Triton Shared Computing Cluster (TSCC) 101 Spring Training

TSCC User Services
San Diego Supercomputer Center
tscc-support@ucsd.edu

University of California San Diego
San Diego
March 3, 2022

Outline

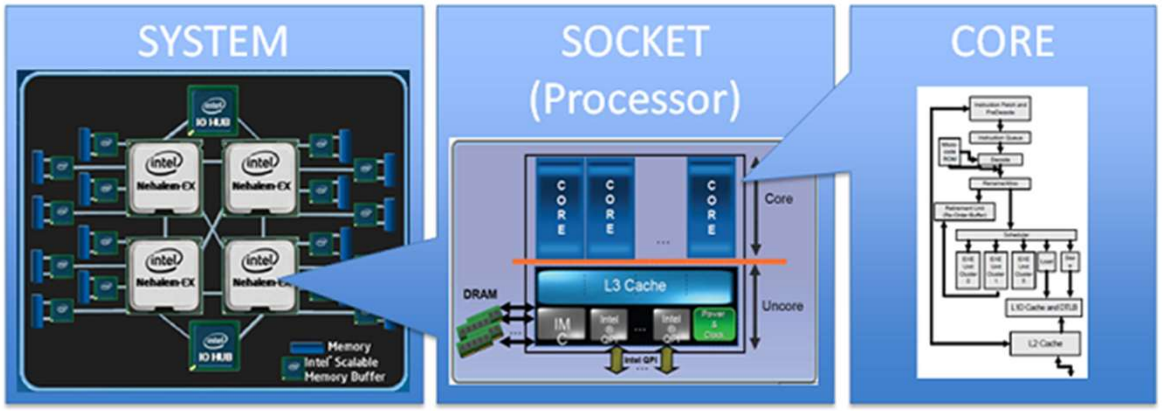
- **General cluster architecture**
- **Account and allocation**
- **Software environment**
    - How to access TSCC cluster
    - File systems
    - Transfer files between cluster and local PC
    - The software management tool Module
- **Job management**
    - Job queue basics
    - Interactive vs Batch jobs
    - Submit and monitor your jobs
- **Understanding job scheduling**

TSCC 101 Spring Training

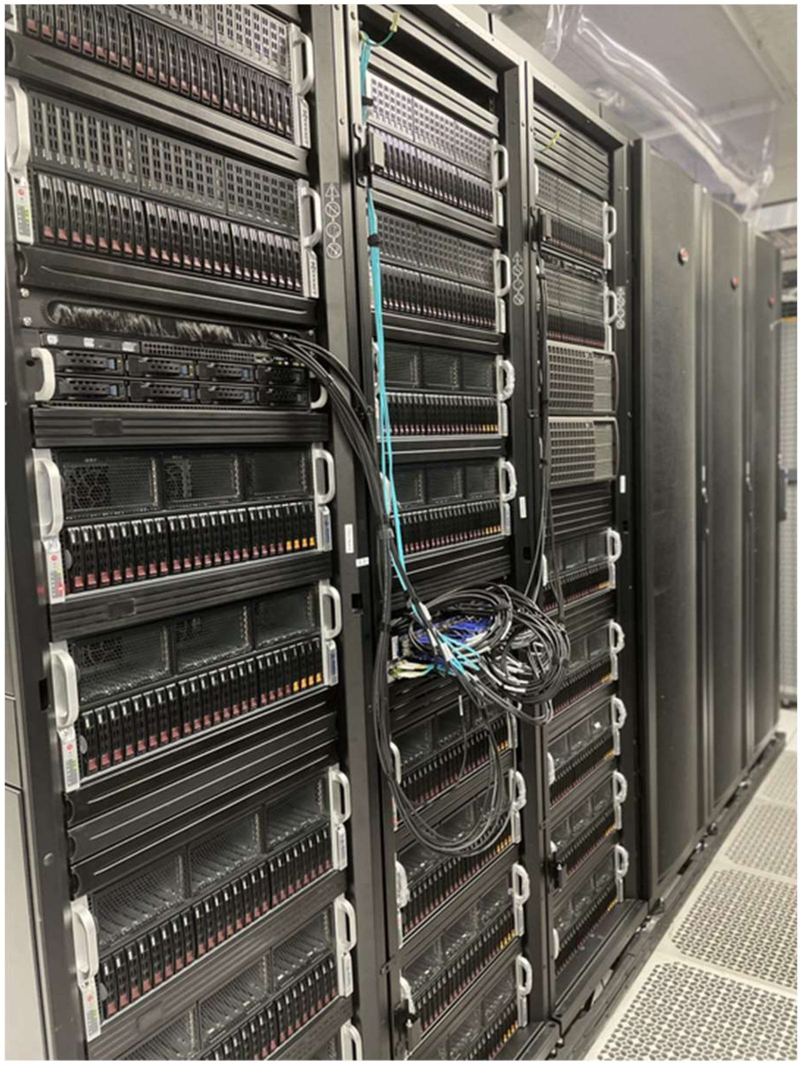
General cluster architecture

Cluster Nomenclature

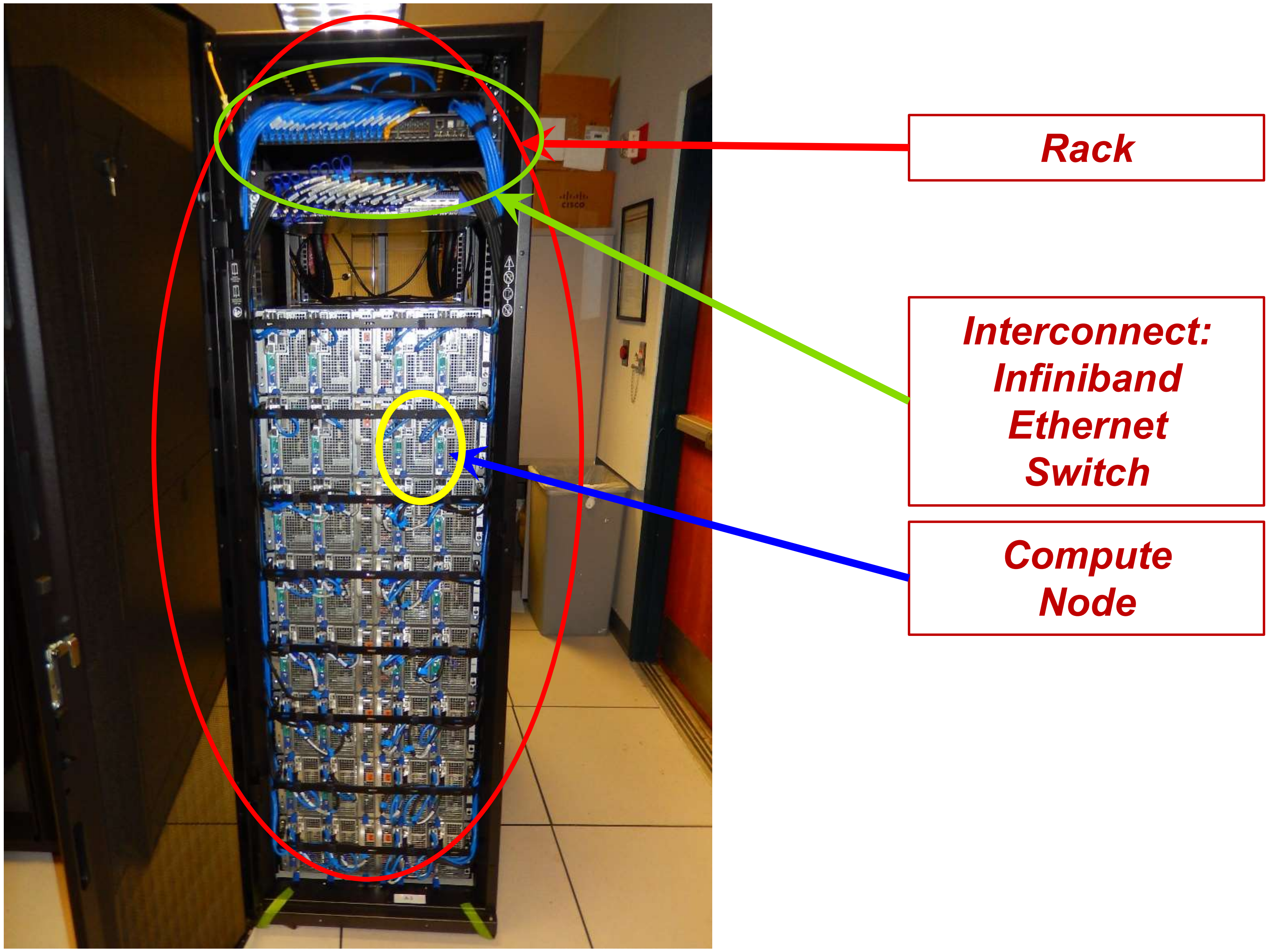
Term	Definition
Cluster	A set of connected computer nodes that work together, each node set to perform the same kind of task (job).
Node	A single, named host machine in the cluster.
Core	The basic computation unit in the processor (CPU). For example, a quad-core processor has 4 cores.
Job	A user's request to use a certain amount of resources for a certain amount of time on cluster for his/her work.



