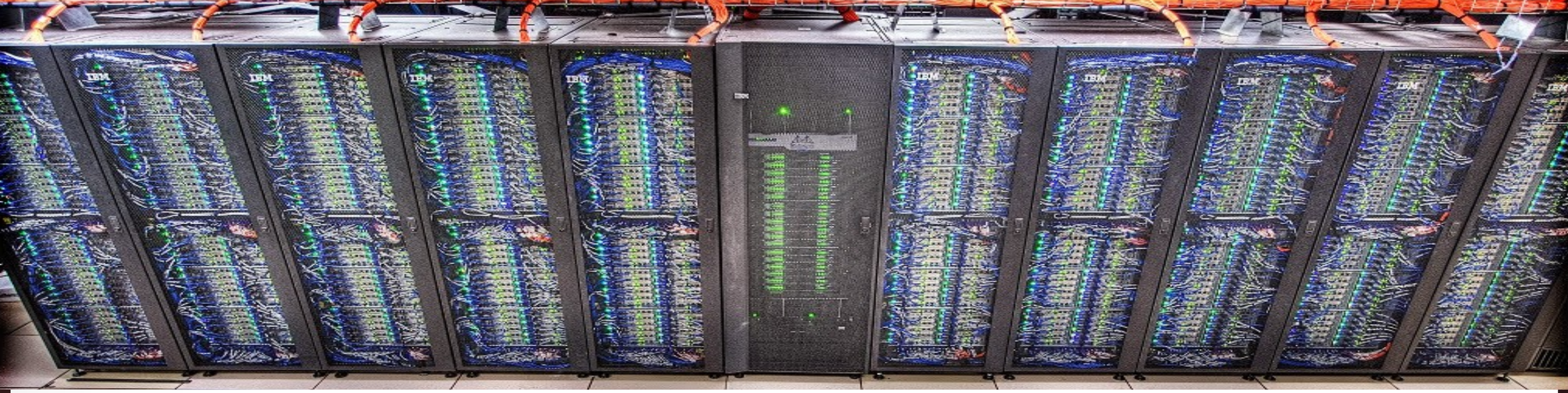


Introduction to Using the Ada Cluster

T. Mark Huang

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

HPRC Short Course – Fall 2017

Outline

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

Introduction

- Prerequisites:
  - Basic knowledge of UNIX/Linux
  - Slides from our UNIX/Linux short course are at:

    https://hprc.tamu.edu/training/intro_ada.html
- Examples:
  - Available in /scratch/training/Intro-to-ada directory
  - Copy these files to your scratch directory!!!

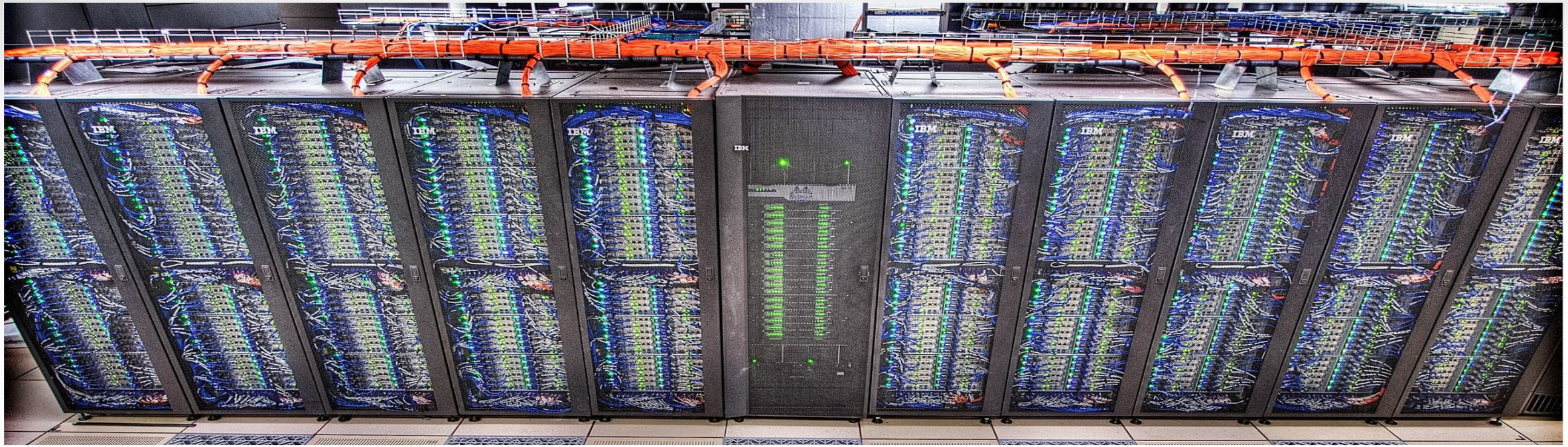
```
cp -r /scratch/training/Intro-to-ada $SCRATCH/
```

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/wiki/index.php/Ada:Policies

