Introduction to Using the Terra Cluster

Francis Dang

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

HPRC Short Course – Spring 2017

Outline

- Usage Policies
- Hardware Overview
- Accessing 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:

    http://hprc.tamu.edu/shortcourses/SC-unix/
- Examples:
  - 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)

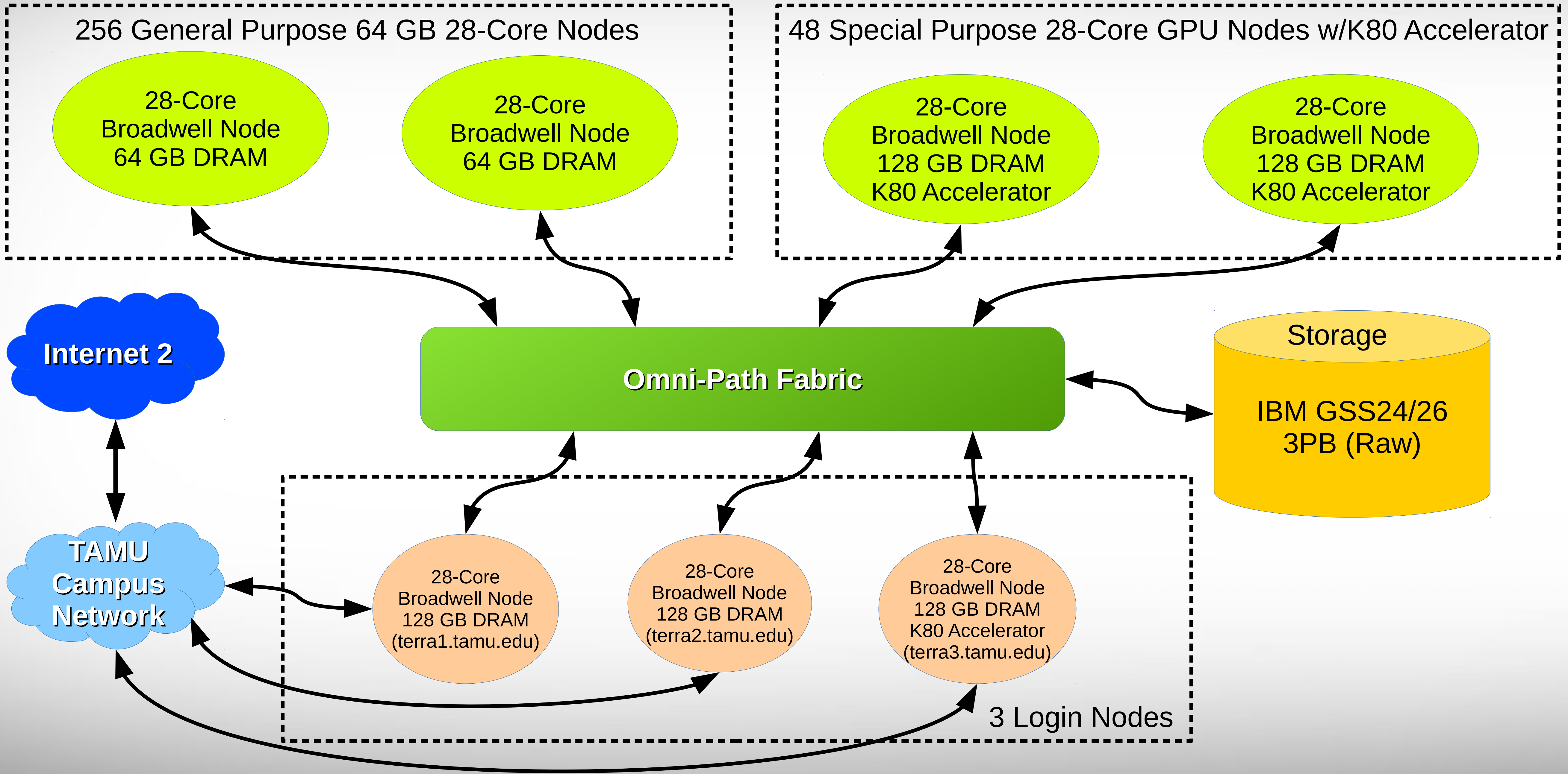
- It is illegal to share computer passwords and accounts by state law and university regulation
- It is prohibited to use Terra in any manner that violates the United States export control laws and regulations, EAR & ITAR
- Abide by the expressed or implied restrictions in using commercial software

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.

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


256 General Purpose 64 GB 28-Core Nodes
28-Core Broadwell Node 64 GB DRAM
28-Core Broadwell Node 64 GB DRAM
48 Special Purpose 28-Core GPU Nodes w/K80 Accelerator
28-Core Broadwell Node 128 GB DRAM K80 Accelerator
28-Core Broadwell Node 128 GB DRAM K80 Accelerator
Internet 2
Omni-Path Fabric
Storage
IBM GSS24/26 3PB (Raw)
TAMU Campus Network
28-Core Broadwell Node 128 GB DRAM (terra1.tamu.edu)
28-Core Broadwell Node 128 GB DRAM (terra2.tamu.edu)
28-Core Broadwell Node 128 GB DRAM K80 Accelerator (terra3.tamu.edu)
3 Login Nodes

Accessing Terra

- SSH is required for accessing Terra:
  - On campus: ***`ssh NetID@terra.tamu.edu`***
  - Off campus:
    - Set up VPN: u.tamu.edu/VPnetwork
    - Then: ***`ssh NetID@terra.tamu.edu`***
- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11)
  - PuTTY SSH
- Terra has 3 login nodes. Check the bash prompt.