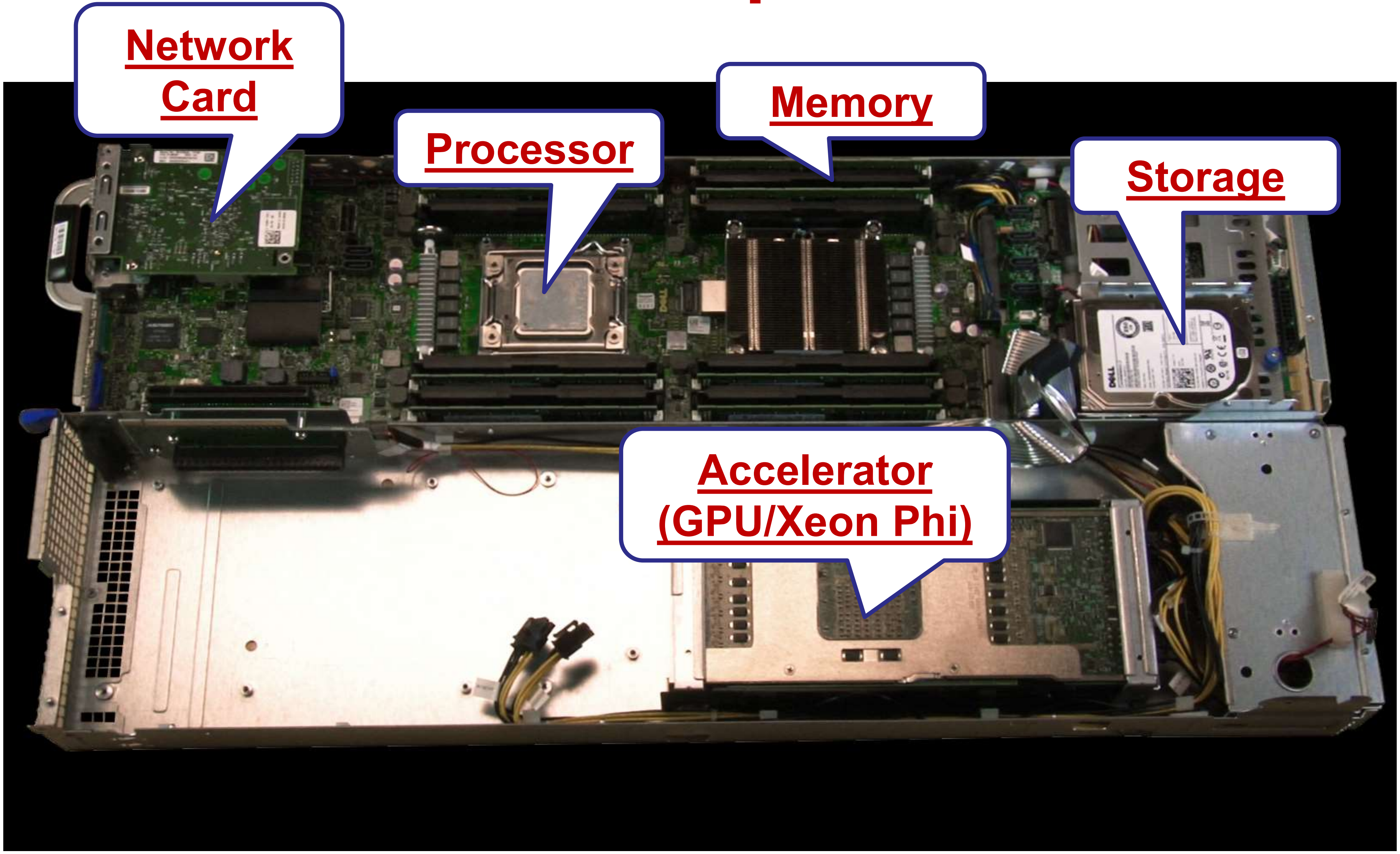
TSCC Cluster Racks



Inside A Cluster Rack

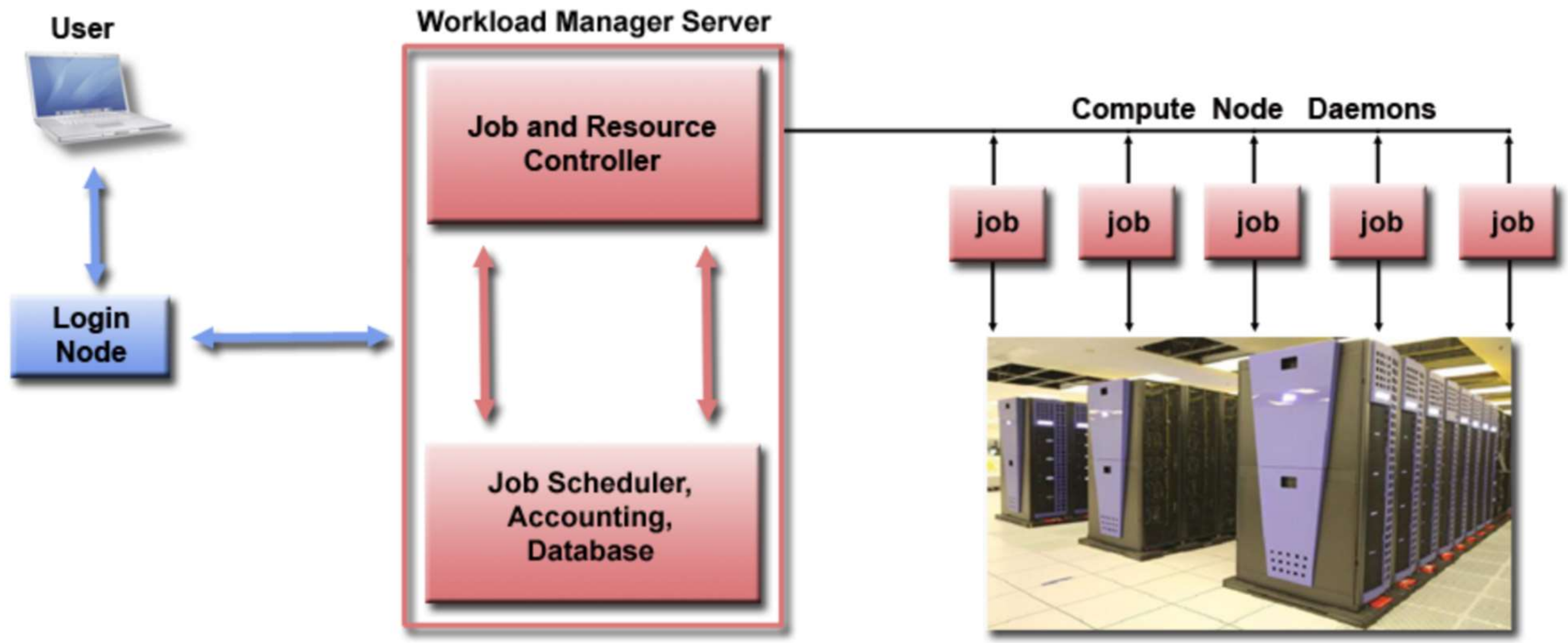


Inside a Compute Node



General Cluster Architecture

- Multiple compute nodes
- Multiple users
- Each user may have multiple jobs running simultaneously



TSCC 101 Spring Training

Account and allocation

Account Type

- **Condo account / hotel account**

- **Trial account**
  - 250 SUs valid for 90 days

- **Training account**
  - Some technical trainings, workshops, bootcamps or classes on TSCC may provide training accounts to their attendees
  - This TSCC 101 training does NOT provide training accounts

How Do I Get a TSCC account?

- **Send your account request to tscc-support@ucsd.edu**
  - Full name
  - Institutional email address (commercial email address such as Gmail will be handled on a case-by-case basis)
  - The group/lab name (if your group is using TSCC)
  - Non-USCD users: please provide an ssh public key for login

- **We may also need a note from your P.I. (or the liaison in your group) approving your account request. Please cc your P.I. in the account request email**

- **P.I.: The person who purchased node or time on TSCC and is responsible for your activities on the cluster.**

Account Eligibility Test

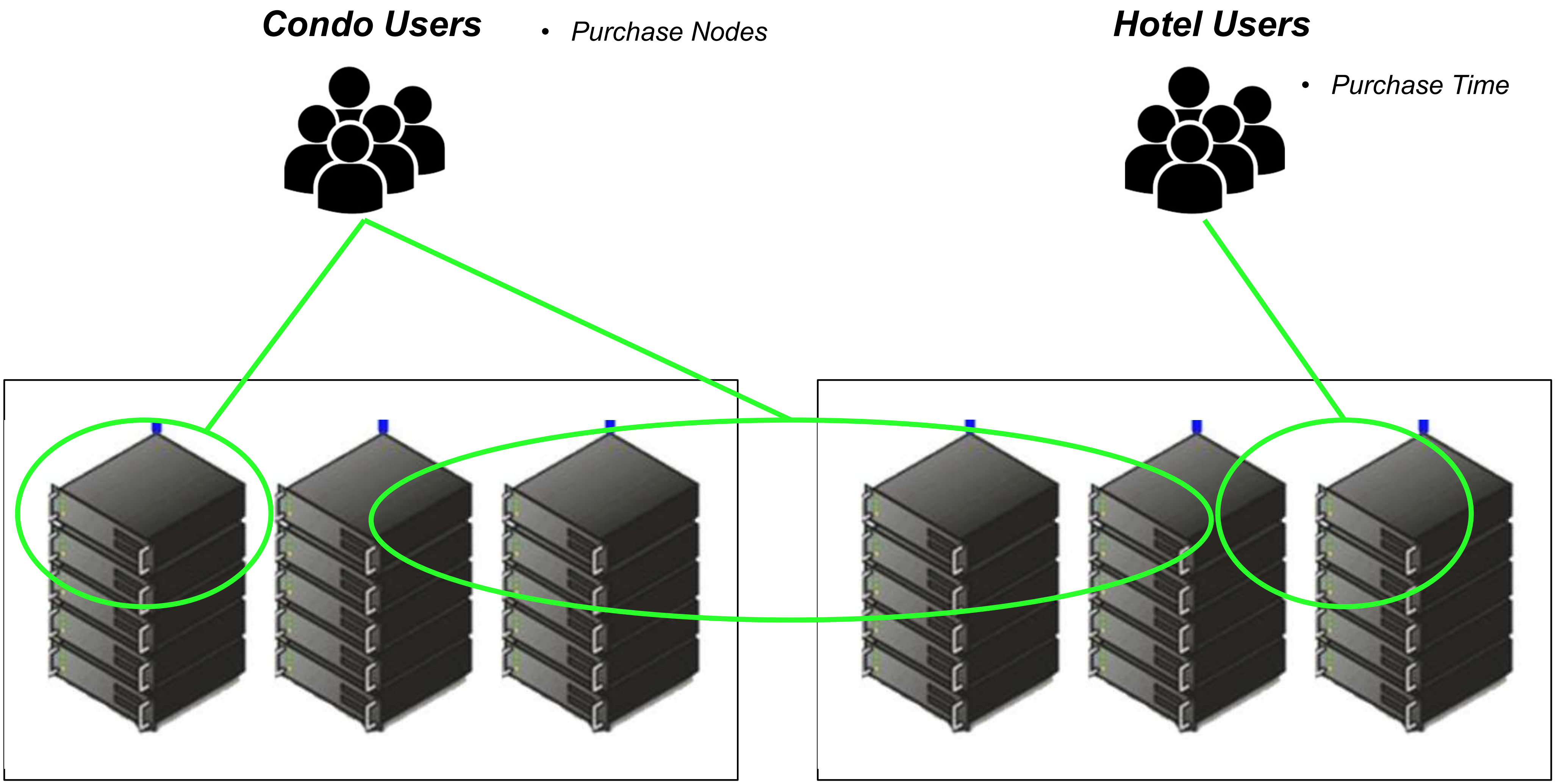
- **I can be granted a TSCC account if:**
  a) I am using TSCC computational resource for my research, the account is sponsored by my advisor (PI)
  b) I am attending this TSCC 101 training session, the account will be provided by the TSCC team
  c) I am attending a technical bootcamp that requires using TSCC resource, the training account will be provided by the TSCC team
  d) a and b
  e) a and c
  f) a, b and c

Allocation

- **An allocation is a block of service unit (SUs) that allows a user to run jobs on a supercomputer cluster**
  - One SU is one core-hour
  - Example:
    - 40 SUs will be charged for a job that runs 10 hours on 4 cores

- **TSCC jobs need to be charged to a valid allocation.**
  - exception: “glean” queue allows free use of spare capacity for short-running or re-startable jobs

TSCC Operations



Allocation Types

- **Condo allocation**
  - Allocation is credited annually at the beginning of the month the nodes were added to TSCC
  - The credited SUs is equal to the full-time usage of node(s) for one year (allowing for 3% maintenance downtime).
    - Example:

    Approximately 140K SUs will be credited annually for a node that has 16 cores
  - Expire one year after. There is no roll over of left-over SUs.

- **Hotel allocation**
  - Allocation can be credited at any time of the year. Pay-as-you-go.
  - For example, the cost for UCSD affiliates is $0.025 per 1 SU. The minimum purchase is 10,000 SUs for $250
  - No expiration date

How to Request or Join an Allocation

- **Send your allocation request to tscc-support@ucsd.edu**
  - Condo allocation: if the allocation is exhausted before the next credit date, TSCC Teams will handle it on a case-by-case basis
  - Hotel allocation: provide the funding information (i.e.: project & task #) to create an allocation or add more SUs to an existing allocation

- **Join an existing allocation**
  - tscc-support@ucsd.edu, we may also need a note from your P.I. (or the liaison in your group) approving your allocation join request. Please cc your P.I. in the email

TSCC 101 Spring Training

Software environment

Accessing TSCC via SSH (Secure Shell)

- **On Linux and macOS**
  - use ssh command on a terminal to connect
- **Windows box (ssh client):**
  - MobaXterm (recommended)
  - Putty
- **Authentication**
  - Username assigned by TSCC admin + UCSD AD password
  - SSH Keys (For non-UCSD users please send public key to tscc-support@ucsd.edu)
- **Hostname**
  - tscc-login.sdsc.edu

Accessing TSCC on Linux and MacOS

- **Open a terminal and type:**

  $ ssh username@tscc-login.sdsc.edu

  - The hostname "tscc-login.sdsc.edu" will bring you to different login node (login1, login2, login11 etc.)