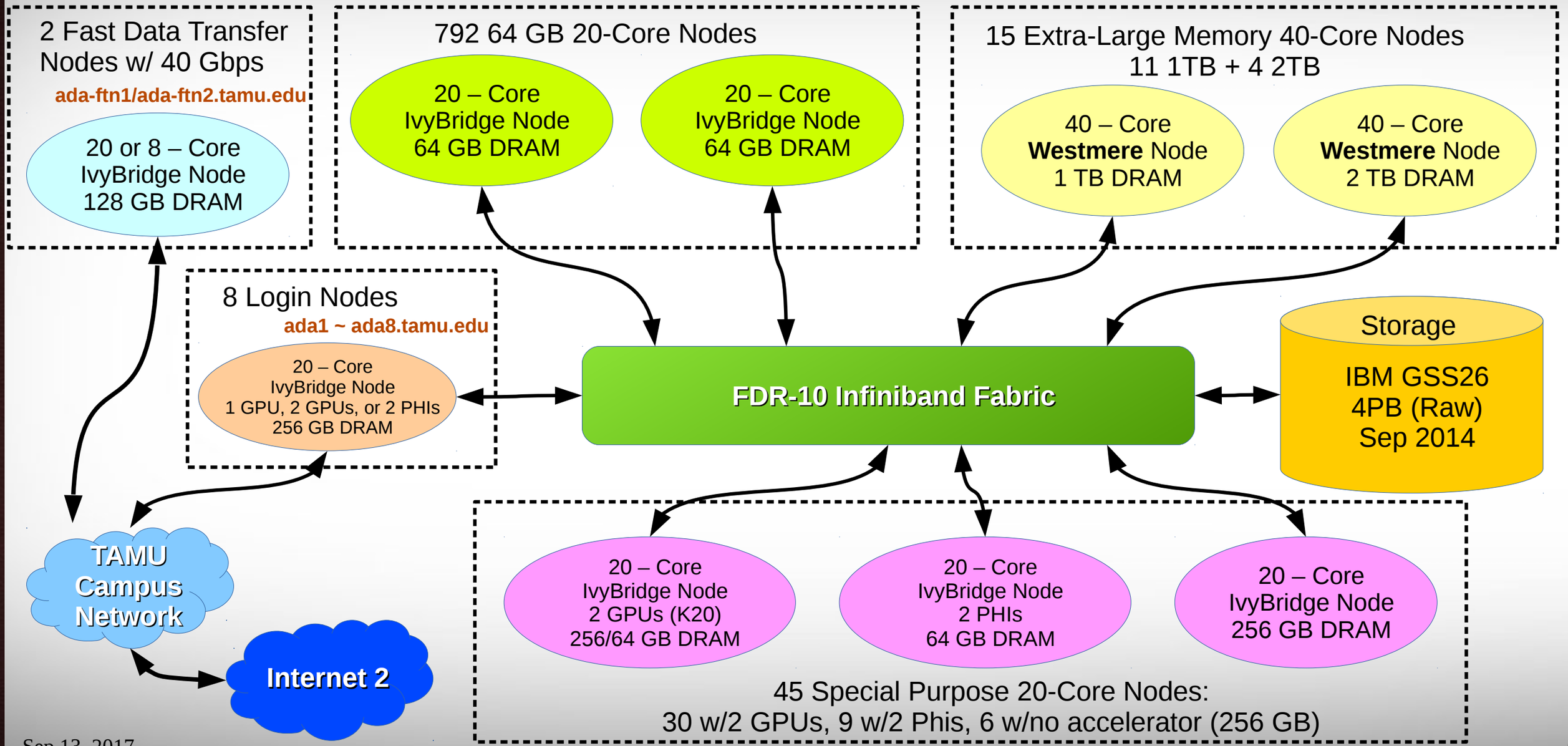
Ada – an x86 Cluster



A 17,500-core, 860-node cluster with:

- **837** 20-core compute nodes with two Intel 10-core 2.5GHz *IvyBridge* processors.
  - Among these nodes, 30 nodes have 2 GPUs (*K20*) each and 9 nodes have 2 *Phi* coprocessors.
- **15** compute nodes are 1TB and 2TB memory, 4-processor SMPs with the Intel 10-core 2.26GHz Westmere processor.
- **8** 20-core login nodes with two Intel 10-core 2.5GHz *IvyBridge* processors and 1 GPU, 2 GPUs, or 2 *Phi* coprocessors
- Nodes are interconnected with FDR-10 InfiniBand fabric in a two-level (core switch shown above in middle rack and leaf switches in each compute rack) fat-tree topology.

