Introduction to Using the HPRC Clusters

T. Mark Huang

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

HPRC Research Computing Week – June 2017

Outline

- Usage Policies
- Hardware Overview of Ada and Terra
- Accessing Ada / Terra
- 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/wiki/index.php/HPRC:SC:Unix
- 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 $SCRATCH/
  - For Terra:
    - Available in /scratch/training/Intro-to-terra directory
    - Copy these files to your scratch directory!!!

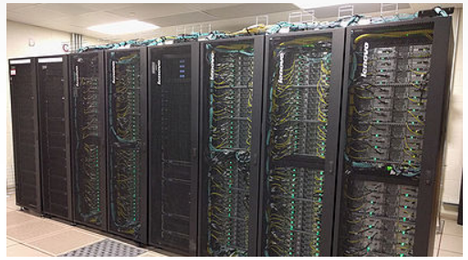
      cp -r /scratch/training/Intro-to-terra $SCRATCH/

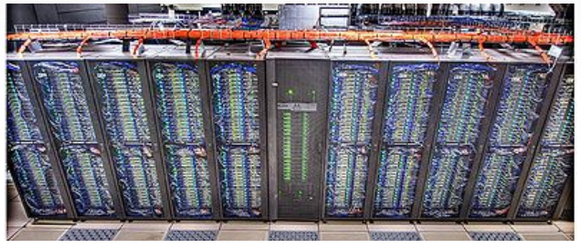
Usage Policies
(Be a good compute citizen)

- 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

HPRC Clusters

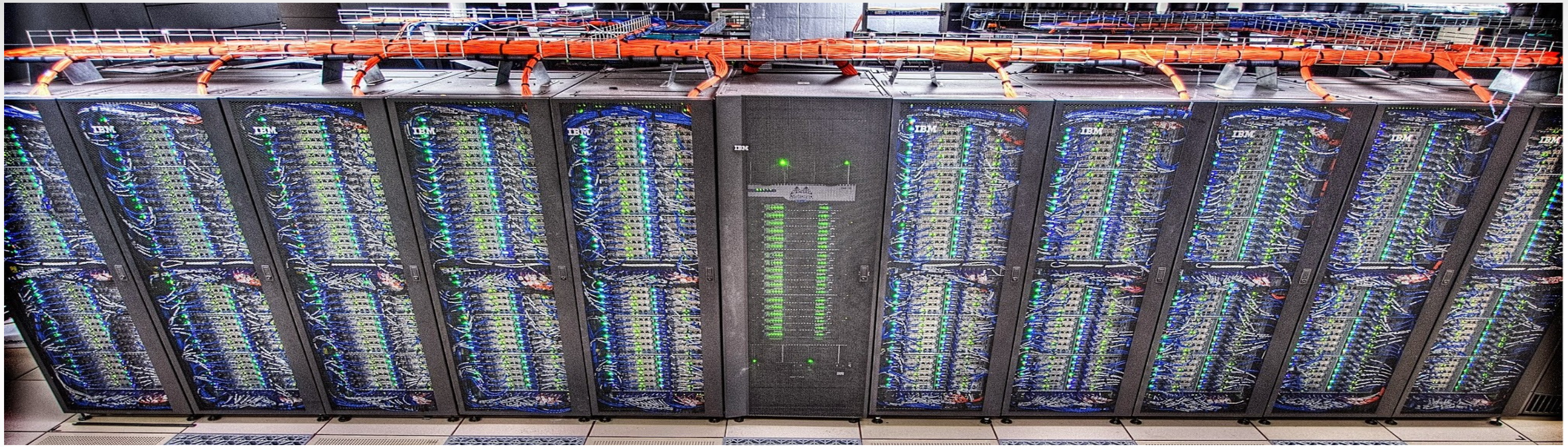




Cluster	Terra	Ada	Curie
Nodes/Cores	307 / 8,512	860 / 17,500	48 / 768
CPU Architecture	*x86_64* Intel 14-core 2.4GHz *Broadwell*	*x86_64* Intel 10-core 2.5 GHz *IvyBridge*	*ppc64* IBM 8-core 4.2 GHz *Power7+*
Interconnect	FDR-10 Infiniband	10 Gbps Ethernet	Onmi-Path
Accelerator	48 nodes w/ Tesla K80	30 nodes w/ 2 Tesla K20 9 nodes w/ 2 *Phi* coprocessors	N/A
Scheduler	Slurm	LSF	LSF (shared with Ada)
File System	GPFS; 3 PB raw	GPFS; 4 PB raw	GPFS (shared with Ada)
Production Date	Spring 2017	Sep 2014	May 2015