- **Enter UCSD password, or login via SSH keys.**

- **Instructions for generating ssh keys on Linux and macOS is in the next slide**

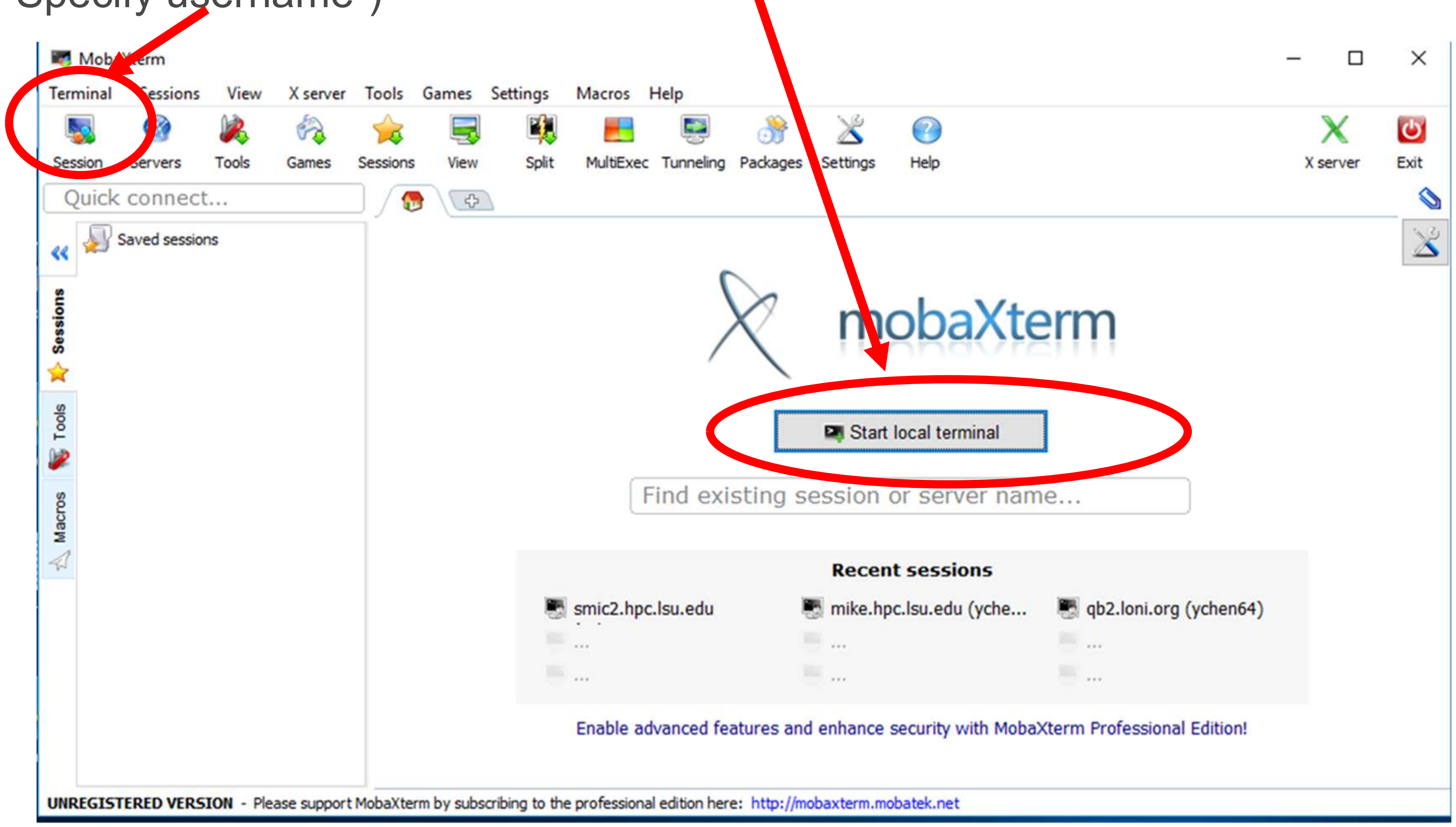
Accessing TSCC on Linux and MacOS

- **Generating a key pair on a Linux local machine:**

  $ ssh-keygen -b 4096 -t rsa
  - A SSH key pair “id_rsa” and “id_rsa.pub” will be created.
  - Make sure you have a password on the private key on your local machine.
- **Installing the private key (don’t share it to anyone)**
  - Keep the private key “id_rsa” in the ~/.ssh on your PC
- **Installing the public key**
  - The public key is for using on servers like TSCC, so it can be shared.
  - copy id_rsa.pub to TSCC's ~/.ssh (if you don’t have access to TSCC, send us your key)
  - Add the content to authorized_keys: $ id_rsa.pub >> ~/.ssh/authorized_keys
- **Use ssh-agent or keychain to avoid repeatedly typing the private key password.**
- **Instructions: https://www.ssh.com/academy/ssh/keygen**

Accessing TSCC on Windows - MobaXterm

- **First time user, choose either one:**
  - Start a local terminal, then use ssh command on a terminal
  - Start a new remote session -> SSH, enter hostname and username (remember to check "Specify username")

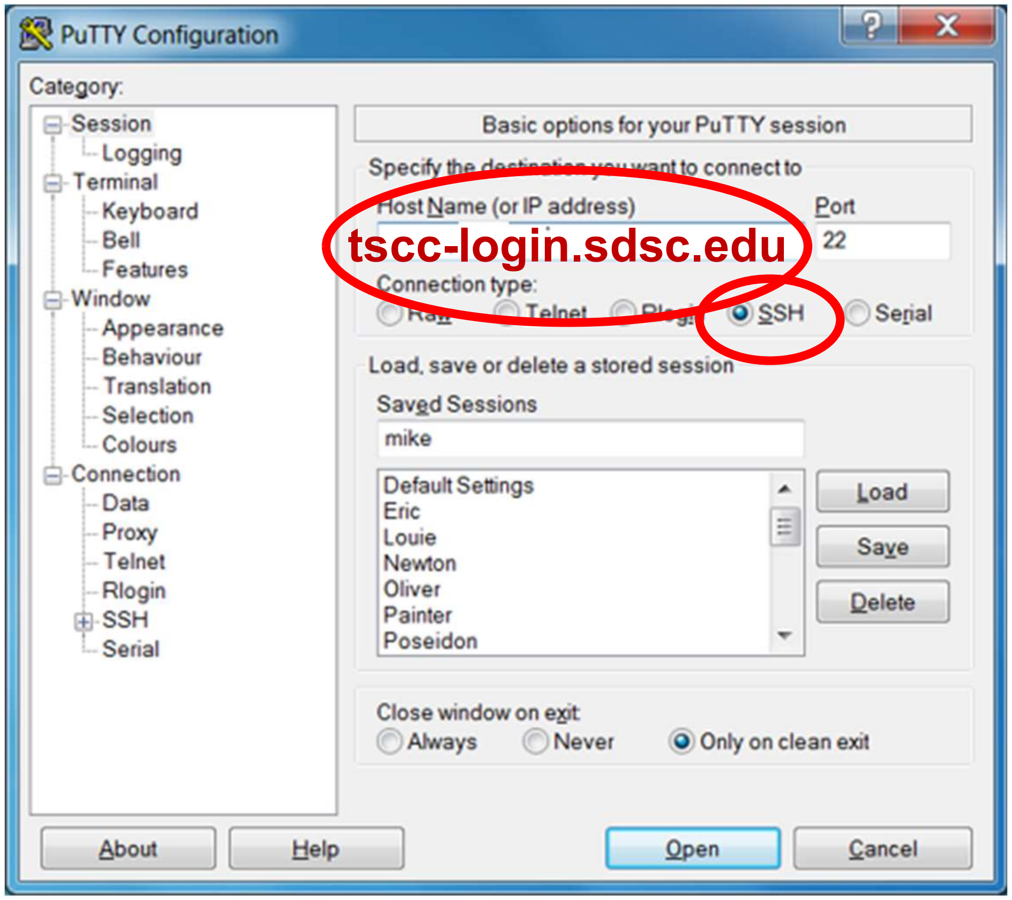


Enable advanced features and enhance security with MobaXterm Professional Edition!

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

Accessing TSCC on Windows - Putty

- Enter Hostname and choose ssh


PuTTY Configuration

Category:

Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): tscc-login.sdsc.edu

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: mike

Default Settings
Eric
Louie
Newton
Oliver
Painter
Poseidon

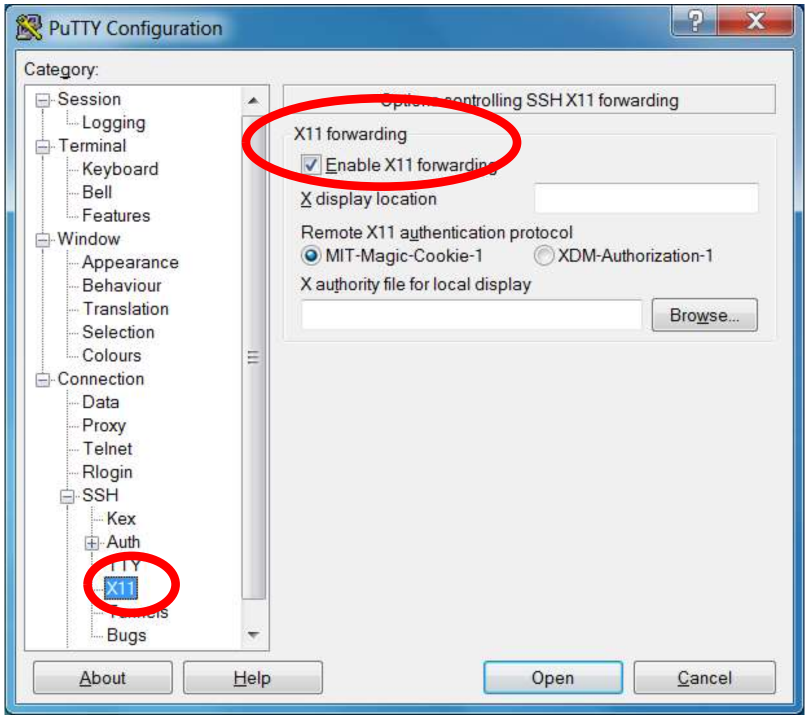
Close window on exit: Always Never [✓ Only on clean exit]

Accessing TSCC via SSH Keys on Windows

- **Generating a key pair with MobaXterm:**
  - Start the MobaXterm SSH Key Generator (MobaKeyGen) found under the MobaXterm Tools menu
  - Make sure you have a password on the private key on your local machine
- **Installing the private key (don't share it to anyone)**
  - MobaXterm: Edit a saved session -> Advanced SSH settings -> Check Use Private key
- **Installing the public key**
  - The public key is for using on servers like TSCC, so it can be shared.
  - copy id_rsa.pub to TSCC's ~/.ssh (if you don't have access to TSCC, send us your key)
  - Add the content to authorized_keys: $ id_rsa.pub >> ~/.ssh/authorized_keys
- **Use ssh-agent or keychain to avoid repeatedly typing the private key password.**
  - MobaXterm: Settings menu -> Configuration -> SSH -> Check Use internal SSH agent -> add the private key

Enable X11 Forwarding

- **On Linux or Mac, simply pass the -X option to the ssh command line**
  - ssh -X username@tscc-login.sdsc.edu
  - ssh -Y: Enables trusted X11 forwarding
- **On Windows using Putty**
  - Connection->SSH->X11->Enable X11 forwarding
  - Install X server (e.g. Xming)
- **On Windows using MobaXterm**
  - X server already set up
  - Automatically start X server at start up (Settings -> X11)