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

- 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 help installing software.

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

File Transfers with 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:
  - ***Will be available at later date via the login nodes.***
  - Recommended methods will likely be:
    - Globus Connect (https://hprc.tamu.edu/wiki/index.php/SW:GlobusConnect)
    - GridFTP
    - bbcp

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.

- 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 directories
- **Do not share your home/scratch directories.** Request a group directory for sharing 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 this 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.

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 Python                         #load the default version of a package
– module load Python/2.7.12-intel-2016D      #load a specific version (recommended way)
```

- It's a good habit to purge unused modules before loading new modules.
- **Avoid loading modules in your *.bashrc***

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
- **Recommended version TBD.** Slide examples will use intel/2016D

  `module load intel/2016D`

- For applications that need gcc/g++, run ***`module spider GCC`*** to find available versions.

https://hprc.tamu.edu/wiki/index.php/SW:Toolchains

Modules and Toolchains

- Use modules based on the same toolchains in your job scripts

```
module load Python/2.7.12-intel-2016D
module load Eigen/3.2.9-intel-2016D
module load Voro++/0.4.6-intel-2016D
```

- Avoid mixing modules from different tool chains in the same job script:

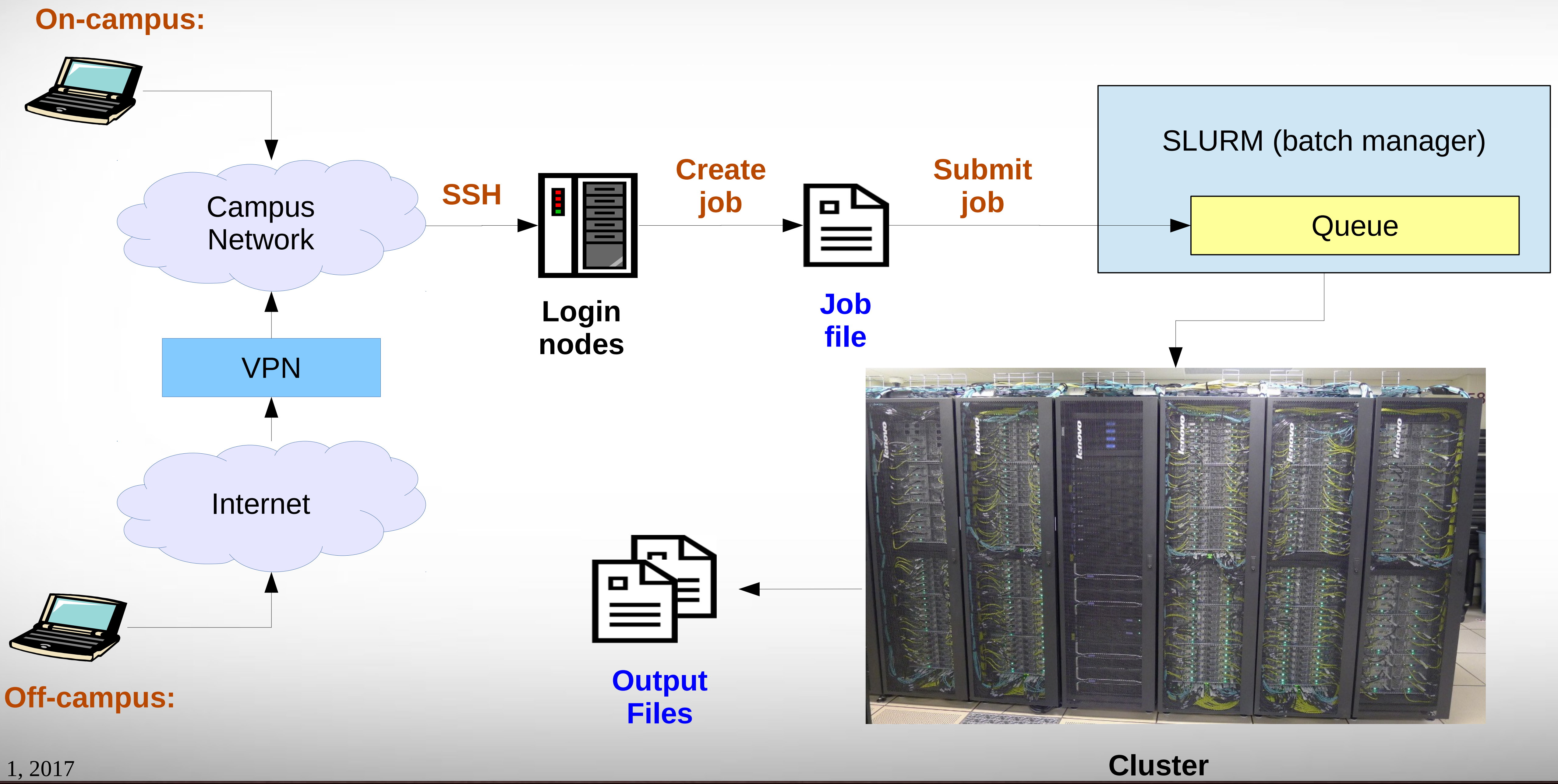
```
module load Python/2.7.12-intel-2016D
module load Eigen/3.2.9-intel-2016C
module load Voro++/0.4.6-intel-2016C
```

- 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).

Batch Computing on Terra



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 GPU nodes are needed, use the gpu queue
- Batch queue policies are used to manage the workload and may be adjusted periodically.

Current Queues