Ada Schematic: 17,500-core 860-node Cluster



Accessing Ada

- SSH is required for accessing Ada:
  - On campus: ***ssh NetID@ada.tamu.edu***
  - Off campus:
    - Set up VPN: u.tamu.edu/VPnetwork
    - Then: ***ssh NetID@ada.tamu.edu***
- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11)
  - PuTTY SSH
- Ada has 8 login nodes. Check the bash prompt.

```
NetID@ada1 ~]$
```

- 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 if you need assistance installing software.

https://hprc.tamu.edu/wiki/index.php/HPRC:Access

File Transfers with Ada

- Simple File Transfers:
  - scp: command line   (Linux, MacOS)
  - rsync: command line (Linux, MacOS)
  - MobaXterm: GUI (Windows)
  - WinSCP: GUI (Windows)
  - FileZilla: GUI (Windows, MacOS, Linux)
- Bulk data transfers:
  - Use fast transfer nodes (FTN; ada-ftn1/ada-ftn2) with:
    - Globus Connect (https://hprc.tamu.edu/wiki/index.php/SW:GlobusConnect)
    - GridFTP

https://hprc.tamu.edu/wiki/index.php/Ada:Filesystems_and_Files#Transfering_Files

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

- View usage and quota limits: the ***`showquota`*** command
- Also, only home directories are backed up daily.
- Quota and file limit increases will only be considered for scratch and tiered directories
- Do not share your home/scratch/tiered directories. Request a group directory for sharing files.

https://hprc.tamu.edu/wiki/index.php/Ada:Filesystems_and_Files

Computing Environment

Try "`echo $PATH`"

- Paths:
  - $PATH: for commands (eg. /bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/netid/bin)
  - $LD_LIBRARY_PATH: for libraries
- Many applications, many versions, and many paths
  ..... How do you manage all these software?!
- The solution: ***module*** (lmod)
  - Each version of an application, library, etc. is available as a module.
  - Module names have the format of package_name/version.

https://hprc.tamu.edu/wiki/index.php/Ada:Computing_Environment#Modules

Application Modules

- Installed applications are available as modules which are available to all users *(except for restricted modules)*
- **`module`** commands
  - **`module avail`** `#show all available modules`
  - **`module spider`** `tool_name` `#search all modules`
  - **`module key`** `genomics` `#search with keyword`
  - **`module load`** `tool_name` `#load a specific module`
  - **`module list`** `#list loaded modules`
  - **`module purge`** `#unload all loaded modules`
  - **`module load`** `Stacks` `#load the default version of a tool`
  - **`module load`** `Stacks/1.37-intel-2015B` `#load a specific version` ***(recommended way)***
- It's a good habit to purge unused modules before loading new modules.
- **Avoid loading modules in *.bashrc***

https://hprc.tamu.edu/wiki/index.php/Ada:Computing_Environment#Modules

Software

- Search module first:
  - ***module avail***
  - ***module spider software_name***
- Check Software wiki page (https://hprc.tamu.edu/wiki/index.php/SW) for instructions and examples
- License-restricted software: contact license owner for approval
- Contact us for software installation help/request

Development Environment - Toolchains

- Intel toolchain (eg. software stack) is recommended, which includes:
  - Intel C/C++/Fortran compilers
  - Intel Math Kernel Library
  - Intel MPI library
- Intel toolchain modules are named intel/version
- To load/use the current recommended Intel toolchain module (as Jan 2017):

  module load intel/2017A

- For applications which must use gcc/g++, run ***module spider GCC*** to find available versions.

https://hprc.tamu.edu/wiki/index.php/Ada:Compile:All#Getting_Started

Modules and Toolchains

- Use the same toolchains in your job scripts
  - The **`intel-2017A`** is the recommended toolchain (**`intel-2015B`** for bio software)

```
module load Bowtie2/2.2.6-intel-2015B
module load TopHat/2.1.0-intel-2015B
module load Cufflinks/2.2.1-intel-2015B
```

- Avoid mixing tool chains if loading multiple modules in the same job script:

```
module load Bowtie2/2.2.2-ictce-6.3.5
module load TopHat/2.0.14-goolf-1.7.20
module load Cufflinks/2.2.1-intel-2015B
```

