



# Introduction to Using the HPRC Clusters

## Ada & Terra



Spring 2019

# Outline

- Usage Policies
- Hardware Overview of Ada and Terra
- Accessing Ada and Terra
- File Transfers
- File systems and User Directories
- Computing Environment
  - short break
- Development Environment
- Batch Processing
- Common Problems
- Need Help?

# Usage Policies

## (Be a good compute citizen)

- It is illegal to share computer passwords and accounts by state law and university regulation
- It is prohibited to use Ada in any manner that violates the United States export control laws and regulations, EAR & ITAR
- Abide by the expressed or implied restrictions in using commercial software

<https://hprc.tamu.edu/policies>

# Introduction

- Prerequisites:

- Basic knowledge of UNIX commands
- Slides from our UNIX short course are at:

[https://hprc.tamu.edu/training/intro\\_linux.html](https://hprc.tamu.edu/training/intro_linux.html)

- Examples:

- For **Ada**:

- Available in /scratch/training/Intro-to-ada directory
- Copy these files to your scratch directory

- ```
cp -r /scratch/training/Intro-to-ada/2019/Spring/ $SCRATCH
```

- For **Terra**:

- Available in /scratch/training/Intro-to-terra directory
- Copy these files to your scratch directory

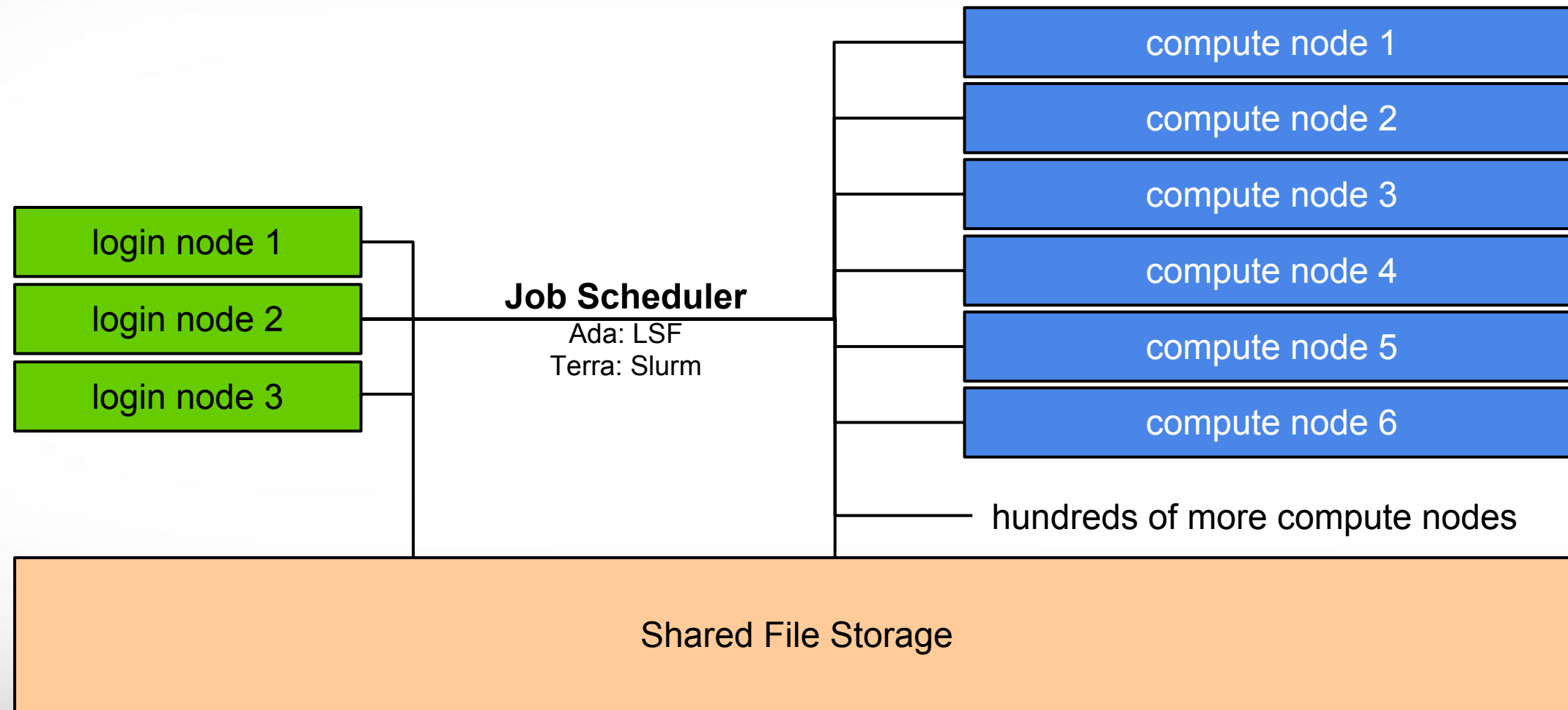
- ```
cp -r /scratch/training/Intro-to-terra/2019/Spring/ $SCRATCH
```

# Laptops/Workstations vs HPC Clusters

	Laptops / Workstations	HPC Clusters
<b>CPU Clock Rate</b>	1.2 ~ 4 GHz	2.4 GHz
<b>CPU Core Counts</b>	4 ~ 12	20 ~ 40 / node
<b>Memory Size</b>	up to 10s GB	64GB ~ 2 TB / node
<b>Node Counts</b>	1 or a few	300 ~ 900
<b>Storage Size</b>	usually up to a few TB	a few PB (1 PB= 1000 TB)
<b>Network or Interconnect Speed</b>	usually $\leq$ 1 Gbps	40 ~ 100 Gbps
<b>User Interface</b>	GUI and Text	mostly Text; some GUI
<b>Running jobs</b>	Exclusive	Batch processing

- When HPC clusters are more suitable
  - Problems are too big to fit in one laptop or workstation, due to limitation on memory, core count, or node count
  - Problems scale well with more CPU cores or memory
  - Single threaded problems with millions of permutations
  - Problems require large high speed storage and/or interconnect

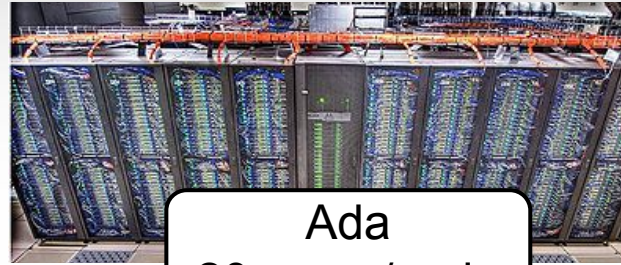
# HPC Diagram



# HPRC Clusters



Terra  
28 cores/node



Ada  
20 cores/node



Curie  
16 cores/node

<b>Nodes</b>	307	860	72
<b>Cores</b>	8,512	17,500	1,152
<b>CPU Architecture</b>	x86_64 Intel 14-core 2.4GHz <i>Broadwell</i>	x86_64 Intel 10-core 2.5 GHz <i>IvyBridge</i>	ppc64 IBM 8-core 4.2 GHz <i>Power7+</i>
<b>Interconnect</b>	Onmi-Path	FDR-10 Infiniband	10 Gbps Ethernet
<b>Accelerator</b>	48 nodes with Tesla K80	30 nodes with 2 Tesla K20 9 nodes with 2 <i>Phi</i> coprocessors	N/A
<b>Job Scheduler</b>	Slurm	LSF	LSF (shared with Ada)
<b>File System</b>	GPFS; 3 PB raw	GPFS; 4 PB raw	GPFS (shared with Ada)
<b>Production Date</b>	Spring 2017	Sep 2014	May 2015

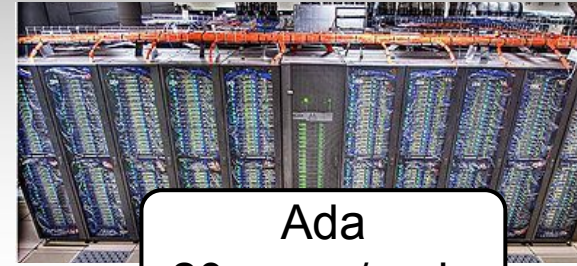
No SU charged on Curie

<https://hprc.tamu.edu/resources>

# HPRC Clusters



Terra  
28 cores/node



Ada  
20 cores/node



Curie  
32 cores/node

<b>Login Nodes</b>	3	8	2
<b>64 GB memory compute nodes</b>	256	792	-
<b>PHI 64 GB memory compute nodes</b>	-	9	-
<b>KNL (Knights Landing) (no SUs charged)</b>	16	-	-
<b>K20 GPU 64 GB memory compute nodes</b>	-	10	-
<b>K80 GPU 128 GB memory compute nodes</b>	48	-	-
<b>K20 GPU 256 GB memory compute nodes</b>	-	20	-
<b>256 GB memory compute nodes</b>	-	6	70
<b>1 TB memory compute nodes</b>	-	11 (40 cores/node)	-
<b>2 TB memory compute nodes</b>	-	4 (40 cores/node)	-

