2nd Hands-On Tutorial on Linux Cluster Computing:

JOBS SCRIPTS AND DATA MANAGEMENT

Efstratios Efstathiadis, Loren Koenig, Eric R. Peskin

Tuesday January 14th, 2014, 2:00 PM to 4:00 PM

Translational Research Building (TRB)
1st floor, Seminar Room 120
2nd Hands-On Tutorial on Linux Cluster Computing:

**JOBS SCRIPTS AND DATA MANAGEMENT**

Efstratios Efstathiadis, Loren Koenig, Eric R. Peskin

- Quick overview of local HPC resources
- Announcements
- Preliminaries (login, etc.)
- Shell Scripts (bash)
- Batch System (SGE)
- Data Transfers

Slides:

HPC Cluster: **Phoenix** (rebuilt in 2013)

- **2 Head “Login” Nodes** *(phoenix.med.nyu.edu)*
  - 2 Intel Sandy Bridge 2.6GHz CPUs, 16 processing cores, 128GB Random Access Memory (RAM) each.

- **64 Compute “Worker” Nodes**
  - 2 Intel CPUs, 16 processing cores, 128GB RAM each, 8 GB/core.

- **5 GPU Compute Nodes**
  - 2 Intel CPUs, 16 processing cores, 128 GB RAM each
  - NVIDIA Tesla Kepler K20 GPU accelerator (2496 cores, 1.17 TFLOPS)

- **1 High Memory Compute Node**
  - 4 Intel CPUs, 32 processing cores, 1 TB RAM

- **Networks:**
  - Private 10Gbit Message Passing Network
  - Private 10Gbit Data (Input-Output) Network to Primary Isilon data storage cluster.
  - Private 1 Gbit Administrative Network
  - Private 1 Gbit Management/Monitoring Network
  - Public 10Gbit interface to the Enterprise Campus Network

**Total CPU Cores:**
1,170

**Total GPU Cores:**
12,480

**Total RAM:**
10TB

**Performance:** (est.)
28TFLOPS
HPCF Data Storage

1 PetaByte ($10^{15}$ Bytes) raw High-throughput, Scalable, EMC/Isilon Data Storage Clusters

- **Primary Storage Cluster**
  - Directly attached to all HPC cluster nodes via a dedicated, private 10Gbit network.
  - Home directories and group-shared data
  - Accessible on the campus network via 1Gbit and 10Gbit interfaces
  - Instrument/Sensor data (Sequencers, slide scanners, etc.)
  - File System Snapshots

[Diagram of Primary and Backup storage clusters]
Upcoming Seminar:

Cloud BioLinux: Pre-configured, On-Demand, Computing for Genomics and Beyond

Dr Constantinos Krampis
J Craig Venter Institute

Friday January 17th, 2014
2:00pm – 3:00pm

NYULMC Skirball Institute
3rd Floor Seminar Room
540 First Avenue, New York, NY 10016
Sackler Institute for Graduate Biomedical Studies

Course #: BMSC-GA 4448
Class #: 20801
Credits: 3

High Performance Computing in Biomedical Informatics

Start date: Thursday, February 6\textsuperscript{th}, 2014 at 2:30pm

Translational Research Building (TRB)
Conference Room 717, 7\textsuperscript{th} floor
227 East 30\textsuperscript{th} Street, New York, NY 10016

http://sackler.med.nyu.edu/graduate/grad-student-info/online-student-handbook/course-catalog
2nd Hands-On Tutorial on Linux Cluster Computing:
Jobs Scripts and Data Management

HPCF Contacts:  hpc_admins@nyumc.org

HPCF Web Site:  http://www.nyuinformatics.org/services/hpcf

HPCF Acknowledgement Statement:
"This work has utilized computing resources at the High Performance Computing Facility of the Center for Health Informatics and Bioinformatics at NYU Langone Medical Center."
2nd Hands-On Tutorial on Linux Cluster Computing:
Jobs Scripts and Data Management

- Quick overview of local HPC resources & Announcements
- Preliminaries (login, etc.)
- Shell Scripts (bash)
- Batch System (SGE)
- Data Transfers
Login to phoenix.med.nyu.edu (from Linux/Mac)

➔ Make sure you are connected to the LMCmobile Wireless Network

$ ssh username@phoenix.med.nyu.edu

Username: your NYULMC Kerberos id

Note: If your private key does not have the default filename, you must use the –i key flag