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

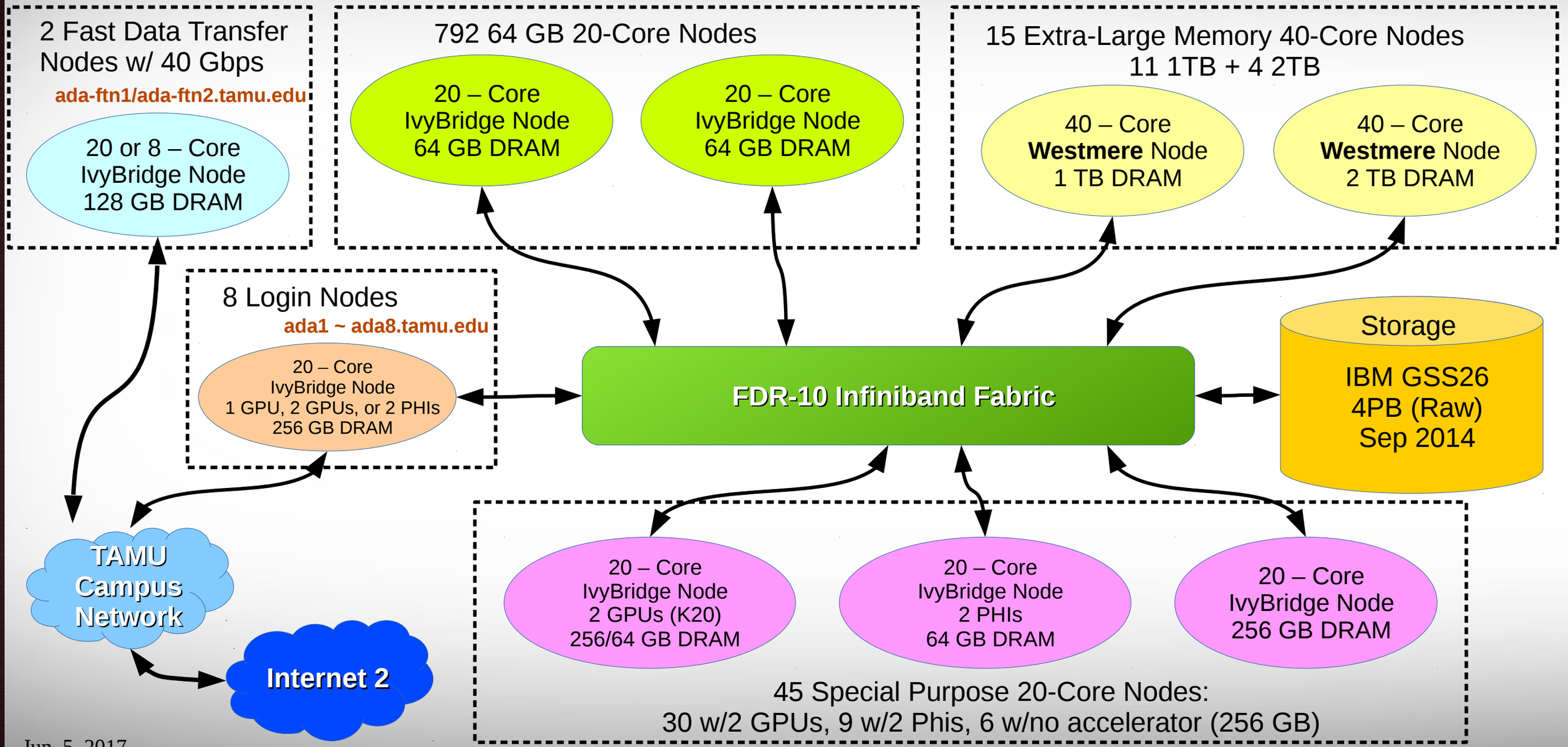
Ada – an x86 Cluster



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



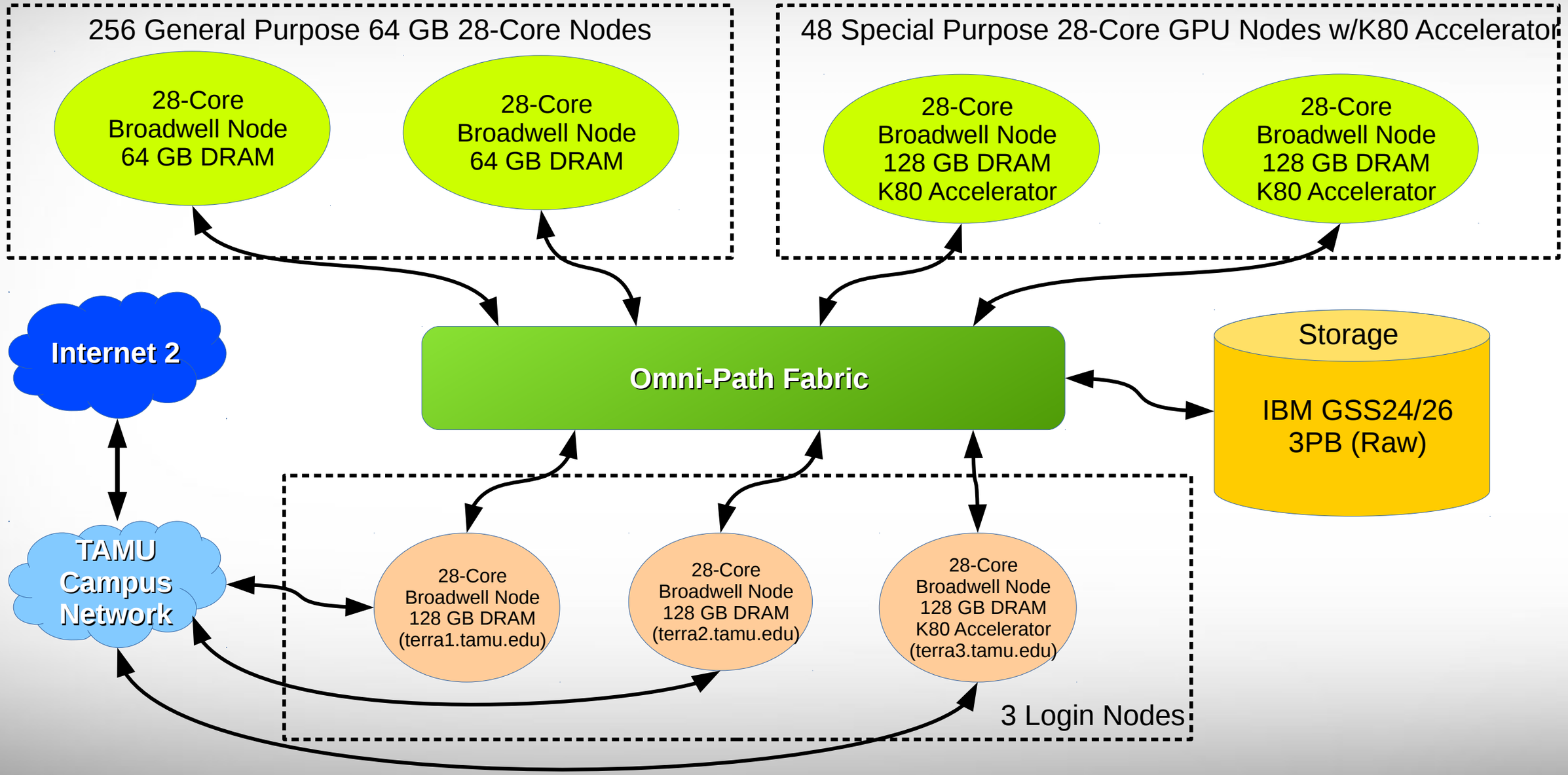
Terra – an x86 Cluster

A 8,512-core, 307-node cluster with:

- **256** 28-core compute nodes with two Intel 14-core 2.4GHz *Broadwell* processors and 64 GB of memory.
- **48** 28-core compute nodes with two Intel 14-core 2.4GHz *Broadwell* processors, 128 GB of memory, and one dual-GPU K80 accelerator.
- **3** 28-core login nodes with two Intel 14-core 2.4GHz *Broadwell* processors.
  - 1 login node has a dual-GPU K80 accelerator (terra3.tamu.edu).
- Nodes are interconnected with Omni-Path fabric in a two-level fat-tree topology.

https://hprc.tamu.edu/wiki/index.php/Terra:Intro

Terra Schematic: 8,512-core, 307-node Cluster



Accessing Ada / Terra

- SSH is required for accessing Ada / Terra:
  - On campus: ***ssh NetID@ada.tamu.edu*** or ***ssh NetID@terra.tamu.edu***
  - Off campus:
    - Set up VPN: 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

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

```
NetID@terra1 ~]$
```

- Ada has 8 login nodes. Terra has 3 login nodes Check the bash prompt.
- 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 / Terra

- 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 on Ada (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 (on Ada/Curie only)

- Home and scratch directories are not shared between Ada and Terra clusters.
- 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
https://hprc.tamu.edu/wiki/index.php/Terra: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
https://hprc.tamu.edu/wiki/index.php/Terra: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
https://hprc.tamu.edu/wiki/index.php/Terra: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 Jun 2017):

  Ada: ***module load intel/2015B***

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

Modules and Toolchains

- Use the same toolchains in your job scripts
  - The **`intel-2015B`** is the recommended toolchain

```
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
```