<https://hprc.tamu.edu/resources>



# Accessing Ada and Terra

- SSH command is required for accessing Ada / Terra:
    - On campus: `ssh NetID@ada.tamu.edu` or `ssh NetID@terra.tamu.edu`
    - Off campus:
      - Set up and start VPN (Virtual Private Network): <http://u.tamu.edu/VPnetwork>
      - Then: `ssh NetID@ada.tamu.edu` or `ssh NetID@terra.tamu.edu`
  - SSH programs for Windows:
    - MobaXTerm (preferred, includes SSH and X11)
    - PuTTY SSH
  - Access through <https://portal.hprc.tamu.edu> (Menu “Clusters” => “Ada Shell Access”)
  - Ada has 8 login nodes. Terra has 3 login nodes Check the bash prompt.
    - `[NetID@ada1 ~]$` `[NetID@terra3 ~]$`
  - Login sessions that are idle for **60** minutes will be closed automatically
  - Processes run longer than **60** minutes on login nodes will be killed automatically.
  - **Do not use more than 8 cores on the login nodes!**
  - **Do not use the sudo command.** Contact us for assistance installing software.
- <https://hprc.tamu.edu/wiki/HPRC:Access>

**Terra portal coming soon**

# File Transfers with **Ada** and **Terra**

- Simple File Transfers:
  - scp: command line (Linux, MacOS)
  - rsync: command line (Linux, MacOS); *can resume transfer*
  - MobaXterm: GUI (Windows)
  - WinSCP: GUI (Windows)
  - FileZilla: GUI (Windows, MacOS, Linux) (use sftp protocol)
  - Portal: <https://portal.hprc.tamu.edu> (Ada only; through menu “Files”; portal for Terra coming soon)
- Bulk data transfers:
  - Use fast transfer nodes
    - data transfer processes will not timeout at 60 minutes
    - on **Ada**: `ada-ftn1.tamu.edu` OR `ada-ftn2.tamu.edu`
    - on **Terra**: `terra-ftn.hprc.tamu.edu`
    - Globus Connect (<https://hprc.tamu.edu/wiki/SW:GlobusConnect>)
    - GridFTP

<https://hprc.tamu.edu/wiki/HPRC:FileTransfers>

# File Systems and User Directories

Directory	Environment Variable	Space Limit	File Limit	Intended Use
/home/\$USER	\$HOME	10 GB	10,000	Small to modest amounts of processing.
/scratch/user/\$USER	\$SCRATCH	1 TB	50,000	Temporary storage of large files for on-going computations. Not intended to be a long-term storage area.
/tiered/user/\$USER	\$ARCHIVE	10 TB	50,000	Intended to hold valuable data files that are not frequently used ( <b>on Ada/Curie only</b> )

- `$HOME` and `$SCRATCH` directories are not shared between Ada and Terra clusters.
- View usage and quota limits using the command: `showquota`
- Quota and file limit increases will only be considered for scratch and tiered directories
- Request a group directory for sharing files.
- **Do not share your home, scratch, tiered directories.**

[https://hprc.tamu.edu/wiki/Ada:Filesystems\\_and\\_Files](https://hprc.tamu.edu/wiki/Ada:Filesystems_and_Files)  
[https://hprc.tamu.edu/wiki/Terra:Filesystems\\_and\\_Files](https://hprc.tamu.edu/wiki/Terra:Filesystems_and_Files)

# Software

- See the Software wiki page for instructions and examples
  - <https://hprc.tamu.edu/wiki/SW>
  - <https://hprc.tamu.edu/software/ada>
  - <https://hprc.tamu.edu/software/terra>
  - <https://hprc.tamu.edu/wiki/Bioinformatics>
- License-restricted software
  - Contact license owner for approval
- Contact us for software installation help/request
  - User can install software in their home/scratch dir
  - Do not run the “*sudo*” command when installing software

# Computing Environment

- Paths:
  - \$PATH: for commands (eg. /bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/netid/bin)
  - \$LD\_LIBRARY\_PATH: for libraries
  - See your \$PATH variable with the command `echo $PATH`
- There is a lot of software, many versions, and many paths to manage  
..... How do you manage all these software versions?
- The solution (lmod) which uses the command: `module`
- Each version of a software, application, library, etc. is available as a module.
  - Module names have the format:

software\_name / version toolchain [Python-version]  
`TopHat/2.1.1-intel-2017A-Python-2.7.12`

[https://hprc.tamu.edu/wiki/Ada:Computing\\_Environment#Modules](https://hprc.tamu.edu/wiki/Ada:Computing_Environment#Modules)  
[https://hprc.tamu.edu/wiki/Terra:Computing\\_Environment#Modules](https://hprc.tamu.edu/wiki/Terra:Computing_Environment#Modules)

# Application Modules

- Installed applications are available as modules which are available to all users
  - (except for restricted modules)
- It's a good habit to purge unused modules before loading new modules.
- It is recommended to load a specific software version instead of the defaults
- **Avoid loading modules in your `~/ .bashrc`**

```
module avail
```

```
# list all available modules (sometimes it is very slow)
# space bar down, page up/down, q to quit
# / for case sensitive search (similar to a UNIX man page)
```

```
module spider boost
```

```
# case insensitive search for modules with 'boost' in name
```

```
module keyword graphics
```

```
# search module descriptions for keyword 'graphics'
# some graphics modules may be missed if
# keyword is not found in description (case insensitive)
```

[https://hprc.tamu.edu/wiki/Ada:Computing\\_Environment#Modules](https://hprc.tamu.edu/wiki/Ada:Computing_Environment#Modules)  
[https://hprc.tamu.edu/wiki/Terra:Computing\\_Environment#Modules](https://hprc.tamu.edu/wiki/Terra:Computing_Environment#Modules)

# The Case (in)sensitive spider command

The following commands will give you different results on **Ada** because the `module spider` command is not case sensitive unless it finds an exact match for the search term

```
module spider python
```

```
module spider Python
```

```
module spider python
```

python:

Description:

Python is a programming language that lets you work more quickly and integrate your systems more effectively. - Homepage: <http://python.org/>

Versions:

python/2.7.6-generic  
python/2.7.6-ictce-7.1.2  
python/2.7.10-intel-2015B.badSSL  
python/2.7.10-intel-2015B  
python/2.7.13-generic

**Other possible modules matches:**

Biopython IPython MySQL-python NGS-Python **Python** ScientificPython bx-python findpython myPython netcdf4-python ...

look for other possible modules

[https://hprc.tamu.edu/wiki/Ada:Computing\\_Environment#Modules](https://hprc.tamu.edu/wiki/Ada:Computing_Environment#Modules)  
[https://hprc.tamu.edu/wiki/Terra:Computing\\_Environment#Modules](https://hprc.tamu.edu/wiki/Terra:Computing_Environment#Modules)

# Module Loading Exercise

1. `module spider trinity` # search for available module names matching trinity  
# not case sensitive unless an exact match is found
2. `module load Trinity/2.5.1-GCCcore-6.3.0-Perl-5.24.0` # load specific module version  
# type Trinity/2.5 then hit tab key
3. `module list` # list all loaded modules
4. `module spider Bowtie2` # see which versions of Bowtie2 are available
5. `module swap Bowtie2 Bowtie2/2.3.4-GCCcore-6.3.0` # change the version of a loaded module
6. `module list` # list all loaded modules
7. `module purge` # remove all loaded modules



# Development Environment - Toolchains

- Intel toolchain (eg. software stack) is recommended
    - Intel C/C++/Fortran compilers (icc, icpc, ifort)
    - Intel Math Kernel Library
    - Intel MPI library
  - For packages that require MPI but not MKL or BLAS/FFTW/LAPACK
    - iimpi/2017A                      iompi/2017A                      gompi/2017A
  - Toolchains that contain MPI, MKL, and BLAS/FFTW/LAPACK
    - intel/2017A                      iomkl/2017A                      foss/2017A
  - To load/use the current recommended Intel toolchain module
- If you do not want to use GCC version in the intel/2017A toolchain, find available gcc versions for applications which must use gcc/g++

```
module load intel/2017A
```

```
module spider GCC
```

<https://hprc.tamu.edu/wiki/SW:Toolchains>

[https://hprc.tamu.edu/wiki/Ada:Compile:All#Getting\\_Started](https://hprc.tamu.edu/wiki/Ada:Compile:All#Getting_Started)

[https://hprc.tamu.edu/wiki/Terra:Compile:All#Getting\\_Started](https://hprc.tamu.edu/wiki/Terra:Compile:All#Getting_Started)

# Modules and Toolchains

- Load modules with the same toolchains in your job scripts
- The `2017A` and `GCCcore-6.3.0` toolchain versions are recommended
  - `intel/2017A`
  - `iomkl/2017A`
  - `foss/2017A`
  - `GCCcore/6.3.0`
- Avoid loading modules in your `.bashrc` and `.bash_profile` files
- Avoid mixing toolchains if loading multiple modules in the same job script

```
module load HISAT2/2.0.4-foss-2016b
module load TopHat/2.1.1-intel-2017A-Python-2.7.12
module load Cufflinks/2.2.1-intel-2015B
```

- Same rule applies to compilers and libraries.