PuTTY Configuration

Category:

Options controlling SSH X11 forwarding

X11 forwarding

☑ Enable X11 forwarding

X display location

Remote X11 authentication protocol

● MIT-Magic-Cookie-1 ○ XDM-Authorization-1

X authority file for local display

Cluster Access Test

- **How do I connect to TSCC?**
  a) Open a web browser, enter the TSCC website address: https://www.sdsc.edu/services/hpc/tscc/index.html

  b) Use ssh command or an ssh (secure shell) client such as MobaXterm/Putty

  c) Go to the machine room in SDSC Datacenter and connect my laptop to the nodes using a cable

Accounts and Allocation Usage Monitoring

- **Useful commands on the login node**
  - id, find out the Linux group(s) of the account

```
[ychen64@tscc-login12 ~]$ id
uid=516315(ychen64) gid=7192(tscc-admin) groups=7192(tscc-
admin),50(sdsc),300(use300),5213(gaussian),6367(matlab-users),7193(builder-group)
```

- **gbalance, check allocation balance**

```
[ychen64@tscc-login2 ~]$ gbalance -u ychen64
Id Name          Amount Reserved Balance CreditLimit Available
-- ------------- ------ -------- ------- ----------- ---------
13 builder-group  78148        0   78148           0     78148
```

- glsproject, check who is in allocation

```
glsproject allocation_name
```

- gstatement, generate a summary of all activity on the account (** PLEASE be sure to always include the start and end time as paraters)

```
gstatement -u user_name -s 2019-01-01 -e 2019-02-28
gstatement -p group_name -s 2019-01-01 -e 2019-02-28
```

TSCC: Filesystems

- **TSCC filesystems are accessible from all nodes (login, compute, gpus, datamover etc.)**
- **Home directory: /home/username**
  - 100GB space limit
  - Source files, small input files
  - NOT for heavy I/O or running jobs
- **Lustre filesystem: /oasis/tscc/scratch/**
  - Good for storing large scale scratch data
  - Shared parallel file system (~1.7PB)
  - Purged after 3 months of inactivity

Filesystem contd..

- **Projects filesystem: /projects**
  - Not managed by TSCC
  - Available for purchase with SDSC RDS group

- **Compute Node's local scratch: $TMPDIR**
  - Local scratch for compute node
  - Good for writing small files.
  - Shared space with around 220-270GB
  - Purged at the end of running jobs

File Transfer (Linux/Mac)

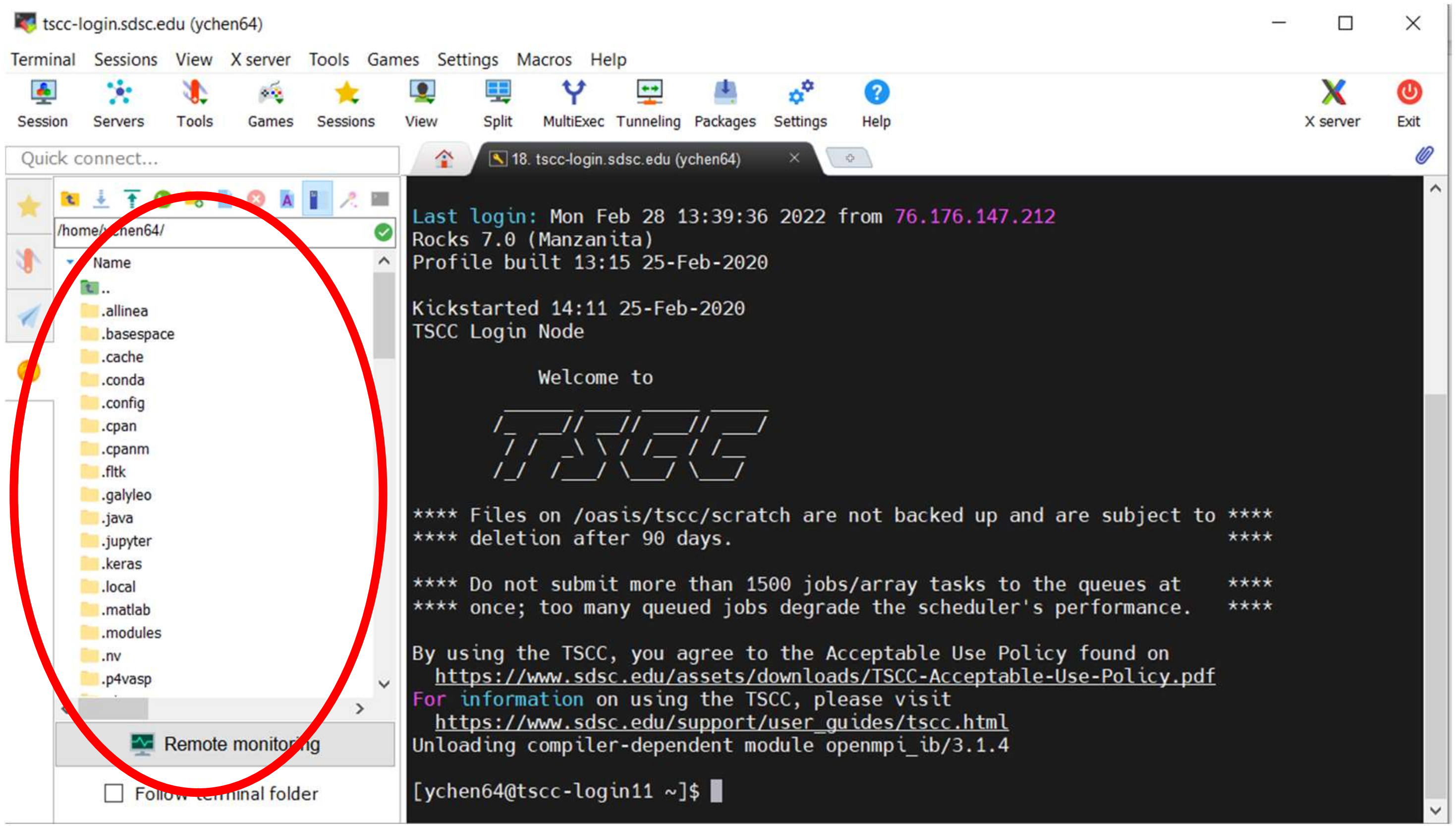
- **From/to a Unix/Linux/Mac machine (including between the clusters)**
  - scp or rsync command
    - Syntax: scp <options> <source> <destination>
- **From a download link on a website (usually in a web browser)**
  - Right click on the link and then copy the link location
  - Run wget command

```
[ychen64@tscc-login1 ~]$ wget <paste_the_copied_link_here>
```

- **Globus**
  - Send us your XSEDE username (account can be created at portal.xsede.org), we will activate it and notify you.
  - Go to www.globus.org, then authenticate with your XSEDE credentials
  - To access TSCC scratch, look for the Comet endpoint: xsede#comet
  - Now you will have access to /oasis/tscc/scratch/<your_username> on TSCC
- **Large files should be transferred on the datamover node**
  - on the login node, "ssh tscc-dm1"

File Transfer (Windows)

- **From/to a Windows machine**
  - Use a client that supports the scp protocol (e.g. MobaXterm)


tscc-login.sdsc.edu (ychen64)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
18. tscc-login.sdsc.edu (ychen64)
/home/ychen64/
Name
..
.allinea
.basespace
.cache
.conda
.config
.cpan
.cpanm
.fltk
.galyleo
.java
.jupyter
.keras
.local
.matlab
.modules
.nv
.p4vasp
Remote monitoring
Follow terminal folder

```
Last login: Mon Feb 28 13:39:36 2022 from 76.176.147.212
Rocks 7.0 (Manzanita)
Profile built 13:15 25-Feb-2020

Kickstarted 14:11 25-Feb-2020
TSCC Login Node

           Welcome to

**** Files on /oasis/tscc/scratch are not backed up and are subject to ****
**** deletion after 90 days.                                           ****

**** Do not submit more than 1500 jobs/array tasks to the queues at    ****
**** once; too many queued jobs degrade the scheduler's performance.   ****

By using the TSCC, you agree to the Acceptable Use Policy found on
  https://www.sdsc.edu/assets/downloads/TSCC-Acceptable-Use-Policy.pdf
For information on using the TSCC, please visit
  https://www.sdsc.edu/support/user_guides/tscc.html
Unloading compiler-dependent module openmpi_ib/3.1.4

[ychen64@tscc-login11 ~]$
```

Application Software

- **Installed Software**
  - Mathematical and utility libraries
    - FFTW, HDF5, NetCDF, PETSc...
  - Applications
    - Amber, Gaussian, NAMD, Gromacs, R, LAMMPS...
  - Visualization
    - VMD
  - Programming Tools
    - DDT, TAU...
- **List of software**

  https://www.sdsc.edu/support/user_guides/tscc.html#software
- **System widely installed under /opt or /projects/builder-group/jpg**
- **User requested packages**
  - Usually installed in user home directory, unless request by a group of users, in which case it will be installed under /projects or /opt

Software Environment: Module

- **Environment variables**
  - PATH: where to look for executables
  - LD_LIBRARY_PATH: where to look for shared libraries
  - LD_INCLUDE_PATH: where to look for header and include files
- **Other environment variables sometimes needed by various software**
    - LIBRARY_PATH, C_LIBRARY_PATH, LDFLAGS, LDLIBS

- **Environment Modules**
  - An application that helps users set up their environment variables. Most supercomputing sites (including XSEDE) use modules. Much more convenient than setting variables in .bashrc

Using Environment Modules

- **Environment Modules is a framework to manage what software is loaded into a user's environment. Its functionality includes**
  - List all software packages currently available in the Environment Modules system,
  - List all software packages loaded into a user's environment,
  - Load/Switch software packages into a user's environment
  - Unload a software package from a user's environment.

Modules: List All Available Packages

- **The command to list all available packages is: module avail (av)**

```
[ychen64@tscc-login1 ~]$ module av

------------------------------------------------------------ /home/ychen64/my_module ------------------------------------------------------------
bzip2/1.0.6/intel-2018.1.163 curl/7.55.1/intel-2018.1.163 miniconda2              r/3.6.1/intel-2018.1.163

--------------------------------------------------------- /opt/modulefiles/applications ---------------------------------------------------------
abinit/8.10.2(default)          dendropy/4.4.0(default)         matlab/2014a                    R/3.6.1(default)
abyss/2.1.5(default)             diamond/0.9.22(default)         matlab/2014b                    randfold/2.0(default)
…
…
---------------------------------------------------------- /opt/modulefiles/compilers ----------------------------------------------------------
cilk/5.4.6(default)      guile/2.2.4(default)      llvm/7.0.1(default)      python/2.7.15(default)
cmake/3.12.1(default)      intel/2018.1.163(default) mono/5.18.0.268(default)  upc/2019.4.4(default)
gnu/7.2.0(default)        julia/1.6.1(default)      pgi/18.10(default)
```

- **The format of the listed packages is <package name>/<package version>. For example, gnu/7.2.0 is version 7.2.0 of gnu compilers.**

Modules: Access more Modules on TSCC

- **On TSCC some software applications are installed at /projects/builder-group/jpg, which modules are not visible by default.**
  - Example modules installed at /projects/builder-group/jpg:

  R/4.0.2

  R/4.1.2

  python/3.7.0
- **To access those modules, add the module location to the MODULEPATH environment variable**

```
[ychen64@tscc-login1 ~]$ export MODULEPATH=/projects/builder-group/jpg/modulefiles/applications:$MODULEPATH
```

