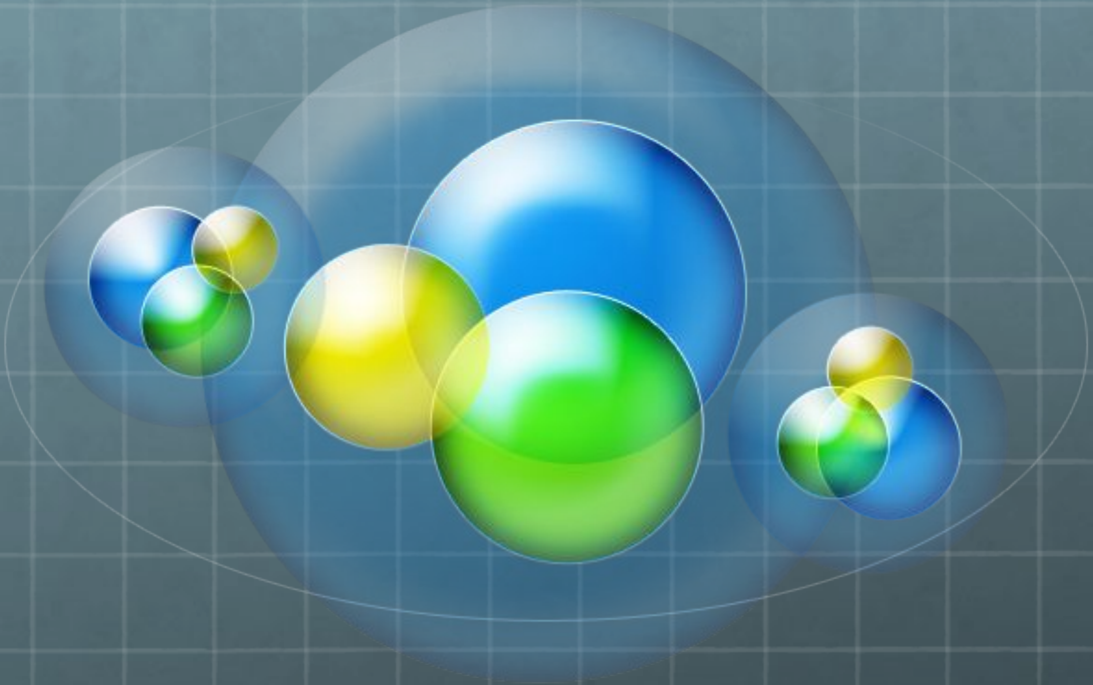
Class begins on Michigan Time. Before class begins, please do the following:

1. Visit **www.umich.edu/~cja/** and click "STATS 531 Introduction to Flux" to view these slides.

2. Login to Flux using the instructions on the first few slides.

STATS 531 Introduction to Flux

Dr Charles J Antonelli, LSA IT ARS
Mark Montague, LSA IT ARS
February, 2018



STATS 531
Introduction to Flux

Dr Charles J Antonelli, LSA IT ARS
Mark Montague, LSA IT ARS
March, 2018

Preliminaries

Connecting to Flux

(Windows)

1. If you don't have PuTTY installed
   Install the PuTTY/WinSCP installer for Windows:
   Compute at the U: http://its.umich.edu/computing/computers-software/compute
   Select + under Get Going
   Get U-M PuTTY (UM_PuTTY_WinSCP.zip)
   Execute the installer (UM_PuTTY_WinSCP.exe)
   Accept all defaults

2. If you don't see this icon in your system tray
   Install the Xming X Server for Windows
   https://sourceforge.net/projects/xming/files/Xming/6.9.0.31/
   Get the Xming installer (Xming-6-9-0-31-setup.exe)
   Execute the installer
   Accept all defaults
   https://sourceforge.net/projects/xming/files/Xming-fonts/7.7.0.10/
   Get the XMing fonts installer (Xming-fonts-7-7-0-10-setup.exe)
   Execute the installer
   Accept all defaults

Connecting to Flux

(Windows)

1. Double-click "UM Internet Access Kit" icon on Desktop
   Double-click PuTTY application within

2. In the Putty Application that appears:
   Enter "flux-login.arc-ts.umich.edu" in the Host Name box
   Under Connection | SSH | X11, ensure Enable X11 Forwarding is checked
   Click "Open" at bottom

3. In the terminal window that appears:
   Login with uniqname, Kerberos password, and Duo

4. This creates an ssh session on Flux

Connecting to Flux

(Linux & macOS)