# The GCCcore Toolchain

- To minimize the number of software builds, the GCCcore-6.3.0 toolchain modules can be loaded alone or with any one of the following 2017A toolchains
  - `intel/2017A`
  - `iomkl/2017A`
  - `foss/2017A`
- Example of loading a GCCcore-6.3.0 module with a 2017A module

```
module load Bowtie2/2.3.3.1-GCCcore-6.3.0
module load TopHat/2.1.1-intel-2017A-Python-2.7.12
```

# Python-version-bare modules

- You need to load a non '-bare' Python version along with the -bare module
  - If you do not, then the older default OS Python version will be used
- Used in conjunction with GCCcore-6.3.0 builds in order to reduce the number of software modules built.

intel/2017A

iomkl/2017A

foss/2017A

Three different examples of loading GCCcore-6.3.0-Python-bare and a Python module with a 2017A toolchain

1.

```
module load Cython/0.25.2-GCCcore-6.3.0-Python-2.7.12-bare
module load Python/2.7.12-foss-2017A
```

2.

```
module load Cython/0.25.2-GCCcore-6.3.0-Python-2.7.12-bare
module load Python/2.7.12-iomkl-2017A
```

3.

```
module load Cython/0.25.2-GCCcore-6.3.0-Python-2.7.12-bare
module load HISAT2/2.1.0-intel-2017A-Python-2.7.12
```

Loads Python indirectly

# Consumable Computing Resources

- Resources specified in a job file:
  - Processor cores
  - Memory
  - Wall time
  - GPU
- Service Unit (SU) - Billing Account
  - Use "myproject" to query  
[https://hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](https://hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)

```
myproject
```

```
-----  
List of YourNetID's Project Accounts  
-----
```

Account	FY	Default	Allocation	Used & Pending SUs	Balance	PI
1228000223136	2019	N	10000.00	0.00	10000.00	Doe, John
1428000243716	2019	Y	5000.00	-71.06	4928.94	Doe, Jane

- Other resources:
  - Software license/token
    - Use "license\_status" to query
    - [https://hprc.tamu.edu/wiki/SW:License\\_Checker](https://hprc.tamu.edu/wiki/SW:License_Checker)

Find available license for "ansys":

```
license_status -s ansys
```

```
License status for ANSYS:
```

```
-----  
| License Name | # Issued | # In Use | # Available |  
-----  
| aa_mcad | 50 | 0 | 50 |  
| aa_r | 50 | 32 | 18 |  
| aim_mp1 | 50 | 0 | 50 |  
| ..... | | | |  
-----
```

Find detail options:

```
license_status -h
```

# Ada: Examples of SUs charged based on Job Cores, Time and Memory Requested

A Service Unit (SU) on **Ada** is equivalent to one core or **2500** MB memory usage for one hour.

	Number of Cores	MB of memory per core	Total Memory (GB)	Hours	SUs charged
1.	1	2500	2.5	1	1
2.	1	2600	2.6	1	2
3.	1	50000	50	1	20
4.	20	2500	50	1	20

[https://hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](https://hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)

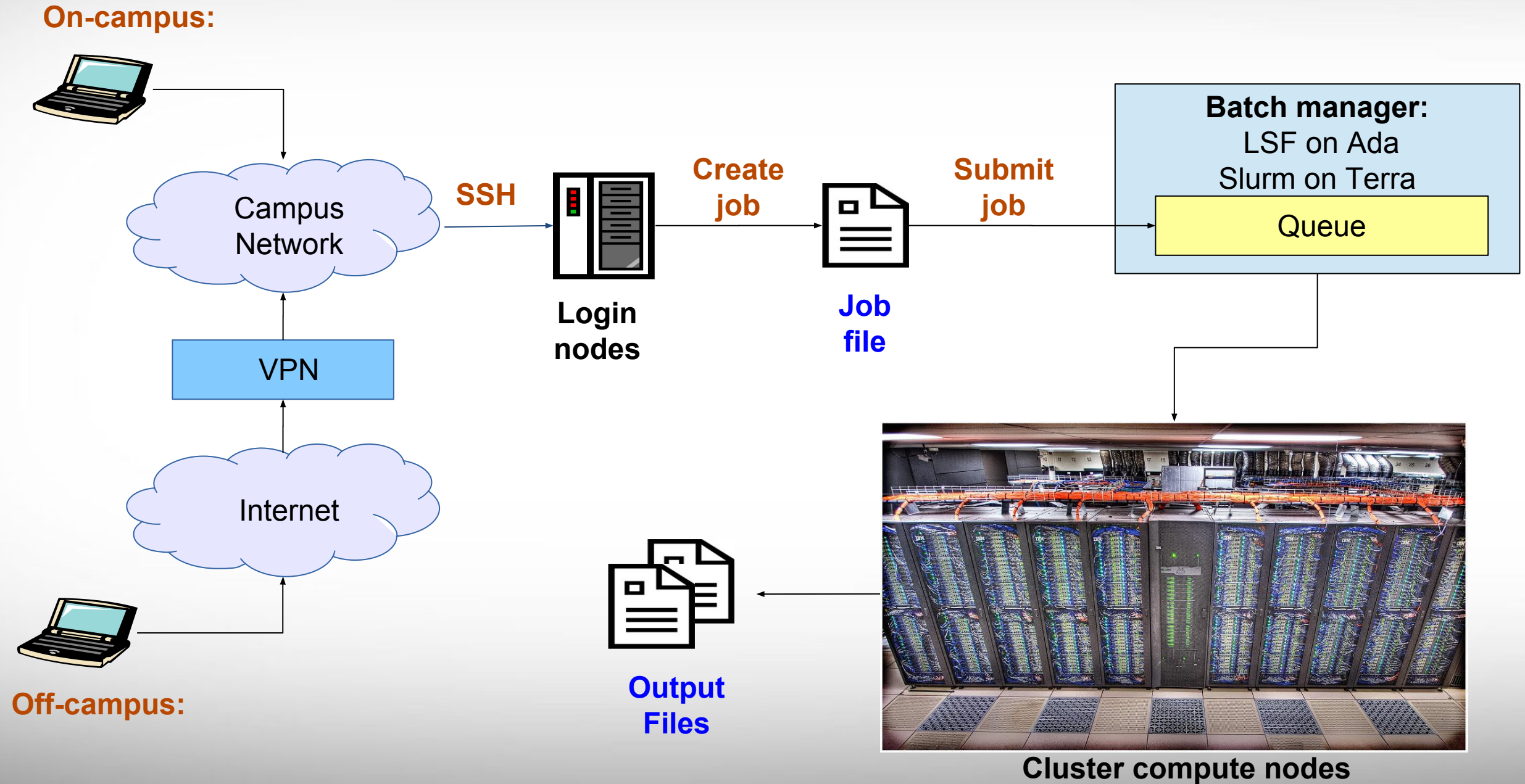
# Terra: Examples of SUs charged based on Job Cores, Time and Memory Requested

A Service Unit (SU) on **Terra** is equivalent to one core or **2** GB memory usage for one hour.

	Number of Cores	GB of memory per core	Total Memory (GB)	Hours	SUs charged
1.	1	2	2	1	1
2.	1	3	3	1	2
3.	1	56	56	1	28
4.	28	2	56	1	28

[https://hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](https://hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)