Modules: List Currently Loaded Packages

- **To see what packages are currently loaded into a user's environment, the command is: module list**

```
[ychen64@tscc-login1 ~]$ module list
Currently Loaded Modulefiles:
  1) intel/2018.1.163   2) openmpi_ib/3.1.4
```

- **The above listing shows that this user has two packages loaded**

Modules: Load/Unload a Package

- The command for loading a package into a user's environment is **module load <package name>**.
- The command for unloading a package is: **module unload <package name>**.
- If a specific version of a package is desired, the command can be expanded to **module load <package name>/<package version>**.

```
[ychen64@tscc-login1 ~]$ module av cuda
---------------------------------------------------------- /opt/modulefiles/applications ------------------------------------------------------------------------
cuda/10.1.243(default) cuda/11.2.0
[ychen64@tscc-login1 ~]$ module load cuda/11.2.0
[ychen64@tscc-login1 ~]$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Mon_Nov_30_19:08:53_PST_2020
Cuda compilation tools, release 11.2, V11.2.67
Build cuda_11.2.r11.2/compiler.29373293_0
```

Modules: Unload all Loaded Packages

- **To unload all loaded module files, use module purge**

```
[ychen64@tscc-login1 ~]$ module list
Currently Loaded Modulefiles:
  1) intel/2018.1.163   2) openmpi_ib/3.1.4   3) cuda/11.2.0
[ychen64@tscc-login1 ~]$ module purge
[ychen64@tscc-login1 ~]$ module list
No Modulefiles Currently Loaded.
```

Modules: Dependencies

- **Note that Modules will load any prerequisites (dependencies) for a package when that package is loaded.**

```
[ychen64@tscc-login1 ~]$ module list
No Modulefiles Currently Loaded.
[ychen64@tscc-login1 ~]$ module load gromacs/2021.1
[ychen64@tscc-login1 ~]$ module list
Currently Loaded Modulefiles:
  1) mkl/2018.1.163      2) gnu/4.9.2          3) intelmpi/2019.8.254   4) gromacs/2021.1
```

Modules: Display the module changes

- The display/show command will detail all changes that will be made to the user's environment: **module disp <package name>** .

```
[ychen64@tscc-login1 ~]$ module disp python/2.7.15
-------------------------------------------------------------------
/opt/modulefiles/compilers/python/2.7.15:

module-whatis    python
module-whatis    Version: 2.7.15 and 3.6.9
setenv           PYTHONROOT /opt/python
prepend-path     PATH /opt/python/bin
prepend-path     LD_LIBRARY_PATH /opt/python/lib
prepend-path     LIBPATH /opt/python/lib
setenv           KMP_INIT_AT_FORK FALSE
module           load gnu mkl
-------------------------------------------------------------------
```

Modules: Load Automatically on Login

- **Modules can be loaded automatically on login by adding the appropriate module load commands to a user's ~/.bashrc file**
- **The following example shows a ~/.bashrc file that automatically loads R and gromacs/2021.1**

```
[ychen64@tscc-login1 ~]$ cat .bashrc
# .bashrc
…
…
module load R/3.6.1
module load gromacs/2021.1
```

Creating Your Own Module File

- **An example of a simple module file (/home/ychen64/my_module/bzip2/1.0.6/intel-2018.1.163):**

```
#%Module
proc ModulesHelp { } {
    puts stderr {   bzip2   }
}
module-whatis {Description: bzip2}
set root  /home/ychen64/packages/bzip2-1.0.6
conflict    bzip2
if { ![is-loaded  intel/2018.1.163 ] } {
    module load intel/2018.1.163
}

prepend-path    CPATH           $root/include
prepend-path    LD_LIBRARY_PATH         $root/lib
prepend-path    LIBRARY_PATH            $root/lib
prepend-path    MANPATH         $root/share/man
prepend-path    PATH            $root/bin
prepend-path    PKG_CONFIG_PATH         $root/lib/pkgconfig
```

Creating Your Own Module File contd..

- **Add the path to the key to the MODULEPATH environment variable:**

```
export MODULEPATH=~/my_module:$MODULEPATH
```

- **Then try loading:**

```
module load bzip2/1.0.6/intel-2018.1.163
```

Homework: Use Modules

- **Find the module for gromacs**
  - Set up your environment to use gromacs you choose (one time change)
  - Check if the module is correctly loaded by which gmx_mpi

- **Find the module for Python-2.7.15**
  - Set up your environment to permanently use Python-2.7.15
  - Check if the variables are correctly set by python --version

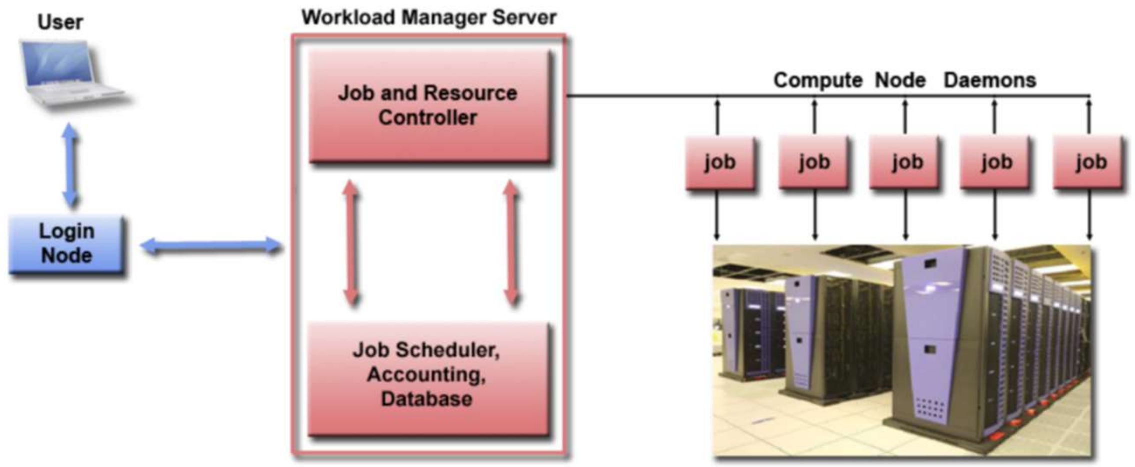
TSCC 101 Spring Training

Job management

Job Submission Basics

- **Do not run your computational programs/applications/codes on the login nodes - even for simple tests.**

- **These login nodes are meant for "login". Besides login, you can use them for file editing, simple data analysis, and other tasks that use minimal computational resources.**
  - Your tests on the login node will adversely impact all the other tasks and login activities of the other users who are logged onto the same node.



Job Submission Basics

1. **Find appropriate queue**
    - Understand the queuing system and your requirements and
    - Queue Querying

2. **Submit job**
    - PBS job scheduler
3. **Monitor jobs during execution**

Job Queues

- **Various queues are available to access the compute nodes. Queue access is through ACL (Linux group-based)**
- **Each job queue differs in**
  - Number of available nodes
  - Max run time
  - Max running jobs per user
  - Nodes may have special characteristics: GPU, large memory, etc.
- **It is recommended to specify resource requirements**
  - Nodes, time, queue and allocation
- **It is called a queue for a reason, but jobs don't run on a "First Come First Served" policy.**
  - This will be detailed in later slides

Queues on TSCC

	Condo	Hotel	Gpu-condo	Gpu-hotel	Glean	Pdafm	Home	Home-xyz
Condo	x	x	*	x	x	x	x	x
Hotel		x		x		x		

* PI needs to purchase gpu nodes in order to use other gpu-condo nodes

Queues on TSCC - Characteristics

- All users have access to

Queue	Max Time/ Job	Max Jobs submitted/User	Max Processor/User
hotel	168:00:0	1500	160
gpu-hotel	168:00:0	-	-
pdafm	168:00:0	50	60

If you are condo participant,

Queue	Max Time/ Job	Max Jobs submitted/User	Max Processor/User
condo	08:00:0	1500	1024
gpu-condo	08:00:0	-	84
glean	08:00:0	500	1024
home	-	-	-

Queue Characteristics

- "qstat -q" will give you more info on the queues

```
 [ychen64@tscc-login1 ~]$ qstat -q

server: tscc-mgr7.local

Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
home-farley        --      --       --      --    0   0 --   E R
home-gibbs         --      --       --      --   38  74 --   E R
home-jogleeson     --      --       --      --   11   9 --   E R
condo              --      --    08:00:00   --   17   2 --   E R
home-yeo           --      --       --      --   56  26 --   E R
home-bafna         --      --       --      --    0   0 --   E R
home-sbpd          --      --       --      --    0   0 --   E R
home-jsebat        --      --       --      --    0   0 --   E R
home-sio           --      --       --      --    3  41 --   E R
home-acms          --      --       --      --    0   0 --   E R
glean              --      --    08:00:00   --  105 508 --   E R
home-leschziner    --      --       --      --    0   0 --   E R
```

Queue Querying – Linux Clusters

- Displays information about active, eligible, blocked, and/or recently completed jobs: `showq` command

```
 $showq

active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME
94                    hocks    Running     8    00:09:53  Fri Apr  3 13:40:43
1 active job                  8 of 16 processors in use by local jobs (50.00%)
                               8 of 8 nodes active      (100.00%)

eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT              QUEUETIME
95                    hocks       Idle     8    00:10:00  Fri Apr  3  13:40:04
96                    hocks       Idle     8    00:10:00  Fri Apr  3  13:40:05
2 eligible jobs

blocked jobs-----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT             QUEUETIME
0 blocked jobs
Total jobs:  3
```

Queue Querying – Free Nodes

- Query free nodes: `lsjobs` command
  - Please add “-node” as suffix:

```
[ychen64@tscc-login1 ~]$ lsjobs --property=gpu-hotel-node
tscc-gpu-10-1: 28155620/leeh7/21:30:52/gpu-hotelx8
28178214/nbnarasi/07:02:37/gpu-hotelx2 DOWNx6
tscc-gpu-5-8: FREEx12
tscc-gpu-7-7: DOWNx12
tscc-gpu-7-8: DOWNx12
tscc-gpu-99-99: DOWNx16
```

Two Job Types

- **On TSCC, all jobs are submitted via TORQUE resource manager (PBS) with Maui scheduler.**
- **Interactive job**
  - Set up an interactive environment on compute nodes for users
    - Advantage: can run programs interactively
    - Disadvantage: must be present when the job starts
  - Purpose: testing, debugging or using GUI
    - Try not to run interactive jobs with large core count, which is a waste of resources
- **Batch job**
  - Executed without user intervention using a job script
    - Advantage: the system takes care of everything
    - Disadvantage: can only execute one sequence of commands which cannot changed after submission
  - Purpose: production run

Submitting Interactive Jobs on TSCC

- **Interactive job example:**
  - PBS for TSCC

```
qsub -I \
     -l walltime=<hh:mm:ss>,nodes=<num_nodes>:ppn=<num_cores> \
     -A <Allocation> \
     -q <queue name> \
      -X to enable X11 forwarding (optional)
```

Submit PBS Interactive Jobs on TSCC

```
[ychen64@tscc-login1 ~]$ qsub -I -X -l walltime=00:30:00,nodes=1:ppn=16 -q condo -A builder-group
qsub: waiting for job 28178208.tscc-mgr7.local to start
qsub: job 28178208.tscc-mgr7.local ready

[ychen64@tscc-1-22 ~]$
```

Interactive job

Enable X11 forwarding to use GUI (optional)

30 minutes walltime

1 node

16 cores per node

submit to condo queue

Allocation name

Submit PBS Interactive Jobs on TSCC

```
[ychen64@tscc-login1 ~]$ qsub -I -X -l walltime=00:30:00,nodes=1:ppn=16 -q condo -A builder-group
qsub: waiting for job 28178208.tscc-mgr7.local to start
qsub: job 28178208.tscc-mgr7.local ready

[ychen64@tscc-1-22 ~]$
```

