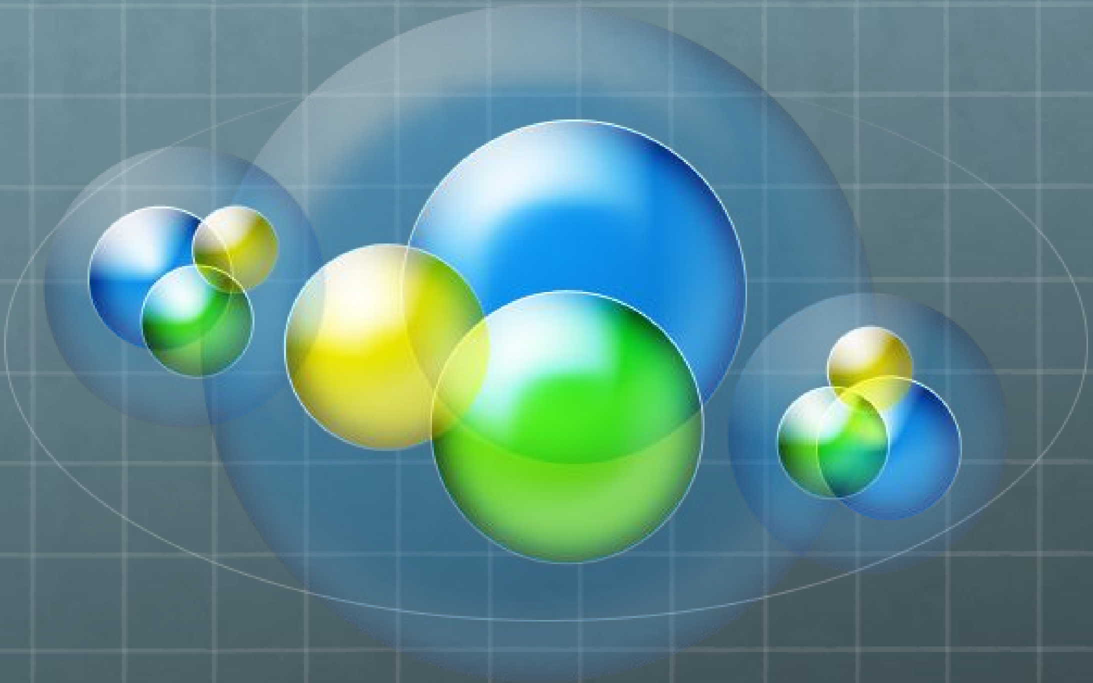


Advanced Batch Computing On the Flux Custer

Dr Charles J Antonelli, LSA IT ARS
Mark Champe, LSA IT ARS
May, 2017

Roadmap

- Flux review
- Advanced PBS
  - Array & dependent scheduling
  - Tools
- Python on Flux
- MPI on Flux
  - Programming
  - Debugging & Profiling

Schedule

1:10 - 2:00	Advanced PBS (Charles)
2:00 - 2:05	Break
2:05 - 3:00	Python (Mark)
3:00 - 3:05	Break
3:05 - 4:00	MPI Programming (Charles)
4:00 - 4:30	Open Q & A

Advanced PBS

Job Arrays

- Submit copies of identical jobs
- In a PBS script use:

  ```
  #PBS -t array-spec
  ```

  As a command line parameter:

  ```
  qsub -t array-spec job.pbs
  ```

  Where array-spec can be

  ```
  m-n
  a,b,c
  m-n%slotlimit
  ```

  e.g.

  ```
  qsub -t 1-50%10 job.pbs
  ```
  Fifty jobs, numbered 1 through 50, only ten can run simultaneously
- $PBS_ARRAYID records array identifier

Lab: Run an array job

1. Copy the files from the examples directory

```
cp -a /scratch/data/workshops/hpc201 ~
cd ~/hpc201/hpc-201-cpu/arrayjob
```