# Batch Computing on HPRC Clusters





# Historical HPRC Cluster Usage

hprc.tamu.edu

## Cluster Status

### Terra

Nodes 228/315 (72%)  
Cores 5889/9408 (63%)  
Jobs 311R-1Q

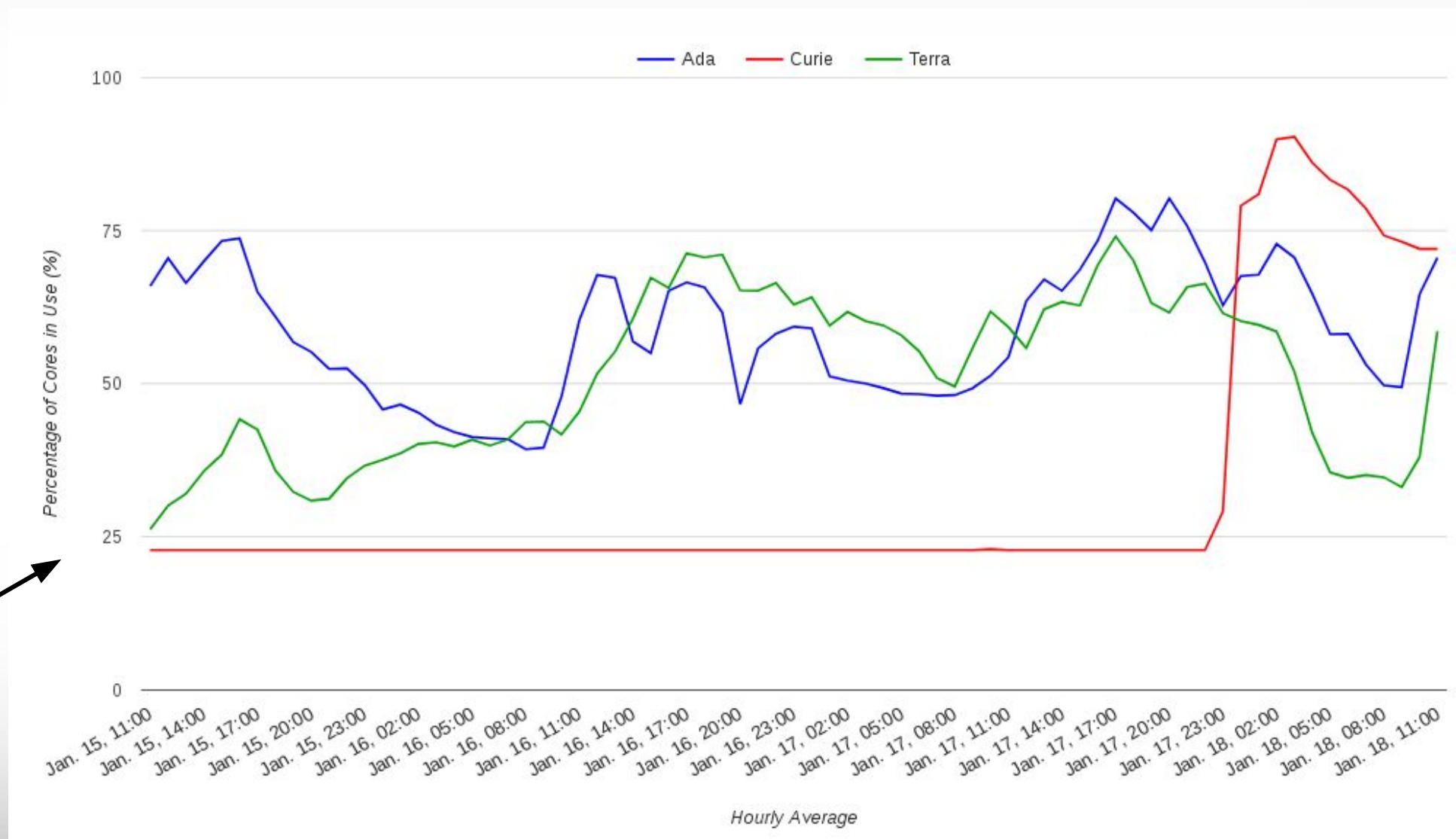
### Ada

Nodes 692/823 (84%)  
Cores 12070/16740 (72%)  
Jobs 318R-77Q

### Curie

Nodes 41/57 (72%)  
Cores 656/911 (72%)  
Jobs 41R-5Q

[Historical Status](#)



# Batch Queues

- Job submissions are auto-assigned to batch queues based on the resources requested (number of cores/nodes and walltime limit)
- Some jobs can be directly submitted to a queue:
  - On **Ada**, if the 1TB or 2TB nodes are needed, use the xlarge queue  
*#BSUB -q xlarge*
  - Jobs that have special resource requirements are scheduled in the special queue (must request access to use this queue)
- Batch queue policies are used to manage the workload and may be adjusted periodically.

[https://hprc.tamu.edu/wiki/Ada:Batch\\_Queues](https://hprc.tamu.edu/wiki/Ada:Batch_Queues)

<https://hprc.tamu.edu/wiki/Terra:Batch#Queues>

# bqueues : Current Queues on **Ada**

```
cmdickens@login7:~  
File Edit View Search Terminal Help  
[ netid @ada7 ~]$ bqueues  
QUEUE_NAME      PRIO STATUS      MAX JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP  
staff            450 Open:Active   -   -   -   -     0     0    0    0  
special          400 Open:Active   -   -   -   -   1076    0  1076  0  
xlarge           100 Open:Active   -   -   -   -    40     0   40    0  
vnc               90  Open:Active   -   -   -   -     6     0    6    0  
sn_short          80  Open:Active   -   -   -   -     0     0    0    0  
mn_short          80  Open:Active  2000 -   -   -     0     0    0    0  
mn_large          80  Open:Active  8000 -   -   -   2960    0  2960  0  
general           50  Closed:Inact  0     -   -   -     0     0    0    0  
sn_regular        50  Open:Active   -   -   -   -   1351    24  1327  0  
sn_long           50  Open:Active   -   -   -   -    577     0   577  0  
sn_xlong          50  Open:Active   -   -   -   -    430     0   430  0  
mn_small          50  Open:Active  7000 -   -   -   3320    0  3320  0  
mn_medium         50  Open:Active  6000 -   -   -   1670    0  1670  0  
curie_devel       40  Open:Active   -   -   -   -     0     0    0    0  
curie_medium      35  Open:Active   -   -   -   -     4     4    0    0  
curie_long        30  Open:Active   -   -   -   -    209     1   208  0  
curie_general     25  Closed:Inact  0     -   -   -     0     0    0    0  
low_priority      1   Open:Active  2500  500 -   -     0     0    0    0
```

[https://hprc.tamu.edu/wiki/Ada:Batch\\_Queues](https://hprc.tamu.edu/wiki/Ada:Batch_Queues)

# Queue Limits on **Ada**

Queue	Min/Default/Max Cores	Default/Max Walltime	Compute Node Types	Pre-Queue Limits	Aggregate Limits Across Queues	Per-User Limits Across Queues	Notes
sn_short	1 / 1 / 20	10 min / 1 hr	64 GB nodes (811) 256 GB nodes (26)		Maximum of <b>6000</b> cores for all running jobs in the single-node (sn_*) queues.	Maximum of <b>1000 cores and 50 jobs per user</b> for all running jobs in the single node (sn_*) queues.	For jobs needing <b>only one compute node</b> .
sn_regular		1 hr / 1 day					
sn_long		24 hr / 4 days					
sn_xlong		4 days / 7 days					
mn_short	2 / 2 / 200	10 min / 1 hr		Maximum of <b>2000</b> cores for all running jobs in this queue.	Maximum of <b>12000</b> cores for all running jobs in the multi-node (mn_*) queues.	Maximum of <b>3000 cores and 150 jobs per user</b> for all running jobs in the multi-node (mn_*) queues.	For jobs needing <b>more than one compute node</b> .
mn_small	2 / 2 / 120	1 hr / 10 days		Maximum of <b>6000</b> cores for all running jobs in this queue.			
mn_medium	121 / 121 / 600	1 hr / 7 days		Maximum of <b>6000</b> cores for all running jobs in this queue.			
mn_large	600 / 601 / 2000	1 hr / 5 days					
xlarge	1 / 1 / 280	1 hr / 10 days	1 TB nodes (11) 2 TB nodes (4)				For jobs needing <b>more than 256GB of memory per compute node</b> .
vnc	1 / 1 / 20	1 hr / 6 hr	GPU nodes (30)				For remote visualization jobs.
special	None	1 hr / 7 days	64 GB nodes (811) 256 GB nodes (26)				Requires permission to access this queue.

Run "*blimits -w*" to show how policies are applied to users and queues.

[https://hprc.tamu.edu/wiki/Ada:Batch\\_Queues](https://hprc.tamu.edu/wiki/Ada:Batch_Queues)

# sinfo : Current Queues on Terra

```
cmdickens@tlogin-0502:~  
File Edit View Search Terminal Help  
[ netid @terra2 ~]$ sinfo  
PARTITION      AVAIL  TIMELIMIT  JOB_SIZE  NODES(A/I/O/T)  CPUS(A/I/O/T)  
short*         up     2:00:00    1-16      156/145/3/304   3667/4761/84/8512  
medium         up     1-00:00:00 1-64      156/145/3/304   3667/4761/84/8512  
long           up     7-00:00:00 1-32      156/145/3/304   3667/4761/84/8512  
gpu            up     2-00:00:00 1-48      48/0/0/48       797/547/0/1344  
vnc            up     12:00:00   1         48/0/0/48       797/547/0/1344  
xlong          up     21-00:00:00 1-32      108/145/3/256   2870/4214/84/7168  
staff          up     infinite   1-infinite 156/145/3/304   3667/4761/84/8512  
low_priority   up     1-00:00:00 1-infinite 156/145/3/304   3667/4761/84/8512  
special        up     7-00:00:00 1-infinite 156/145/3/304   3667/4761/84/8512  
knl            up     7-00:00:00 1-8       0/14/2/16       0/980/140/1120
```

For the NODES and CPUS columns:  
A = Active (in use by running jobs)  
I = Idle (available for jobs)  
O = Offline (unavailable for jobs)  
T = Total

# Queue Limits on Terra

Queue	Job Max Cores / Nodes	Job Max Walltime	Compute Node Types	Per-User Limits Across Queues	Notes
short	448 cores / 16 nodes	2 hrs	64 GB nodes (256) 128 GB nodes with GPUs (36)	1800 cores per user	
medium	1792 cores / 64 nodes	1 day			
long	896 cores / 32 nodes	7 days			
xlong	448 cores / 16 nodes	21 days	64 GB nodes (256)	448 cores per user	--partition xlong
gpu	1344 cores / 48 nodes	2 days	128 GB nodes with GPUs (48)		For jobs requiring GPUs.
vnc	28 cores / 1 node	6 hours	128 GB nodes with GPUs (48)		For remote visualization jobs
knl	68 cores / 8 nodes 72 cores / 8 nodes	7 days	96 GB nodes with KNL processors (16)		For jobs requiring a KNL processor