$ ssh –i private_key username@phoenix.med.nyu.edu
Login to phoenix.med.nyu.edu (from Windows)

- Make sure you are connected to the **LMCmobile** Wireless Network

![PuTTY Configuration](image)

1. **Session**
   - Specify the destination you want to connect to
     - Host Name (or IP address)
     - Port
     - Connection type: SSH

2. **Saved Sessions**
   - Default Settings

3. **Save**
Remote Access

(1) VPN Access
http://www.atnyulmc.org/

(2) SSH Access (Requires MCIT approval)
http://www.nyuinformatics.org/services/hpcf/remote-access
2nd Hands-On Tutorial on Linux Cluster Computing:
Jobs Scripts and Data Management

• Quick overview of local HPC resources & Announcements
• Preliminaries (login, etc.)
• Shell Scripts (bash)
• Batch System (SGE)
• Data Transfers
Some Shell Commands

• First try the following in the shell:
  
  \texttt{wc -l /ifs/data/tutorials/linux2/data/s1\_sequence.fastq}
  \texttt{wc -l /ifs/data/tutorials/linux2/data/s2\_sequence.fastq}
  \texttt{wc -l /ifs/data/tutorials/linux2/data/s3\_sequence.fastq}

• \texttt{wc -l} counts the number of lines in a file.
Same Commands as a Script

#!/bin/bash
echo "==================================="
date
wc -l /ifs/data/tutorials/linux2/data/s1_sequence.fastq
wc -l /ifs/data/tutorials/linux2/data/s2_sequence.fastq
wc -l /ifs/data/tutorials/linux2/data/s3_sequence.fastq
date
echo "==== The End ============"
Copy Example Script

• Now copy, examine and run script:
  mkdir tutorial
  cd tutorial
  cp /ifs/data/tutorials/linux2/scripts/script1.sh .
  cat script1.sh
  chmod +x script1.sh
  ./script1.sh

• Have fun running it again and again!
  ./script1.sh
  cd ..
  tutorial/script1.sh
  /ifs/data/tutorials/linux2/scripts/script1.sh (same?)
About that Script…

#!/bin/bash

echo "=====================

date

cwc -l /ifs/data/tutorials/linux2/data/s1_sequence.fastq

cwc -l /ifs/data/tutorials/linux2/data/s2_sequence.fastq

cwc -l /ifs/data/tutorials/linux2/data/s3_sequence.fastq

date

echo "==== The End ==========

Hard-coded. Repetitive. What if files move?
Script with Variable

#!/bin/bash

echo "=====================

date

# Define Variables
DIR="/ifs/data/tutorials/linux2/data"

echo "File Directory is: $DIR"

wc -l $DIR/s1_sequence.fastq
wc -l $DIR/s2_sequence.fastq
wc -l $DIR/s3_sequence.fastq

date

Can change in one place and use in many places. Can also give meaningful name.
About Variables

• Bash variables are named entities that can hold string values for the duration of the session/script.
• Defining a variable: `varname=value`
  – Case sensitive – `$Dir` and `$DIR` are different variables!
• Evaluating ("referencing") a variable: `_${varname}_`
  – e.g., `echo ${DIR}`
• Bash variables support integer arithmetic:
  – `$((var_1 + var_2))`, e.g.: `a=3; b=4; sum=$((a+b))`
• Remove variable definition: `unset varname`
Fun with Variables

- tutorial=/ifs/data/tutorials/linux2
- ls ${tutorial}/scripts
- ls ${tutorial}/data
Environment Variables

• Like shell variables, but with a wider scope:
  – Available to other programs you call.

• Defining an environment variable:
  – `export VARNAME=value`

• By convention, environment variables have uppercase names.
Environment Variable Example

#!/bin/bash
#$ -S /bin/bash
module load freesurfer/5.3.0
export SUBJECTS_DIR=${HOME}/fs_subjects
x=5
recon-all -all -subjid bert
Because of the export, SUBJECTS_DIR can impact recon-all, but x cannot.
Predefined shell variables

• Environment Variables that are created when bash is first started and control some of the shell features.
• E.g., $OSTYPE, $PATH, $HOME, $USER
• Try the following:
  - `echo $USER`
  - `echo $PWD`
  - `echo $?`
  - `echo $HOME`
PATH