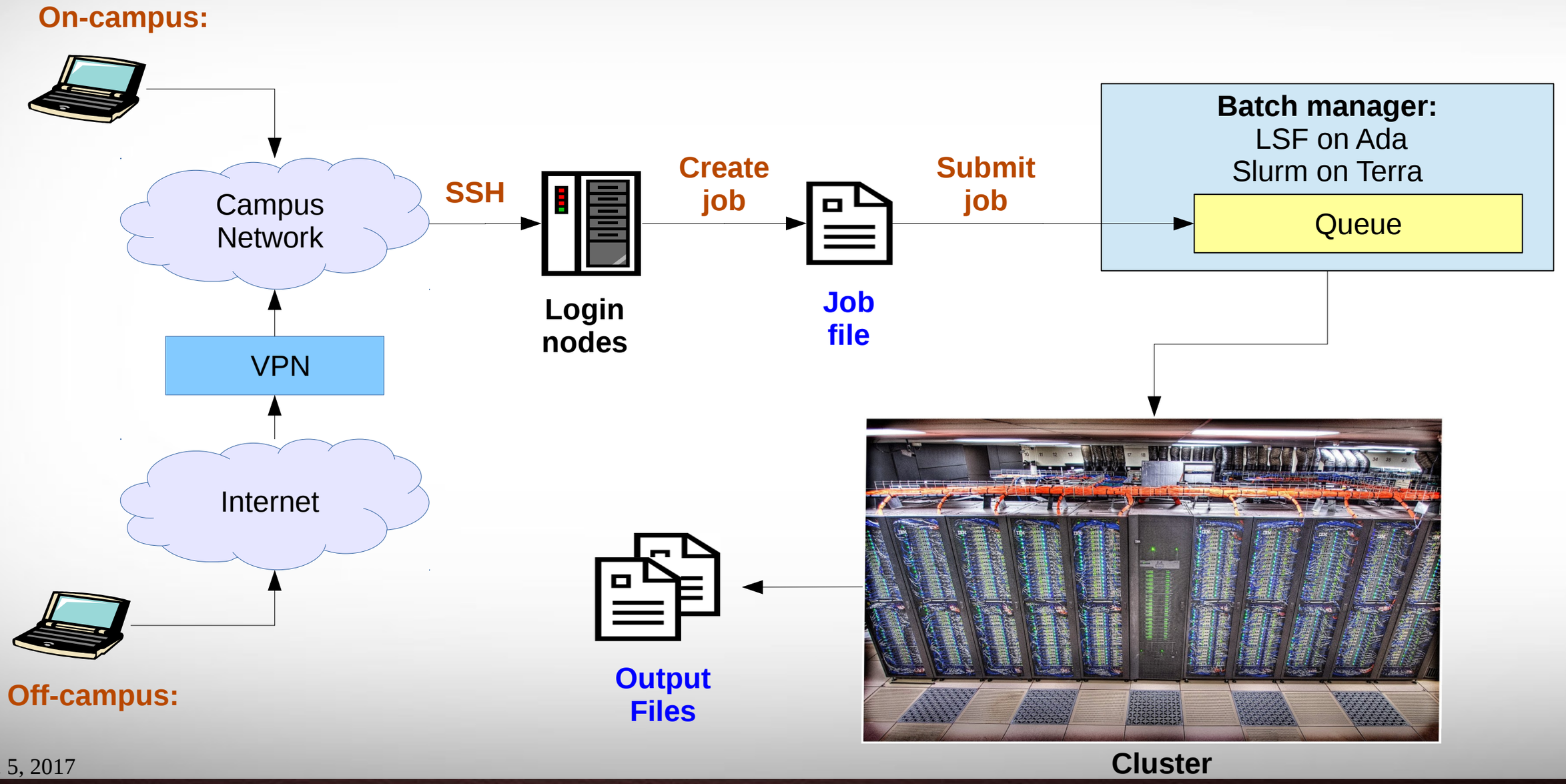
- Same rule applies to compilers and libraries.

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

Batch Computing on HPRC Clusters



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:
  - On Ada, 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.

https://hprc.tamu.edu/wiki/index.php/Ada:Batch_Queues
https://hprc.tamu.edu/wiki/index.php/Terra:Batch#Queues

Current Queues on Ada

$ 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 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 **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.

Current Queues on Terra

```
% sinfo
PARTITION  AVAIL  TIMELIMIT    JOB_SIZE    NODES(A/I/O/T)   CPUS(A/I/O/T)
short*     up     2:00:00      1-16        33/249/10/292    668/7228/280/8176
medium     up     1-00:00:00   1-64        33/249/10/292    668/7228/280/8176
long       up     7-00:00:00   1-32        33/249/10/292    668/7228/280/8176
gpu        up     2-00:00:00   1-48        0/48/0/48        0/1344/0/1344
vnc        up     12:00:00     1           0/48/0/48        0/1344/0/1344
```

- **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			
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

Batch Queue Policies also at:
https://hprc.tamu.edu/wiki/index.php/Terra:Batch#Queues

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/index.php/HPRC:AMS:Service_Unit

```
myproject -l
```

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

- 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)

```
#!/bin/bash

##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE
#SBATCH --get-user-env=L

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name JobExample1
#SBATCH --time 01:30:00
#SBATCH --ntasks 2
#SBATCH --ntasks-per-node=2
#SBATCH --mem=2048M
#SBATCH --output Example1Out.%j

# this intel toolchain is just an example.  recommended toolchain is TBD
module load intel/2016D

# run program
./myprogram
```

This script will use the bash shell.

These lines (directives) describe your job to the job scheduler

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

Load the required module(s) first

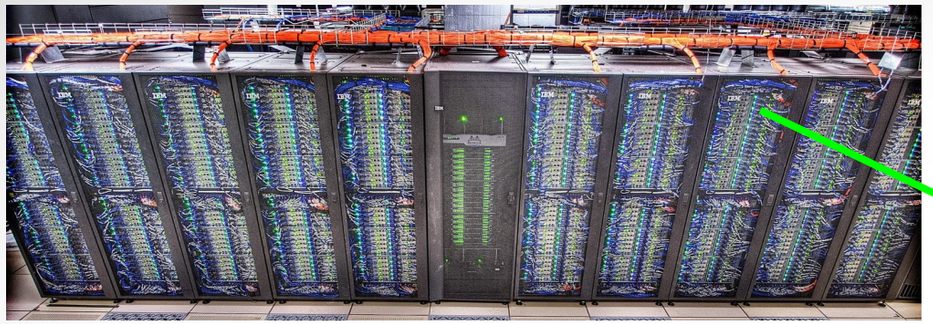
This is a command that is executed by the job

Important Job Parameters

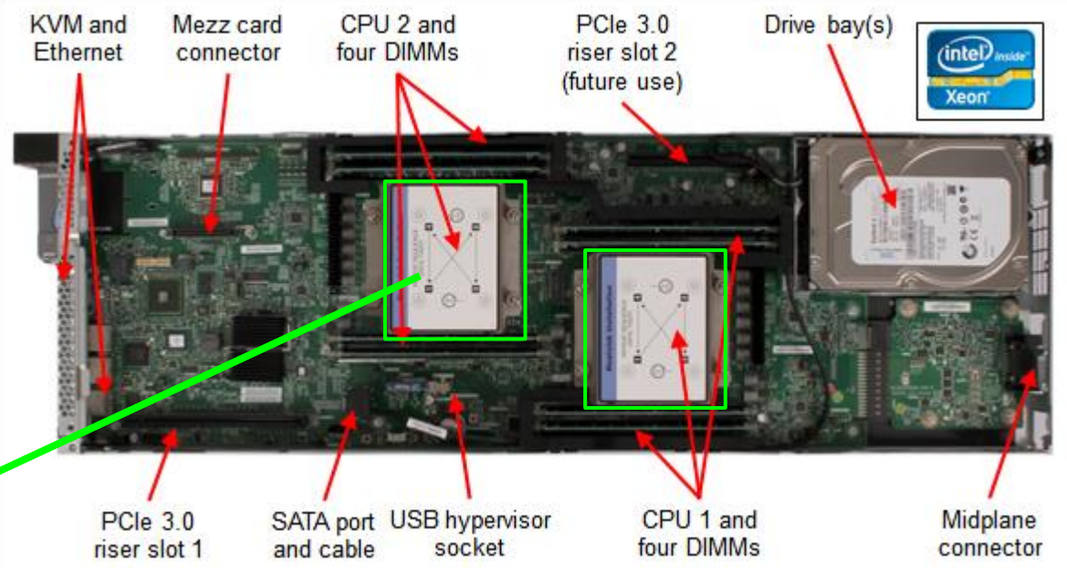
Ada	Terra	Comment
`#BSUB -L /bin/bash`	`#SBATCH --export=NONE` `#SBATCH --get-user-env=L`	Initialize job environment.
`#BSUB -W HH:MM or MM`	`#SBATCH --time HH:MM:SS`	Specifies the time limit for the job.
`#BSUB -n NNN`	`#SBATCH --ntasks NNN`	Total number of tasks for the job.
`#BSUB -R "span[ptile=XX]"`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

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

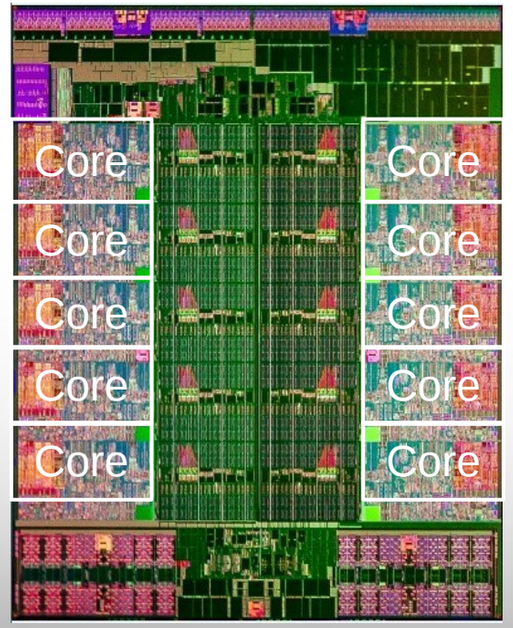
Ada Compute nodes



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



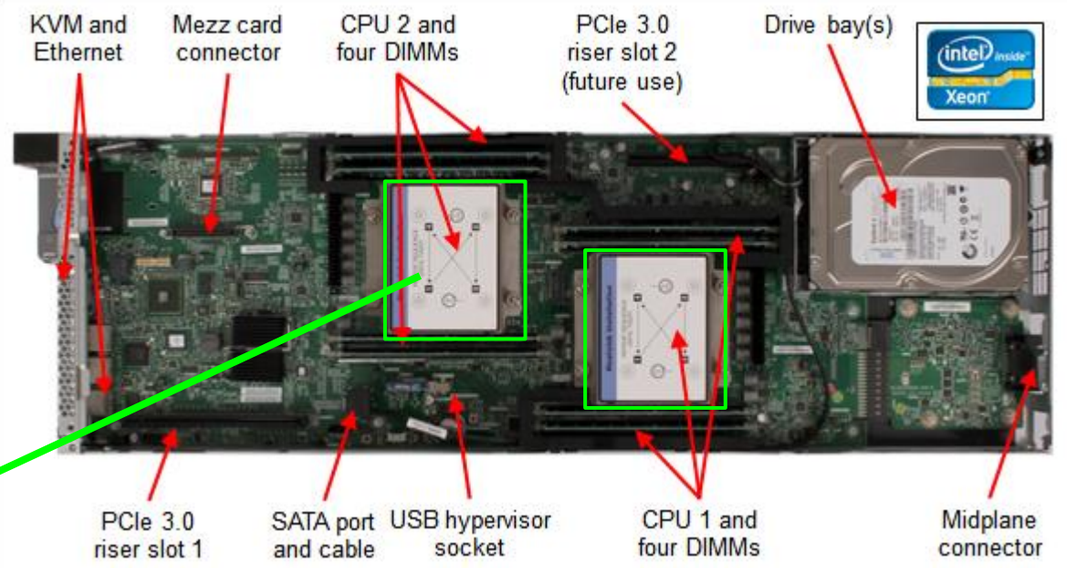
Each node has 2 sockets.



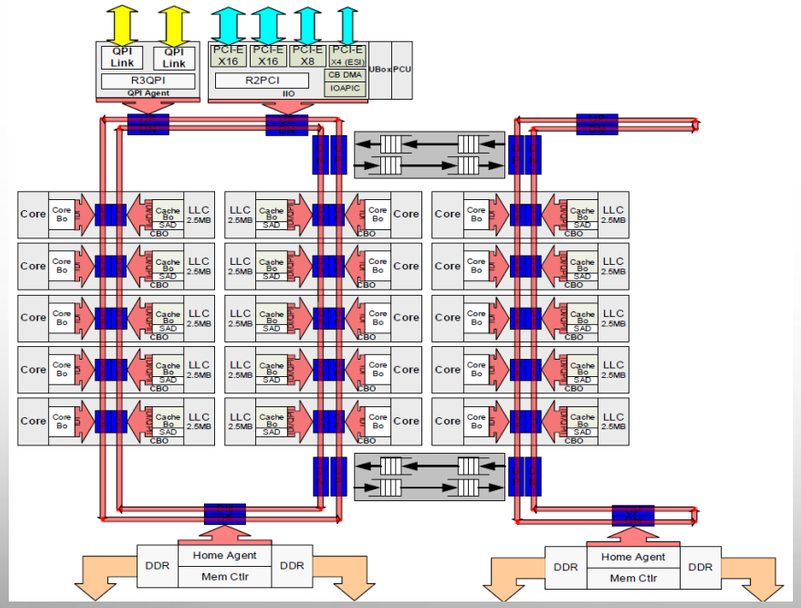
Each socket/CPU has 10 processor cores.
So, each node has 20 processor cores.

Terra Compute Nodes

Part of Terra cluster.
Each green light is a node.

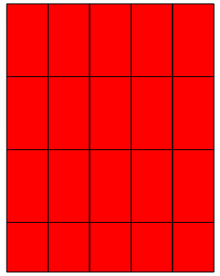


Each node has 2 sockets.



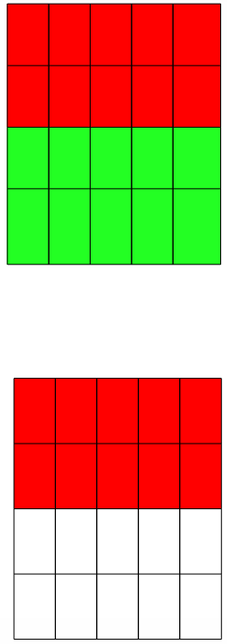
Each socket/CPU has 14 processor cores.
So, each node has 28 processor cores.

Mapping Jobs to Nodes on Ada



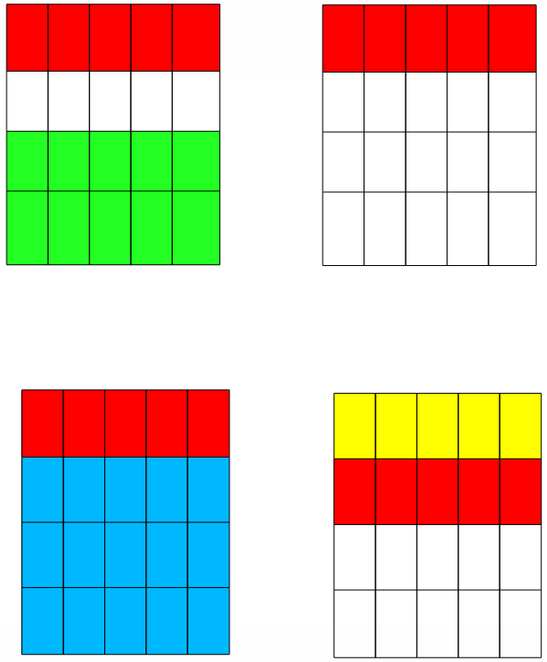
20 cores
on 1 node

#BSUB -n 20
#BSUB -R “span[ptile=20]



20 cores on 2 nodes

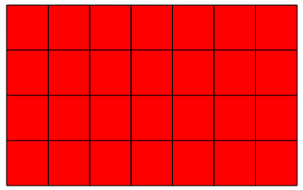
#BSUB -n 20
#BSUB -R “span[ptile=10]



20 cores on 4 nodes

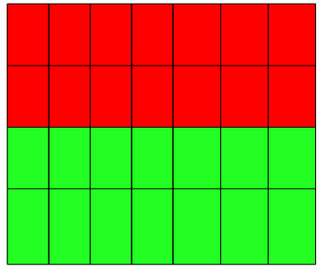
#BSUB -n 20
#BSUB -R “span[ptile=5]

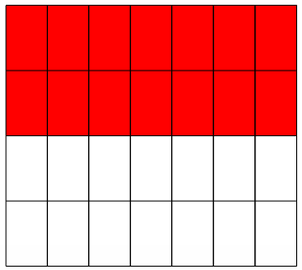
Mapping Jobs to Nodes on Terra



28 cores
on 1 node

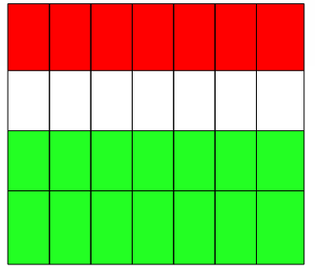
#SBATCH --ntasks 28
#SBATCH --tasks-per-node=28

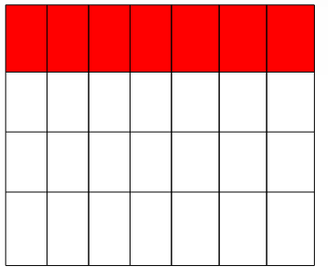


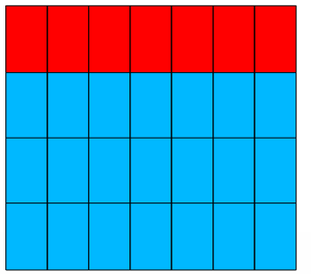


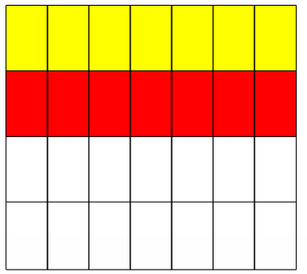
28 cores on 2 nodes

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









28 cores on 4 nodes

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

Job Resource Examples (node vs memory)

Terra

Requests 8 tasks (2 per node). The job will span 4 nodes. The job can use up to 4 GB per node.

```
#SBATCH --ntasks=8
#SBATCH --tasks-per-node=2
#SBATCH --mem=4096M
```

Request 4 whole nodes (112 cores, 28 cores per node). The job can use up to 56 GB per node.

```
#SBATCH –ntasks=112
#SBATCH --tasks-per-node=28
#SBATCH --mem=57344M
```

Ada

Requests 8 tasks (2 per node). The job will span 4 nodes. The job can use up to 4 GB per node (2048M per core).

```
#BSUB -n 8
#BSUB -R "span[ptile=2]"
#BSUB -R "rusage[mem=2048]" -M 2048
```

Request 4 whole nodes (80 cores, 20 cores per node). The job can use up to 54 GB per node (54*1000/20 = 2700M per core).

```
#BSUB -n 80
#BSUB -R "span[ptile=20]"
#BSUB -R "rusage[mem=2700]" -M 2700
```

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 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 Memory Requests on Terra

- Must use one of the following lines to request memory for your job:

#SBATCH --mem=XXXXM # memory per node in MB

#SBATCH --mem-per-cpu=XXXXM # memory per cpu in MB

- On 64GB nodes, usable memory is at most 56 GB. The per-process memory limit should not exceed 2048 MB for a 28-core job.
- On 128GB nodes, usable memory is at most 112 GB. The per-process memory limit should not exceed 4096 MB for a 28-core job.

Ada 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/2015B                 # loads the Intel software tool chain
prog.exe < input1 >& data_out1          # both input1 and data_out1 reside in the job submission dir
```

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
#SBATCH --mem=2560M                  #Request 2560MB (2.5GB) per node
#SBATCH --output=Example1Out.%j      #Send stdout/err to "Example1Out.[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

# this intel toolchain is just an example.  recommended toolchain is TBD
module load intel/2016D

# run program
./myprogram
```

Ada 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/2015B           # load intel module
./my_multicore_prog.exe           # run your program
```

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

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=8          #Request 8 tasks/cores per node
#SBATCH --mem=8G                     #Request 8GB per node
#SBATCH --output=Example2Out.%j      #Send stdout/err to "Example2Out.[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

# this intel toolchain is just an example.  recommended toolchain is TBD
module load intel/2016D

# run program
./my_multicore_program
```

Ada 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/2015B               # load intel module
./my_multicore_multinode_prog.exe     # run your program
```

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

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
#SBATCH --ntasks-per-node=2          #Request 2 tasks/cores per node
#SBATCH --mem=4096M                  #Request 4096MB (4GB) per node
#SBATCH --output=Example3Out.%j      #Send stdout/err to "Example3Out.[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

# this intel toolchain is just an example.  recommended toolchain is TBD
module load intel/2016D

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

Ada 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

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=01:30:00              #Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=1                   #Request 1 task
#SBATCH --mem=2560M                  #Request 2560MB (2.5GB) per node
#SBATCH --output=Example4Out.%j      #Send stdout/err to "Example4Out.[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
#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/2016D CUDA/8.0.44-unsupportedCC

# run program
./my_gpu_program
```

Other Type of Jobs

- MPI and OpenMP

  Introduction to parallel computing using MPI and OpenMP on Ada and Terra

  3:45 ~ 4:45 PM, June 5th

- Visualization:
  - https://hprc.tamu.edu/wiki/index.php/Ada:Remote-Viz
  - https://hprc.tamu.edu/wiki/index.php/HPRC:SC:Visualization_Portal
- Huge number of concurrent single core jobs
  - Check out ***tamulanucher*** https://hprc.tamu.edu/wiki/index.php/Ada: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

Job Submission and Tracking

Ada	Terra	Description
`bsub < jobfile1`	`sbatch jobfile1`	Submit jobfile1 to batch system
`bjobs [-u all or user_name] [[-l] job_id]`	`squeue [-u user_name] [-j job_id]`	List jobs
`bkill job_id`	`scancel job_id`	Kill a job
`bhist [-l] job_id`	`sacct -X -j job_id`	Show information for a job (can be running or finished)
	`sacct -X -S YYYY-HH-MM`	Show information for all of your jobs since YYYY-HH-MM
`lnu [-l] -j job_id`	`lnu job_id`	Show resource usage for a job

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

Submit the Job and Check Status

- Submit your job to the job scheduler

Ada: `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>.
```

Terra: `sbatch sample01.job`

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

- Summary of the status of your running/pending jobs

Ada: `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
```

Terra: `squeue -u $USER`

```
% 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]
```

- A more detailed summary of a running job on Ada: `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

Ada: `#BSUB -o output.ex03.python_mem.%J`

Terra: `#SBATCH -o output.ex03.python_mem.%j`

Ada:

```
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
```

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

- Show the SU Balance of your Account(s)

```
myproject -l
```

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

- To specify a project ID to charge in the job file
  - Ada: Use "**`#BSUB -P project_id`**"
  - Terra: Use "**`#SBATCH -A project_id`**"
- 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)

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

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

Node Utilization on Ada: *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

Node Utilization on Terra: *lnu*

lnu jobid # lists on stdout the CPU utilization and free memory across all nodes for an executing job.

Example:

```
% lnu 64033
JOBID  NAME     USER      PARTITION  NODES  CPUS  STATE    TIME    TIME_LEFT  START_TIME          REASON     NODELIST
64033  somejob  someuser  medium     4      112   RUNNING  4:42    15:18      2017-01-30T19:51:5 None       tnxt-[0401-0404]

HOSTNAMES    CPU_LOAD   FREE_MEM   MEMORY     CPUS(A/I/O/T)
tnxt-0401    24.17      36104      57344      28/0/0/28
tnxt-0402    25.78      33999      57344      28/0/0/28
tnxt-0403    26.29      36777      57344      28/0/0/28
tnxt-0404    25.36      36706      57344      28/0/0/28
```

Note: SLURM updates the node information every few minutes.

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
    - $TMPDIR is local to each assigned compute node for the job
- **Terra:**
  - ***$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
    - Local disk space is about 850GB
    - Use of $TMPDIR is recommended for jobs that use many small temporary files

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

Common Job Problems

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

- 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 -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`**

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) or Terra User Guide (https://hprc.tamu.edu/wiki/index.php/Terra) 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.

Upcoming Tutorials

Topics	Date/Time
Introduction to parallel computing using MPI and OpenMP on Ada and Terra	3:45 ~ 4:45 PM, Mon, Jun 5
Introduction to Python for scientific programming	9 AM ~ 12 PM, Tue, Jun 6
Data Literacy and Data Management	3:45 ~ 4:45 PM, Tue, Jun 6

See full list of activities of Research Computing Week at

- https://sites.google.com/a/tamu.edu/rcompweek/home

HIGH PERFORMANCE RESEARCH COMPUTING

TEXAS A&M UNIVERSITY

Thank you.

Any question?