1. Start a terminal window
2. In the terminal window that appears, type
   ```
   ssh -X uniqname@flux-login.arc-ts.umich.edu
   ```
3. Login to Flux with Kerberos password and Duo

Roadmap

- Preliminaries
- Introduction to Flux
- The command line
- Flux mechanics
- Parallel R and pomp

Introduction to Flux

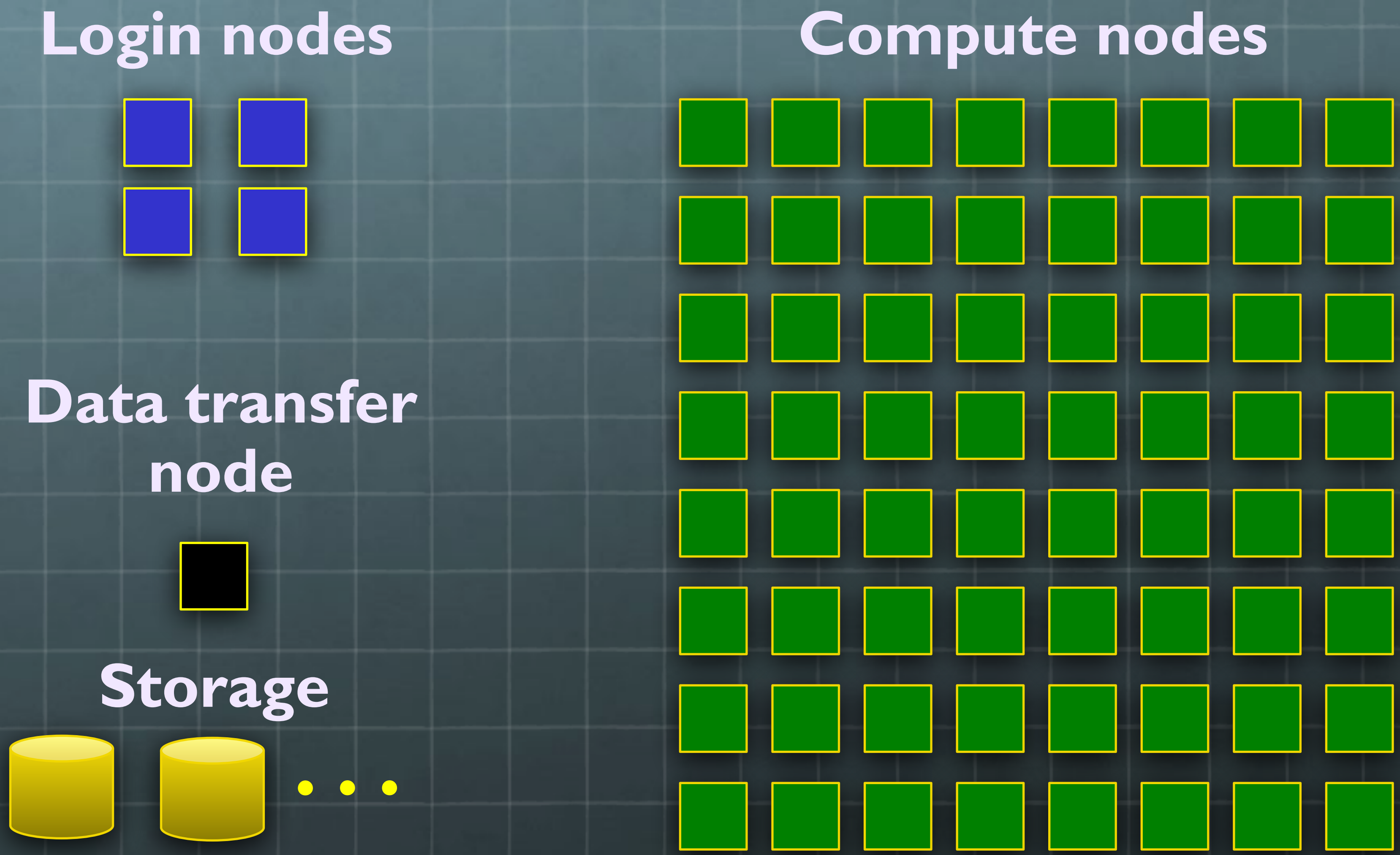
Flux

Flux is a university-wide shared computational discovery / high-performance computing service.