- Contains the user’s (your) command search path.
- A list of directories, separated by “:”
- Given a command name, shell scans list (in order) to find commands.
- Try: `echo $PATH`
- The `which` command searches the paths of executables in $PATH. Try:
  ```
  which cat
  which bwa
  ```

```bash
oldpath=$PATH
realdir=/local/apps/bwa/0.7.5a
export PATH=${oldpath}:${realdir}  # but there’s a better way … modules …
```

```
echo $PATH
bwa
which bwa
```
Exercise

local=${HOME}/mybin
mkdir ${local}
cat > ${local}/bwa
#!/bin/bash

echo "OOOPS!"
(Control-D)
chmod +x ${local}/bwa
export PATH=${oldpath}:${local}:${realdir}
bwa
which bwa

After previous slide, try this. →
What happens?
Why?
Environmental Impact of Modules

• Try:
  
  echo $PATH
  module load bwa
  echo $PATH
  which bwa
  module show bwa
  module unload bwa
  echo $PATH

• Brave? Try:
  module show qiime

• Craving simplicity? Try:
  module show igenomes
Environmental Impact of Modules

• Each module sets variable \texttt{APPNAME\_ROOT}
  – e.g., \texttt{BWA\_ROOT}, \texttt{TOPHAT\_ROOT}, etc.
  – useful if documentation refers to subdirectory of “installation directory”
  – For igenomes, that’s all the module does!
#!/bin/bash

```
# Read Command Line Argument
DIR=$1

echo "File Directory is: $DIR"

wc -l $DIR/s1_sequence.fastq

wc -l $DIR/s2_sequence.fastq

wc -l $DIR/s3_sequence.fastq

date
```

$1 is first argument to script. ($2 would be second, $3 third,… $* would be all the arguments)

Try it:
```
cd tutorials
cp script1.sh script3.sh
eemacs script3.sh
... make changes ...
./script3.sh /ifs/data/tutorial/linux2/data
```

Try creating new files s1_sequence.fastq, etc. in some other directory and then calling

```
./script3.sh myotherdir
```
General Form of Linux Commands

```
your command

- echo -n Hello world!
```

name of the command

arguments

options/switches

other arguments

separated by spaces:
Commands, Programs, and Processes

Commands identify programs; entering a command invokes (calls, executes) the program.

An individual execution of a program is a process.

If the same program is executed multiple times, there exist multiple processes of the same program.

How does input of data to and output of data from processes work?
General Functional View on Linux Command Line Processes

- Environment variables
- Command line arguments
- Standard input "stdin"
- Standard output "stdout"
- Standard error output "stderr"
- Return code
Redirection: Input

Input can be redirected so as to make `command` read from `file`:

```
command < file
```

ex.:

```
wc < s1_sequence.fastq
```
Redirection: Output

Output can be redirected so as to make \textit{command} write to \textit{file}:

\begin{verbatim}
command > file
\end{verbatim}

ex.:

\begin{verbatim}
ls -l > filelist.txt
\end{verbatim}
Redirection: To Different Standard Stream

Standard output can be redirected so as to appear as standard error:

```
command >&2
```

Ex.:

```
echo Error: This example sucks. >&2
```

(useful in producing controlled error messages)
Pipelines
("Piping")

Pipelines can be created by feeding the output from one command into the input to another command:

```
command1 | command2
```

Ex.:
```
cat s1_sequence.fastq | wc
```
Compound Commands: AND lists

AND list: If and only if the first command succeeds, the second command is executed:

```
command1 && command2
```

Ex.:

```
[ -f ${file} ] && process ${file}
```

(useful, e.g., for checking preconditions and chains of dependant commands)
**Compound Commands: OR lists**

OR list: If and only if the first command fails, the second command is executed:

```
command1 || command2
```

Ex.:

```
[ -f ${file} ] || echo missing >&2
```

(useful, e.g., for error reporting)
Compound Commands: For Loop

The variable varname is set to each of the words in list in turn, and command is executed each time:

```bash
for varname in list;
  do command;
done
```

Ex.:

```bash
for file in *.txt;
  do mv ${file} ${file%txt}text;
  done
```

(useful, e.g., to process a number of files)
Compound Commands: Conditional Execution

If and only if ifcmd succeeds, then cmd is executed; otherwise, else cmd is executed:

```
if ifcmd; then then cmd;
else else cmd; fi
```