❖ ***Note the hostname change.***

Node Properties

- **To guarantee that the job will use some node properties, e.g.: InfiniBand interconnect ib, specify all nodes on one switch or group**
    - #PBS –l nodes=1:ppn=16:ib
    - Use information obtained from pbsnodes (showing in the next slide)

Find out Node Properties

- **Check properties of all of the nodes using pbsnodes -a | less**
- **Check properties of a certain node using pbsnodes <node_name>**

```
[ychen64@tscc-login12 orca]$ pbsnodes tscc-10-12
tscc-10-12
     state = free
     np = 36
     properties = davidson-node,cascade,noib,mem192,rack10,condo-node,glean-node
     ntype = cluster
     jobs = 0/28243157.tscc-mgr7.local,1/28243157.tscc-mgr7.local,2/28243157.tscc-mgr7.local,3/28243157.tscc-
mgr7.local,4/28243157.tscc-mgr7.local,5/28243157.tscc-mgr7.local,6/28243157.tscc-mgr7.local,7/28243157.tscc-
mgr7.local,8/28243157.tscc-mgr7.local,9/28243157.tscc-mgr7.local,10/28243157.tscc-
mgr7.local,11/28243157.tscc-mgr7.local,12/28243157.tscc-mgr7.local,13/28243157.tscc-
mgr7.local,14/28243157.tscc-mgr7.local,15/28243157.tscc-mgr7.local
     status = rectime=1645742890,varattr=,jobs=28243157.tscc-
mgr7.local,state=free,netload=11175197371258,gres=,loadave=1.01,ncpus=36,physmem=196296216kb,availmem
=176095452kb,totmem=198393364kb,idletime=4466,nusers=1,nsessions=1,sessions=163902,uname=Linux tscc-
10-12.sdsc.edu 3.10.0-1160.45.1.el7.x86_64 #1 SMP Wed Oct 13 17:20:51 UTC 2021 x86_64,opsys=linux
     mom_service_port = 15002
     mom_manager_port = 15003
```

Homework Exercise

- **Start an interactive job session for 1 hour (or a time with 30-min increment)**
  - Find out your allocation name with gbalance if you don't remember
  - Decide how many node and which queue to use
  - Use "`qsub -I`" to submit including all necessary options
  - Once job started, how to verify that you are NOT on the login node?

- ***Why 30 min for the interactive jobs?***
  - *The requested time for the interactive jobs should have 30-min increment*
  - *A 30-min job is the easiest job to be fit into the job queue.*
  - *Based on the actual test needs, longer time can be requested.*

Homework Exercise contd..

- **When your interactive job starts, try the following below:**
- **Computing an approximate value for PI**
  - cd to your /scratch directory
    ```
    $ cd /oasis/tscc/scratch/$USER
    ```
  - Copy the tarball from HPC website to your /scratch directory
    ```
    $ cp /projects/builder-group/tscc101/pi.tar.gz .
    ```
  - Untar it
    ```
    $  tar –xvzf pi.tar.gz
    ```
  - cd to the directory “pi”
    ```
    $ cd pi
    ```
  - Use “module list” to make sure the openmpi_ib/3.1.4 is loaded.
  - Execute serial or mpi version
    ```
    #serial version, if no argument given, default value 1000000000
    $ ./serialpi.out
    # MPI version:
    $ mpirun -np 16 ./mpi_pi.out 100000000000  # default 1000000000000
    ```

Submit a Batch Job

- **PBS batch Job example:**

```
[ychen64@tscc-login1 pi]$ qsub qsub.submit
```

- ***Carefully prepare the PBS job script***
  - *examples in the next few slides*

PBS Job Script – Parallel Job

```
#!/bin/bash
#PBS -l nodes=2:ppn=16            #Number of nodes and processors per node
#PBS -l walltime=24:00:00         #Maximum wall time
#PBS -N myjob                     #Job name
#PBS -o <file name>               #File name for standard output
#PBS -e <file name>               #File name for standard error
#PBS -j oe                        # Combine stdout and stderr (optional)
#PBS -q hotel                     #Queue name
#PBS -A <allocation_if_needed>    #Allocation name
#PBS -m e                         #Send mail when job ends
#PBS -M <email address>           #Send mail to this address

<shell commands>
mpirun -machinefile $PBS_NODEFILE -np 32 <path_to_executable> <options>
<shell commands>
```

Tells the scheduler how much resource you need.

How will you use the resources?

True or False?

- **I have the below job script on TSCC, since I used nodes=2:ppn=20, my script will run in parallel using 2 nodes with 40 cores.**
  a) True
  b) False

```
#!/bin/bash
#PBS -l nodes=2:ppn=20
#PBS -l walltime=24:00:00
#PBS -N myjob
#PBS -j oe
#PBS -q condo
#PBS -A my_allocation

./my_executable.out
```

Job Monitoring - PBS

- Check details on your job using `qstat`

```
$ qstat -n -u $USER  : For quick look at nodes assigned to you
[ychen64@tscc-login11 ~]$ qstat -nu ychen64

tscc-mgr7.local:
                                                                                  Req'd     Req'd       Elap
Job ID                   Username    Queue    Jobname          SessID  NDS   TSK    Memory    Time     S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
28308989.tscc-mgr7.loc   ychen64     condo    STDIN             26056     1      8     --    06:00:00 R  01:28:39
   tscc-3-0/16+tscc-3-0/17+tscc-3-0/18+tscc-3-0/19+tscc-3-0/20+tscc-3-0/21
   +tscc-3-0/22+tscc-3-0/23
```

- Delete job using `qdel`

```
$ qdel <jobid>
```

- Check details of your job using `checkjob`

```
$ checkjob <jobid>
```

Job Monitoring - PBS

- Diagnose why submitted jobs remain queued **yqd**

```
$ yqd --user=mac157
27461468 (mac157 home-visres 1x5 ['gpuK80'] 24:43:57): 0 nodes free

$ yqd --user=tbaddour
28274922 (tbaddour hotel 1x8 [] 54:00:23): unknown account aguado-lab

$ yqd --user=yuw044
28168502 (yuw044 hotel 2x4 [] 51:41:18): insufficient SU balance -740;
384 needed

$ yqd --user=rlarsen
28309803 (rlarsen hotel 1x16 [] 0:13:57): max of 15 jobs to be run ahead
of this job or job has a reservation
```

❖ **Please pay close attention to the CPU load and the memory consumed by your job!**

Using the “top” Command

- The Linux `top` program provides a dynamic real-time view of a running system.
- **Should be used on the compute node assigned to you (ssh to it first)**

```
top - 19:39:56 up 89 days,  4:13,  1 user,  load average: 0.63, 0.18, 0.06
Tasks: 489 total,   2 running, 487 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.3%us,  0.0%sy,  0.0%ni, 93.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  65909356k total,  3389616k used, 62519740k free,   151460k buffers
Swap: 207618040k total,     5608k used, 207612432k free,   947716k cached

   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 39595 fchen14   20   0  266m 257m  592 R 99.9  0.4   0:06.94 a.out
 39589 fchen14   20   0 17376 1612  980 R  0.3  0.0   0:00.05 top
 38479 fchen14   20   0  108m 2156 1348 S  0.0  0.0   0:00.03 bash
 39253 fchen14   20   0  103m 1340 1076 S  0.0  0.0   0:00.00 236297.mike3.SC
 39254 fchen14   20   0  103m 1324 1060 S  0.0  0.0   0:00.00 bm_laplace.sh
 39264 fchen14   20   0 99836 1908  992 S  0.0  0.0   0:00.00 sshd
 39265 fchen14   20   0  108m 3056 1496 S  0.0  0.0   0:00.03 bash
```

Using the "free" Command

- The Linux `free` displays the total amount of free and used physical and swap memory in the system
- Should be used on the compute node assigned to you (ssh to it first)

```
[ychen64@tscc-4-59 ~]$ free -h
              total        used        free      shared  buff/cache   available
Mem:           125G        6.1G        112G        200M        7.0G        118G
Swap:          2.0G        1.9G        144M
```

PBS Environment Variables

$PBS_ENVIRONMENT	$PBS_MOMPORT	$PBS_NUM_PPN	$PBS_O_MAIL
$PBS_QUEUE	$PBS_WALLTIME	$PBS_GPUFILE	**$PBS_NODEFILE**
$PBS_O_HOME	$PBS_O_PATH	$PBS_SERVER	$PBS_JOBCOOKIE
$PBS_NODENUM	$PBS_O_HOST	$PBS_O_QUEUE	$PBS_TASKNUM
$PBS_JOBID	$PBS_NP	$PBS_O_LANG	$PBS_O_SHELL
$PBS_VERSION	$PBS_JOBNAME	$PBS_NUM_NODES	$PBS_O_LOGNAME
$PBS_O_WORKDIR	$PBS_VNODENUM		

Homework Exercise

- **Submit a batch job**
  - cd to the directory “pi”

    ```
    $ cd pi
    ```
  - edit qsub.submit (change allocation name, email etc.)

    ```
    $ vi qsub.submit
    ```
  - `submit job`

    ```
    $ qsub qsub.submit
    ```
- **Check details on your job using `qstat`**

  ```
  $ qstat -n -u $USER   #PBS
  ```

- ***Monitor the job***
  - ***`top`*** *(must ssh to the compute node assigned to your job)*
  - ***`free`*** *(must ssh to the compute node assigned to your job)*

GPU Usage

- **The GPUs on a GPU node are allocated proportionally to the total number of cores present.**
  - To use 1 gpu on a GPU node with 12 cores and 4 GPUs, use a command similar to

    # qsub -I -q gpu-hotel -l nodes=1:ppn=3

- **Allocated GPUs are referenced by the CUDA_VISIBLE_DEVICES environment variable. Applications using the CUDA libraries will discover GPU allocations through that variable. Users do not need to set CUDA_VISIBLE_DEVICES variable.**

More Things to be Noticed

- The purpose of pdafm queue is for jobs costing big memory not for jobs using more number of cores.
- If not your home queue, GPU is available in gpu-condo or gpu-hotel queues (need to purchase gpu node to use gpu-condo)
- **Why is my job not get accelerated when running on cluster?**
  - Is your job utilizing the parallel resource on the cluster?
  - Does you job have lots of I/O tasks?

TSCC 101 Spring Training

Understanding job scheduling

SDSC SAN DIEGO SUPERCOMPUTER CENTER

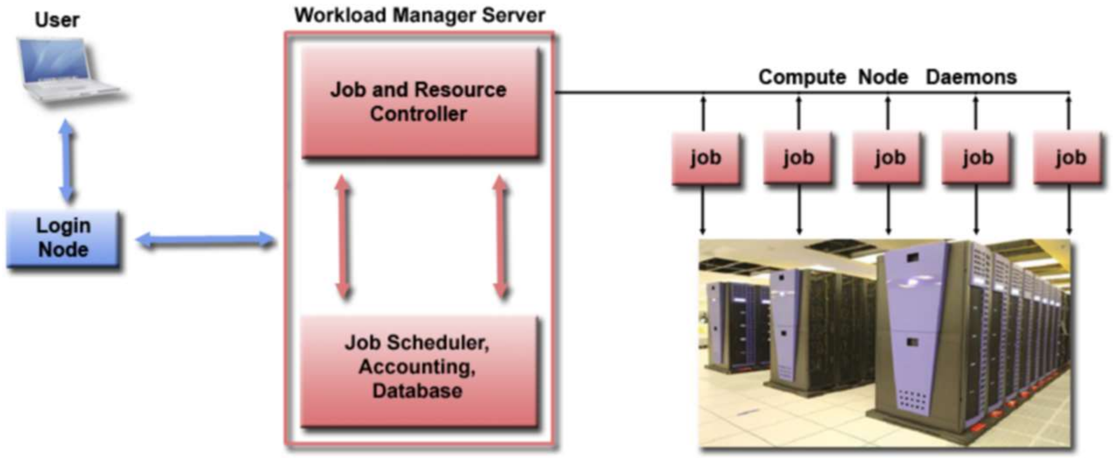
UNIVERSITY OF CALIFORNIA

Frequently Asked Questions

- I submitted job A before job B. Why job B started earlier than job A?
- There are free nodes available, and I am sure I have enough SUs for the job, but why my job is still waiting and not running?