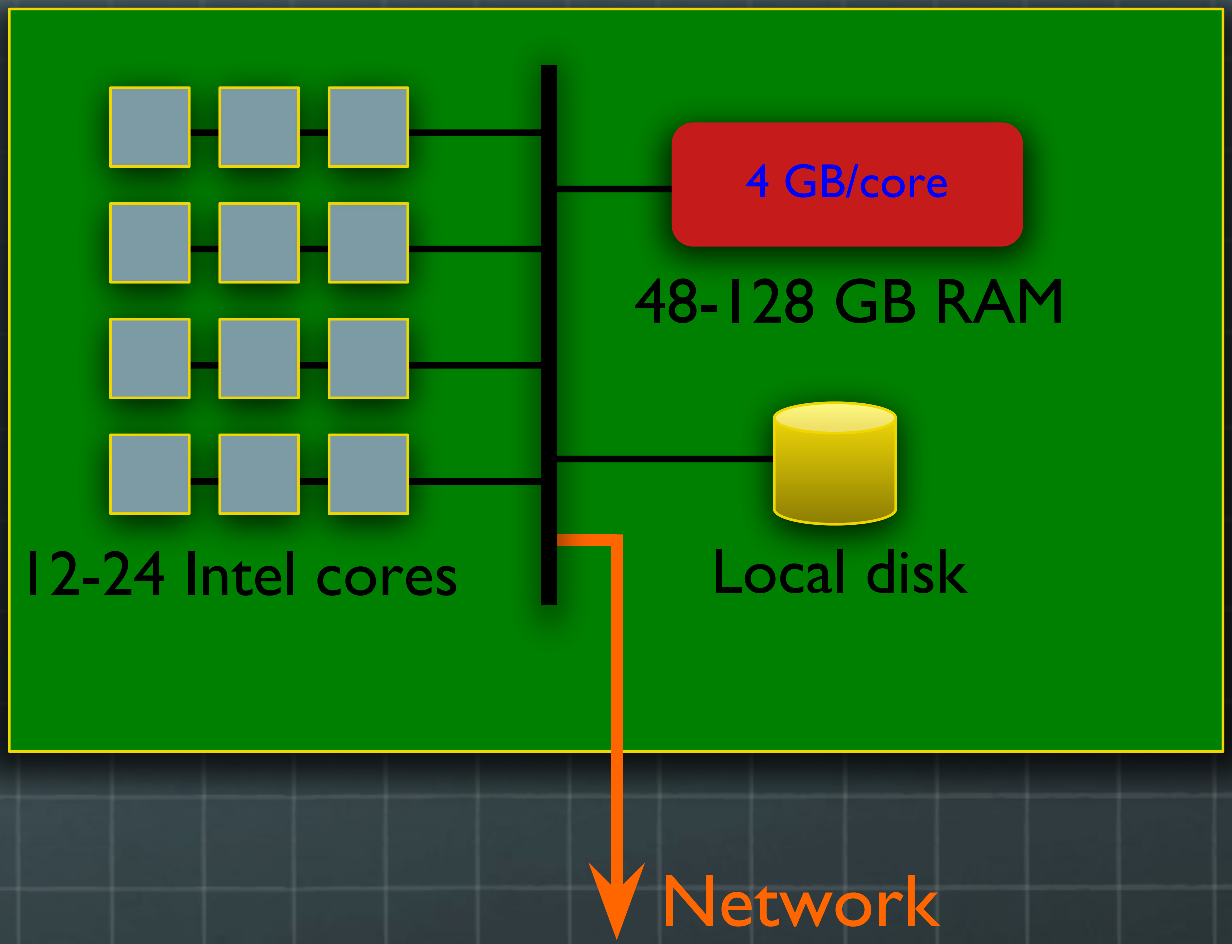
- Provided by Advanced Research Computing at U-M
- Procurement, licensing, billing by U-M ITS
- Interdisciplinary since 2010

http://arc-ts.umich.edu/resources/compute-resources/

The Flux cluster



A Standard Flux node



Using Flux

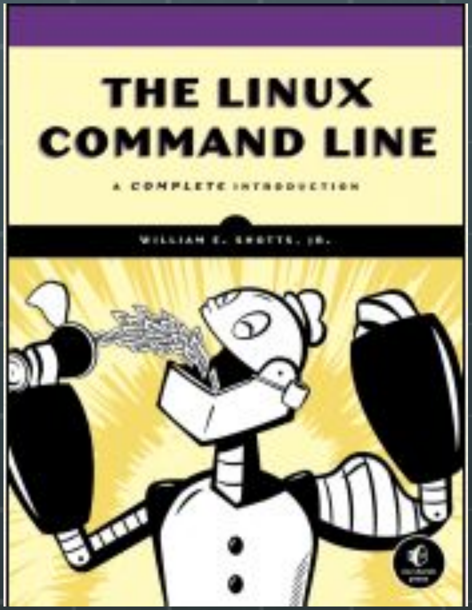
- Three basic requirements:
  A Flux login account
    https://arc-ts.umich.edu/fluxform
  A Flux allocation
    `stats531w18_flux`
  A Duo app on your smartphone or tablet
    http://its.umich.edu/two-factor-authentication
- Logging in to Flux
    `ssh -X` *`login`*`@flux-login.arc-ts.umich.edu`
    PuTTY
  Campus wired or MWireless
    Otherwise use VPN, or
    `ssh login.itd.umich.edu` first