Ex.:
```
if [ -f '${file}' ]; then
    process '${file}'
else
    echo invalid: '${file}' >&2
fi
```

The else branch can be omitted.
Defining the Exit Code

The exit code can only be set when actually exiting from a script:

`exit code`

Ex.:

`exit 1`
Example Script

#!/bin/bash
ret=0
for file in "$@"; do
  if [ -f "$file" ]; then
    numLines=$(wc -l < "$file")
    echo File $file has $numLines lines.
    (($numLines % 4)) && { echo File $file seems suspicious.; ret=1; } || \\
    echo File $file seems to contain $((($numLines / 4) reads.
  else
    echo $file is not a simple file. >&2
    ret=1
  fi
done
exit $ret

usage, e.g.: ./readcount.bash ${tutorial}/data/*fastq
Wildcards

A mechanism to enumerate existing paths matching a given name pattern. Patterns consist of:
  literals
  * matches a sequence of characters of any length
  ? matches exactly one character
Ex.:
  file?.* matches (and expands to)
  file1.txt
  file1.bam
  file1.sam
  file1.sam
  file2.txt ...
Usage: ls -l file?.*
SGE: The Batch System

HPC Linux Cluster
Head Node

Compute Nodes

Researchers

NYU Langone Medical Center
Multiple Related Jobs: Array Jobs

job 76475.1
script.sh

Same script

job 76475.2
script.sh

job 76475.3
script.sh
Multiple Related Jobs: Array Jobs

Same script…
Different task IDs

job 76475.1
script.sh
SGE_TASK_ID=1

job 76475.2
script.sh
SGE_TASK_ID=2

job 76475.3
script.sh
SGE_TASK_ID=3

SGE defines variable SGE_TASK_ID with unique value for each task. Can be used to customize behavior.
Array Job Example

#!/bin/bash
#$ -S /bin/bash
date
hostname
DIR=$1  # First argument
echo "File Directory is: $DIR"
echo "Task ID is:  $SGE_TASK_ID"
wc -l $DIR/s${SGE_TASK_ID}_sequence.fastq
date

Submit with:
qsub -t 1-3 script-array.sh /ifs/data/tutorials/linux2/data
Array Job Output

qsub -t 1-n script

<table>
<thead>
<tr>
<th>Task ID</th>
<th>stdout</th>
<th>stderr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>script.ojobnum.1</td>
<td>script.ejobnum.1</td>
</tr>
<tr>
<td>2</td>
<td>script.ojobnum.2</td>
<td>script.ejobnum.2</td>
</tr>
<tr>
<td>3</td>
<td>script.ojobnum.3</td>
<td>script.ejobnum.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>script.ojobnum.n</td>
<td>script.ejobnum.n</td>
</tr>
</tbody>
</table>

General form:  <jobname>.'['e'|'o']<job_id>'.'<task_id>
Array job submission

• In general, can use \(-t \ n-\ m:s\)
• Task ID goes from \(n\) to \(m\) in steps of \(s\).
• Example \(-t\ 2-\ 16:3\)
  – five tasks with IDs of 2, 5, 8, 11, 14.
• \(n\) must be at least 1
Array Job Idioms (inside script)

• Same command, different files:
  
  prog < in.${SGE_TASK_ID} > out.${SGE_TASK_ID}

• Same command, different parameters:
  
  prog –n ${SGE_TASK_ID} > out.${SGE_TASK_ID}

• Arithmetic, for example, to start from 0:
  
  i=$(expr $SGE_TASK_ID - 1)
  wc -l $DIR/s${i}_sequence.fastq
More Array Job Idioms

- Bash arrays for arbitrary sequences:

  ```bash
  values = (30220 13681 15754 28146 26143)
  i=$(expr $SGE_TASK_ID - 1)
  n=${values[$i]}
  prog -n $n > out.n.$n
  ```
Multi-threaded Jobs

• Several applications can use multiple threads:
  – a form of shared-memory parallelism
  – uses multiple CPU cores within one node

• Use SGE threaded parallel environment.
Threaded Environment Examples

• `qsub -pe threaded 4 script.sh`
  – waits until 4 cores available on one node
  – starts on that node, claiming 4 slots

• `qsub -pe threaded 6-13 script.sh`
  – waits until at least 6 cores on one node
  – prefers up to 13
  – starts the node with most available (up to 13)
  – claims as many as available on that node (up to 13)
How Many Did I Get?