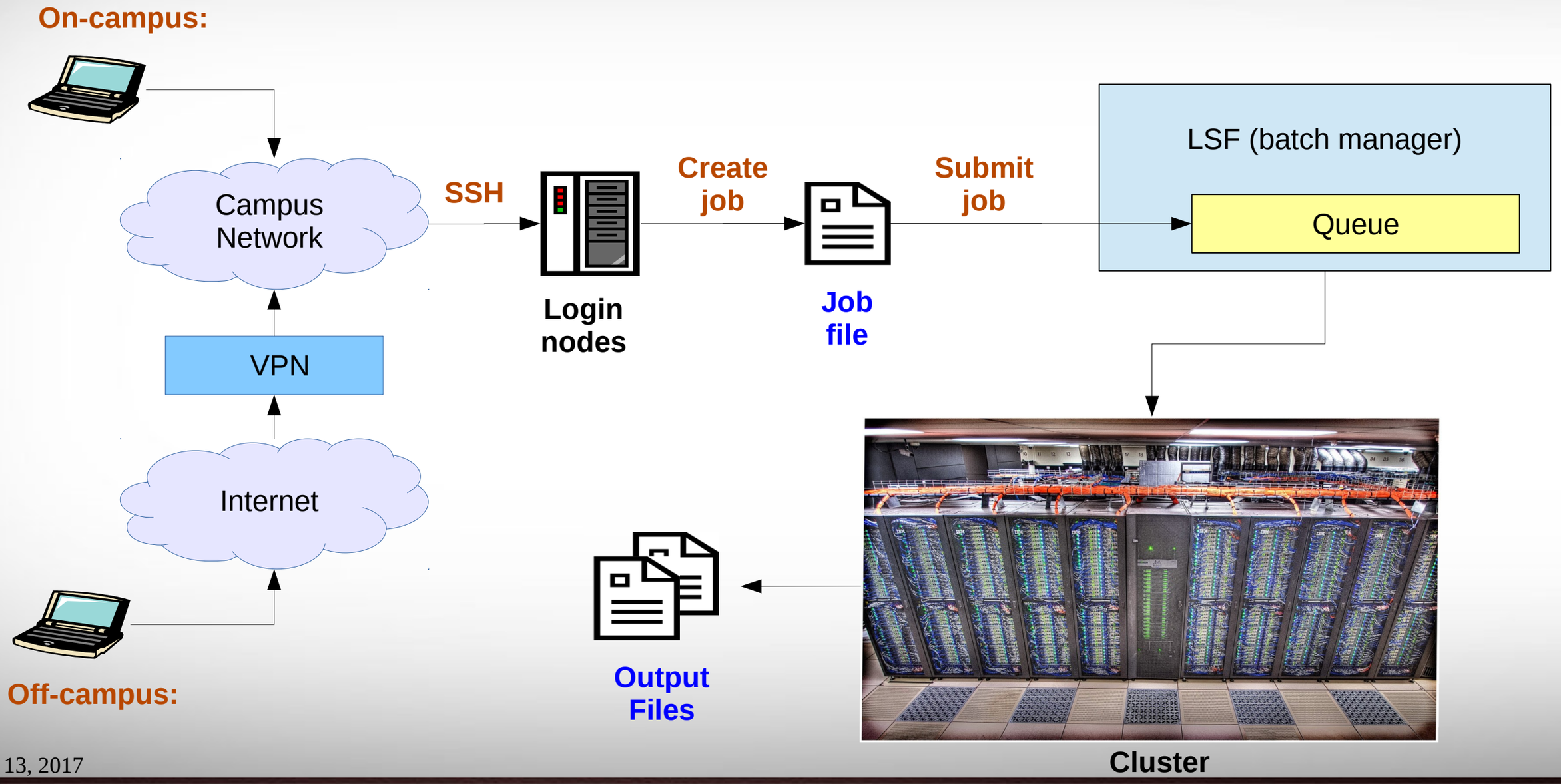
- Same rule applies to compilers and libraries.

Development Environment: Compilers

- The commands to invoke each compiler are:
  - ***icc*** for C
  - ***icpc*** for C++
  - ***ifort*** for Fortran
- Man pages (documentation) are available for each compiler:
  - ***man icc***
- Help for compiler options also available with *-help* option.
  - Also organized by categories (see ***icc -help help*** for more information).

https://hprc.tamu.edu/wiki/index.php/Ada:Compile:All#Getting_Started

Batch Computing on Ada



Batch Queues

- Job submissions are assigned to batch queues based on the resources requested (number of cores/nodes and wall-clock limit)
- Some jobs can be directly submitted to a queue:
  - If the 1TB or 2TB nodes are needed, use the xlarge queue (via ***`#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.

Current Queues

$ 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	-	-	-	-	2124	0	2124	0
xlarge	100	Open:Active	-	-	-	-	240	160	80	0
vnc	90	Open:Active	-	-	-	-	4	0	4	0
sn_short	80	Open:Active	-	-	-	-	20	0	20	0
mn_short	80	Open:Active	2000	-	-	-	0	0	0	0
mn_large	80	Open:Active	5000	-	-	-	0	0	0	0
general	50	Closed:Inact	0	-	-	-	0	0	0	0
sn_regular	50	Open:Active	-	-	-	-	1224	108	1116	0
sn_long	50	Open:Active	-	-	-	-	3599	290	3309	0
sn_xlong	50	Open:Active	-	-	-	-	56	10	46	0
mn_small	50	Open:Active	6000	-	-	-	5369	1320	4049	0
mn_medium	50	Open:Active	6000	-	-	-	6160	1240	4920	0
curie_devel	40	Open:Active	32	32	-	-	0	0	0	0
curie_medium	35	Open:Active	512	192	-	-	1328	1136	192	0
curie_long	30	Open:Active	192	64	-	-	2448	2256	192	0
curie_general	25	Closed:Inact	0	-	-	-	0	0	0	0
preempt_medium	10	Open:Active	-	-	-	-	0	0	0	0
low_priority	1	Open:Active	2500	500	-	-	0	0	0	0
preempt_low	1	Open:Active	40	-	-	-	0	0	0	0

https://hprc.tamu.edu/wiki/index.php/Ada:Batch_Queues

Queue Limits

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 **6000** cores for all running jobs in the single-node (sn_*) queues.	Maximum of **1000 cores and 50 jobs per user** for all running jobs in the single node (sn_*) queues.	For jobs needing **only one compute node**.
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 **2000** cores for all running jobs in this queue.	Maximum of **12000** cores for all running jobs in the multi-node (mn_*) queues.	Maximum of **3000 cores and 150 jobs per user** for all running jobs in the multi-node (mn_*) queues.	For jobs needing **more than one compute node**.
mn_small	2 / 2 / 120	1 hr / **7 days**		Maximum of **6000** cores for all running jobs in this queue.			
mn_medium	121 / 121 / 600	1 hr / **7 days**		Maximum of **6000** cores for all running jobs in this queue.			
mn_large	600 / 601 / 2000	1 hr / **5 days**		Maximum of **5000** cores for all running jobs in this queue.			
xlarge	1 / 1 / 280	1 hr / **10 days**	1 TB nodes (11) 2 TB nodes (4)				For jobs needing **more than 256GB of memory per compute node**.
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.