The command line

Command Line Reference

William E Shotts, Jr.,
"The Linux Command Line: A Complete Introduction,"
No Starch Press, January 2012.
http://linuxcommand.org/tlcl.php .

Download Creative Commons Licensed version at
http://downloads.sourceforge.net/project/linuxcommand/TLCL/13.07/TLCL-13.07.pdf



At the command prompt

- Basic input line editing commands
  - Backspace erases previous character
  - Left and right arrow move insertion point on the line
  - Control-U erases the line to the insertion point, so you can start over
  - Enter executes the line you typed
  - Control-C interrupts whatever command you started and returns you to the shell prompt (usually)
  - Up and down arrow will access your *command history*
  - Type “`exit`” without the quotes to exit the shell

Flux mechanics

Cluster batch workflow

- You create a batch script and submit it to PBS
- PBS schedules your job, and it enters the flux queue
- When its turn arrives, your job will execute the batch script
- Your script has access to all Flux applications and data
- When your script completes, anything it sent to standard output and error are saved in files stored in your submission directory
- You can ask that email be sent to you when your jobs starts, ends, or aborts
- You can check on the status of your job at any time, or delete it if it's not doing what you want
- A short time after your job completes, it disappears from PBS

Multi-threaded batch script

```
#PBS -N yourjobname
#PBS -V
#PBS -A youralloc_flux
#PBS -q flux
#PBS -l nodes=1:ppn=12,mem=47gb,walltime=00:05:00
#PBS -m abe
#PBS -j oe

#Your Code Goes Below:
cd $PBS_O_WORKDIR
R CMD BATCH --vanilla myscript.R myscript.out
```

Copying data

Using command line programs:

scp: copies files between hosts on a network over ssh

```
scp localfile uniqname@flux-xfer.arc-ts.umich.edu:remotefile
scp -r localdir uniqname@flux-xfer.arc-ts.umich.edu:remotedir
scp uniqname@flux-login.arc-ts.umich.edu:remotefile localfile
```

Use "." as destination to copy to your Flux home directory:

```
scp localfile uniqname@flux-xfer.arc-ts.umich.edu:.
```

... or to your Flux scratch directory:

```
scp localfile
uniqname@flux-xfer.arc-ts.umich.edu:/scratch/allocname/uniqname
```

sftp: an interactive file transfer program over ssh (a secure ftp)

```
sftp uniqname@flux-xfer.arc-ts.umich.edu
```

Using graphical (GUI) applications:

FileZilla (cross-platform):	http://filezilla-project.org/
Cyberduck (Mac):	https://cyberduck.io/
WinSCP (Windows):	http://www.itcs.umich.edu/bluedisc/

Basic batch commands

- Once you have a script, submit it:
  `qsub` *`scriptfile`*

```
$ qsub singlenode.pbs
6023521.nyx.engin.umich.edu
```

- You can check on the job status
  `qstat` *`jobid`*
  `qstat -u` *`user`*

```
$ qstat -u cja
nyx.engin.umich.edu:
                                                                         Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- -
-----
6023521.nyx.engi     cja      flux     hpc101i              --     1   1     -- 00:05 Q   --
```

- To delete your job
  `qdel` *`jobid`*

Modules

- The module command allows you to specify what Flux software you want to use

```
module list                -- Show loaded modules
module load name           -- Load module name for use
module avail               -- Show all available modules
module avail  name         -- Show versions of module name
module key string          -- Search for string in module descrip
module spider string       -- Search for string in all module doc
module unload name         -- Unload module name
module use path            -- Add path to module search path
module                     -- List all options
```

- Enter these commands at any time during your session
- You can load multiple modules in the same login session
- Software modules remain available throughout the session

Modules

- Once loaded, you can group a set of modules into a *module set*
  `module save myset`
- A module set can be restored at any time
  `module restore myset`
- List all module sets you've defined
  `module savelist`
- Create a module set to be loaded each time you log in
  `module save`