- SGE defines **NSLOTS** variable to tell you:

```bash
#!/bin/bash
#$ -S /bin/bash
#$ -cwd
module load cufflinks/2.1.1
cufflinks -p $NSLOTS ...
```
Threads: Best Practices

• Read application documentation carefully:
  – Are multiple threads supported?
  – Do they happen by default?
  – What are the tradeoffs?
  – What flag controls them?

• Don’t hard-code number of threads.
  – use $NSLOTS
Message Passing: MPI

- Some applications support *message passing* through MPI.
  - A form of distributed memory parallelism.
  - Uses multiple slots *across different nodes*.

- Use SGE **openmpi** *parallel environment*…
  - or **mpich** … or … we have a few flavors
MPI Examples

• `qsub -pe openmpi 4 script.sh`
  – waits until 4 cores available across cluster
  – starts jobs, claiming 4 slots total

• `qsub -pe openmpi 6-13 script.sh`
  – waits until at least 6 cores available across cluster
  – prefers up to 13
  – claims as many as available (up to 13)
How Many Did I Get?

• SGE defines **NSLOTS** variable to tell you:
  
  ```bash
  #!/bin/bash
  #$ -S /bin/bash
  #$ -cwd

  module load meme/4.9.1
  meme -p $NSLOTS ...
MPI: Best Practices

• Read application documentation carefully:
  – Is MPI supported (fairly rare)?
  – Does it happen by default?
  – What are the tradeoffs?
  – What flag controls it?

• Don’t hard-code number of processes.
  – use $NSLOTS
MPI Best Practices Continued

- MPI goes across multiple nodes, so be careful about local files, *e.g.*, in $TMP$
- Which flavor of MPI?
  - Check application documentation and module.
  - module help *app* (try it with meme)
  - module show *app* (try it with meme)
Which MPI flavor?

[root@phoenix1 ~]# module show meme
/local/modulefiles/meme/4.9.1:

module-whatis   The MEME Suite: Motif-based sequence analysis tools
module          load openmpi/gcc/64/1.4.5
prereq          openmpi/gcc/64/1.4.5
conflict        meme
append-path     PATH /local/apps/meme/4.9.1/bin
setenv          MEME_ROOT /local/apps/meme/4.9.1
SGE Directives

- SGE Directives (e.g., -cwd) can go in several places:
  - on the `qsub` command line: `qsub -cwd`
  - In the script: `#$ -cwd`
  - In a `.sge_request` file, with a line “-cwd”
    - Directory from which script is submitted.
    - Your home directory.
    - Global directory.
Hands-On Tutorial on Linux Cluster Computing:
Jobs Scripts and Data Management

- Quick overview of local HPC resources & Announcements
- Preliminaries (login, etc.)
- Shell Scripts (bash)
- Batch System (SGE)
- Data Transfers
Remote file copying using secure copy `scp`

Format: `scp` source destination

source, destination: filename or directory or `username@hostname:directory`

`username`: usually the user’s kerberos id
`hostname`: remote host, for our purposes: `phoenix.med.nyu.edu`

Note: if your private key does not have one of the default names (ex. `id_rsa`) you must use the `-i` flag with `scp` to specify the name of your private key.

Example: `~> scp -i key file1 efstae01@phoenix.med.nyu.edu:`
Remote file copying using secure copy \texttt{scp}

Examples:

\texttt{scp} \texttt{brain.bam} efstae01@phoenix.med.nyu.edu

(common mistake! Missed the colon ":" at the end)

\texttt{scp} \texttt{brain.bam} efstae01@phoenix.med.nyu.edu: \\

(file copied to default directory \~{}efstae01/, home directory)

\texttt{scp} \texttt{brain.bam} efstae01@phoenix.med.nyu.edu: \texttt{data/} \\

(file copied to \~{}efstae01/data/, target directory must exist)

\texttt{scp} \texttt{brain.bam} efstae01@phoenix.med.nyu.edu::\texttt{data/} \\

(file copied to \texttt{/data/}, target directory must exist)
Remote file copying using secure copy `scp`

Examples:

```
scp *fastq efstae01@phoenix.med.nyu.edu:data/
```

(using wild characters on the local machine)

```
scp efstae01@phoenix.med.nyu.edu:\*\.bam .
```

(Be careful using wild characters on remote machines)

```
scp -r efstae01@phoenix.med.nyu.edu:data/ ./data/
```

(recursive copying. All files and directories under `data/` will be copied. Not very efficient with links)

`man scp`
Interactive, Remote file copying using `sftp`

dhcp-10-193-37-109:~ stratos$ sftp efstae01@phoenix.med.nyu.edu
Connecting to phoenix.med.nyu.edu...