Consumable Computing Resources

- Resources specified in a job file:
  - Processor cores
  - Memory
  - Wall time
  - GPU
- Service Unit (SU) - Billing Account
- Other resources:
  - Software license/token
    - Use "**`license_status`**" to query

      https://hprc.tamu.edu/wiki/index.php/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
```

Sample Job Script (structure)

```
#BSUB -L /bin/bash
#BSUB -J blastx
#BSUB -n 2
#BSUB -R "span[ptile=2]"
#BSUB -R "rusage[mem=1000]"
#BSUB -M 1000
#BSUB -W 2:00
#BSUB -o stdout.%J
#BSUB -e stderr.%J

module load BLAST+/2.2.31-intel-2015B-Python-3.4.3

echo "BLAST manual: http://www.ncbi.nlm.nih.gov/books/NBK279690/"

#blastx: search protein databases using a translated nucleotide query

blastx -query mrna_seqs_nt.fasta -db /scratch/datasets/blast/nr \
-outfmt 10 -out mrna_seqs_nt_blastout.csv
```

These parameters are read by the job scheduler

Load the required module(s) first

Add a comment to the output

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

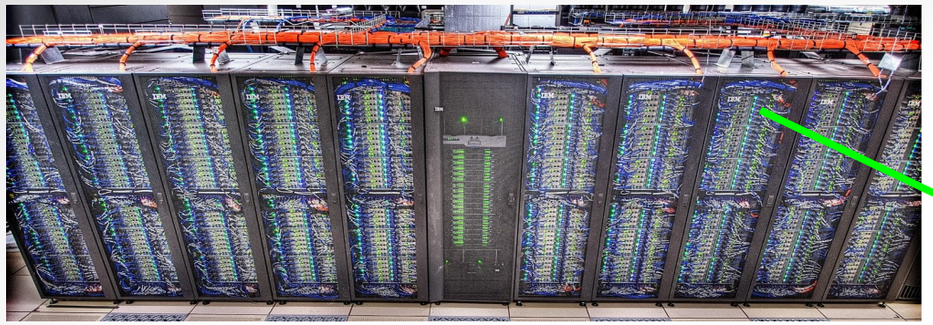
This is the command to run the application

This means the command is continued on the next line;
The space before the \ is required. Do not put a space after the \

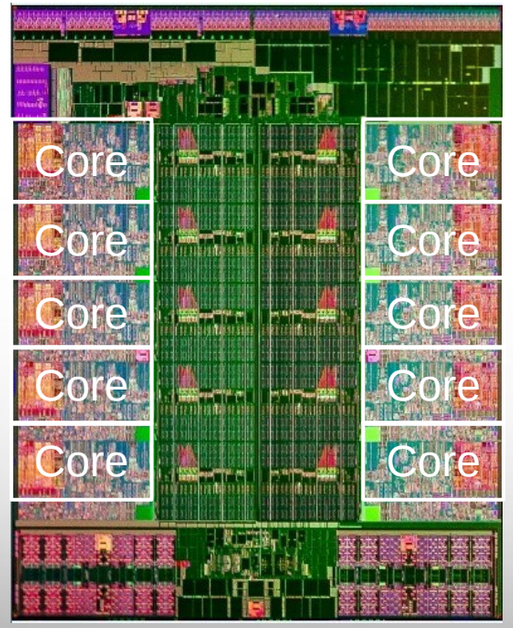
Important Job Parameters

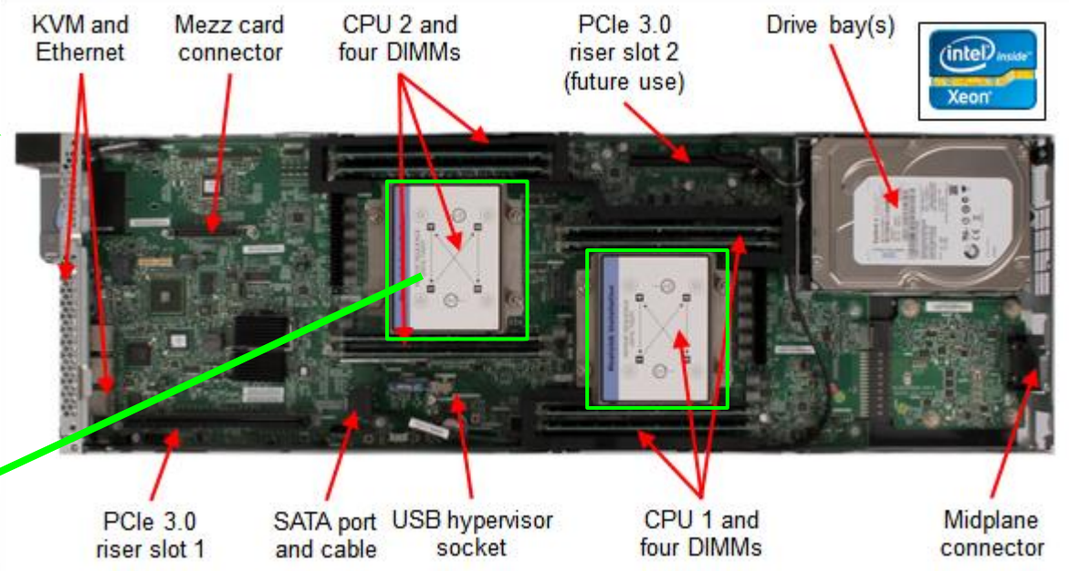
```
#BSUB -n NNN
      # NNN: total number of cores or job slots to allocate for the job
#BSUB -R "span[ptile=XX]"
      # XX:  number of cores or job slots per node to use
#BSUB -R "rusage[mem=nnn]"
      # reserves nnn MBs per core or job slot for the job
#BSUB -M nnn
      # enforces (XX cores * nnn MB) as memory limit
      # per node for the job
#BSUB -W hh:mm or mm
      # sets job's runtime wall-clock limit in hours:minutes or just minutes
```

Node / Socket / Core



Part of Ada cluster.
Each blue light is a node.

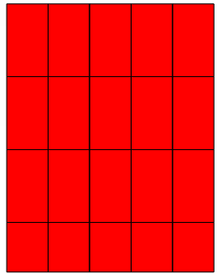




Each node has 2 sockets.

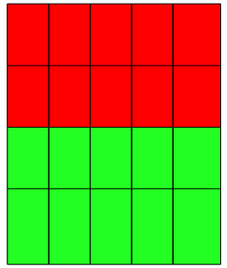
Each socket/CPU has 10 processor cores.
So, each node has 20 processor cores.

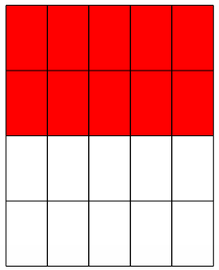