Lab

Task: Use the R multicore package

- Copy sample code to your login directory

```
cd
cp
/scratch/data/workshops/stats/stats-sampl
e-code.tar.gz .
tar -zxvf stats-sample-code.tar.gz
cd ./stats-sample-code
```

- Examine `lab3.pbs` and `lab3.R`

Lab

Task: Use the R multicore package

- `module load R`
- Submit your job to Flux
  `qsub lab3.pbs`
- Watch the progress of your job
  `qstat -u` *`uniqname`*
  where *`uniqname`* is your own uniqname
- When complete, look at the job's output
  `less lab3.out`

Parallel R and pomp

Random numbers

Serial pseudorandom sequence
- Usually based on linear recurrences modulo m
- Initialize the generator with some seed s
- Generate stream of pseudorandom numbers

Parallel pseudorandom sequence
- Performance, Reproducibility, Serializability

Install pomp

Task: Install pomp package

- `module load R/3.4.1`
- `R`
  ```
  install.packages("pomp",repos="https://cran.mtu.edu/")
  ```
  (answer y to "Would you like to use a personal library instead?" and "Would you like to create a personal library")

  List of mirrors: https://cran.r-project.org/mirrors.html

Multicore example

```
rm(list=ls())
library(doParallel)
set.seed(2018,kind="L'Ecuyer")
cores <- as.numeric(Sys.getenv('PBS_NP', unset='8'))
cl <- makeCluster(cores)
registerDoParallel(cl)
trials <- 100
```

Multicore example

```
system.time(
 r <- foreach(icount(trials),
 .inorder=FALSE,
 .options.multicore=list(set.seed=TRUE)
 ) %dopar% {
  library(pomp)
  for (i in 1:100) {
   pompExample("gompertz")
   simulate(gompertz)
  }
 }
)
stopCluster(cl)
```

MPI example

```
system.time(
 r <- foreach(icount(trials),
 .inorder=FALSE,
 .options.multicore=list(set.seed=TRUE)
 ) %dopar% {
  library(pomp)
  for (i in 1:100) {
   pompExample("gompertz")
   simulate(gompertz)
  }
 }
)
stopCluster(cl)
```

Interactive jobs

- You can submit jobs *interactively*:

```
qsub -I -X -V -l nodes=1:ppn=2 -l walltime=15:00
  -A youralloc_flux -l qos=flux -q flux
```

- This queues a job as usual
  - Your terminal session will be blocked until the job runs
  - When your job runs, you'll get an interactive shell
  - When you exit the shell your job is deleted
- Interactive jobs allow you to
  - Develop and test on cluster node(s)
  - Execute GUI tools on a cluster node
  - Utilize a parallel debugger interactively

Lab

Task: Use an interactive PBS session

- `module load R`
- Start an interactive session
```
qsub -I -V -l nodes=1:ppn=2 -l walltime=30:00 -A
stats531w18_flux -q flux
```
- Run R in the interactive shell
```
cd $PBS_O_WORKDIR
R
```

Troubleshooting

- System-level
  - freenodes — # aggregate node/core busy/free
  - pbsnodes [-l] — # nodes, states, properties
    # with -l, list only nodes marked down
- Account-level
  - mdiag -a *acct* — # cores & users for account *acct*
  - showq [-r][-i][-b][-w acct=*acct*] — # running/idle/blocked jobs for *acct*
    # *with -r|i|b* show more info for that job state
  - freealloc [--jobs] *acct* — # free resources in *acct*
    # *with –jobs, shows resources in use*
  - idlenodes *acct [property]* — # shows available nodes for *acct* with *property*
- User-level
  - mdiag -u *uniq* — # allocations for user *uniq*
  - showq [-r][-i][-b][-w user=*uniq*] — # running/idle/blocked jobs for *uniq*
- Job-level
  - qstat -f *jobno* — # full info for job *jobno*
  - qstat -n *jobno* — # show nodes/cores where *jobno* running
  - checkjob [-v] *jobno* — # show why *jobno* not running
  - qpeek *jobno* — # peek at script output while *jobno* is running

Resources

- ARC User Guide http://arc-ts.umich.edu/flux-user-guide/
- ARC Flux pages http://arc-ts.umich.edu/flux/
- Software Catalog http://arc-ts.umich.edu/software/
- Quick Start Guide http://arc-ts.umich.edu/flux/using-flux/flux-in-10-easy-steps/
- Flux FAQs http://arc-ts.umich.edu/flux/flux-faqs/
- For assistance, send email to: hpc-support@umich.edu
  - Read by a team of people including unit support staff
  - Can help with Flux operational and usage questions
  - Programming support available