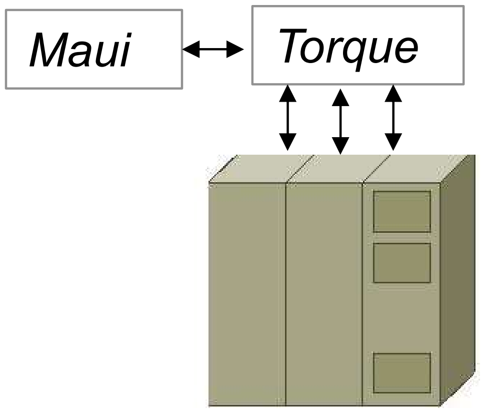
Back to Cluster Architecture

- **As a user, you interact with the scheduler and/or resource manager whenever you submit a job, or query on the status of your jobs or the whole cluster or seek to manage your jobs.**
- **Resource managers give access to compute resource**
  - Takes in a resource request (job) on login node
  - Finds appropriate resource and assigns you a priority number
  - Positions your job in a queue based on the priority assigned.
  - Starts running jobs until it cannot run more jobs with what is available.



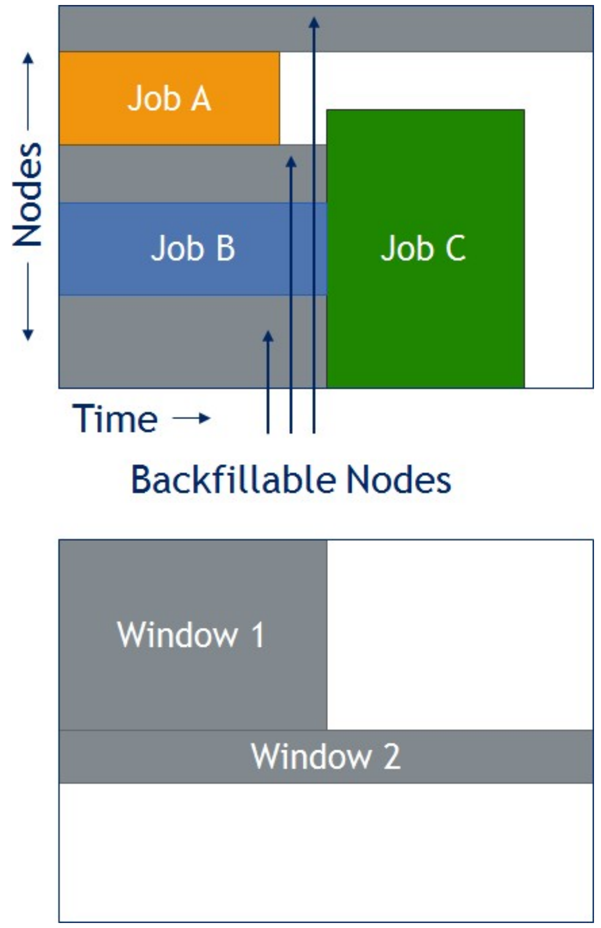
Job Scheduler

- **TSCC Linux clusters use TORQUE, an open source version of the Portable Batch System (PBS) together with the Maui Scheduler, to manage user jobs.**
- **Resource Manager - Torque**
  - Manages a queue of jobs for a cluster of resources
  - Launches job to a simple FIFO job queue
- **Workload Manager - Maui**
  - A scheduler that integrates with one or more Resource Managers to schedule jobs across domains of resources (servers, storage, applications)
  - Prioritizes jobs
  - Provides status of running and queued jobs, etc.
- **The batch queuing system determines**
  - The order jobs are executed
  - On which node(s) jobs are executed



Job Management Philosophy

- **Working Philosophy**
  - Prioritize workload into a queue for jobs
  - ***Backfill*** idle nodes to maximize utilization
    - Will be detailed later...



Job Priorities

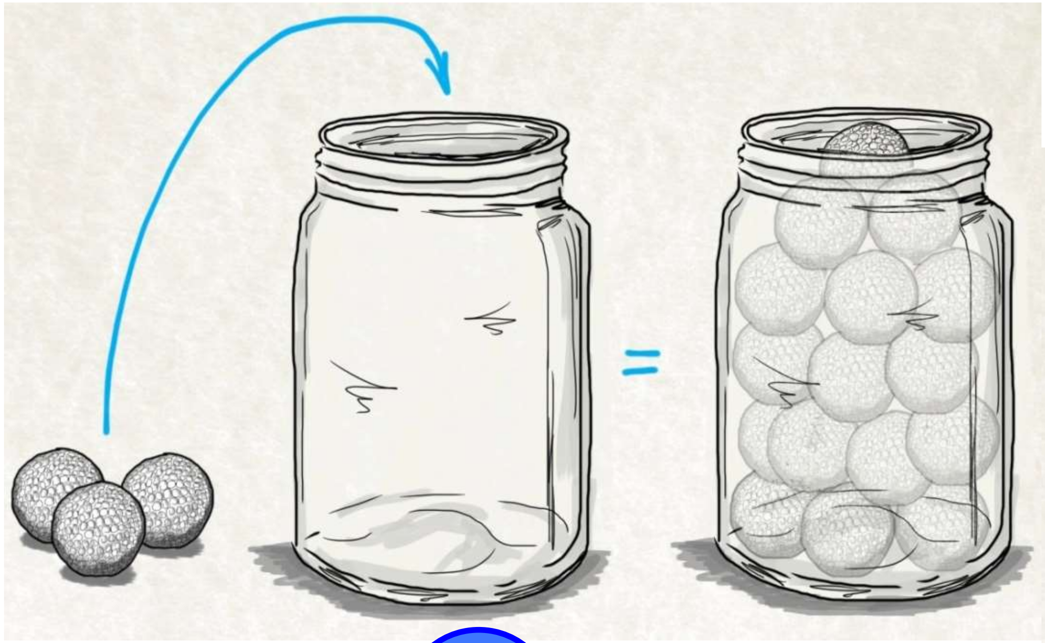
- **Jobs with a higher job priority are scheduled ahead of jobs with a lower priority.**
- **Job priorities have contributions from the following:**
  - credential priority
  - fairshare priority
  - service priority
  - resource priority

How to Get Higher Priority?

- **Do not submit too many jobs within one week.**
- **Submit your job early to accumulate the queue time.**
- **More on resource priority**:
  - Request proper number of compute nodes.
  - Request a smaller walltime limit.
  - see next few slides...

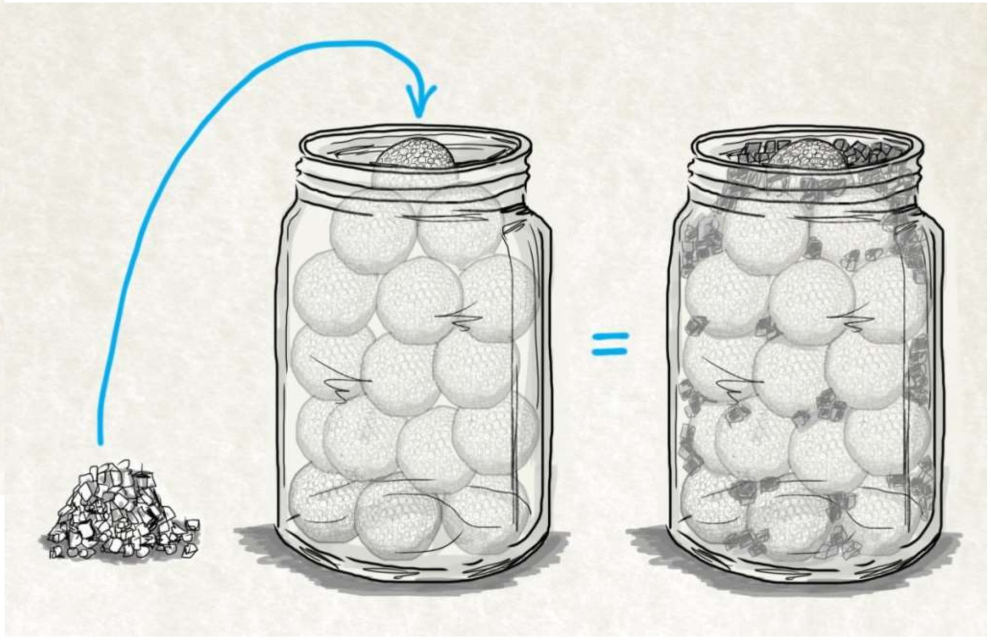
How to Maximize the Usage of a Cluster?

- Fill in high-priority (large) jobs
- Backfill low-priority (small) jobs



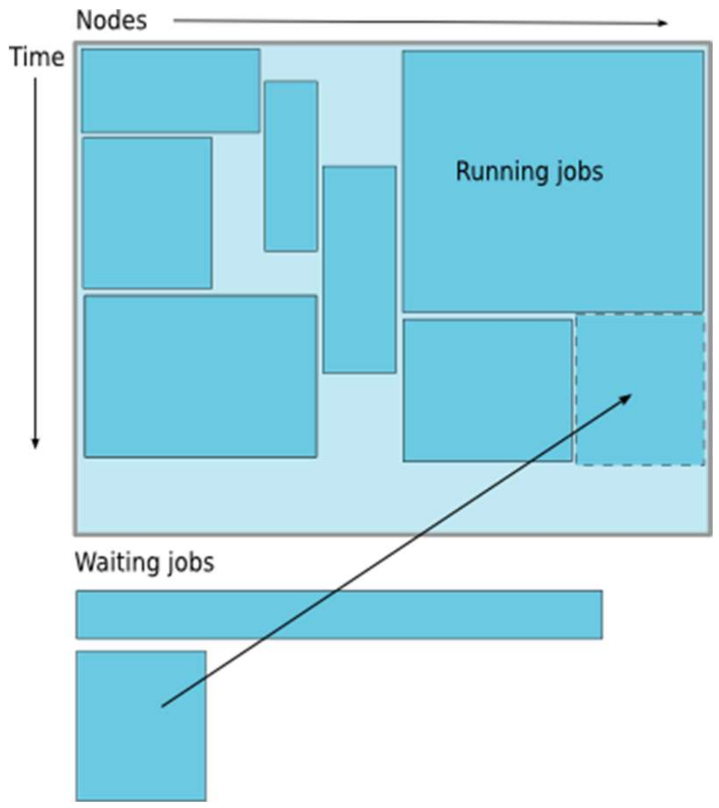
1

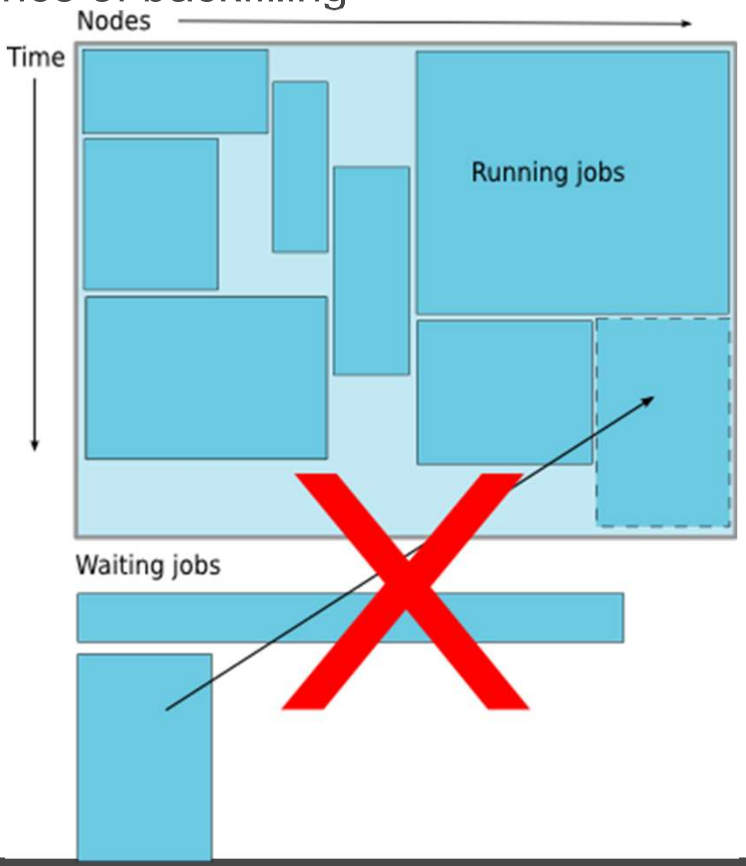
2



How Much Time Should I Ask for?