Processor Cores Mapping



20 cores
on 1 node

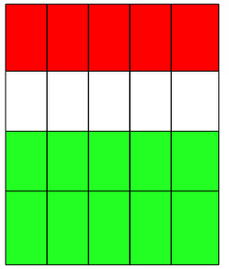
#BSUB -n 20
#BSUB -R “span[ptile=20]

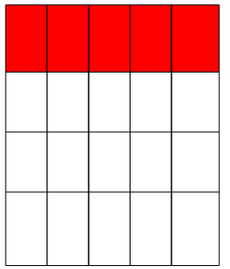


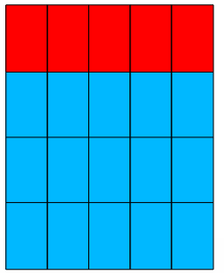


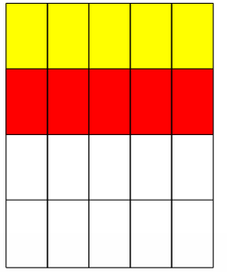
20 cores on 2 nodes

#BSUB -n 20
#BSUB -R “span[ptile=10]









20 cores on 4 nodes

#BSUB -n 20
#BSUB -R “span[ptile=5]

Job Resource Examples (node vs memory)

#BSUB -n 10 -R “span[ptile=2]”

#BSUB -R “rusage[mem=500]” -M 500 ...

Requests 10 job slots (2 per node). The job will span 5 nodes. The job can use up to 1000 MB per node.

#BSUB -n 80 -R “span[ptile=20]”

#BSUB -R “rusage[mem=2500]” -M 2500

Request 4 whole nodes (80/20), not including the xlarge memory nodes. The job can use up to 50GB per node.

Job Memory Requests

- Must specify both parameters for requesting memory:

  *#BSUB **-R** "rusage[mem=process_alloc_size]"*

  *#BSUB **-M** process_size_limit*

- Default value of 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 -R "rusage[mem=12300]" or -R “select[mem256gb]”
  - If need up to 1 or 2 TB of memory or up to 40 cores:
    - use -R “select[mem1tb]” (40 cores) or -R “select[mem2tb]” with the -q xlarge option
    - The mem1tb and mem2tb nodes are accessible only via the *xlarge* queue.

Job Parameters Example
For Job Scripts on xlarge queue

```
#BSUB -L /bin/bash                  # use the bash login shell
#BSUB -J stacks_S2                  # job name
#BSUB -n 40                         # assigns 40 cores for execution
#BSUB -R "span[ptile=40]"           # assigns 40 cores per node
#BSUB -q xlarge                     # required if using mem1tb or mem2tb
#BSUB -R "rusage[mem=25000]"        # reserves 25GB memory per core
#BSUB -M 25000                      # sets to 25GB process limit
#BSUB -W 48:00                      # sets to 48 hours the job's limit
#BSUB -o stdout.%J                  # job standard output to stdout.jobid
#BSUB -e stderr.%J                  # job standard error to stderr.jobid
```

Specified xlarge queue to use 1TB or 2TB memory nodes

Job File (Serial Example)