```
% 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

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

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

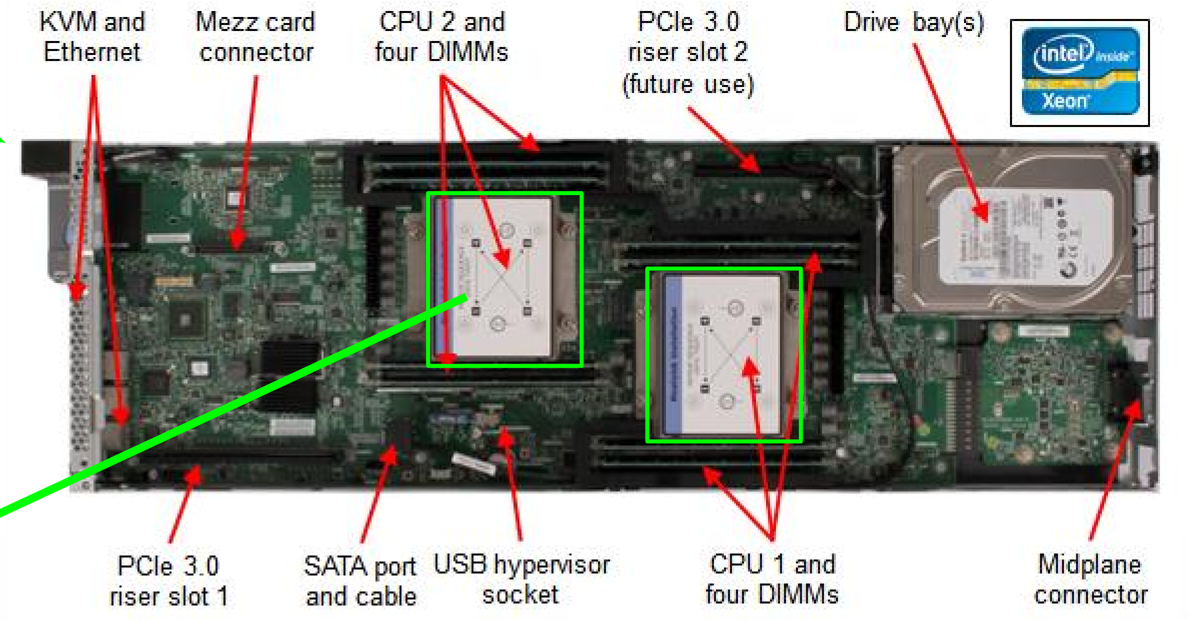
This is a command that is executed by the job

Important Job Parameters

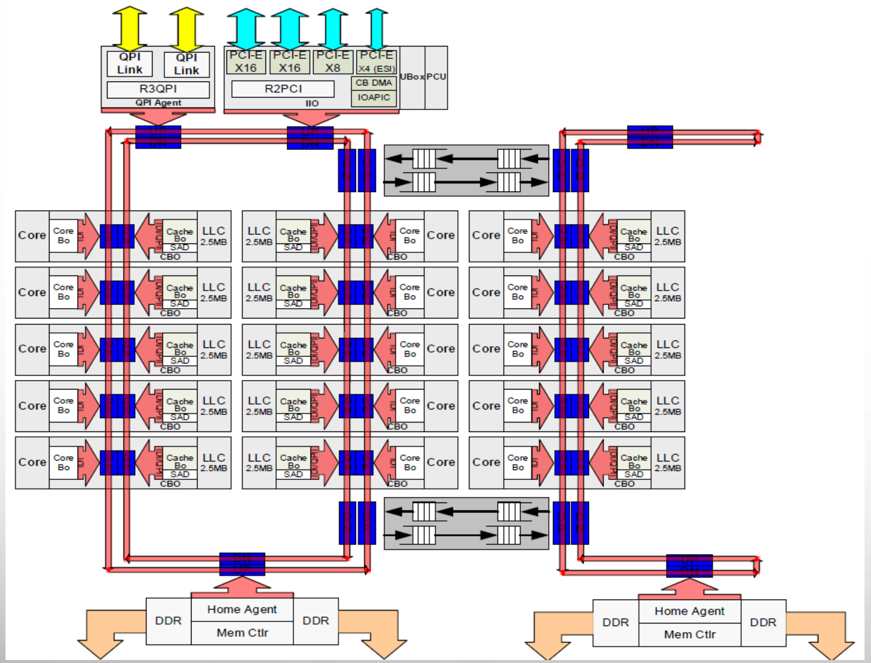
#SBATCH --export=NONE **#SBATCH --get-user-env=L**	Initialize job environment.
#SBATCH --time HH:MM:SS	Specifies the time limit for the job.
#SBATCH --ntasks MM	Total number of tasks for the job.
#SBATCH --ntasks-per-node=NN	Specifies the maximum number of tasks to allocate per node
#SBATCH --mem=XXXXM	Sets the maximum amount of memory (MB) the job can use per node.

Compute Nodes

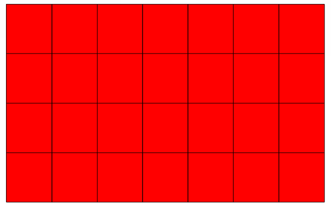
Part of Terra cluster.
Each green light is a node.



Each node has 28 processor cores.