- **It should be**
  - Long enough for your job to complete
  - As short as possible to increase the chance of backfilling





Take-home message

- **Condo and Hotel on TSCC**
- **Account and allocation**
- **Infrastructure**
- **Practice on the cluster**
  - How to login via SSH
  - How to check your allocation balance
  - How to transfer files, add software by Modules
- **Find appropriate queue**
  - Nodes are organized into queues. Nodes can be shared.
  - Nodes may have special characteristics: GPU, Large memory, etc
- **Submit job**
  - interactively, use “-I” option in the qsub command
  - in batch with a carefully-prepared PBS script
- **Monitor core and memory usage with top and free commands**
- **Job schedule basics**

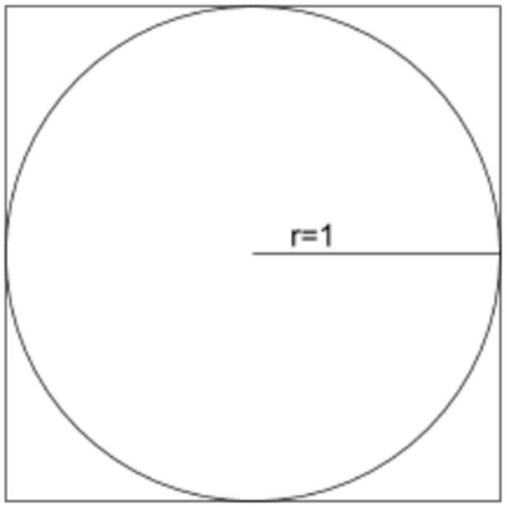
TSCC@SDSC User Services

- **Hardware resources**
  - Currently manages ~ 400 nodes
- **Software stack**
  - Communication software
  - Programming support: compilers and libraries
  - Application software
- **Contact user services**
  - Email Help Ticket: ***tscc-support@ucsd.edu***
- **TSCC website:**

https://www.sdsc.edu/services/hpc/tscc/index.html

Appendix
Computing an approximate value for PI

- The executables in this training calculate the value for PI based on the math which is actually quite simple: Imagine a square dartboard with circle inscribed within it such that the diameter of the circle is the length of a side of the square.

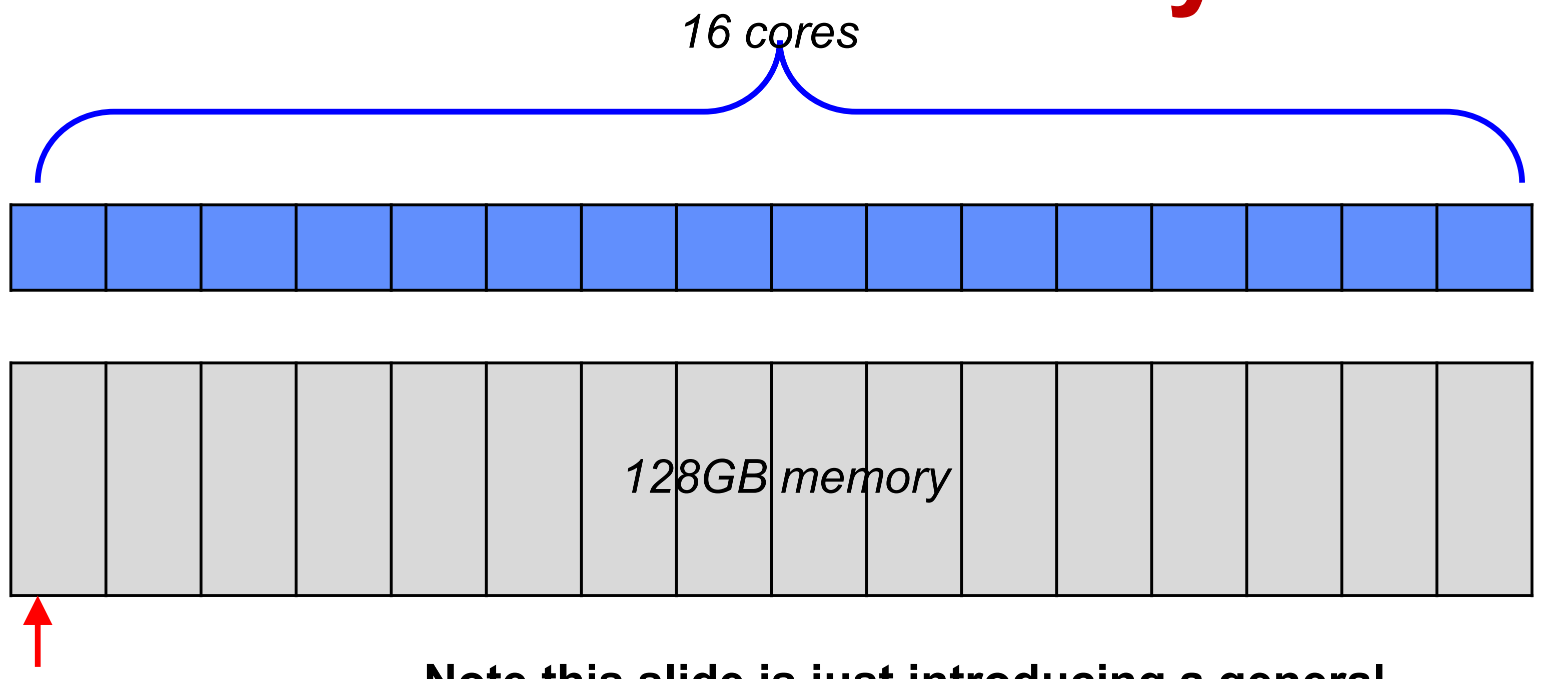


- We can observe that the ratio of the area of the circle to the area of the square is equal to some constant, π/4 (since the square's area is 2*2 = 4 and area_circle = π*r^2 = π). If we randomly place many points (darts) inside the square, we can count how many are also inside the circle (satisfy x^2+y^2 <= 1) vs the total number of points and compute an estimate for the value of π. (Problem description is from Jared Baker, UW; Ben Matthews, NCAR)

Appendix
Pay Attention to Memory Usage

- Note this slide is just introducing a general rule of memory allocation on PBS system, which hasn't been verified on TSCC
- Pay attention to memory used for jobs that will only execute on a part of node, i.e. `nodes=1:ppn=1/2/4/6/8`…
- The allocated amount of memory is proportional to the number of cpu cores asked by a job (explain in detail in the next slide).
- If applications require more memory, scale the number of cores (ppn) to the amount of memory required.

Appendix
Core and Memory



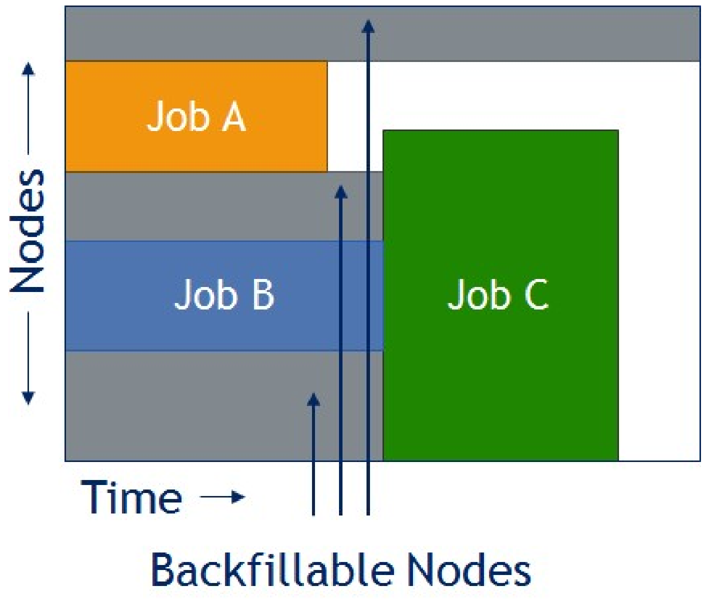
Note this slide is just introducing a general rule of memory allocation on PBS system, which hasn't been verified on TSCC

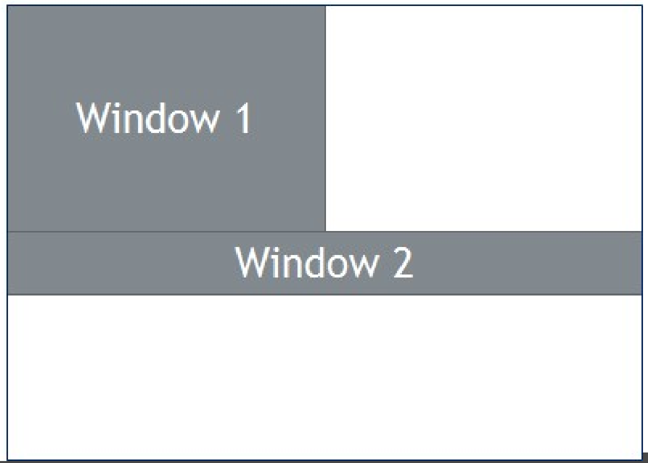
Question:

On TSCC, if my serial job needs 12GB memory, what ppn value should I use?

Appendix
An Overview of Backfilling (1)

- Backfill is a scheduling optimization that allows a scheduler to make better use of available resources by running jobs out of order.
- Enabling backfill allows the scheduler to start other, lower-priority jobs so long as they do not delay the highest priority job.
- **If the FIRSTFIT algorithm is applied, the following steps are taken:**
  - The list of feasible backfill jobs is filtered, selecting only those that will actually fit in the current backfill window.
  - The first job is started.
  - While backfill jobs and idle resources remain, repeat step 1.





Appendix
An Overview of Backfilling (2)

- **Although by default the start time of the highest priority job is protected by a reservation, there is nothing to prevent the third priority job from starting early and possibly delaying the start of the second priority job.**
- **Command to show current backfill windows:**
  - `showbf`
    - Shows what resources are available for immediate use.
    - This command can be used by any user to find out how many processors are available for immediate use on the system. It is anticipated that users will use this information to submit jobs that meet these criteria and thus obtain quick job turnaround times.
  - Example:

```
[ychen64@tscc-login11 ~]$ showbf
Partition      Tasks  Nodes      Duration   StartOffset       StartDate
---------      -----  -----  ------------  ------------  --------------
ALL               40      5      18:50:35      00:00:00  11:16:49_09/04
ALL                8      1      INFINITY      00:00:00  11:16:49_09/04
```