```
##NECESSARY JOB SPECIFICATIONS
#BSUB -J ExampleJob1                    #Set the job name to "ExampleJob1"
#BSUB -L /bin/bash                      #Uses the 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
#BSUB -R "span[ptile=1]"                #Request 1 core per node.
#BSUB -R "rusage[mem=5000]"             #Request 5000MB per process (CPU) for the job
#BSUB -M 5000                           #Set the per process enforceable memory limit to 5000MB.
#BSUB -o Example1Out.%J                 #Send stdout and stderr to "Example1Out.[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)
#First Executable Line
module load intel/2017A                 # loads the Intel software tool chain
prog.exe < input1 >& data_out1          # both input1 and data_out1 reside in the job submission dir
```

https://hprc.tamu.edu/wiki/index.php/Ada:Batch#Job_File_Examples

Job File (multi core, single node)

```
##NECESSARY JOB SPECIFICATIONS
#BSUB -J ExampleJob2          #Set the job name to "ExampleJob2"
#BSUB -L /bin/bash            #Uses the bash login shell to initialize the job's execution environment.
#BSUB -W 6:30                 #Set the wall clock limit to 6hr and 30min
#BSUB -n 10                   #Request 10 cores
#BSUB -R "span[ptile=10]"     #Request 10 cores per node.
#BSUB -R "rusage[mem=2560]"   #Request 2560MB per process (CPU) for the job
#BSUB -M 2560                 #Set the per process enforceable memory limit to 2560MB.
#BSUB -o Example2Out.%J       #Send stdout and stderr to "Example2Out.[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)
#First Executable Line
module load intel/2017A       # load intel module
./my_multicore_prog.exe       # run your program
```

https://hprc.tamu.edu/wiki/index.php/Ada:Batch#Job_File_Examples

Job File (multi core, multi node)

```
##NECESSARY JOB SPECIFICATIONS
#BSUB -J ExampleJob3                  #Set the job name to "ExampleJob3"
#BSUB -L /bin/bash                    #Uses the bash login shell to initialize the job's execution environment.
#BSUB -W 24:00                        #Set the wall clock limit to 24hr
#BSUB -n 40                           #Request 40 cores
#BSUB -R "span[ptile=20]"             #Request 20 cores per node.
#BSUB -R "rusage[mem=2560]"           #Request 2560MB per process (CPU) for the job
#BSUB -M 2560                         #Set the per process enforceable memory limit to 2560MB.
#BSUB -o Example3Out.%J               #Send stdout and stderr to "Example3Out.[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)
#First Executable Line
module load intel/2017A               # load intel module
./my_multicore_multinode_prog.exe     # run your program
```

https://hprc.tamu.edu/wiki/index.php/Ada:Batch#Job_File_Examples

Job File (serial GPU)

```
##NECESSARY JOB SPECIFICATIONS
#BSUB -J ExampleJob4                  #Set the job name to "ExampleJob4"
#BSUB -L /bin/bash                    #Uses the 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 cores
#BSUB -R "span[ptile=1]"              #Request 1 core per node.
#BSUB -R "rusage[mem=2560]"           #Request 2560MB per process (CPU) for the job
#BSUB -M 2560                         #Set the per process enforceable memory limit to 2560MB.
#BSUB -o Example4Out.%J               #Send stdout and stderr to "Example4Out.[jobID]"
#BSUB -R "select[gpu]"                #Request a node with a GPU
##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)
#First Executable Line
module load CUDA                      # load CUDA module
./my_CUDA_prog.exe                    # run your program
```

https://hprc.tamu.edu/wiki/index.php/Ada:Batch#Job_File_Examples

OpenMP Jobs

- Must set ***OMP_NUM_THREADS*** to take advantage of the requested cores
- All processes run on the same node.
  - Submit to the xlarge queue if you need up to 40 cores per node
- Example job:

```
#BSUB -n 20 -R 'rusage[mem=300] span[ptile=20]' -M 300
#BSUB -J omp_helloWorld
#BSUB -o omp_helloWorld.%J -L /bin/bash -W 1:00
module load intel/2015B
ifort -openmp -o omp_helloWorld.exe omp_helloWorld.f90
export OMP_NUM_THREADS=20
./omp_helloWorld.exe
```

MPI Jobs

- MPI programs may be run in batch jobs on multiple nodes
- Note, the mpiexec -np option must match the number of cores requested by the job (#BSUB -n option).