# Batch Job Scripts

# Sample Job Script Structure (Ada)

## ##NECESSARY JOB SPECIFICATIONS

```
#BSUB -L /bin/bash
#BSUB -J ExampleJob
#BSUB -W 24:00
#BSUB -n 1
#BSUB -R "span[ptile=1] "
#BSUB -R "rusage [mem=2500] "
#BSUB -M 2500
#BSUB -o stdout.%J
#BSUB -e stderr.%J
```

These parameters describe your job to the job scheduler

## ##OPTIONAL JOB SPECIFICATIONS

```
#BSUB -P 123456
#BSUB -u email_address
#BSUB -B -N
```

This is single line comment and not run as part of the script

```
# load required module(s)
module load Python/3.5.2-intel-2017A
```

Load the required module(s) first

```
./my_program.py
```

This is a command that is executed by the job



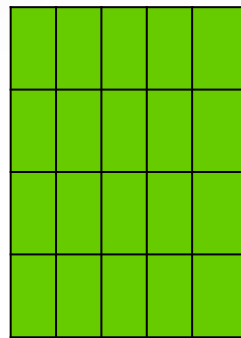
# Important Batch Job Parameters

Ada	Terra	Comment
#BSUB -L /bin/bash	#SBATCH --export=NONE #SBATCH --get-user-env=L	Initialize job environment.
#BSUB -W HH:MM or #BSUB -W MM	#SBATCH --time=HH:MM:SS	Specifies the time limit for the job. Must specify seconds SS on Terra
#BSUB -n NNN	#SBATCH --ntasks=NNN	Total number of tasks (cores) for the job.
#BSUB -R "span[ptile=XX]"	#SBATCH --ntasks-per-node=XX	Specifies the maximum number of tasks (cores) to allocate per node
#BSUB -R "rusage [mem=nnnn]" #BSUB -M nnnn  (memory per CORE)	#SBATCH --mem=nnnnM or #SBATCH --mem=nG  (memory per NODE)	Sets the maximum amount of memory (MB).  G for GB is supported on Terra

[https://hprc.tamu.edu/wiki/HPRC:Batch\\_Translation](https://hprc.tamu.edu/wiki/HPRC:Batch_Translation)

# Mapping Jobs to Cores per Node on **Ada**

**A.**

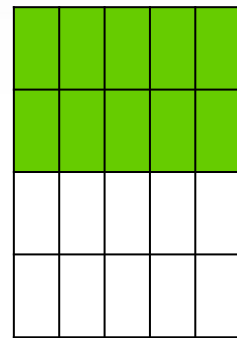


20 cores on  
1 compute node

#BSUB -n 20  
#BSUB -R "span[ptile=20]"

Preferred Mapping  
(if applicable)

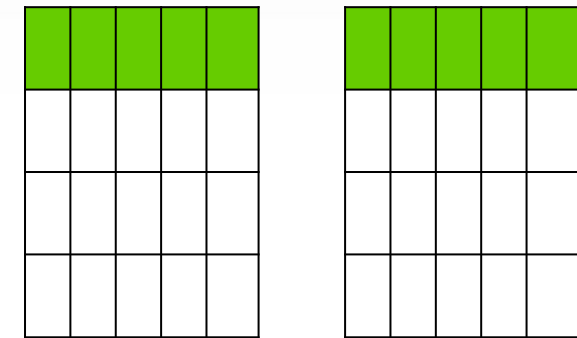
**B.**



20 cores on  
2 compute nodes

#BSUB -n 20  
#BSUB -R "span[ptile=10]"

**C.**

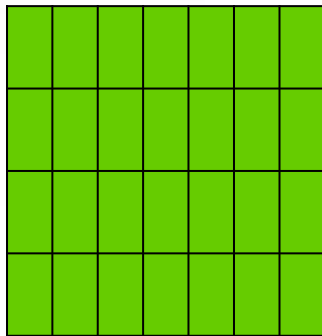


20 cores on  
4 compute nodes

#BSUB -n 20  
#BSUB -R "span[ptile=5]"

# Mapping Jobs to Cores per Node on Terra

A.

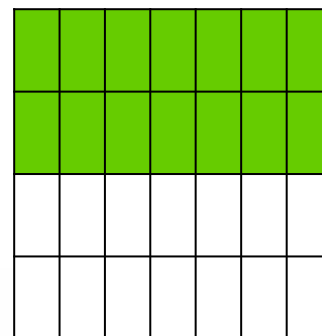
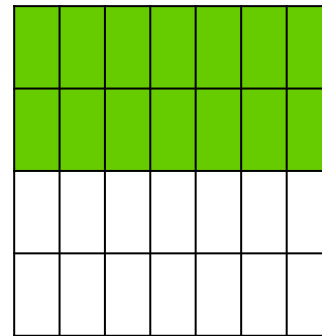


28 cores on  
1 compute node

#SBATCH --ntasks 28  
#SBATCH --tasks-per-node=28

Preferred Mapping  
(if applicable)

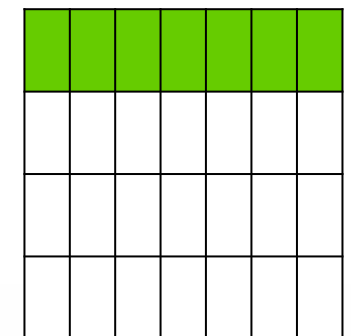
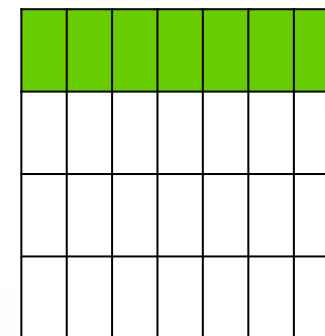
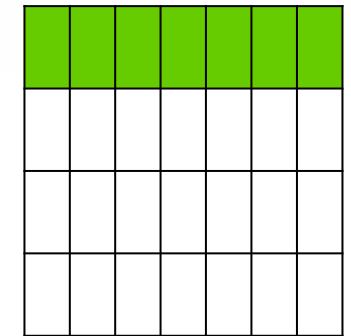
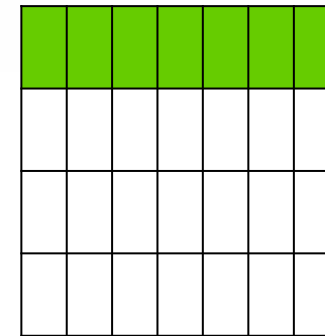
B.



28 cores on  
2 compute nodes

#SBATCH --ntasks 28  
#SBATCH --tasks-per-node=14

C.



28 cores on  
4 compute nodes

#SBATCH --ntasks 28  
#SBATCH --tasks-per-node=7

# Job Memory Requests on **Ada**

- Must specify both parameters for requesting memory:
  - **#BSUB -R "rusage[mem=process\_alloc\_size]"**
  - **#BSUB -M process\_size\_limit**
- Default value of 2500 MB (2.5 GB) per job slot if -R/-M not specified, but it might cause memory contention when sharing a node with other jobs.
- On 64GB nodes, usable memory is at most **54 GB** (where 10 GB is used by the system).
  - The per-process memory limit should not exceed **2700 MB** for a 20-core job.
- If more memory is needed, request the large memory nodes:
  - If under 256 GB and up to 20 cores per node use:
    - **#BSUB -R "select[mem256gb]"**
  - If you need up to 1 or 2 TB of memory and up to 40 cores:
    - use the **-q xlarge** option with either **-R "select[mem1tb]"** or **-R "select[mem2tb]"**
    - The mem1tb and mem2tb nodes are accessible only via the *xlarge* queue.

# Ada: Software for Large Memory Nodes

- The large memory nodes (1TB and 2TB) on Ada have their own separate modules
- To see the list of available large memory node modules use the following commands:

```
module load Westmere  
module avail
```

- Look in the Westmere section:

```
----- /software/easybuild/Westmere/modules/all -----  
ABINIT/8.0.8-intel-2016a                               Tcl/8.6.3-foss-2015a  
ABYSS/1.9.0-foss-2016a                               Tcl/8.6.3-intel-2015B  
ABYSS/1.9.0-intel-2015B-Python-2.7.10                (D) Tcl/8.6.4-foss-2016a
```

- Or type **module spider tool\_name/version** to see if Westmere needs to be loaded

```
module spider Canu/1.7-intel-2017A-Perl-5.24.0
```

```
You will need to load all module(s) on any one of the lines below before the  
"Canu/1.7-intel-2017A-Perl-5.24.0" module is available to load.
```

```
Westmere/0.devel  
Westmere/1
```

- You can just load Westmere along with the other module(s)

```
module load Westmere  
module load Canu/1.7-intel-2017A-Perl-5.24.0
```

# Sample Job Script for 1TB Node on **Ada**

```
##NECESSARY JOB SPECIFICATIONS
#BSUB -L /bin/bash # Uses bash to initialize the job's execution environment.
#BSUB -J my_job_script # Set the job name to "my_job_script"
#BSUB -W 24:00 # Set the wall clock limit to 24hr
#BSUB -q xlarge # Request xlarge queue
#BSUB -R "select [mem1tb] " # Request 1TB memory node
#BSUB -n 40 # Request 40 core
#BSUB -R "span [ptile=40] " # Request 40 core per node.
#BSUB -R "rusage [mem=24500] " # Request 24500MB (24.5GB) per process (CPU) for the job
#BSUB -M 24500 # Set the per process enforceable memory limit to 24500MB.
#BSUB -o stdout.%J # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J # Send stderr to "stderr.[jobID]"
#BSUB -u email_address # (optional) Send all emails to email_address
#BSUB -B -N # (optional) Send email on job begin (-B) and end (-N)

# load required module(s) for use on the large memory nodes
module load Westmere
module load Canu/1.7-intel-2017A-Perl-5.24.0

# run your commands
canu -assemble *fastq
```

# Job Memory Requests on Terra

- Specify memory request based on memory per node:  
**#SBATCH --mem=xxxxM**                      **# memory per node in MB**  
or  
**#SBATCH --mem=xG**                              **# memory per node in GB**
- On 64GB nodes, usable memory is at most 56 GB. The per-process memory limit should not exceed 2000 MB for a 28-core job.
- On 128GB nodes, usable memory is at most 112 GB. The per-process memory limit should not exceed 4000 MB for a 28-core job.