```
sftp> cd /ifs/data/tutorials/linux2/scripts

sftp> ls
script1.sh  script1.sh~  script2.sh  script2.sh~  script3.sh

sftp> get script1.sh
Fetching /ifs/data/tutorials/linux2/scripts/script1.sh to script1.sh
/ifs/data/tutorials/linux2/scripts/script1.sh                               100%  253     0.3KB/s   00:00

sftp> quit
```
Free, GUI-based file transfer tools

- fugu
- cyberduck
- filezilla
Options for Windows Users:

- **Winscp**: [http://winscp.net/eng/index.php](http://winscp.net/eng/index.php)

- **Putty scp (pscp) and Putty sftp (psftp)**: [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)

Example:

C:\> pscp -i C:\Keys\my-key-pair.ppk C:\Sample_file.txt
efstae01@phoenix.med.nyu.edu:
Remote file copying using *rsync*

rsync is more efficient than scp/sftp because:

1. It can transfer just the differences between two sets of files (local and remote).
2. It can compress (on the source) and decompress (at the destination).

Local Machine: ~> `mkdir scripts/`
Local Machine: ~> `rsync -azv --progress efstae01@phoenix.med.nyu.edu:/ifs/data/tutorials/linux2/scripts/ ./scripts/`

- `z`: compress file data
- `a`: archive mode: allows copying files recursively, preserves symbolic links, file permissions, timestamps, user and group ownership.
- `v`: verbose mode

`--progress`: Show progress while transferring the data
`--delete`: Sync Source with Target. Any additional files that exist at the destination will be deleted.

~> `touch scripts/test.txt`
~> `rsync -azv --delete --progress efstae01@phoenix.med.nyu.edu:/ifs/data/tutorials/linux2/scripts/ ./scripts/`

- `--exclude`: Specify files to exclude
- `--include`: Specify files to include
- `--dry-run`: Do not make any changes
The **`tar`** utility

- The name “*tar*” stands for “tape archiver”.
- **Creates** `tar` archives.
- Also **updates** archives, or **lists** the files in an archive
- On a previously created `tar` archive, **extracts** files
- It is very common to archive AND compress at the same time combining `tar` with a compression utility, such as `gzip` or `bzip2`.

**Examples:**
- `tar --cvf file.tar file1 file2 file3`
- `tar --xvf file.tar`
- `tar --cvjf file.tar.bz2 file1 file2 file3`
- `tar --xjvf file.tar.bz2`

**Benefits when copying files:**
- less data to transfer (compression). Text files can compress ~ 90%

**Be Aware of different versions of `tar` on source and target machines.**
The *screen* utility

*screen* separates a program from a terminal shell. It can later detach or attach to the terminal session.
md5 checksum:

• An alphanumeric string associated with a single file.
• A corrupted file is very unlikely to have the same md5 checksum.

To use:

~phoenix> md5sum /ifs/data/tutorials/linux2/data/s* > checklist.md5
Distributing data sets to remote collaborators

- Data sets can be transferred using a URL: https://genome.nyumc.org/LabName/
- The same URL is available inside and outside NYULMC.
- Enables the sharing of TeraBytes of data.
- HPCF manages access and authorization.
- One-hop data transfers.

Command Line access to genome.nyumc.org:

```bash
~> curl -u user https://genome.nyumc.org/results/testlab/2013-12-11/
lane1_NoIndex_L001_R2_001.fastq.gz -o myfile

~> wget --user=user --ask-password https://genome.nyumc.org/results/testlab/2013-12-11/
lane1_NoIndex_L001_R2_001.fastq.gz
```
Issues with transferring large data using Basic data transfer tools (scp, sftp, rsync, etc.)

(1) Firewalls
(2) Performance limited by underlying protocol (TCP/IP)
(3) Lack of Monitoring, Error Recovery, and Notifications
(4) Do not support multiple streams
BaBar Copy (bbcp)  http://www.slac.stanford.edu/~abh/bbcp/

Aspera  www.asperasoft.com

Globus Online:  www.globus.org