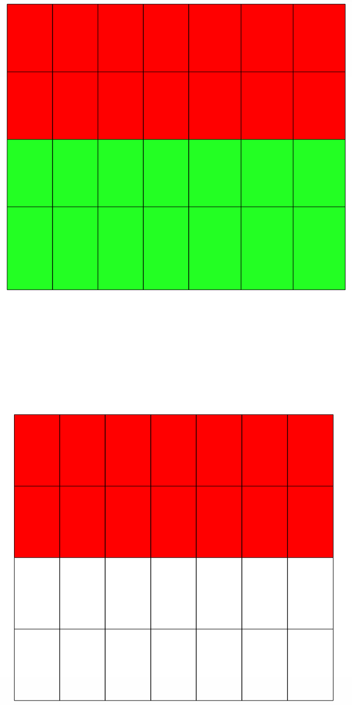


Mapping Jobs to Nodes



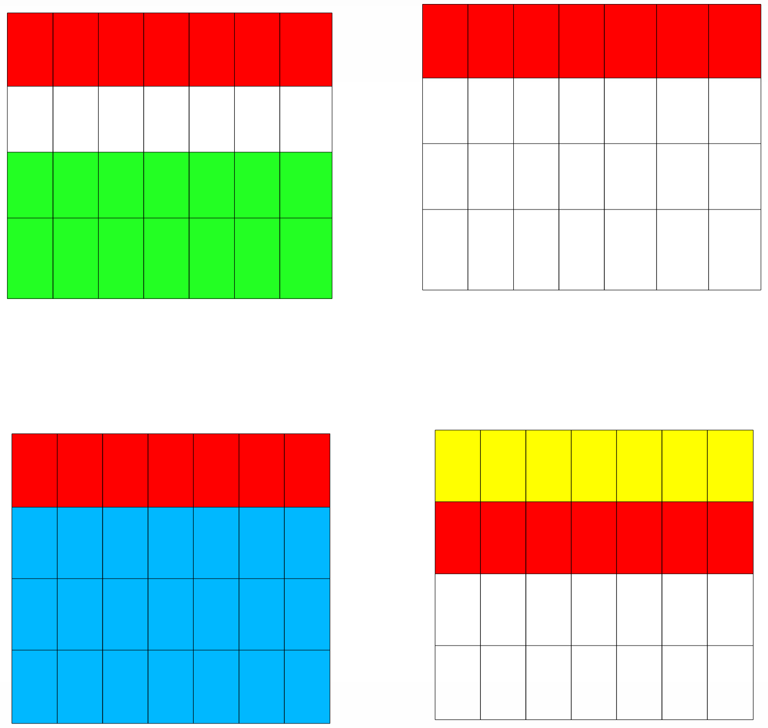
28 cores on
1 node

```
#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)

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
```

Job Memory Requests

- 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.

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
```

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
```

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
```

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
```

OpenMP Jobs

- Must set ***OMP_NUM_THREADS*** to take advantage of the requested cores
- All processes run on the same 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=JobExample5       #Set the job name to "JobExample2"
#SBATCH --time=6:30:00               #Set the wall clock limit to 6hr and 30min
#SBATCH --ntasks=1                   #Request 1 task
#SBATCH --cpus-per-task=8            #Request 8 cpus/cores per task
#SBATCH --mem=8192M                  #Request 8192MB (8GB) per node
#SBATCH --output=Example5Out.%j      #Send stdout/err to "Example2Out.[jobID]"

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

# set OpenMP number of threads to match job request
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# run program
./my_multicore_program
```

MPI Jobs

- MPI programs may be run in batch jobs on multiple nodes
- Note, the mpirun command will know how many MPI tasks to launch from SLURM's node, task, and/or task per node directives.

```
#!/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=JobExample6       #Set the job name to "JobExample2"
#SBATCH --time=6:30:00               #Set the wall clock limit to 6hr and 30min
#SBATCH --ntasks=24                  #Request 24 tasks
#SBATCH --ntasks-per-node=8          #Request 8 tasks/cores per node
#SBATCH --mem=8192M                  #Request 8192MB (8GB) per node
#SBATCH --output=Example6Out.%j      #Send stdout/err to "Example2Out.[jobID]"

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

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

Pop Quiz #1

```
#!/bin/bash

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

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1
#SBATCH --time=24:00:00
#SBATCH --ntasks=20
#SBATCH --ntasks-per-node=10
#SBATCH --mem=1000M
#SBATCH --output=Example1Out.%j
```

- How much total memory is requested for this job?

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

Pop Quiz #2

```
#!/bin/bash

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

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1
#SBATCH --time=24:00:00
#SBATCH --ntasks=40
#SBATCH --ntasks-per-node=40
#SBATCH --mem=200000M
#SBATCH --output=Example1Out.%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

```
sbatch sample01.job
```

```
Submitted batch job 64152
```

- Summary of the status of your running/pending jobs

```
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]
```

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

Job Submission and Tracking

Command	Description
sbatch jobfile1	Submit jobfile1 to batch system
squeue [-u user_name] [-j job_id]	List jobs
scancel job_id	Kill a job
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 job_id	Show resource usage for a job

Node Utilization: *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

- ***$SLURM_JOBID*** = job id
- ***$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

Check your Service Unit (SU) Balance

- Show the SU Balance of your Account(s)

```
myproject -l
```

```
==============================================================================
                      List of username's Project Accounts
------------------------------------------------------------------------------
|   Account     | Default | Allocation |Used & Pending SUs|    Balance    |
------------------------------------------------------------------------------
|122728110918|         N|     50000.00|                -10.38|     49989.62|
------------------------------------------------------------------------------
```

- Use "**`#SBATCH -A project_id`**" to charge SUs 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 Issues (SUs)