# Ada Pop Quiz

```
#BSUB -L /bin/bash
#BSUB -J stacks_S2
#BSUB -n 10
#BSUB -R "span [ptile=10] "
#BSUB -R "rusage [mem=2500] "
#BSUB -M 2500
#BSUB -W 36:00
#BSUB -o stdout.%J
#BSUB -e stderr.%J
```

How much total memory is requested for this job?

- A. 2.5 GB
- B. 2500 MB
- C. 25 GB
- D. 250 GB



# Ada Pop Quiz

```
#BSUB -L /bin/bash
#BSUB -J stacks_S2
#BSUB -n 80
#BSUB -R "span [ptile=20] "
#BSUB -R "rusage [mem=2500] "
#BSUB -M 2500
#BSUB -W 48:00
#BSUB -o stdout.%J
#BSUB -e stderr.%J
```

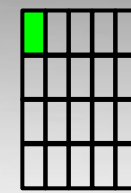
How many nodes is this job requesting?

- A. 1600
- B. 80
- C. 20
- D. 4

# Example Batch Job Scripts

[https://hprc.tamu.edu/wiki/Ada:Batch\\_Processing\\_LSF#Job\\_File\\_Examples](https://hprc.tamu.edu/wiki/Ada:Batch_Processing_LSF#Job_File_Examples)

# Ada Job File (Serial Example)



```
##NECESSARY JOB SPECIFICATIONS
#BSUB -L /bin/bash           # Uses bash to initialize the job's execution environment.
#BSUB -J ExampleJob1        # Set the job name to "ExampleJob1"
#BSUB -W 2:00                # Set the wall clock limit to 2hr
#BSUB -n 1                   # Request 1 core
#BSUB -R "span[ptile=1]"     # Request 1 core per node.
#BSUB -R "rusage[mem=2500]"  # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J          # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J          # Send stderr to "stderr.[jobID]"

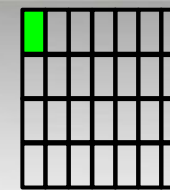
##OPTIONAL JOB SPECIFICATIONS
#BSUB -P 123456              # Set billing account to 123456
#BSUB -u email_address       # Send all emails to email_address
#BSUB -B -N                  # Send email on job begin (-B) and end (-N)

# load required module(s)
module load Python/3.5.2-intel-2017A

# run your program
./my_program.py
```

SUs = 2

# Terra Job File (Serial Example)



```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE          # Do not propagate environment
#SBATCH --get-user-env=L       # Replicate login environment

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1 # Set the job name to "JobExample1"
#SBATCH --time=01:30:00        # Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=1             # Request 1 task (core)
#SBATCH --mem=2G               # Request 2GB per node
#SBATCH --output=stdout.%j     # Send stdout and stderr to "stdout.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
#SBATCH --account=123456       # Set billing account to 123456
#SBATCH --mail-type=ALL        # Send email on all job events
#SBATCH --mail-user=email_address # Send all emails to email_address

# load required module(s)
module load intel/2017A

# run your program
./myprogram
```

SUs = 1.5

# Submitting Your Job and Check Job Status

Submit job

```
cd $SCRATCH/Spring
```

**Ada:** `bsub < example01.job`

```
Verifying job submission parameters...
Verifying project account...
  Account to charge:    082792010838
  Balance (SUs):       4871.5983
  SUs to charge:       0.0333
Job <2470599> is submitted to default queue <sn_short>.
```

**Terra:** `sbatch example01.job`

```
Submitted batch job 161997
(from job_submit) your job is charged as below
  Project Account: 122792016265
  Account Balance: 1687.066160
  Requested SUs:   3
```

Check status

**Ada:** `bjobs`

**Ada (more detailed):** `bjobs -l 2470599`

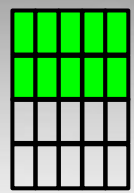
(dash lower case l as in *list*)

JOBID	STAT	USER	QUEUE	JOB_NAME	NEXEC_HOST	SLOTS	RUN_TIME	TIME_LEFT
2470599	RUN	tmarkhuang	sn_short	sample01	1	1	0 second(s)	0:5 L

**Terra:** `squeue -u $USER`

JOBID	NAME	USER	PARTITION	NODES	CPUS	STATE	TIME	TIME_LEFT	START_TIME	REASON	NODELIST
64039	somejob	someuser	medium	4	112	PENDING	0:00	20:00	2017-01-30T21:00:4	Resources	
64038	somejob	someuser	medium	4	112	RUNNING	2:49	17:11	2017-01-30T20:40:4	None	tnxt-[0401-0404]

# Ada Job File (multi core, single node)



## ##NECESSARY JOB SPECIFICATIONS

```
#BSUB -L /bin/bash           # Use bashto initialize the job's execution environment.
#BSUB -J ExampleJob2        # Set the job name to "ExampleJob2"
#BSUB -W 6:30                # Set the wall clock limit to 6hr and 30min
#BSUB -n 10                  # Request 10 cores total for the job
#BSUB -R "span[ptile=10]"    # Request 10 cores per node.
#BSUB -R "rusage[mem=2500]"  # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J          # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J          # Send stderr to "stderr.[jobID]"
```

SUs = 65

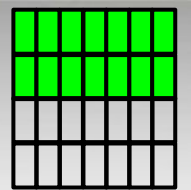
## ##OPTIONAL JOB SPECIFICATIONS

```
#BSUB -P 123456             # Set billing account to 123456 #find your account with "myproject"
#BSUB -u email_address      # Send all emails to email_address
#BSUB -B -N                  # Send email on job begin (-B) and end (-N)
```

```
# load required module(s)
module load intel/2017A
```

```
# run your program
./my_multicore_program
```

# Terra Job File (multi core, single node)



```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE           # Do not propagate environment
#SBATCH --get-user-env=L        # Replicate login environment

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample2  # Set the job name to "JobExample2"
#SBATCH --time=6:30:00         # Set the wall clock limit to 6hr and 30min
#SBATCH --nodes=1              # Request 1 node
#SBATCH --ntasks-per-node=14   # Request 14 tasks(cores) per node
#SBATCH --mem=28G              # Request 28GB per node
#SBATCH --output=stdout.%j     # Send stdout and stderr to "stdout.[jobID]"

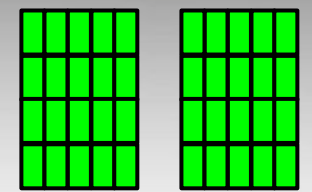
##OPTIONAL JOB SPECIFICATIONS
#SBATCH --account=123456       # Set billing account to 123456 #find your account with "myproject"
#SBATCH --mail-type=ALL        # Send email on all job events
#SBATCH --mail-user=email_address # Send all emails to email_address

# load required module(s)
module load intel/2017A

# run your program
./my_multicore_program
```

SUs = 91

# Ada Job File (multi core, multi node)



## ##NECESSARY JOB SPECIFICATIONS

```
#BSUB -J ExampleJob3           # Set the job name to "ExampleJob3"
#BSUB -L /bin/bash             # Use bash to initialize the job's execution environment.
#BSUB -W 24:00                 # Set the wall clock limit to 24hr
#BSUB -n 40                    # Request 40 cores total for the job
#BSUB -R "span[ptile=20] "     # Request 20 cores per node.
#BSUB -R "rusage[mem=2500] "   # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                  # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J            # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J            # Send stderr to "stderr.[jobID]"
```

SUs = 960

## ##OPTIONAL JOB SPECIFICATIONS

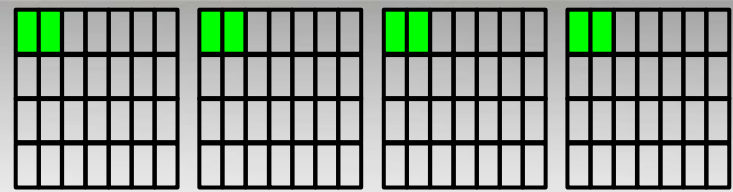
```
#BSUB -P 123456                # Set billing account to 123456 #find your account with " myproject"
#BSUB -u email_address         # Send all emails to email_address
#BSUB -B -N                    # Send email on job begin (-B) and end (-N)
```

```
# load required module(s)
module load intel/2017A
```

```
# run your program
./my_multicore_multinode_program
```