2. Inspect `arr.m` and `[123]/seed.txt`

3. Edit `submit.pbs`

```
$ nano submit.pbs
```

4. Submit the batch job

```
$ qsub submit.pbs
```

5. Inspect the results

Dependent scheduling

- Submit job to become eligible for execution at a given time
- Invoked via qsub -a:

  qsub -a [[[[CC]YY]MM]DD]hhmm[.SS] ...

```
qsub -a 201712312359 j1.pbs
```
j1.pbs becomes eligible one minute before New Year's Day 2018

```
qsub -a 1800 j2.pbs
```
j2.pbs becomes eligible at six PM today (or tomorrow, if submitted after six PM)

Dependent scheduling

- Submit job to run after specified job(s)
- Invoked via qsub -W:

    qsub -W *depend*=type:jobid[:jobid]…

  Where depend can be

after	Schedule this job after *jobids* have started
afterany	Schedule this job after *jobids* have finished
afterok	Schedule this job after *jobids* have finished with no errors
afternotok	Schedule this job after *jobids* have finished with errors

```
JOBID=`qsub first.pbs`   # JOBID receives first.pbs’s jobid
qsub -W depend=afterany:$JOBID second.pbs
```

Schedule second.pbs after first.pbs completes

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

Python on Flux

Scientific computing tools and practices

Python Distributions and core libraries

- The two major python distributions are:
  - **Anaconda Python**
    Open Source modern analytics platform powered by Python. Anaconda Python is recommended because of optimized performance (special versions of numpy and scipy) , and it has the largest number of preinstalled scientific Python packages.
    ```
    module load python-anaconda2/latest
    ```
  - **EPD**
    The Enthought Python Distribution provides scientists with a comprehensive set of tools to perform rigorous data analysis and visualization.
    ```
    module load python-epd/7.6-1
    ```

Debugging & Profiling

Debugging with GDB

- Command-line debugger
  - Start programs or attach to running programs
  - Display source program lines
  - Display and change variables or memory
  - Plant breakpoints, watchpoints
  - Examine stack frames
- Excellent tutorial documentation
  - http://www.gnu.org/s/gdb/documentation/

Compiling for GDB

- Debugging is easier if you ask the compiler to generate extra source-level debugging information
  - Add -g flag to your compilation
    ```
    icc -g serialprogram.c -o serialprogram
    ```
    or
    ```
    mpicc -g mpiprogram.c -o mpiprogram
    ```
  - GDB will work without symbols
    - Need to be fluent in machine instructions and hexadecimal
- Be careful using -O with -g
  - Some compilers won't optimize code when debugging
  - Most will, but you sometimes won't recognize the resulting source code at optimization level -O2 and higher
  - Use -O0 -g to suppress optimization

Running GDB

Two ways to invoke GDB:

- Debugging a serial program:
  `gdb ./serialprogram`

- Debugging an MPI program:
  `mpirun -np N xterm -e gdb ./mpiprogram`
  - This gives you *N* separate GDB sessions, each debugging one rank of the program
  - Remember to use the -X or -Y option to ssh when connecting to Flux, or you can't start xterms there

Useful GDB commands

gdb exec	start gdb on executable exec
gdb exec core	start gdb on executable exec with core file core
l [m,n]	list source
disas	disassemble function enclosing current instruction
disas func	disassemble function func
b func	set breakpoint at entry to func
b line#	set breakpoint at source line#
b *0xaddr	set breakpoint at address addr
i b	show breakpoints
d bp#	delete beakpoint bp#
r [args]	run program with optional args
bt	show stack backtrace
c	continue execution from breakpoint
step	single-step one source line
next	single-step, don't step into function
stepi	single-step one instruction
p var	display contents of variable var
p *var	display value pointed to by var
p &var	display address of var
p arr[idx]	display element idx of array arr
x 0xaddr	display hex word at addr
x *0xaddr	display hex word pointed to by addr
x/20x 0xaddr	display 20 words in hex starting at addr
i r	display registers
i r ebp	display register ebp
set var = expression	set variable var to expression
q	quit gdb

Debugging with DDT

- Allinea's Distributed Debugging Tool is a comprehensive graphical debugger designed for the complex task of debugging parallel code
- Advantages include
  - Provides GUI interface to debugging
    - Similar capabilities to Eclipse or Visual Studio
  - Supports parallel debugging of MPI programs
    - Scales much better than GDB