```
#BSUB -n 12 -R 'rusage[mem=150] span[ptile=4]' -M 150
#BSUB -J mpi_helloWorld -o mpi_helloWorld.%J
#BSUB -L /bin/bash -W 1:00
#
module load intel/2015B
mpiifort -o mpi_helloWorld.exe mpi_helloWorld.f90
mpiexec.hydra -np 12 ./mpi_helloWorld.exe
```

Pop Quiz #1

```
#BSUB -L /bin/bash
#BSUB -J stacks_S2
#BSUB -n 10
#BSUB -R "span[ptile=10]"
#BSUB -R "rusage[mem=2000]"
#BSUB -M 2000
#BSUB -W 36:00
#BSUB -o stdout.%J
#BSUB -e stderr.%J
```

- How much total memory is requested for this job?

- What is the maximum time this job is allowed to run?

Pop Quiz #2

```
#BSUB -L /bin/bash
#BSUB -J stacks_S2
#BSUB -n 80
#BSUB -R "span[ptile=80]"
#BSUB -R "rusage[mem=50000]"
#BSUB -M 50000
#BSUB -W 48:00
#BSUB -o stdout.%J
#BSUB -e stderr.%J
```

- Find two parameters that are either missing or not configured correctly.

Submit the Job and Check Status

- Submit your job to the job scheduler

```
bsub < sample01.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>.
```

- Summary of the status of your running/pending jobs

```
bjobs
```

```
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
```

- A more detailed summary of a running job

```
bjobs -l 2470599
```

Try yourself; copy examples: *cp -r /scratch/training/Intro-to-ada $SCRATCH/*

Debug job failures

- Debug job failures using the stdout and stderr files
  - `cat output.ex03.python_mem.2447336`

This job id was created by the parameter in your job script file
`#BSUB -o output.ex03.python_mem.%J`

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

Resource usage summary:

    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
```

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

Check your Service Unit (SU) Balance

- Show the SU Balance of your Account(s)

```
myproject -l
```

```
==========================================================================
                  List of tmarkhuang's Project Accounts
--------------------------------------------------------------------------
|  Account    | Default | Allocation |Used & Pending SUs|    Balance   |
--------------------------------------------------------------------------
|082792010838|        N|    50000.00|            -10.38|    49989.62|
--------------------------------------------------------------------------
```

- Use "**#BSUB -P project_id**" to charge SU to a specific project
- Run "**myproject -d accountNo**" to change default project account
- Run "**myproject -h**" to see more options

https://hprc.tamu.edu/wiki/index.php/HPRC:AMS:Service_Unit
https://hprc.tamu.edu/wiki/index.php/HPRC:AMS:UI

Job submission issue (SU)

```
$ 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.
```

- Insufficient SU
  - Ask PI to transfer SU to you
  - Apply for more SU (if you are eligible, as a PI or permanent researcher)

See Accounts section in FAQ: https://hprc.tamu.edu/wiki/HPRC:CommonProblems#Accounts

Job Submission and Tracking

Command	Description
bsub < jobfile1	Submit jobfile1 to batch system
bjobs [-u all or user_name] [[-l] job_id]	List jobs
bpeek [-f] job_id	View job's output and error files
bkill job_id	Kill a job
bhist [-l] job_id	Show historical information about a job
lnu [-l] -j job_id	Show resource usage for a job
blimits -w	Show how policies are applied to users and queues

https://hprc.tamu.edu/wiki/index.php/Ada:Batch#Job_tracking_and_control_commands

Node Utilization: *lnu*

lnu [-h] [-l] -j jobid # lists on stdout the utilization across all nodes for an executing job.

Examples:

Run "*lnu -h*" to see more options

```
$ 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
================================================================================================
$ lnu -l -j 753454
Job           User                    Queue         Status Node  Cpus
753454        ajochoa                 long               R    1    20
   HOST_NAME   status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem  Assigned Cores
   nxt1222 (L)      ok   4.3   4.5   6.2  20%   0.0   0 54464  422M  4.7G 52.9G    20
================================================================================================
```

The utilization (**ut**) and memory paging (**pg**), overall, are probably the most significant. Note that the **tmp**, **swp**, and **mem** refer to available amounts respectively. See "*man lsload*" for explanations on labels.

https://hprc.tamu.edu/wiki/index.php/Ada:Batch#Job_tracking_and_control_commands

Job Environment Variables

- ***$LSB_JOBID*** = job id
- ***$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

Concurrent Program Execution in Jobs via Tamulauncher

- Useful for running many programs 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
- Preferred over LSF job arrays

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 parameter
- Insufficient memory specified in #BSUB -M and -R "rusage[mem=xxx]" parameters
- No matching resource (-R rusage[mem] too large)
- Running OpenMP jobs across nodes
- Insufficient SU: See your SU balance: **myproject -l**
- 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**

```
$ file jobfile.txt
jobfile.txt: ASCII text, with CRLF line terminators
$ dos2unix jobfile.txt
dos2unix: converting file jobfile.txt to UNIX format ...
$ file jobfile.txt
jobfile.txt: ASCII text
```

FAQ: https://hprc.tamu.edu/wiki/index.php/HPRC:CommonProblems

Need Help?

- Check the FAQ (https://hprc.tamu.edu/wiki/index.php/HPRC:CommonProblems) or the Ada User Guide (https://hprc.tamu.edu/wiki/index.php/Ada) for possible solutions first.
- Email your questions to **help@hprc.tamu.edu**. (Now 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?