```
$ 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
```

- Insufficient SUs?
  - Ask PI to transfer SUs to you
  - Apply for more SUs (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

Debugging 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

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

```
slurmstepd: error: Exceeded job memory limit at some point.
```

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

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

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 #SBATCH -t parameter
- Insufficient memory specified in #SBATCH --mem or --mem-per-cpu parameters
- Memory specified is 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 terra`
  - Or use VNC
- Software license availability: check license status with `license_status -s softwarename`

```
$ 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
```

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 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 Programming Short Courses

Course Title	Times
Introduction to Using Ada Cluster	3-5 PM, Fri., Feb. 3
Introduction to Python	3-5 PM, Wed., Feb. 15
Introduction to Perl	3-5 PM, Wed., Feb 22
Intermediate MATLAB Programming	3-5 PM, Wed., Mar. 1
Next Generation Sequencing Data Analysis on the Ada Cluster	3-5 PM, Wed., Mar. 22
Introduction to Code Parallelization using OpenMP	3-5 PM, Wed., Mar. 29
Introduction to Code Parallelization using MPI	3-5 PM, Wed., Apr 5
Introduction to FORTRAN	3-5 PM, Wed., Apr 19

- Register or see a full list of short courses at:
  - https://hprc.tamu.edu/register/classlist.php

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

Thank you.

Any questions?

Brief Introduction to Parallel Computing

Parallelism

<u>Parallelism</u> means doing multiple things at the same time: you can get more work done in the same time.

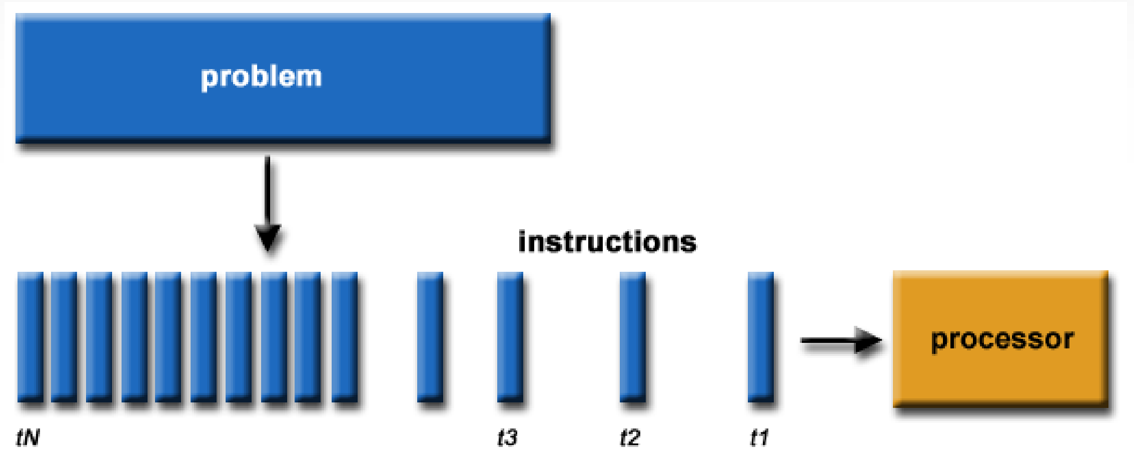
Less fish …

More fish!

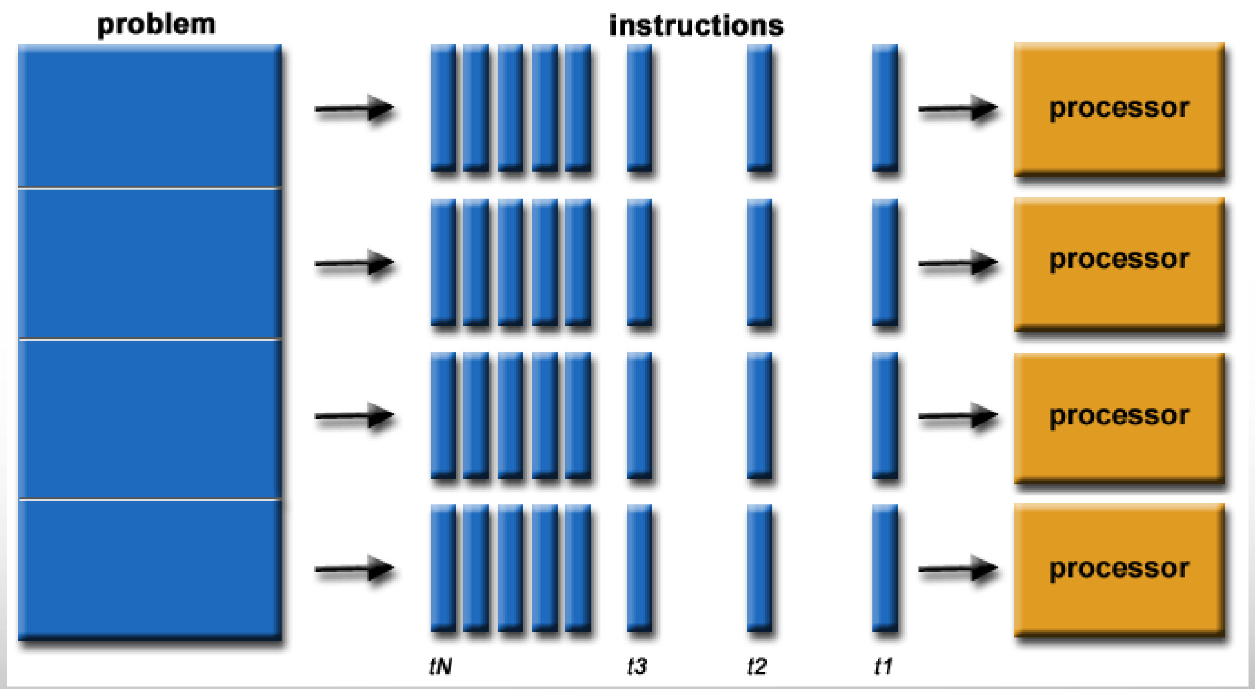