Running DDT

- Compile with -g:
  `mpicc -g mpiprogram.c -o mpiprogram`

- Load the DDT module:
  `module load ddt`

- Start DDT:
  `ddt mpiprogram`
  - This starts a DDT session, debugging all ranks concurrently
  - Remember to use the -X or -Y option to ssh when connecting to Flux, or you can't start ddt there

- http://arc-ts.umich.edu/software/

- http://content.allinea.com/downloads/userguide-forge.pdf

Application Profiling with Allinea Performance Reports

- Allinea Performance Reports provides a summary profile
- Advantages include
    - Less complex to use & interpret
    - Provides (heuristic) advice to improve performance in five areas
    - Handles all of the details under the covers

Running Allinea Performance Reports

- Compile with -g:
  `mpicc -g mpiprogram.c -o mpiprogram`
- Load the Performance Reports module:
  `module load allinea_reports`
- Start Performance Reports:
  `perf-report mpirun -np N mpiprogram`
  - This runs your program, gathers profile data , and produces a performance report
    - .txt and .html versions
- http://content.allinea.com/downloads/userguide-reports.pdf

Application Profiling with MAP

- Allinea's MAP Tool is a statistical application profiler designed for the complex task of profiling parallel code
- Advantages include
  - Provides GUI interface to profiling
    - Observe cumulative results, drill down for details
  - Supports parallel profiling of MPI programs
  - Handles most of the details under the covers

Running MAP

- Compile with -g:
  `mpicc -g mpiprogram.c -o mpiprogram`
- Load the MAP module:
  `module load ddt`
- Start MAP:
  `map mpiprogram`
  - This starts a MAP session
    - Runs your program, gathers profile data, displays summary statistics
  - Remember to use the -X or -Y option to ssh when connecting to Flux, or you can't start ddt there
- http://content.allinea.com/downloads/userguide-forge.pdf

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.research.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

References

1. Supported Flux software, http://arc-ts.umich.edu/software/, (accessed May 2015)
2. Free Software Foundation, Inc., "GDB User Manual," http://www.gnu.org/s/gdb/documentation/ (accessed May 2015).
3. Intel C and C++ Compiler 14 User and Reference Guide, https://software.intel.com/en-us/compiler_15.0_ug_c (accessed May 2015).
4. Intel Fortran Compiler 14 User and Reference Guide,https://software.intel.com/en-us/compiler_15.0_ug_f(accessed May 2015).
5. Torque Administrator's Guide, http://www.adaptivecomputing.com/resources/docs/torque/5-1-0/torqueAdminGuide-5.1.0.pdf (accessed May 2015).
6. Submitting GPGPU Jobs, https://sites.google.com/a/umich.edu/engin-cac/resources/systems/flux/gpgpus (accessed May 2015).
7. http://content.allinea.com/downloads/userguide.pdf (accessed May 2015)

Addendum

Reference Materials

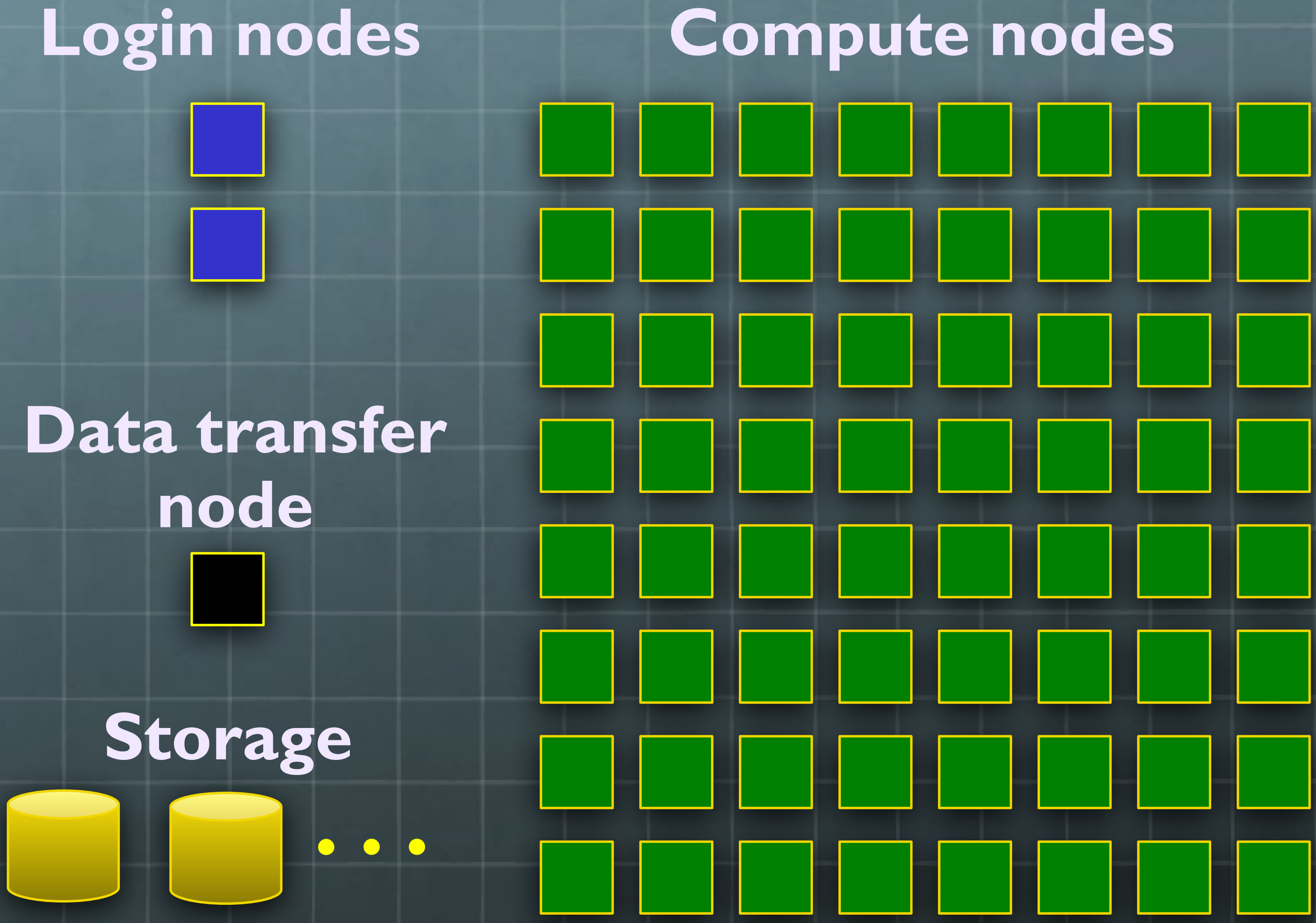
Flux

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

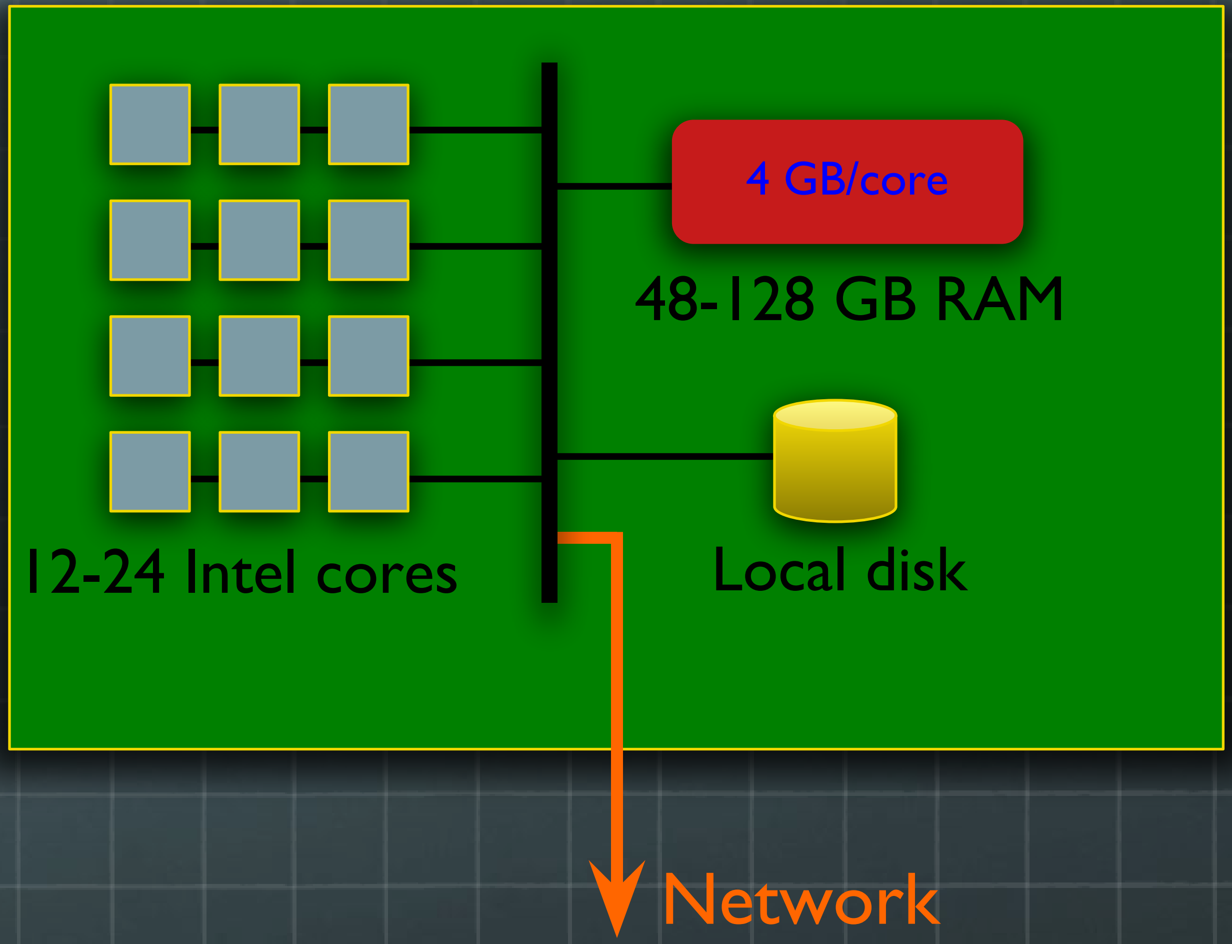
- Provided by Advanced Research Computing at U-M
- Procurement, licensing, billing by U-M ITS
- Interdisciplinary since 2010

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

The Flux cluster



A Standard Flux node



Other Flux services

- Flux Larger Memory (FluxM)
  - 360 cores across some 14 nodes
  - Each core comes with 25 GB of memory, total node memory up to 1.5 TB
- Flux GPUs (FluxG)
  - 5 nodes: Standard Flux, plus 8 NVIDIA K20X GPUs with 2,688 CUDA cores each
  - 6 nodes: Standard Flux, plus 4 NVIDIA K40X GPUs with 2,880 CUDA cores each
- Flux on Demand (FoD)
  - Pay only for resources wallclock *consumed*, at a higher cost rate
  - You do pay for cores and memory *requested* while job is running
- Flux Operating Environment (FoE)
  - For grants that require the purchase of computing hardware, nodes are added to the Flux environment (same management,
- http://arc-ts.umich.edu/flux-configuration

- http://arc-ts.umich.edu/rates/

Programming Models

- Two basic parallel programming models
  - Multi-threaded
    The application consists of a single process containing several parallel threads that communicate with each other using synchronization primitives
    - Used when the data can fit into a single process, and the communications overhead of the message-passing model is intolerable
    - "Fine-grained parallelism" or "shared-memory parallelism"
    - Implemented using OpenMP (Open Multi-Processing) compilers and libraries
  - Message-passing
    The application consists of several processes running on different nodes and communicating with each other over the network
    - Used when the data are too large to fit on a single node, and simple synchronization is adequate
    - "Coarse parallelism" or "SPMD"
    - Implemented using MPI (Message Passing Interface) libraries
  - Both

Using Flux

- Three basic requirements:
  A Flux login account
  https://arc-ts.umich.edu/fluxform
  A Flux allocation
  training_flux
  Duo two-factor authentication
  http://its.umich.edu/two-factor-authentication

- Logging in to Flux

```
$ ssh -X uniqname@flux-login.arc-ts.umich.edu
```

  Directly accessible over wired networks, MWireless, or from UM VPN
  Otherwise, you can SSH to campus login nodes first then to Flux

```
$ ssh uniqname@login.itd.umich.edu
dukenukem% ssh flux-login.arc-ts.umich.edu
```

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 executes the script
- Your script has access to all Flux applications and data
- Your script's standard output and error are saved in files stored in your submission directory
- Email can be sent when your job starts, ends, or fails
- You can check job status or delete your job at any time
- A short time after your job completes, it disappears

Tightly-coupled batch script

```
#PBS -N job_name
#PBS -M your_email
#PBS -m bea
#PBS -j oe
#PBS -A youralloc_flux
#PBS -l qos=flux
#PBS -q flux
#PBS -l nodes=1:ppn=12,mem=47gb,walltime=00:05:00
#PBS -V

# Your Code Goes Below:
cat $PBS_NODEFILE
cd $PBS_O_WORKDIR
matlab -nodisplay -r script
```

Loosely-coupled batch script

```
#PBS -N job_name
#PBS -M your_email
#PBS -m bea
#PBS -j oe
#PBS -A youralloc_flux
#PBS -l qos=flux
#PBS -q flux
#PBS -l procs=12,pmem=1gb,walltime=00:05:00
#PBS -V

# Your Code Goes Below:
cat $PBS_NODEFILE
cd $PBS_O_WORKDIR
mpirun ./c_ex01
```

Flux scratch

- 1.5 PB of high speed *temporary* storage
  - **Not backed up!**
- `/scratch/alloc_name/user_name`
- Files stored in `/scratch` will be deleted when they have not been accessed in 90 days
- Moving data onto and off of `/scratch`
  - < ~100 GB: scp or SFTP
  - > ~100 GB: Globus Online

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 login@flux-xfer.arc-ts.umich.edu:.
```

... or to your Flux scratch directory:

```
scp localfile
login@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/

Globus Online

- Features
  - High-speed data transfer, *much faster* than copying with scp or SFTP
  - Reliable & persistent
  - Minimal, polished client software: Mac, Linux, Windows
- Globus Endpoints
  - GridFTP Gateways through which data flow
    - XSEDE, OSG, National labs, …
    - Umich Flux: `umich#flux`
  - Add your own server endpoint: contact hpc-support@umich.edu
  - Add your own client endpoint!
- Share folders via Globus Plus

http://arc-ts.umich.edu/systems-and-services/globus/

ARC Connect

- Provides *performant* GUI access to Flux
  - Easily use graphical software
  - Do high performance, interactive visualizations
  - Share and collaborate with colleagues on HPC-driven research
- Currently supports VNC desktop, Jupyter Notebook, RStudio
- Browse to
  https://connect.arc-ts.umich.edu/
- Documentation
  http://arc-ts.umich.edu/arc-connect/
- Comments on the service and the documentation are welcome!

Advanced PBS options

Some workflows, such as those creating many small files, function more efficiently by using node-local disks

```
#PBS -l ddisk=200gb
```

Selects nodes with /tmp at least 200 GB in size

However

- It does not check if there is that much *free* space
- It does not account for disk space being consumed by other jobs running on the node along with yours

Dependent scheduling

- Submit job to run before specified job(s)
- Requires dependent jobs to be scheduled first
- Invoked via qsub -W:

  qsub -W *depend*=type:jobid[:jobid]…

  Where depend can be

before	*jobids* scheduled after this job starts
beforeany	*jobids* scheduled after this job completes
beforeok	*jobids* scheduled after this job completes with no errors
beforenotok	*jobids* scheduled after this job completes with errors
on:*N*	wait for *N* job completions

```
JOBID=`qsub -W depend=on:1 second.pbs`
qsub -W depend=beforeany:$JOBID first.pbs
```

Schedule second.pbs after first.pbs completes