel and select Session. (window 1)
5. Name your session e.g. BBC_cluster and click save tab to save.
6. Your session name should appear in saved sessions. Double click on your session name to connect to server with SSH session.
Login: From outside the network

Use VPN (Open Pulse secure)

1. Open Pulse secure
2. Add new connection
3. Set Server URL to: sslvpn.uconn.edu
4. Save
5. Connect and login with NetID and Passwd
Login: (using terminal on mac)

```
$ ssh vsingh@xanadu-submit-ext.cam.uchc.edu
vsingh@xanadu-submit-ext.cam.uchc.edu's password: _
```

Logged on ext-submit node:

```
$ ssh vsingh@xanadu-submit-ext.cam.uchc.edu
vsingh@xanadu-submit-ext.cam.uchc.edu's password:
Last failed login: Thu Jun 1 17:04:42 EDT 2017 from d88h208.public.uconn.edu on ssh:notty
There were 3 failed login attempts since the last successful login.
Last login: Thu Jun 1 09:11:14 2017 from 137.99.88.208
xanadu-submit-ext - $ _
```
Common Linux commands

`pwd` : Present Working directory
`cd destination` : Change directory to destination
`cd` : Change directory to $HOME
`ls` : List contents of directory
`cp source/file destination/file` : Copy file from source in destination folder
`mv source/file destination/file` : Move file from source to destination folder
`mv name name2` : Rename file from name to name2
`touch filename` : Create an empty file with name filename
`mkdir directory` : Make directory
`rm file` : delete file
`rm -r directory` : delete file with its content
`~` : Home directory
`cat` : Read contents of file
`less` : Contents of file, scroll, q to quit it
`head -10 file` : first 10 lines of file
`tail -10 file` : Bottom 10 lines of file

Resources: [http://linuxcommand.org/writing_shell_scripts.php](http://linuxcommand.org/writing_shell_scripts.php)
Common Linux commands to edit file

**vi filename**

- **esc i**: Open file in vim to edit
- **esc q !** : Insert or edit file
- **esc w q !** : quit file without saving changes
- **esc dd** : Save and quit file
Environment Modules:
The Environment Modules package provides for the dynamic modification of a user's environment via modulefiles.

module avail : List modules that are available
module load modulefile : Loads the module to user environment
module list : List modules that are loaded
module unload modulefile : unloads module from user environment
module display modulefile : Displays information on module
swap [modulefile1] modulefile2 : Switch loaded modulefile1 with modulefile2.
### Xanadu Resources

#### Partitions

```
xanadu-submit-ext ~ $ sinfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
general* up infinite 4 mix xanadu-[20-22,25]
general* up infinite 13 idle xanadu-[01-11,23-24]
xeon up infinite 11 idle xanadu-[01-11]
amd up infinite 4 mix xanadu-[20-22,25]
amd up infinite 2 idle xanadu-[23-24]
himem up infinite 4 idle xanadu-[30-33]
xanadu-submit-ext ~ $```

<table>
<thead>
<tr>
<th>NODELIST</th>
<th>NODES</th>
<th>PARTITION</th>
<th>STATE</th>
<th>CPUS</th>
<th>S:C:T</th>
<th>MEMORY</th>
<th>TMP_DISK</th>
<th>WEIGHT</th>
<th>AVAIL_FE</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>xanadu-01</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-02</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-03</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-04</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-05</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-06</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-07</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-08</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-09</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>257767</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-10</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>128532</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-11</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>36</td>
<td>2:18:1</td>
<td>128532</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-20</td>
<td>1</td>
<td>general*</td>
<td>mixed</td>
<td>32</td>
<td>4:8:1</td>
<td>128745</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-21</td>
<td>1</td>
<td>general*</td>
<td>mixed</td>
<td>32</td>
<td>4:8:1</td>
<td>128745</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-22</td>
<td>1</td>
<td>general*</td>
<td>mixed</td>
<td>32</td>
<td>4:8:1</td>
<td>257760</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-23</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>32</td>
<td>4:8:1</td>
<td>257760</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-24</td>
<td>1</td>
<td>general*</td>
<td>idle</td>
<td>32</td>
<td>4:8:1</td>
<td>249696</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-25</td>
<td>1</td>
<td>general*</td>
<td>mixed</td>
<td>32</td>
<td>4:8:1</td>
<td>209380</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
</tbody>
</table>
## Partition: himem

```
xanadu-submit-ext ~ $ sinfo -N -l -p himem
Thu Jun 1 22:42:13 2017

<table>
<thead>
<tr>
<th>NODELIST</th>
<th>NODES</th>
<th>PARTITION</th>
<th>STATE</th>
<th>CPUS</th>
<th>S:C:T</th>
<th>MEMORY</th>
<th>TMP_DISK</th>
<th>WEIGHT</th>
<th>AVAIL_FE</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>xanadu-30</td>
<td>1</td>
<td>himem</td>
<td>idle</td>
<td>32</td>
<td>4:8:1</td>
<td>515792</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-31</td>
<td>1</td>
<td>himem</td>
<td>idle</td>
<td>32</td>
<td>4:8:1</td>
<td>515792</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-32</td>
<td>1</td>
<td>himem</td>
<td>idle</td>
<td>32</td>
<td>4:8:1</td>
<td>515792</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
<tr>
<td>xanadu-33</td>
<td>1</td>
<td>himem</td>
<td>idle</td>
<td>32</td>
<td>4:8:1</td>
<td>515792</td>
<td>15620</td>
<td>1</td>
<td>(null)</td>
<td>none</td>
</tr>
</tbody>
</table>
```
Composing a script for cluster

Script:
1. Resource request
   • number of CPUs,
   • computing expected duration,
   • amounts of RAM or disk space, etc
2. Job commands
   • describe tasks that must be done, software which must be run
Resource request:

```
#!/bin/bash
#SBATCH --job-name=myscript
#SBATCH -n 1 Request number of cores for your job
#SBATCH -N 1 This line requests that the cores are all on node.
#SBATCH --partition=general
#SBATCH --mail-type=END
#SBATCH --mail-user=first.last@uconn.edu
#SBATCH -o myscript.out
#SBATCH -e myscript.err
```

- `#!/bin/bash`: This is the shebang line that specifies the interpreter to use.
- `#SBATCH --job-name=myscript`: This sets the job name to `myscript`.
- `#SBATCH -n 1`: Requesting one core for the job.
- `#SBATCH -N 1`: Requesting that all cores are on the same node.
- `#SBATCH --partition=general`: Specifies the SLURM partition to use, which will be the general partition in this case.
- `#SBATCH --mail-type=END`: Mailing options to indicate the state of the job. In this instance, it will send a notification at the end.
- `#SBATCH -o myscript.out`: Specifies the file to which standard output will be appended.
- `#SBATCH -e myscript.err`: Specifies the file to which standard error will be appended.

The `--mail-type=END` parameter is used to send a notification at the end of the job. Only change this to `>1` if you know your code uses a message passing protocol like MPI. SLURM makes no assumptions on this parameter -- if you request more than one core (`-n > 1`) and forget this parameter, your job may be scheduled across nodes; and unless your job is MPI (multinode) aware, your job will run slowly, as it is oversubscribed on the master node and wasting resources on the other(s).

Only change this to >1 if you know your code uses a message passing protocol like MPI. SLURM makes no assumptions on this parameter -- if you request more than one core (`-n > 1`) and forget this parameter, your job may be scheduled across nodes; and unless your job is MPI (multinode) aware, your job will run slowly, as it is oversubscribed on the master node and wasting resources on the other(s).
#!/bin/bash
#SBATCH --time=10-01:00:00 # days-hh:mm:ss
#SBATCH --job-name=masurca_KG
#SBATCH --mail-user=user@uconn.edu
#SBATCH --mail-type=ALL
#SBATCH --comment=dataset_wuith_jump_libraries
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=10240 # 10GB
or #SBATCH --mem=100G
#SBATCH -o filterGTF-%j.output
#SBATCH -e filterGTF-%j.error
Job commands:
They are regular linux/module commands

```
echo "Hello World"
```

Final script:
```
#!/bin/bash
#SBATCH --job-name=myscript
#SBATCH -n 1
#SBATCH -N 1
#SBATCH --partition=general
#SBATCH --mail-type=END
#SBATCH --mail-user=first.last@uconn.edu
#SBATCH -o myscript.out
#SBATCH -e myscript.err

echo "Hello World"
```

save the script as myscript.sh
Script submission and other commands

sbatch myscript.sh : Submit script for execution
squeue : Status of Jobs currently running on cluster (all users)
squeue -j jobIDNUMBER : Status of job with jobIDNumber
squeue -u UserID : Status of all the jobs submitted by user
scancel jobID_number : Delete job with jobID_number
scancel -u UserID : Delete all the jobs of a user
#!/bin/bash
#SBATCH --mail-user=user@uconn.edu
#SBATCH --mail-type=ALL
#SBATCH --ntasks=1
#SBATCH --time=00:15:00
#SBATCH --mem=1G
#SBATCH --array=1-1002%100
#SBATCH --output=fastqc_%A_%a.out

cd /NGSseq/data
module load fastqc/0.11.5

arrayfile=`ls | awk -v line=${SLURM_ARRAY_TASK_ID} '{if (NR == line) print $0}'`

fastqc $arrayfile

This line will create 1002 jobs, but it instructs slurm to limit the total number of simultaneously running jobs to 100. This avoids swamping the queue, and shares bursting level with others in the group.

This will create 1002 files to catch stdin, stdout and stderr for each respective job in the array. If the array job ID is 23678, we will fine 1002 files starting with fastqc_23678_1.out ... fastqc_23678_1002.out

Start: Slurm job ID and increase with each array job
Slurm job ID
Array job ID : 1-1002

This will list all the files from the directory (/NGSseq/data) and then pick up one file at a time and then run it through fastqc application.
Thank you