Source: http://oscer.ou.edu/Workshops/Overview/sipe_overview_20090201.ppt

Serial vs Parallel Computing

Serial Computing



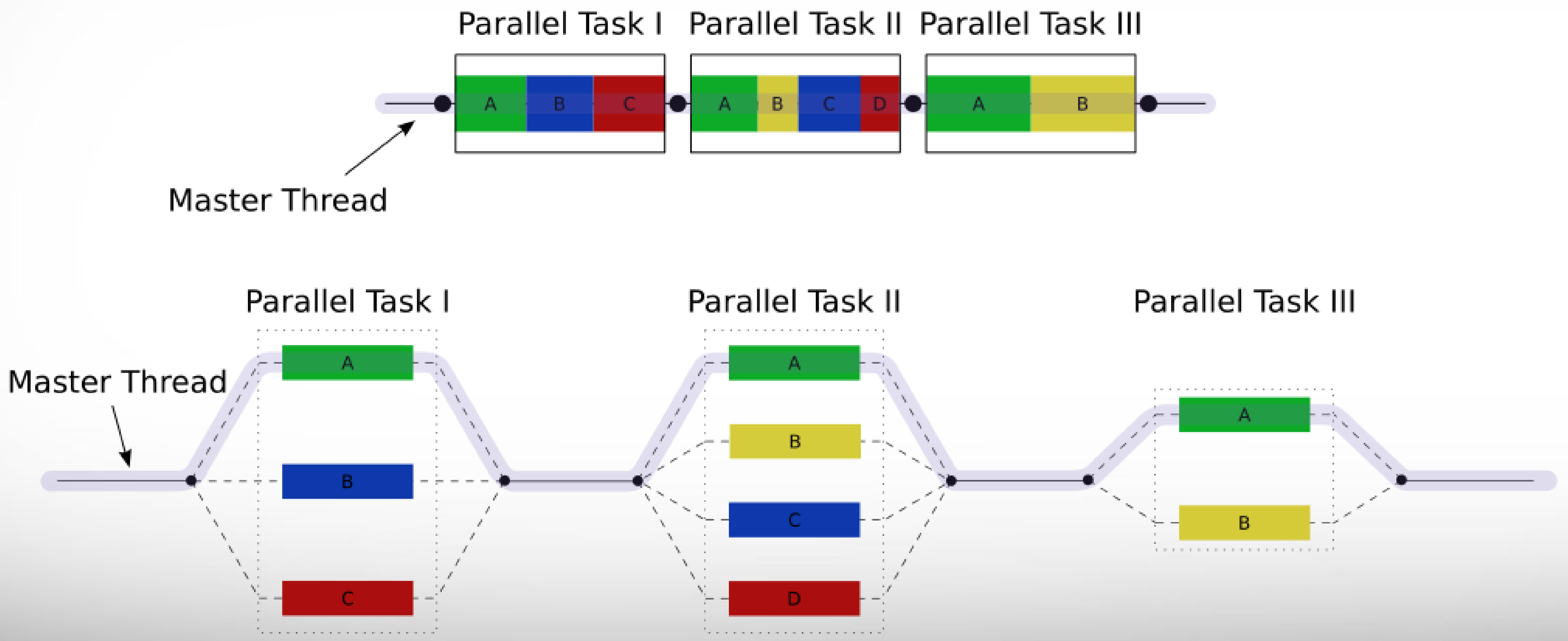
Parallel Computing



Source: https://computing.llnl.gov/tutorials/parallel_comp/

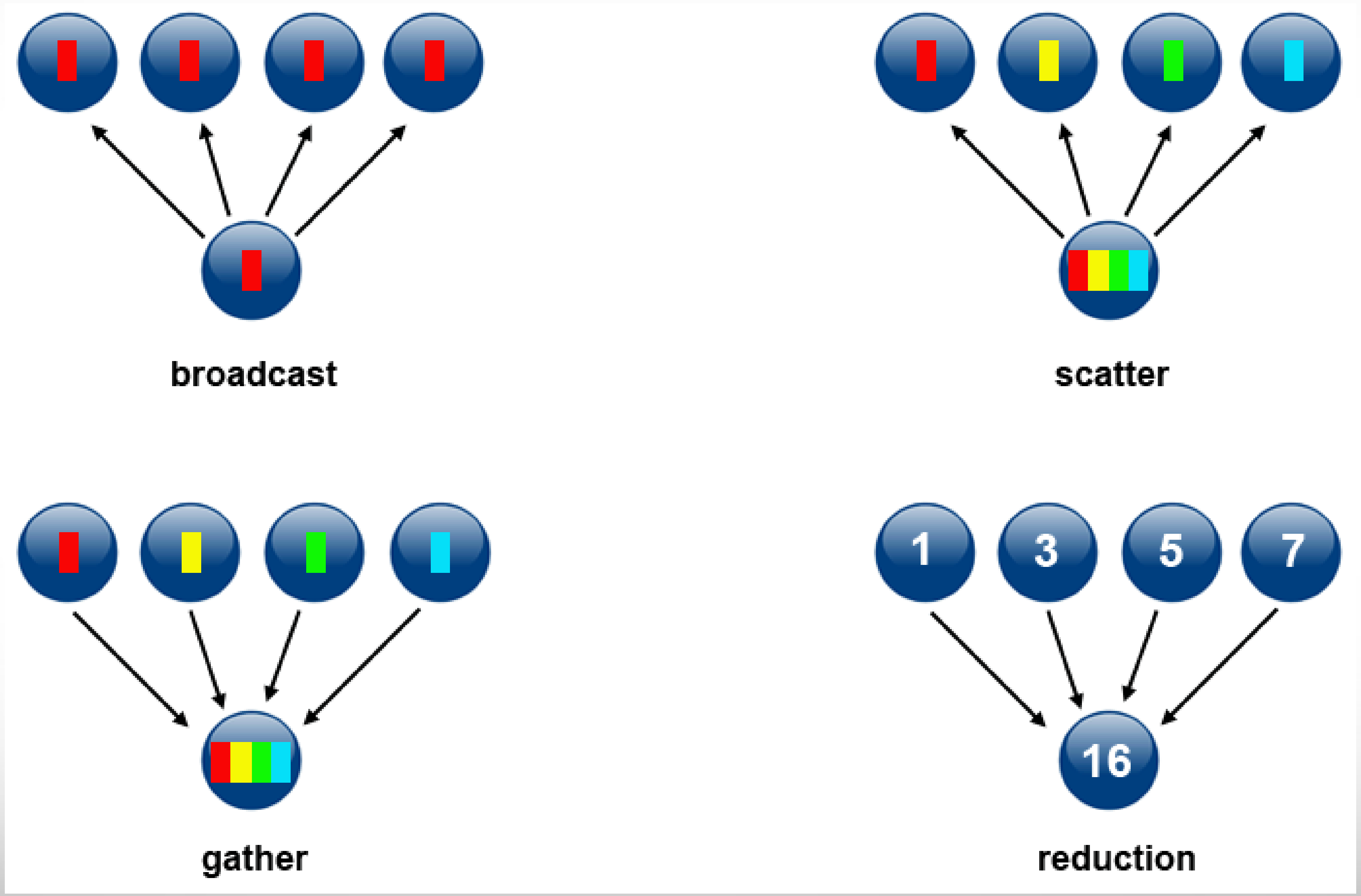
Multi-threading

Some tasks can be split and executed on process cores in a compute node.



Source: https://en.wikipedia.org/wiki/OpenMP

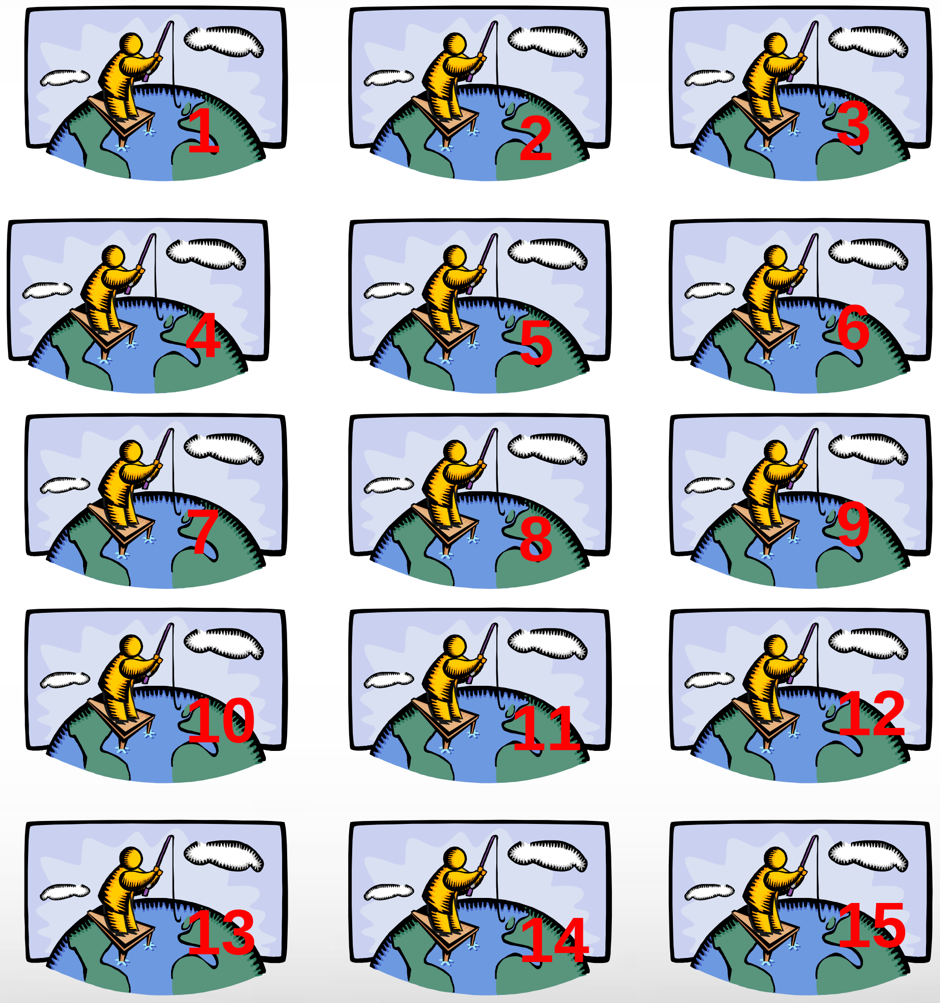
Distributed Computing - Collective Communication



Source: https://computing.llnl.gov/tutorials/mpi/

High Throughput Computing

- Each worker solve a subset of problems
- No dependency/communication among workers
- Parameter sweeping
- Scripting is your friend
- ***Also consider tamulauncher***



Still More fish!

Compiling Basics

- Generally provide the compiler:
  - source file(s) and/or object file(s)
  - compilation option(s)
  - optionally a name for the resulting executable. Default executable name is *a.out* if no name provided.
- Example:

```
icc objfile.o subroutine.c main.c
```

Basic Compiler Flags

Flag	Description
`-help [category]`	Shows all available compiler options or all options under a specified category
`-o <file>`	Specifies the name for an object file. For an executable, the -output filename will be <file> instead of a.out
`-c`	Only compile the source file(s). Linking phase will be skipped.
`-L <dir>`	Tells the linker to search for libraries in directory <dir> ahead of the standard library directories.
`-l<name>`	Tells the linker to search for library named lib**name**.so or lib**name**.a

Examples:

```
icc -o mprog.x subroutine.c myobjs.o main.c
icc -L mylibs -lmyutils main.c
```

Compiler Optimization Flags

Flag	Description
-O2	Default optimization level (includes inlining, constant/copy propagation, loop unrolling,peephole optimizations, etc)
-O3	Enables more aggressive loop transformations in addition to *-O2* optimizations.
-xHost	Tells the compiler to generate vector instructions for the highest instruction set available on the host machine.
-fast	Shortcut for *-ipo*, *-O3*, *-no-prec-div*, *-static*, and *-xHost* flags.
-ip	Perform inter-procedural optimization within the same file.
-ipo	Perform inter-procedural optimization between files.
-parallel	Enable automatic parallelization by the compiler (very conservative)
-opt-report=[n]	Generate optimization report. n represent the level of detail (0 ..3, 3 being most detailed)
-vec-report[=n]	Generate vectorization report. n represents the level of detail (0..7 , 7 being most detailed)

For more information, consult the *opt*, *advanced*, and *ipo* compiler help categories.

Other Compiler Flags

- Debugging flags:
  - https://hprc.tamu.edu/wiki/index.php/Terra:Debugging
  - See also the ***`icc -help command`*** which includes debugging and other flags.
- Flags affecting floating point operations:
  - https://hprc.tamu.edu/wiki/index.php/Terra:Compile:All#Flags_affecting_floating_point_operations
  - See also the ***`icc -help float help`*** or the ***`ifort -help float`*** commands. Some floating point flags are specific to Fortran.
- Many more compiler flags. Consult each compiler's man page or the output from the compiler's ***-help*** option.

Compiling OpenMP Programs

- OpenMP programming:
  - Use compiler directives to specify which code regions to run in parallel
  - Compiler generates multi-threaded code for these code regions
- Example:

```
module load intel/2015B
ifort -qopenmp -o omp_helloWorld.exe omp_helloWorld.f90
```

Running OpenMP Programs

- Common environment variables:
  - OMP_NUM_THREADS:
    - Sets the maximum number of threads per nesting level
    - Default value is 1
  - OMP_STACKSIZE:
    - Sets the size for the private stack of each worker thread. Suffix can be B,K,M,G
    - Default value is 4 MB
- Example using 4 threads and 16 MB stack size per thread

```
$ export OMP_NUM_THREADS=4
$ export OMP_STACKSIZE=16M
$ ./omp_helloWorld.exe
```

- **Do not use more than 8 cores on the login nodes!**

Compiling MPI Programs

- Use a MPI compiler wrapper to compile MPI codes.
  - Wrapper invokes underlying compiler and adds linker flags specific for MPI programs
  - Intel MPI provides wrappers for both Intel and GNU compilers
  - Any flags not recognized by the wrapper are passed to the underlying compiler.
- Example to compile MPI C program with the Intel compiler's *-O3* optimization flag

```
mpiicc -o mpi_prog.x -O3 mpi_prog.c
```

Running MPI Programs

- Requires a MPI launcher (mpirun) to run MPI programs

  mpirun [mpi_flags] executable [executable params]

- Example:

```
module load intel/2015B
mpirun -np 4 ./mpi_helloWorld.exe
```

- **Do not use more than 8 cores on the login nodes!**

CUDA Programming

- Compiling programs to use GPU accelerators
  - Load a CUDA module
  - Can compile CUDA codes on any login node but can only run CUDA programs on the GPU login node (**terra3.tamu.edu)**
  - Use -arch=compute_37 -code=sm_37 to compile your code specifically for Terra's K80 GPUs
- Example:

```
module load CUDA/8.0.44

nvcc -o cuda_prog.exe -arch=compute_37 -code=sm_37 cuda_prog.cpp
```

- For possibly better code performance, optionally load an intel toolchain and add -ccbin=icc to compilation flags.
  - If using an intel toolchain with GCC 5.0+, use the CUDA/7.5.18-unsuppportedCC or CUDA/8.0.44-unsuppportedCC modules (which do not have strict GCC version checks)

Intel Math Library (MKL)

- Provides optimized and threaded math routines such as BLAS, LAPACK, sparse solvers, FFTs, vector math, and more.
- Offers sequential, parallel, and cluster versions.
- Examples:

```
module load intel/2016D
ifort example.f -mkl=sequential -o example.exe
icc example.c -mkl=parallel -o example.exe
mpiifort example.f -mkl=cluster -o example.exe
```

- Consult Intel MKL Link advisor for usage help: https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor