TEXAS TECH UNIVERSITY

CS4379: Parallel and Concurrent Programming
CS5379: Parallel Processing

Lecture 9

Guest Lecture by Misha Ahmadian

Dr. Yong Chen

Associate Professor

Computer Science Department

Texas Tech University

Course Info

- **Lecture Time**: TR, 12:30-1:50
- **Lecture Location**: ECE 217
- **Sessions**: CS4379-001, CS4379-002, CS5379-001, CS5379-D01
- **Instructor:** Yong Chen, Ph.D., Associate Professor
- **Email:** yong.chen@ttu.edu
- **Phone:** 806-834-0284
- **Office**: Engineering Center 315
- **Office Hours**: 2-4 p.m. on Wed., or by appointment
- **TA:** Mr. Ghazanfar Ali, Ghazanfar.Ali@ttu.edu
- **TA Office hours:** Tue. and Fri., 2-3 p.m., or by appointment
- **TA Office:** EC 201 A
- **More info:**
  - http://www.myweb.ttu.edu/yonchen
  - http://discl.cs.ttu.edu; http://cac.ttu.edu/; http://nsfcac.org

Outline

- Using Compilers on HPCC clusters
- Job Submission on HPCC clusters
- Transferring Data
- HPCC Policies

Outline

- Using Compilers on HPCC clusters
- Job Submission on HPCC clusters
- Transferring Data
- HPCC Policies

Sample Source Code

1. Log in to Quanah cluster
2. Copy the following directory into your $HOME directory

   - `$ cp -r /lustre/work/examples/hello-world ./`

3. You can see three C,C++, and Fortan version of Hello-World program besides “pthreads” and “openmp” directories:
   - `$ cd hello-world`
   - `$ ls`

```
hello-world.c
Hello-world.cpp
Hello-world.f
pthreads/
openmp/
```

Compiling and Running Programs

- **Load GNU Compiler Module**
  - `$ module load gnu7/7.3.0`
- **Compile**
  - `$ gcc –o hello-world-c hello-world.c`
  - `$ g++ –o hello-world-c++ hello-world.cpp`
  - `$ gfortran –o hello-world-f hello-world.f`
- **Run**
  - `$ ./hello-world-c`

Do the same with **Intel** Compilers

- **Redirect output (Optional)**
  - `$ ./hello-world-c > output`

Compiling and Running Multi-thread Programs

- **Load GNU Compiler module**
  - `$ module load gnu7/7.3.0`
- **Compile Pthread Program:**
  - `$ cd pthread`
  - `$ gcc –o hello-pthread –lpthread hello-pthread.c`
  - `$ ./hello-pthread`
- **Compile OpenMP Program:**
  - `$ cd ../openmp`
  - `$ gcc –o hello-openmp –fopenmp hello-openmp.c`
  - `$ ./hello-openmp`

Using MPI Compilers on HPCC

intel/18.0.3.222	• **OpenMPI/1.10.7** • impi/2018.3.222 • mvapich2/2.2	• mpicc • mpic++ • mpifort • mpirun
gnu/5.4.0	• OpenMPI/1.10.6 • impi/2018.3.222 • mvapich2/2.2	
gnu7/7.3.0	• **OpenMPI/1.10.7** • impi/2018.3.222 • mvapich2/2.2	

- **Load MPI Modules**
  - `$ module load intel openmpi`
- **Compile MPI program:**
  - `$ mpicc -o mpi-hello-world mpi-hello-world.c`
- **Run MPI program:**
  - `$ mpirun --machinefile machinefile.txt \`
    `-np 12 ./mpi_hello_world`

Outline

- Using Compilers on HPCC clusters
- Job Submission on HPCC clusters
- Transferring Data
- HPCC Policies

Why Using Job Scheduler

- When you login to HPCC Quanah or Ivy clusters, you only access to one machine which acts as a login gateway.
  - E.g. `quanah.Hpcc.ttu.edu` is one machine with limited resources
  - Users are not allowed to execute any program on Login nodes
- Job Scheduler provides a fair-shared resource allocation mechanism and allows your programs to be executed on worker nodes instead of login node.
  - Users can submit a request for specific computing resources they need for executing their program.
  - The job scheduler will take care of assigning fair priorities to users' jobs waiting in the queue and managing the resource allocation.

Job Submission Script Layout

UNIVA *Grid Engine*

-V	instructs the scheduler to keep the current environment settings
-cwd	instructs the scheduler to start the commands from the current directory
-S /bin/bash	instructs the scheduler to use the /bin/bash as the shell for the batch session
-N <job name>	sets the name for the job. Can be referenced in the script using $JOB_NAME
-o $JOB_NAME.o$JOB_ID	indicates the name of the standard output file.
-e $JOB_NAME.e$JOB_ID	indicates the name of the standard error file.

Job Submission Script Layout (Cont.)

-q <queue name>	instructs the scheduler to use the queue defined by <queue name>.
-pe <parallel environment>	instructs the scheduler to use the parallel environment defined by <parallel environment>.
-l h_vmem = <float>G	instructs the scheduler to reserve <float> gb of memory per slot
-l h_rt = HH:MM:SS	instructs the scheduler to set the maximum job time to HH:MM:SS
-P <project name>	instructs the scheduler to use the project defined by <project name>.

$NSLOTS	Number of slots that were defined by -pe option
$JOB_ID	The unique ID assigned to this job
$JOB_NAME	The Job name that was defined by –N option

Parallel Environment

- **Quanah:**
  - Omni Queue (-q omni)
    - -pe mpi <# of CPU cures>
      - Must request in increments of 36.
      - All MPI jobs should use this PE.
    - -pe sm <# of CPU cores>
      - Can request between 1 and 36.
      - Slots are guaranteed to be on one node.
- **Ivy:**
  - Ivy Queue (-q ivy)
    - -pe ivy <# of CPU cores>
      - Must request in increments of 20.

Requesting Memory

- Quanah limits memory usage per slot (essentially per core).
- By default, your processes will be limited to ~5.3 GB per slot.
  - Each node has 192GB of memory and 36 slots:

$$192 \div 36 = 5.333\ldots$$

- You can increase or decrease amount using the –l h_vmem=#G option.
- Examples:

```
#!/bin/sh
#$ -V
#$ -cwd
#$ -S /bin/bash
#$ -N MPI_Test_Job
#$ -o $JOB_NAME.o$JOB_ID
#$ -e $JOB_NAME.e$JOB_ID
#$ -q omni
#$ -pe mpi 36
#$ -l h_vmem=5.3G
#$ -l h_rt=48:00:00
#$ -P quanah
```

Uses 190.8 GB

```
#!/bin/sh
#$ -V
#$ -cwd
#$ -S /bin/bash
#$ -N MPI_Test_Job
#$ -o $JOB_NAME.o$JOB_ID
#$ -e $JOB_NAME.e$JOB_ID
#$ -q omni
#$ -pe mpi 36
#$ -l h_vmem=192G
#$ -l h_rt=48:00:00
#$ -P quanah
```

Uses 6.75 TB

Requesting Runtime Limit

- Recommended that you set the max runtime you expect a job will take.
- The scheduler behaves more efficiently as more jobs give it accurate runtime, slot and memory requirements.
  - Setting an accurate runtime allows your job to possibly be scheduled earlier than it would otherwise.
- You can set the max runtime for a job using -l h_rt=HH:MM:SS
- Examples:

```
#!/bin/sh
#$ -V
#$ -cwd
#$ -S /bin/bash
#$ -N MPI_Test_Job
#$ -o $JOB_NAME.o$JOB_ID
#$ -e $JOB_NAME.e$JOB_ID
#$ -q omni
#$ -pe mpi 36
#$ -l h_vmem=5.3G
#$ -l h_rt=48:00:00
#$ -P quanah
```

Max Runtime: 48 hours

```
#!/bin/sh
#$ -V
#$ -cwd
#$ -S /bin/bash
#$ -N MPI_Test_Job
#$ -o $JOB_NAME.o$JOB_ID
#$ -e $JOB_NAME.e$JOB_ID
#$ -q omni
#$ -pe mpi 36
#$ -l h_vmem=5.3G
#$ -l h_rt=3:00:00
#$ -P quanah
```

Max Runtime: 3 hours

Projects

What is Project?

Construct in Univa Grid Engine that provides a means to organize joint computational tasks from multiple users.

What does it do?

Defines resource usage policies for all jobs that belong to a project.

Projects on Quanah

- **quanah**
  - Maximum amount of running cores: 16,812
  - Default run time: 48 hours
  - Maximum run time: 48 hours
  - Allowed Parallel Environments: ‘sm’ and ‘mpi’

- **xlquanah**
  - Maximum amount of running cores: 144
  - Default run time: 72 hours
  - Maximum run time: 120 hours
  - Allowed Parallel Environments: ‘sm’

Projects on Ivy

- **hrothgar**
  - Available only on the 'ivy' queues.
  - Maximum amount of running cores: 2,000
  - Default run time: 48 hours
  - Maximum run time: 48 hours
  - Allowed Parallel Environments: 'ivy'
- communitycluster
  - Will not be used for this class!

Submitting Jobs

- Example Script
  - Set the name of our job to “MPI_TEST_JOB”
  - Requests to run on the Quanah cluster using the omni queue.
  - Sets the parallel environment to “mpi” and requests 36 cores.

```
#!/bin/sh
#$ -V
#$ -cwd
#$ -S /bin/bash
#$ -N MPI_Test_Job
#$ -o $JOB_NAME.o$JOB_ID
#$ -e $JOB_NAME.e$JOB_ID
#$ -q omni
#$ -pe mpi 36
#$ -l h_vmem=5.3G
#$ -l h_rt=48:00:00
#$ -P quanah

module load intel impi

mpirun --machinefile machinefile.$JOB_ID -np $NSLOTS ./mpi_hello_world
```

Submitting Jobs (cont.)

- Submit job
  - **qsub** <job submission script>
- Check job status
  - Command: **qstat**
  - job-ID, prior, name, user, state, submit/start at, queue, jclass, slots, ja-task-ID
    - "**r**": running
    - "**qw**": waiting in the queue
    - "**E**": error

```
quanah:/mpi_tutorial$ qsub mpi.sh
Your job 51516 ("MPI_Test_Job") has been submitted
quanah:/mpi_tutorial$ qstat
job-ID     prior   name       user     state submit/start at     queue                          jclass     slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
     51516 0.00000 MPI_Test_J Test     qw    07/26/2017 15:41:16                                    36
quanah:/mpi_tutorial$ qstat
job-ID     prior   name       user     state submit/start at     queue                          jclass     slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
     51516 8.79812 MPI_Test_J Test     r     07/26/2017 15:41:16 omni@compute-20-4.localdomain          36
quanah:/mpi_tutorial$
```

Interactive Jobs

- Sometimes it might be necessary to use the clusters in interactive mode :
  - Software Development
  - Testing Programs
  - Debugging codes
  - Observing Output/Error files

- Start an interactive job using `qlogin` command
  - **qlogin -P <cluster> -q <queue> -pe <pe> <#>**
  - Examples:
    - **qlogin** -P quanah -q omni -pe sm 1
    - **qlogin** -P hrothgar -q ivy -pe ivy 20

Interactive Jobs

- While using QLOGIN
  - Make sure the prompt changes to compute-#-#
  - If it doesn't change, check your qlogin and try again.
  - Make sure to run "exit" when you are finished
  - Keep in mind: Runtime limits apply to qlogin!

```
quanah:$ qlogin -P quanah -q omni -pe sm 1
Your job 75025 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 75025 has been successfully scheduled.
Establishing /export/uge/scripts/qlogin_wrapper session to host compute-8-24.localdomain ...
Last login: Tue Nov  7 11:06:15 2017
compute-8-24:$
```

Current Cluster Usage

- Viewing the current queue status and running/pending jobs
  - For a queue overview, run the command: "qstat -g c"
  - Visit the queue status webpage: http://charlie.hpcc.ttu.edu/qstat/qstat.html (Updates every 2 minutes)

Job Runtime Limits

Cluster	Queue	Project	Runtime Limit	# of Cores per node	Memory per node
Quanah	omni	quanah	48 hours	36 cores	192 GB
Quanah	omni	xlquanah	120 hours	36 cores	192 GB
Quanah	omni	hep / cbg	∞ hours	36 cores	192 GB
Ivy	ivy	hrothgar	48 hours	20 cores	64 GB
Ivy	community cluster queues (Chewie, R2D2, Yoda, ancellcc, blawzcc, caocc, dahlcc, phillipscc, tangcc, tang256cc)	communitycluster	∞ hours	vary	vary

Debugging Failed Jobs

- Job output
  - Standard: $JOB_NAME.o$JOB_ID
  - Error: $JOB_NAME.e$JOB_ID

- When debugging:
  - Check the output files for errors
  - Check the output of qacct –j <job_ID>
    - failed
    - exit_status
    - maxvmem
    - start_time & end_time (<runtime limit)
    - low

Outline

- Using Compilers on HPCC clusters
- Job Submission on HPCC clusters
- Transferring Data
- HPCC Policies

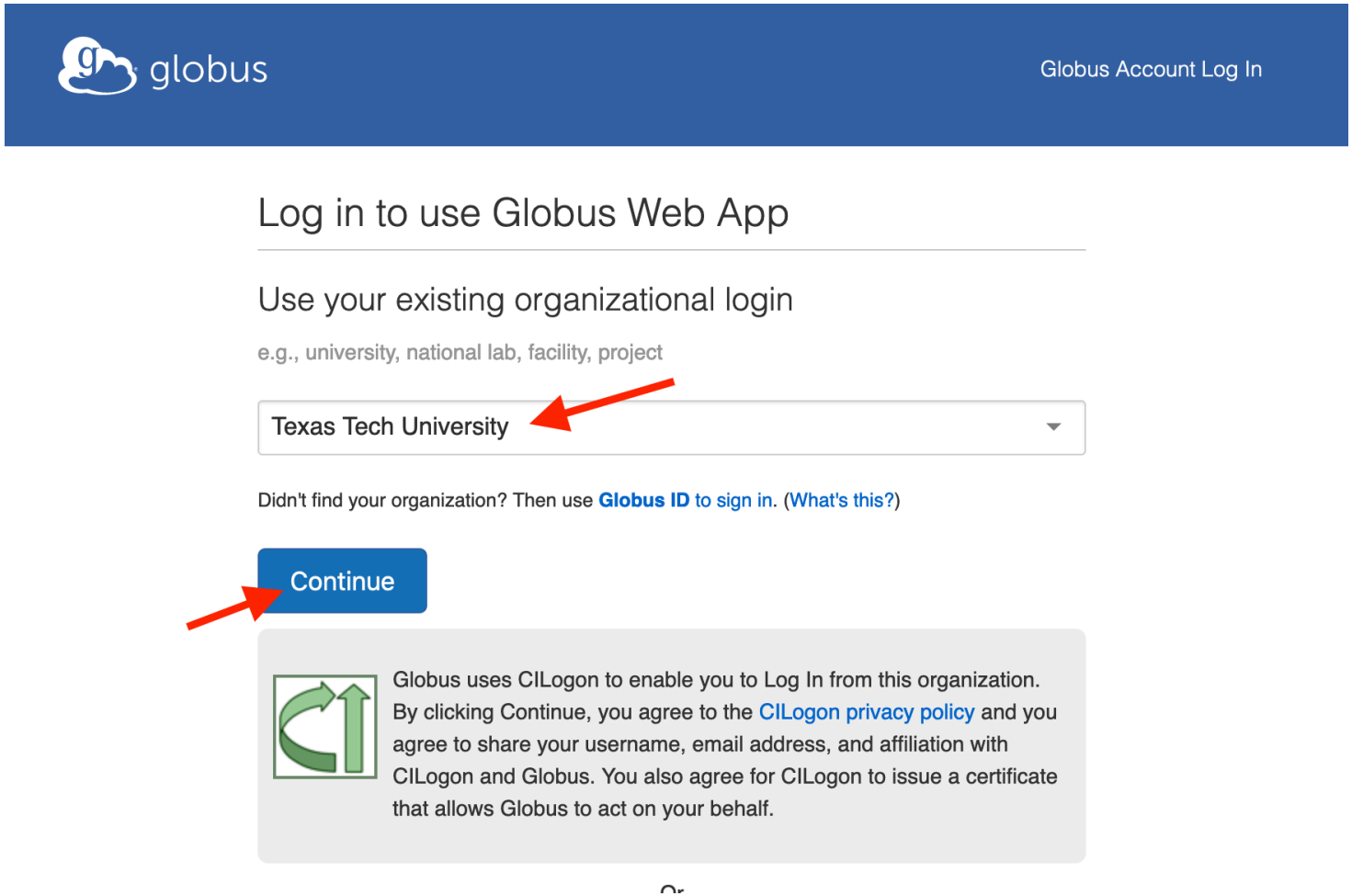
Transferring Data

- Whenever possible, refrain from using:
  - scp,
  - sftp,
  - rsync,
  - Or any other data transfer tool.
- Using Globus Connect

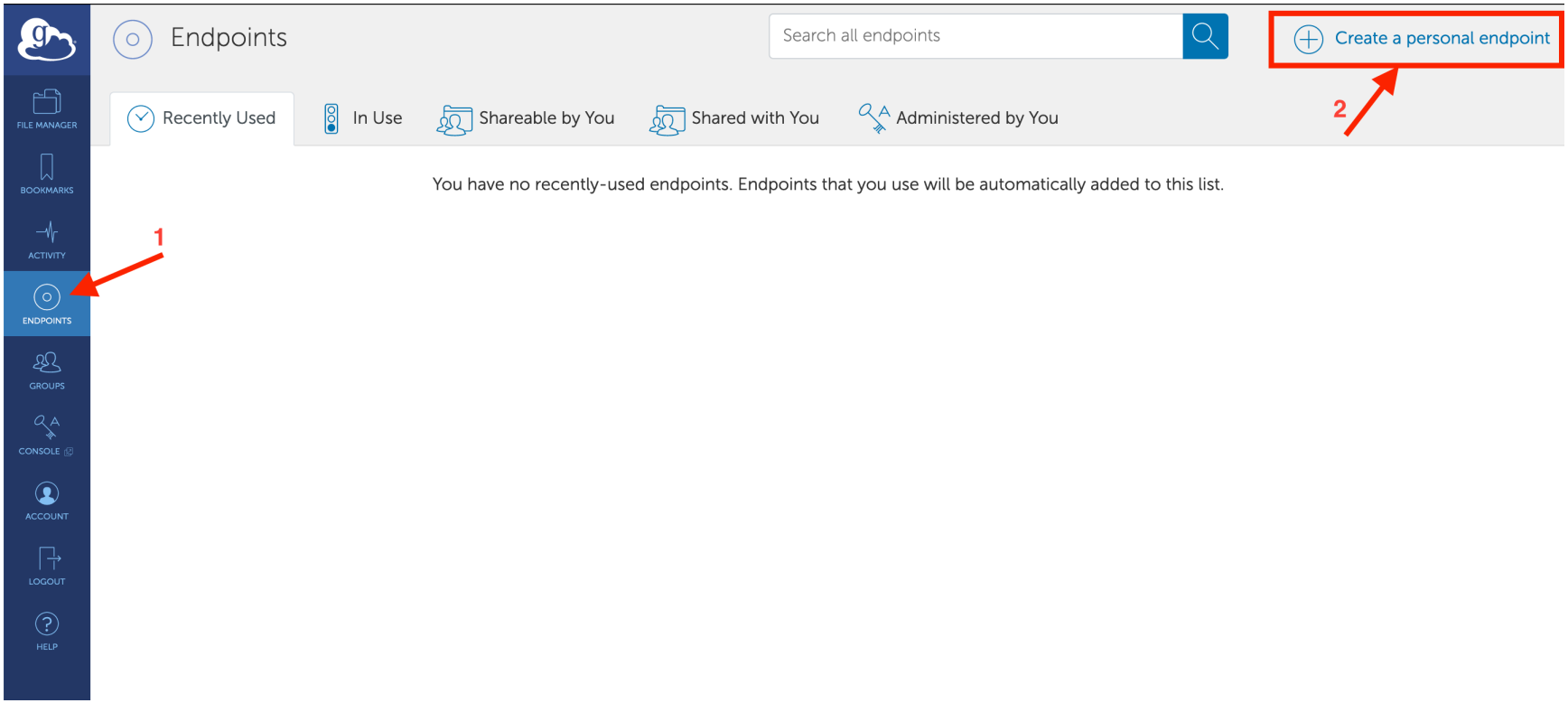
Transferring Data via Globus Connect

- Why use Globus?
  - Globus Connect service is well connected to the campus network.
  - The data transfer nodes are better positioned for transferring user data.
  - Globus connect service eliminates the data transfer load from the cluster login node.
  - Globus connect works with Linux, Mac and Windows and is controlled through a web GUI.
  - Numerous other sites (including TACC) support Globus Connect data transfers.

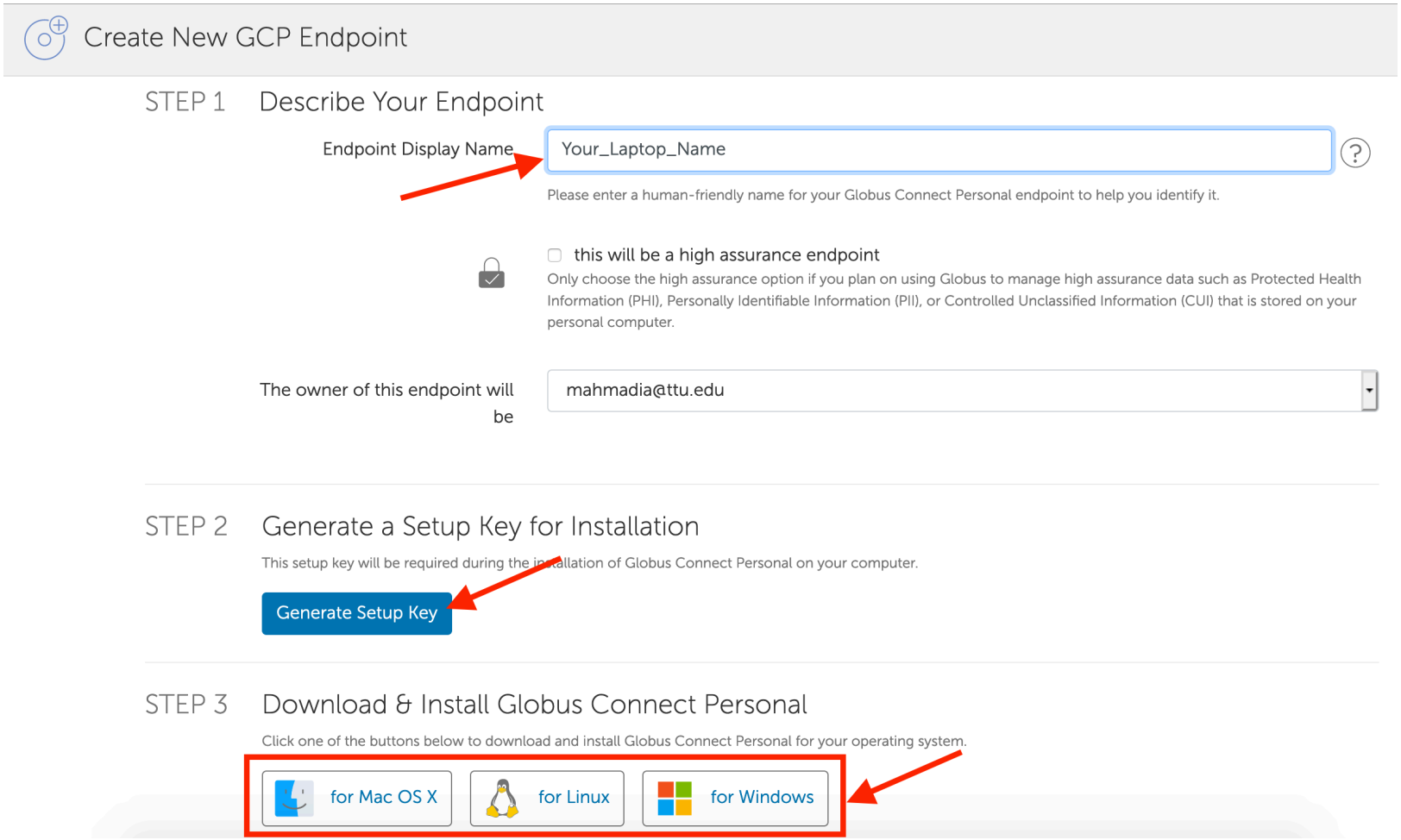
Login to Globus



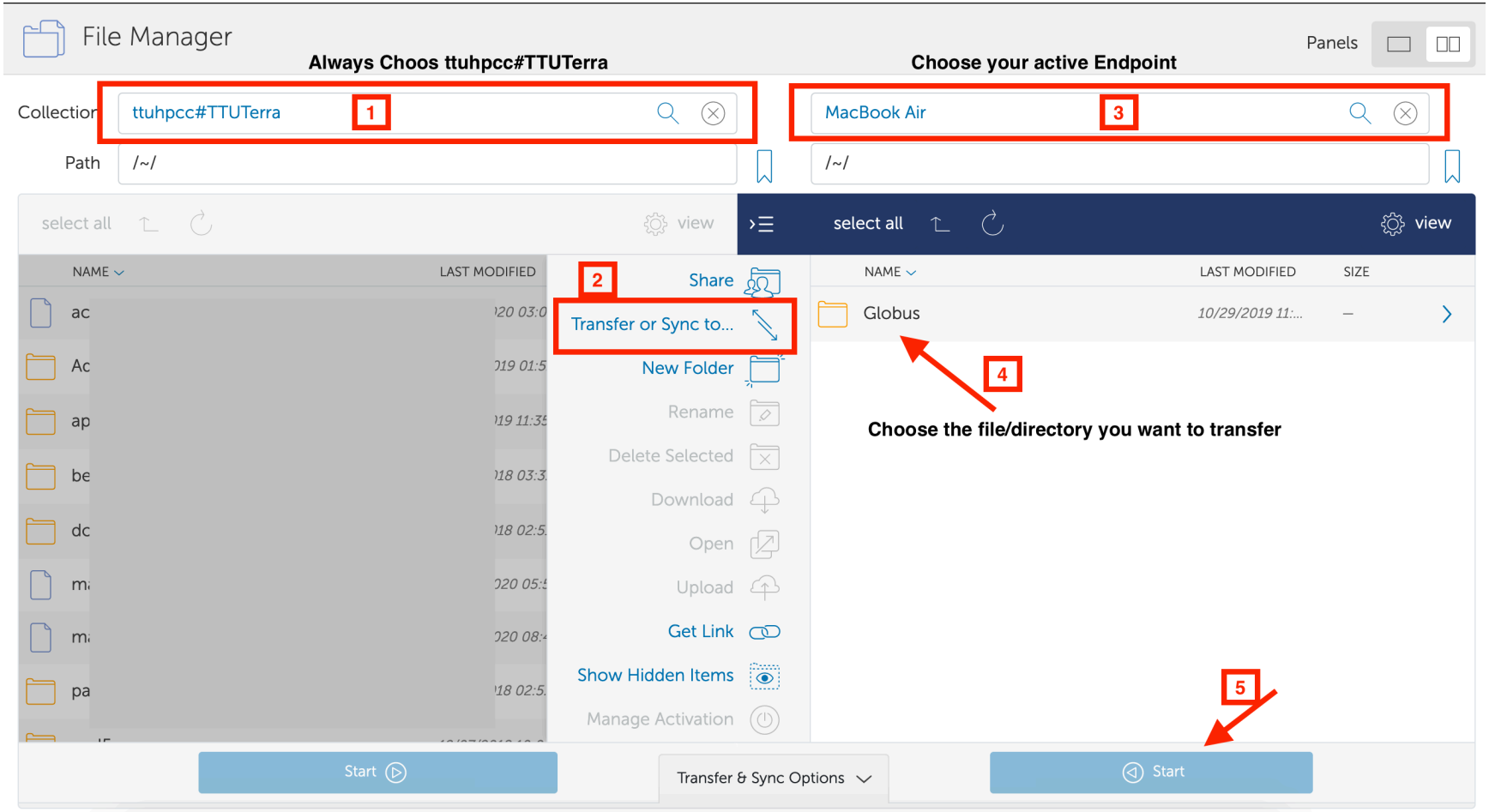
Create Endpoint on Your Local Machine



Create Endpoint on Your Local Machine (cont.)



Transfer Data Between HPCC Endpoint and Yours



Outline

- Using Compilers on HPCC clusters
- Job Submission on HPCC clusters
- Transferring Data
- HPCC Policies

HPCC Policies

- Login nodes (Quanah and Ivy)
  - No jobs are allowed to run on the login node.
- SSH Access
  - No direct SSH access allowed to the nodes.
- Scratch will be purged monthly.
  - Automatic removal of all files not accessed within the past year
  - Purge will aim to drop scratch usage below 65%
  - This may remove files accessed in past year

www.hpcc.ttu.edu

hpccsupport@ttu.edu

Questions?

Questions/Suggestions/Comments are always welcome!

Write me: yong.chen@ttu.edu
Call me: 806-834-0284
See me: ENGCTR 315

If you write me an email for this class, please start the email subject with [CS4379] or [CS5379].