# Terra Job File (multi core, multi node)



```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE           # Do not propagate environment
#SBATCH --get-user-env=L       # Replicate login environment

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample3 # Set the job name to "JobExample3"
#SBATCH --time=1-12:00:00      # Set the wall clock limit to 1 Day and 12hr
#SBATCH --ntasks=8             # Request 8 tasks (cores)
#SBATCH --ntasks-per-node=2    # Request 2 tasks(cores) per node
#SBATCH --mem=2.5G             # Request 2.5 GB per node
#SBATCH --output=stdout.%j     # Send stdout and stderr to "stdout.[jobID]"

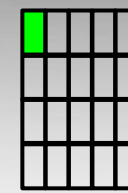
##OPTIONAL JOB SPECIFICATIONS
#SBATCH --account=123456       # Set billing account to 123456 #find your account with "myproject"
#SBATCH --mail-type=ALL        # Send email on all job events
#SBATCH --mail-user=email_address # Send all emails to email_address

# this intel toolchain is just an example.  recommended toolchain is TBD
module load intel/2017A

# run program with MPI
mpirun ./my_multicore_multinode_program
```

SUs = 288

# Ada Job File (serial GPU)



```
##NECESSARY JOB SPECIFICATIONS
#BSUB -J ExampleJob4           # Set the job name to "ExampleJob4"
#BSUB -L /bin/bash             # Use bash login shell to initialize the job's execution environment.
#BSUB -W 2:00                  # Set the wall clock limit to 2hr
#BSUB -n 1                      # Request 1 core total for the job
#BSUB -R "span[ptile=1]"       # Request 1 core per node.
#BSUB -R "rusage[mem=2500]"    # Request 2500MB per process (CPU) for the job
#BSUB -M 2500                  # Set the per process enforceable memory limit to 2500MB.
#BSUB -o stdout.%J            # Send stdout to "stdout.[jobID]"
#BSUB -e stderr.%J           # Send stderr to "stderr.[jobID]"

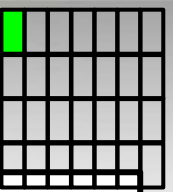
#BSUB -R "select[gpu]"        # Request a node with a GPU
##OPTIONAL JOB SPECIFICATIONS
#BSUB -P 123456                # Set billing account to 123456 #find your account with "myproject"
#BSUB -u email_address         # Send all emails to email_address
#BSUB -B -N                    # Send email on job begin (-B) and end (-N)

# load required module(s)
module load CUDA/9.2.148.1

# run your program
./my_gpu_program
```

SUs = 1

# Terra Job File (serial GPU)



```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE           # Do not propagate environment
#SBATCH --get-user-env=L       # Replicate login environment

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample4  # Set the job name to "JobExample4"
#SBATCH --time=01:00:00        # Set the wall clock limit to 1hr
#SBATCH --ntasks=1             # Request 1 task (core)
#SBATCH --mem=2G               # Request 2GB per node
#SBATCH --output=stdout.%j     # Send stdout and stderr to "stdout.[jobID]"
#SBATCH --gres=gpu:1           # Request 1 GPU
#SBATCH --partition=gpu        # Request the GPU partition/queue

##OPTIONAL JOB SPECIFICATIONS
#SBATCH --account=123456       # Set billing account to 123456 #find your account with "myproject"
#SBATCH --mail-type=ALL       # Send email on all job events
#SBATCH --mail-user=email_address # Send all emails to email_address

# load required module(s)
module load intel/2017A
module load CUDA/9.2.148.1

# run your program
./my_gpu_program
```

SUs = 28

# Other Type of Jobs

- MPI and OpenMP
- Visualization:
  - <https://portal.hprc.tamu.edu> (currently visualization jobs only on Ada; portal coming soon to Terra)
- Large number of concurrent single core jobs
  - Check out *tamulauncher*
    - <https://hprc.tamu.edu/wiki/SW:tamulauncher>
    - Useful for running many single core commands concurrently across multiple nodes within a job
    - Can be used with serial or multi-threaded programs
    - Distributes a set of commands from an input file to run on the cores assigned to a job
    - Can only be used in batch jobs
    - If a tamulauncher job gets killed, you can resubmit the same job to complete the unfinished commands in the input file

# Job Submission and Tracking

Ada	Terra	Description
<code>bsub &lt; jobfile1</code>	<code>sbatch jobfile1</code>	Submit jobfile1 to batch system
<code>bjobs [-u all or user_name] [[-l] job_id]</code>	<code>squeue [-u user_name] [-j job_id]</code>	List jobs
<code>bkill job_id</code>	<code>scancel job_id</code>	Kill a job
<code>bhist [-l] job_id</code>	<code>sacct -X -j job_id</code>	Show information for a job (can be when job is running or recently finished)
-	<code>sacct -X -S YYYY-HH-MM</code>	Show information for all of your jobs since YYYY-HH-MM
<code>lnu [-l] -j job_id</code>	<code>lnu job_id</code>	Show resource usage for a job
-	<code>pestat -u \$USER</code>	Show resource usage for a running job

[https://hprc.tamu.edu/wiki/HPRC:Batch\\_Translation](https://hprc.tamu.edu/wiki/HPRC:Batch_Translation)

# Debug job failures

```
cd $SCRATCH/Spring
```

- Debug job failures using the stdout and stderr files

```
bsub < example03.python_memory.job
```

```
cat output.ex03.python_mem.2447336
```

This job id was created by the parameter in your job script file

**Ada:** #BSUB -o output.ex03.python\_mem.%J

**Terra:** #SBATCH -o output.ex03.python\_mem.%j

```
TERM_MEMLIMIT: job killed after reaching LSF memory usage limit.  
Exited with signal termination: Killed.
```

```
Resource usage summary:
```

**Ada:**

CPU time :	1.42 sec.
Max Memory :	10 MB
Average Memory :	6.50 MB
Total Requested Memory :	10.00 MB
Delta Memory :	0.00 MB
Max Processes :	5
Max Threads :	6

**Terra:** slurmstepd: error: Exceeded job memory limit at some point.

Make the necessary adjustments to BSUB parameters in your job script and resubmit the job

# Check your Service Unit (SU) Balance

- List the SU Balance of your Account(s)

```
myproject
```

```
=====
List of YourNetID's Project Accounts
-----
| Account | FY | Default | Allocation | Used & Pending SUs | Balance | PI |
-----
| 1228000223136 | 2019 | N | 10000.00 | 0.00 | 10000.00 | Doe, John |
-----
| 1428000243716 | 2019 | Y | 5000.00 | -71.06 | 4928.94 | Doe, Jane |
-----
| 1258000247058 | 2019 | N | 5000.00 | -0.91 | 4999.09 | Doe, Jane |
-----
```

- To specify a project ID to charge in the job file
  - **Ada:** `#BSUB -P Account#`
  - **Terra:** `#SBATCH -A Account#`
- Run `"myproject -d Account#"` to change default project account
- Run `"myproject -h"` to see more options

[https://hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](https://hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)

<https://hprc.tamu.edu/wiki/HPRC:AMS:UI>

# Job submission issue: insufficient SUs

Ada:

```
$ bsub < myjob
Verifying job submission parameters...
Verifying project account...
  Account to charge:    082792010838
  Balance (SUs):       342.5322
  SUs to charge:       480.0000
-----
|ERROR! Your project account does not have sufficient balance to submit your job! |
-----
Request aborted by esub. Job not submitted.
```

Terra:

```
$ sbatch myjob
sbatch: error: (from job_submit) your account's balance is not sufficient to submit your job
  Project Account: 123940134739
  Account Balance: 382.803877
  Requested SUs:  18218.666666667
```

- What to do if you need more SUs
  - Ask your PI to transfer SUs to your account
  - Apply for more SUs (if you are eligible, as a PI or permanent researcher)

[https://hprc.tamu.edu/wiki/HPRC:AMS:Service\\_Unit](https://hprc.tamu.edu/wiki/HPRC:AMS:Service_Unit)

<https://hprc.tamu.edu/wiki/HPRC:AMS:UI>



# List Node Utilization on **Ada**: *lnu*

**lnu** [-h] [-l] -j **jobid**

# lists the node utilization across all nodes for a running job.

# to see more options use: **lnu -h**

## Example:

```
lnu -l -j 795375
```

```
Job          User          Queue          Status Node  Cpus
795375      jomber23      medium         R      4    80
HOST_NAME   status  r15s  r1m  r15m  ut  pg  ls  it  tmp  swp  mem  Assigned Cores
nxt1417      ok      20.0  21.0  21.0  97%  0.0  0  94976  366M  3.7G  41.6G  20
nxt1764 (L)  ok      19.7  20.0  20.0  95%  0.0  0  95040  366M  3.7G  41.5G  20
nxt2111      ok      20.0  20.0  20.0  98%  0.0  0  91712  370M  4.2G  41.5G  20
nxt2112      ok      20.0  21.1  21.0  97%  0.0  0  91712  370M  4.2G  41.6G  20
=====
```

The % utilization (**ut**) in conjunction with Assigned Cores is the most useful. Note that the **tmp**, **swp**, and **mem** refer to available amounts respectively and not usage. See "*man lsload*" for explanations on labels.

[https://hprc.tamu.edu/wiki/Ada:Batch\\_Processing\\_LSF#Job\\_Tracking](https://hprc.tamu.edu/wiki/Ada:Batch_Processing_LSF#Job_Tracking)

# List Node Utilization on **Terra**: *lnu*

**lnu jobid**

# lists the node utilization across all nodes for a running job.  
# to see more options use: **lnu -h**

## Example:

```
lnu 565849
```

Note: Slurm updates the node information every few minutes

```
JOBID    NAME      USER      PARTITION  NODES  CPUS  STATE    TIME    TIME_LEFT  START_TIME
565849   somename  someuser   long        3      84    RUNNING  17:37   6-23:42:23 2018-01-25T15:19:55

HOSTNAMES  CPU_LOAD  FREE_MEM  MEMORY  CPUS (A/I/O/T)
tnxt-0703  26.99    53462    57344   28/0/0/28
tnxt-0704  26.93    52361    57344   28/0/0/28
tnxt-0705  26.95    47166    57344   28/0/0/28
```

Note: CPU\_LOAD is not the same as % utilization

### For the CPUS columns:

**A = Active (in use by running jobs)**  
**I = Idle (available for jobs)**  
**O = Offline (unavailable for jobs)**  
**T = Total**

# Monitor Node Utilization on Terra: *pestat*

`pestat [-u username]`

# lists the node utilization across all nodes for a running job.

# to see more options use: `pestat -h`

Example:

```
pestat -u $USER
```

Hostname	Partition	Node	Num_CPU	CPUload	Memsize	Freemem	Joblist
		State	Use/Tot		(MB)	(MB)	JobId User ...
tnxt-0703	xlong	alloc	28 28	16.23*	57344	55506	565849 someuser
tnxt-0704	xlong	alloc	28 28	19.60*	57344	53408	565849 someuser
tnxt-0705	xlong	alloc	28 28	19.56*	57344	53408	565849 someuser

Low CPU load utilization highlighted in Red  
( Freemem should also be noted )

```
pestat -u $USER
```

Hostname	Partition	Node	Num_CPU	CPUload	Memsize	Freemem	Joblist
		State	Use/Tot		(MB)	(MB)	JobId User ...
tnxt-0703	xlong	alloc	28 28	27.54	57344	55506	565849 someuser
tnxt-0704	xlong	alloc	28 28	27.50	57344	53408	565849 someuser
tnxt-0705	xlong	alloc	28 28	26.47*	57344	53408	565849 someuser

Good CPU load utilization highlighted in Purple  
Ideal CPU load utilization displayed in White

# Job Environment Variables

## ● Ada:

- **\$LSB\_JOBID** = job id
- **\$LS\_SUBCWD** = directory where job was submitted from
- **\$SCRATCH** = /scratch/user/NetID
- **\$TMPDIR** = /work/\$LSB\_JOBID.tmpdir
  - \$TMPDIR is local to each assigned compute node for the job and is about 750 GB
  - Use of \$TMPDIR is recommended for jobs that use many small temporary files
  - Do not use \$TMPDIR for software that has checkpoints to restart where it left off

## ● Terra:

- **\$SLURM\_JOBID** = job id
- **\$SLURM\_SUBMIT\_DIR** = directory where job was submitted from
- **\$SCRATCH** = /scratch/user/NetID
- **\$TMPDIR** = /work/job.\$SLURM\_JOBID
  - \$TMPDIR is local to each assigned compute node for the job and is about 850GB

[https://hprc.tamu.edu/wiki/Ada:Batch\\_Processing\\_LSF#Environment\\_Variables](https://hprc.tamu.edu/wiki/Ada:Batch_Processing_LSF#Environment_Variables)

[https://hprc.tamu.edu/wiki/Terra:Batch#Environment\\_Variables](https://hprc.tamu.edu/wiki/Terra:Batch#Environment_Variables)

# Common Job Problems

- Control characters (^M) in job files or data files edited with Windows editor
  - remove the ^M characters with: `dos2unix my_job_file`
- Did not load the required module(s)
- Insufficient walltime specified in `#BSUB -W` or `#SBATCH --time` parameter
- Insufficient memory specified in `#BSUB -M` and `-R "rusage [mem=xxx]"`, or `#SBATCH --mem` or `--mem-per-cpu` parameters
- No matching resource (`-R rusage [mem]` or `--mem` too large)
- Running OpenMP jobs across nodes
- Insufficient SU: See your SU balance: `myproject`
- Insufficient disk or file quotas: check quota with `showquota`
- Using GUI-based software without setting up X11 forwarding
  - Enable X11 forwarding at login `ssh -X user@ada.tamu.edu`
  - Or use VNC
- Software license availability  
`license_status -a`

FAQ: <https://hprc.tamu.edu/wiki/HPRC:CommonProblems>

# CRLF Line Terminators

Windows editors such as Notepad will add hidden Carriage Return Line Feed (CRLF) characters that will cause problems with many applications

```
cd $SCRATCH/Spring
```

```
file dos_text.txt
```

# use file command to check

```
dos_text.txt: ASCII English text, with CRLF line terminators
```

```
cat -v dos_text.txt
```

# use cat command to see CRLF characters

```
dos2unix dos_text.txt  
file dos_text.txt
```

# use dos2unix command to correct

```
dos_text.txt: ASCII English text
```

# portal.hprc.tamu.edu



OnDemand provides an integrated, single access point for all of your HPC resources.

## Message of the Day

### IMPORTANT POLICY INFORMATION

- Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.
- Use of HPRC resources in violation of United States export control laws and regulations is prohibited. Current HPRC staff members are US citizens and legal residents.
- Sharing HPRC account and password information is in violation of State Law. Any shared accounts will be DISABLED.
- Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu/policies>

!! WARNING: There are NO active backups of user data. !!

The HPRC portal allows users to do the following

- Browse files on the Ada (Curie) filesystem
- Access the ada, terra or curie UNIX command line
  - no SUs charged
- Launch jobs
  - SUs charged
- Compose job scripts
- Launch interactive GUI apps (SUs charged) on Ada such as
  - ANSYS Workbench
  - Abaqus/CAE
  - IGV
  - LS-PREPOST
  - MATLAB
  - ParaView
  - VNC
  - Galaxy
  - Jupyter Notebook
  - RStudio
- Monitor and stop running jobs and interactive sessions

# Need Help?

- First check the FAQ <https://hprc.tamu.edu/wiki/HPRC:CommonProblems>
  - Ada User Guide <https://hprc.tamu.edu/wiki/Ada>
    - Exercises <https://hprc.tamu.edu/wiki/Ada:Exercises>
  - Terra User Guide <https://hprc.tamu.edu/wiki/Terra>
    - Exercises <https://hprc.tamu.edu/wiki/Terra:Exercises>
- Email your questions to [help@hprc.tamu.edu](mailto:help@hprc.tamu.edu). (Managed by a ticketing system)
- Help us, help you -- we need more info
  - Which Cluster
  - UserID/NetID (*UIN is not needed!*)
  - Job id(s) if any
  - Location of your jobfile, input/output files
  - Application used if any
  - Module(s) loaded if any
  - Error messages
  - Steps you have taken, so we can reproduce the problem
- Or visit us @ 114A Henderson Hall
  - Making an appointment is recommended.





**HIGH PERFORMANCE  
RESEARCH COMPUTING**  
TEXAS A&M UNIVERSITY

**Thank you.**

*Any question?*

# Upcoming HPRC Short Courses

<b>Jan 22</b>	<b>1:30 to 4:00 PM</b>	Introduction to Next Generation Sequencing
<b>Jan 25</b>	<b>1:30 to 4:00 PM</b>	Data Management and Practices
<b>Jan 29</b>	<b>1:30 to 4:00 PM</b>	NGS Assembly
<b>Feb 1</b>	<b>9:00 AM to 5:00 PM</b>	NVIDIA DLI Deep Learning for Computer Vision Workshop
<b>Feb 5</b>	<b>1:30 to 4:00 PM</b>	NGS Metagenomics
<b>Feb 8</b>	<b>10:00 AM to 12:30 PM</b>	Introduction to Python
<b>Feb 8</b>	<b>1:30 to 4:00 PM</b>	Introduction to Scientific Python
<b>Feb 12</b>	<b>1:30 to 4:00 PM</b>	NGS RADSeq/GBS
<b>Feb 15</b>	<b>10:00 AM to 12:30 PM</b>	Introduction to Fortran
<b>Feb 15</b>	<b>1:30 to 4:00 PM</b>	Introduction to Julia
<b>Feb 22</b>	<b>10:00 AM to 12:30 PM</b>	Introduction to MATLAB
<b>Feb 22</b>	<b>1:30 to 4:00 PM</b>	Introduction to the MATLAB Parallel Toolbox
<b>Mar 1</b>	<b>10:00 AM to 12:30 PM</b>	Introduction to R
<b>Mar 1</b>	<b>1:30 to 4:00 PM</b>	Introduction to Perl
<b>Mar 8</b>	<b>10:00 AM to 12:30 PM</b>	Introduction to Python for MATLAB Users
<b>Mar 8</b>	<b>1:30 to 4:00 PM</b>	Introduction to Spark

Check our webpage for additional courses

<https://hprc.tamu.edu/training>

