Introduction to Computing on Raad2

Research Computing

Nov 2017

TEXAS A&M UNIVERSITY at QATAR

Overview

- Introduction to supercomputers
- Our supercomputer “Raad2”
- Resource management & job scheduling
- Using SLURM to “submit” jobs
- Managing the user environment with the “module” utility

The “Introduction to Linux” short course offered by Research Computing should be treated as a prerequisite for this one.

How & Where to Seek Assistance

- Research Computing Website
  - https://rc.qatar.tamu.edu
  - Primarily for general information about our work & services; account applications
- Research Computing Wiki
  - https://rc-docs.qatar.tamu.edu
  - Technical content for active users of our systems; user guides, training material, etc
- Service desk (email queries)
  - servicedesk@qatar.tamu.edu
  - Email queries should:
    - Have informative and relevant subject headers
    - Mention the name of the HPC system & software package with which you are having problems
    - Include relevant error/warning messages
    - Mention locations of relevant files… job files, error output, etc.
    - A clear description of the problem
- Walk-in assistance
  - TAMUQ building, suite 125
  - We will assist if available, or schedule an appointment with you if busy

TEXAS A&M UNIVERSITY at QATAR

Introduction to Supercomputers

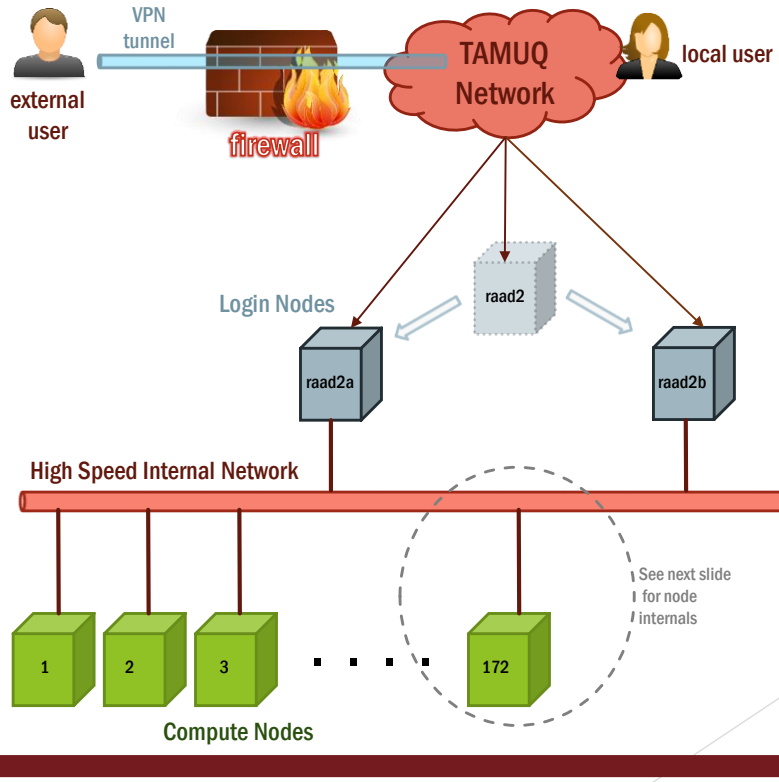
What is a Supercomputer?

- Generic "big picture" description
  - A type of parallel or distributed processing system, which consists of a collection of interconnected computing elements working together as a single computing resource
    - **A supercomputer is not a general purpose computer, and should not be expected to be one**

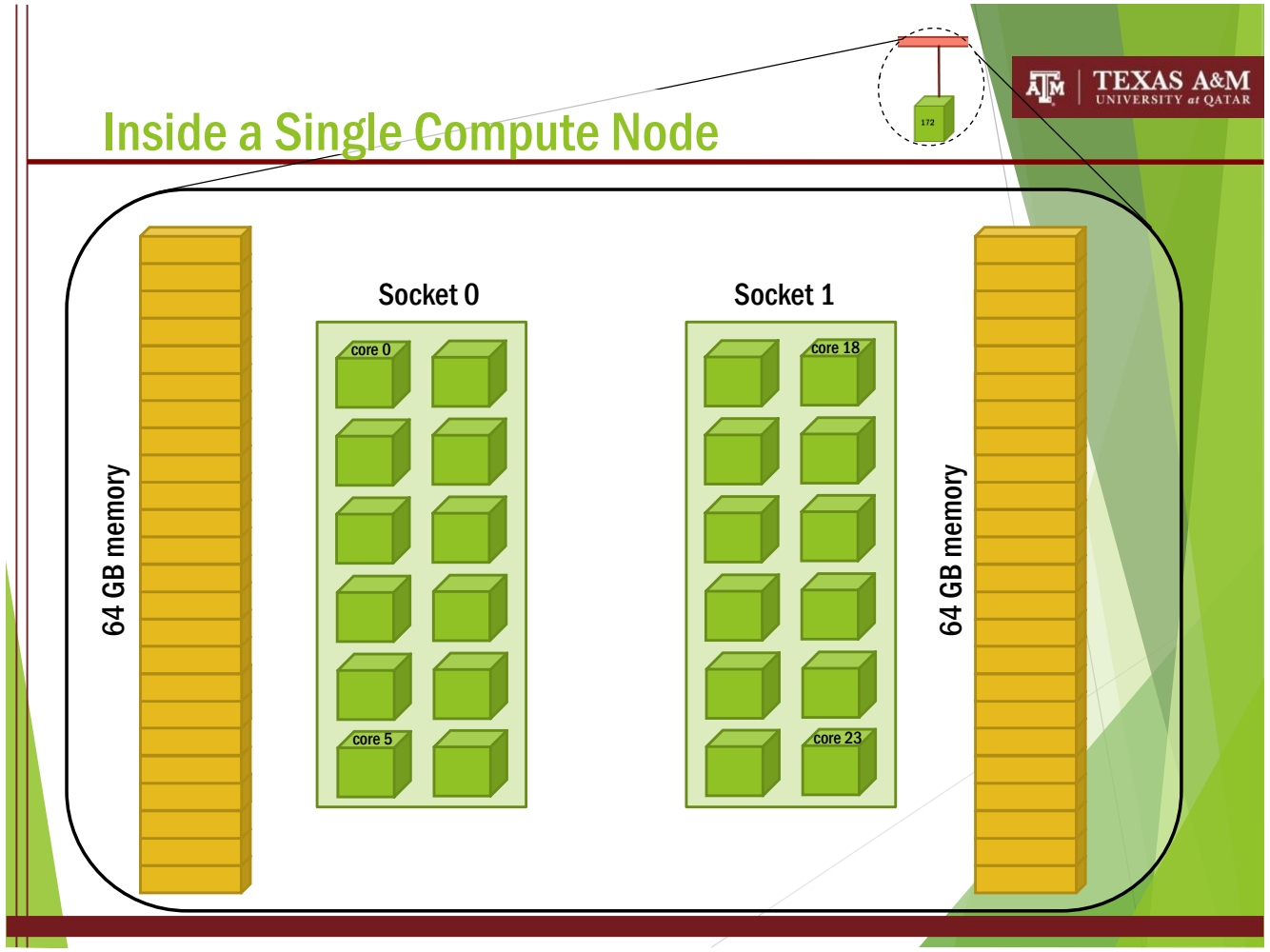
What is a Supercomputer?

- Generic "big picture" description
  - A type of parallel or distributed processing system, which consists of a collection of interconnected computing elements working together as a single, integrated computing resource
    - A supercomputer is not a general purpose computer, and should not be expected to be one
    - **A supercomputer will normally have the capability to effectively bring a large amount of computing resources to bear on a single large problem, and to solve that problem in the shortest amount of time**

Raad2 -- Logical Architecture



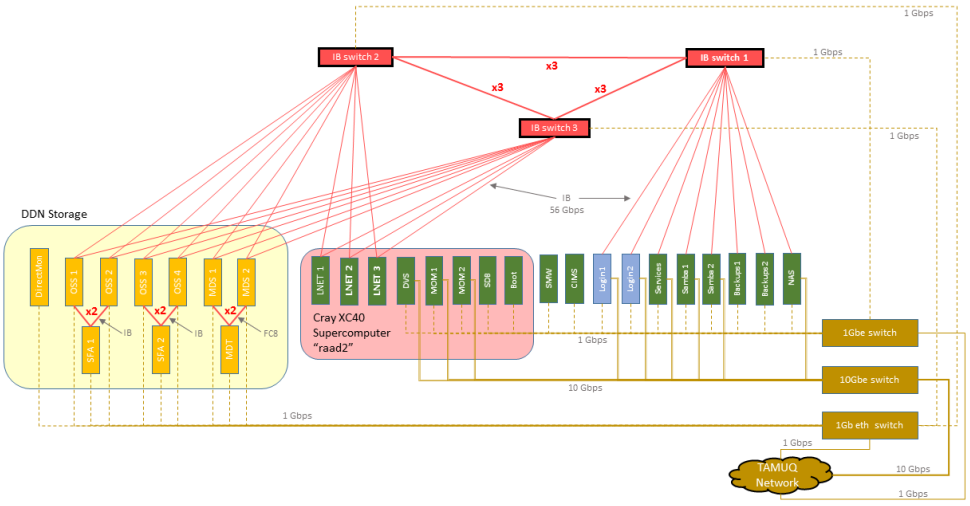
Inside a Single Compute Node



TEXAS A&M UNIVERSITY at QATAR

Our Supercomputer: Raad2

Raad2 -- Physical Architecture



Why Use Supercomputers?

- Processing power
  - Supercomputer compute nodes usually exceed the computing horsepower of desktop workstations. For highly parallel workloads, there is no comparison between supercomputers and individual workstations.
- Scalability
  - Specialized hardware enables parallel workloads to span hundreds of processors, and beyond. This is not feasible even with networked collections of workstations.
- Availability
  - Typically, there is no single point of failure. Compute node failures are isolated, and affected jobs can be re-started on other available nodes.
- Reduced cost
  - If system utilization is kept high, sharing a single supercomputer among a large group of users can ultimately be more cost effective for the organization.

Supercomputer "Ingredients"

- Hardware
  - Login & service nodes
  - Compute nodes
  - Interconnect fabric (a high-speed private network)
  - Shared storage system
- Systems Software
  - Operating system
  - Supercomputer management software
  - Batch system (or "resource manager")
  - Parallel file system
- User Applications
  - Compilers, profilers, debuggers, parallel libraries & related tools
  - ISV packages (Ansys, Matlab, Gaussian, LAMMPS, VASP, Gromacs, etc.)

Raad2 Details

Technical Specs	
Host Name	raad2.qatar.tamu.edu
Batch/Resource Scheduler	SLURM 15.08
Operating System	Cray Linux Environment (CLE) 5.2 – SLES 11 SP3
Number of Nodes	172
Aggregate Number of CPU Cores	4,128
Interconnect	Aries
Aggregate System Memory	22 TB
Peak Performance	120+ TFLOPS
Local Compute Node Disks	none
Shared Storage System Capacity	800 TB (usable)
Parallel Filesystem	Lustre

Raad2 Compute Nodes

- Compute nodes are named **nid00xyz**
  - xyz is a non-contiguous set of 3 digit integers (e.g. 008, 243, etc)
  - each node contains 24 CPU cores & 128 GB of RAM
  - there is NO local disk in any of the compute nodes
- Users cannot directly access any compute node (e.g. `ssh nid00008` not allowed)
  - Access only via the login nodes
  - Access only by issuing requests to the SLURM batch system

User Home & Scratch Directories

- Your home directory resides at `/lustre/home/username`
  - e.g. `/lustre/home/fachaud74`
- Your default disk quota is 500GB
  - Applies to everything you own under `/lustre`, including `/lustre/home`
  - Quota extensions can be considered based on strength of justification
- Home directory backups are performed frequently
  - Full backups after every 4 week interval, on a Friday
  - Incremental backups after every 2 day interval
  - At any given time, the 2 most recent full backups & associated incrementals remain available
- A scratch directory for use by batch jobs is available under `/lustre/scratch`
  - Job specific sub-directories need to be created & deleted manually by the requesting job
  - Consult documentation on the RC Wiki on how to do this within your job file
- The `/lustre/scratch` directories are NOT backed up.

Available Software

- Compilers
  - Cray: cc, ftn, CC
  - GNU: gcc, gfortran, g++
  - Intel: icc, ifort, icpc
  - PGI: pgcc, pgfortran, pgc++
- MPI libraries
  - Intel MPI
  - Cray MPI
- User applications
  - Ansys, Fluent, CFX
  - Gaussian 09, GROMACS, VASP, LAMMPS
  - Matlab, Mathematica
  - HDF, NetCDF
- Complete list: https://rc-docs.qatar.tamu.edu/index.php/Hpc_soft2

Remote Access to Raad2

- VPN required for off-campus access
  - First "log in" to the TAMUQ network, then log in to the Raad2 system
  - VPN enhances protection against network-based attacks
  - Detailed instructions on establishing a VPN connection
    - https://rc-docs.qatar.tamu.edu/index.php/VPN
- SSH (secure shell)
  - The only protocol allowed for login access on Raad2
  - Provides encrypted communication
  - Freely available for a variety of systems
- MobaXterm
  - Remote access software for Windows PCs (freeware)
  - The SSH protocol comes built-in
  - The recommended remote access utility for RC users
- Raad2 runs the Linux OS. For Linux training, see:
  - https://rc.qatar.tamu.edu/Pages/support/training/training.aspx

EXERCISE: Log in to Raad2

- Following the instructor, set up your session to raad2 using MobaXterm
  - Free software to remotely access linux systems from windows PCs
  - https://mobaxterm.mobatek.net/download-home-edition.html
- Login in to raad2 using the session created earlier
  - Supercomputer username will have a format like `fachaud74`
  - Password for supercomputer account is independent of password for TAMUQ domain account (the one used for VPN access when outside the TAMUQ building)
- Following the instructor, log in to raad2 using WinSCP to transfer some files
  - Free software to transfer files between your PC and a remote linux system
  - https://winscp.net/eng/download.php


TEXAS A&M
UNIVERSITY at QATAR
Resource Management &
Job Scheduling

Typical Workflow

Login in to the supercomputer

(If required, establish VPN, then use any SSH client -- MobaXterm recommended)

Decide what resources you need

(in other words, write a job file)

Ask system for your resources

("submit" the job… its like placing an order)

Let the system start your computation

(when your turn comes, this will happen on its own)

Monitor the computation

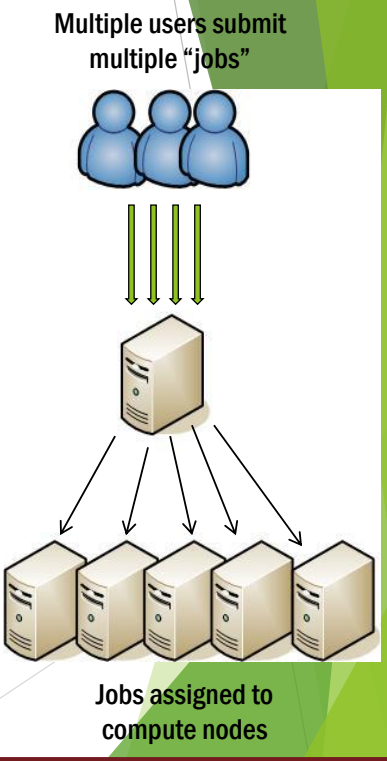
(track job status with specific commands)

Access your results

(examine output files in your home directory to see what happened)

Sharing Can Be Complicated

- Since a supercomputer is a shared computing platform, how do we…
  - keep track of availability of resources?
  - account for usage by users and jobs?
  - maximize system utilization?
  - prioritize jobs, determine order of execution, enforce "fairness"?
- This implies we need a "resource manager" and a "job scheduler"



SLURM

- SLURM originally stood for “Simple Linux Utility for Resource Management”
- Free and open source
- Used on majority of the systems on the Top 500 list

The Role of SLURM

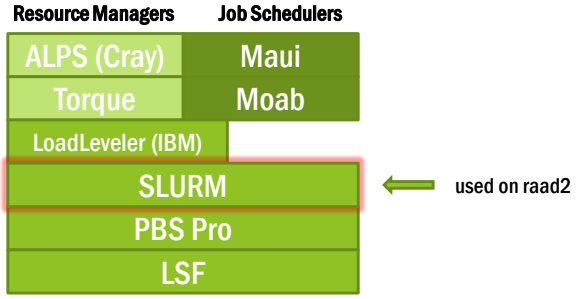
- SLURM: "This job is ready to run. Which specific resources should I assign for this particular job to run on?"
  - **Resource allocation** is one of the basic functions of SLURM
    - memory
    - sockets
    - cores
    - threads
    - and more…
- SLURM: "There are so many jobs waiting to run. Which one is most deserving to run next?"
  - **Job scheduling** is another basic function of SLURM
    - Supports complex scheduling algorithms
    - Can prioritize jobs based on configurable parameters (job age, job size, job QOS, etc)
- SLURM also provides a mechanism for starting, executing & monitoring jobs

The Role of SLURM

- Incoming jobs are prioritized based on multiple factors
- Job priority changes over time due to some dynamic factors (job age, recent resource consumption of requesting user, etc)
- Jobs are kept “pending” until the system is able to run them (i.e. until available resources are found)
- Periodically, the highest ranked jobs are selected from the “pending” list to begin execution
- Order of job execution is not necessarily “first in first out”
- Scheduling policy can be tuned to meet the needs of a given site
- Different types of workloads may have access to varying amounts and types of resources
- The system tracks resource consumption by job, so various limits can be enforced
- Users can also use SLURM commands to track resource availability and job status
- Resource usage is logged, potentially helping admins tune policies in the future

Other Resource Managers

- On our older system "raad" we used PBS Pro for resource allocation & job scheduling. On raad2 we use SLURM.
- Some other similar products (from competing vendors/developers) include:



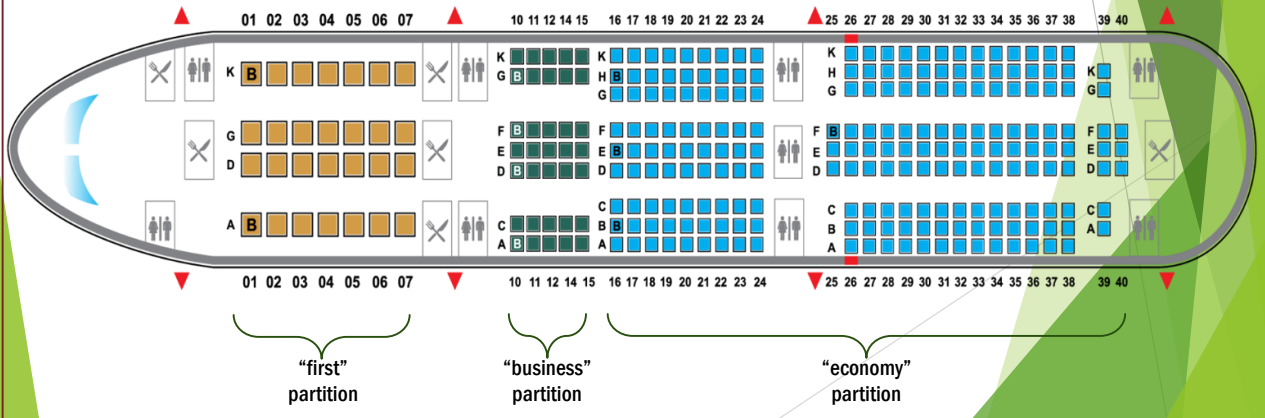
- However, basic idea of how these products work and the core functionality remains conceptually similar.
- SLURM and PBS are functionally very much alike for most use cases. Syntax for job files is different though.

What Resources are Managed by SLURM?

- Resources are allocated to (and consumed by) jobs.
- Basic types of resources requested by SLURM jobs include…
  - Amount of walltime (i.e. total run time) requested for a job
  - Amount of physical memory requested per node
  - Number of individual CPU's (or compute nodes) requested for a job
- SLURM uses the concepts of partitions & Quality Of Service to
  - Control the allocation of hardware resources to jobs
  - Enforce resource limits on various categories of jobs

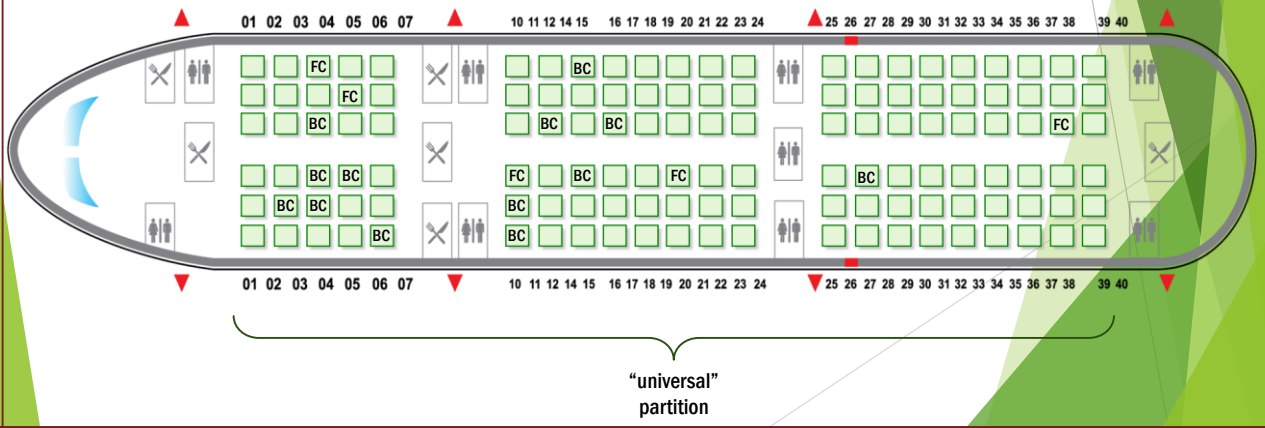
Partitions & Quality of Service

- The airplane cabin analogy
  - A "partition" would equate to a particular cabin area (a specific set of seats)
  - "Quality of Service" (QOS) would equate to "First Class", "Business Class", or "Economy" depending on the type of boarding pass presented
    - Metric 1: Seat dimensions & comfort
    - Metric 2: Quality of food served



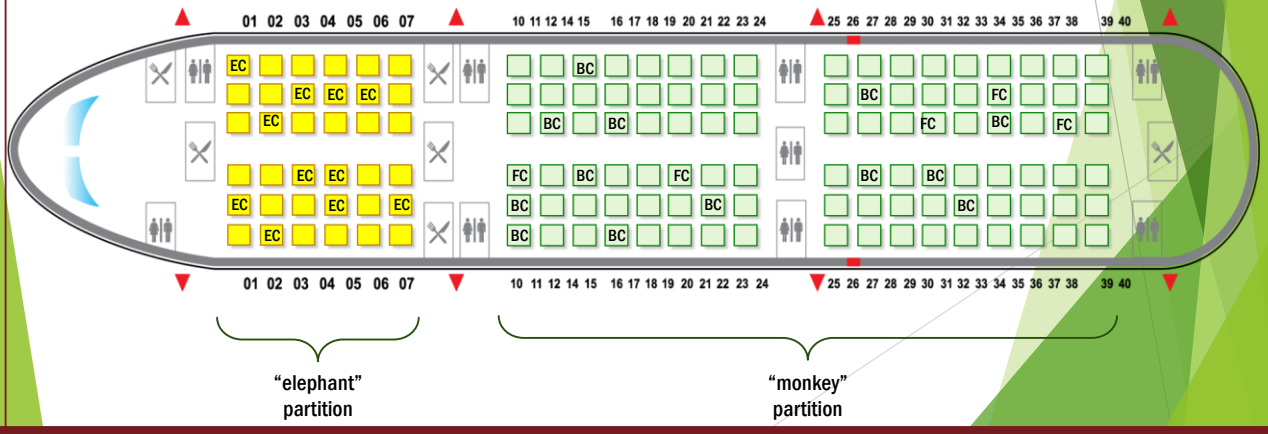
Partitions & Quality of Service

- Imagine a "universal" cabin area (i.e. partition)
  - Any seat can instantly transform into economy, business, or first class
  - Any class of passenger can be seated in any seat
  - Seat configuration & food quality will depend on type of boarding pass (i.e. QOS level)
- We can also define certain limitations for each QOS
  - "no more than 10 FC passengers in the universal partition"
  - "a single EC customer can't purchase more than 6 seats per flight"



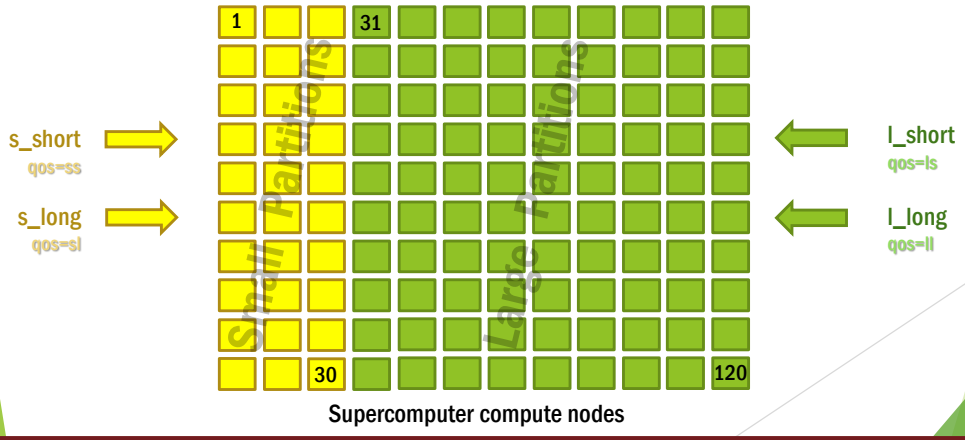
Partitions & Quality of Service

- In this variant of our example, certain limitations on the Business Class and First Class QOS passengers might make sense (for the "monkey" partition)...
  - "No more than 30 of the 102 seats may be occupied by FC passengers"
  - "No more than 72 of 102 seats may be occupied by BC passengers"



Partitions & Quality of Service

- Like the airline cabin, we can divide the supercomputer into different partitions
- Partitions can overlap; in fact, two partitions can be mapped to the same set of nodes
  - At TAMUQ, partitions s_short & s_long map to the same set of nodes
  - l_short & l_long map to a distinct second set of nodes
- At our site, every partition is configured to accept jobs subscribing only to one associated QOS
  - Technically, of course, a single partition can accept jobs subscribing to different QOS's



Quality of Service Metrics

- What defines a specific QOS on our system?
  - Maximum # of CPUs that can be used in total at any given point in time
    - "We can't have more than 100 Economy Class passengers on this plane"
  - Maximum # of CPUs that a single user can use at any given time
    - "A single individual can't purchase more than 6 Economy tickets/seats"
- Other metrics can also be added to this list, but at our site only the above two are used to distinguish between different QOS's.

Partitions & QOS Levels Defined on Raad2

TEXAS A&M UNIVERSITY at QATAR

overlapping partition

overlapping partition

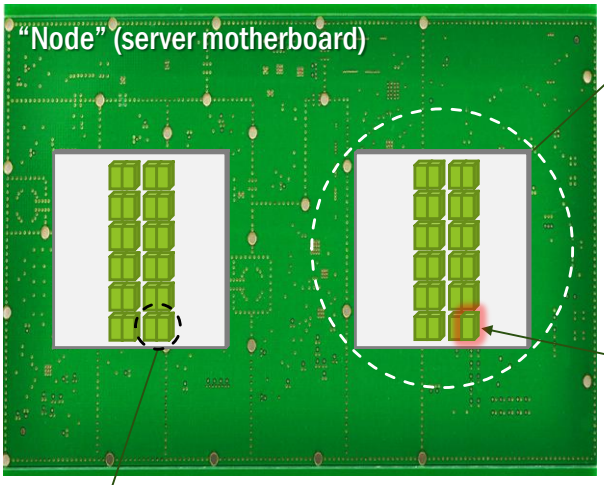
Partition Name (QOS)	express (ex)	s_short (ss)	s_long (sl)	s_debug (sd)	l_short (ls)	l_long (ll)
Max walltime	1 hour	8 hrs	168 hrs	4 hrs	8 hrs	168 hrs
Max CPUs per user	48	24	48	24	2 nodes	10 nodes
Max CPUs per QOS	96	144	288	48	8 nodes	86 nodes
Default walltime	30 mins	2 hrs	24 hrs	1 hrs	2 hrs	24 hrs
Default memory	5,350 MB per allocated core when using `#SBATCH --hint=nomultithread` or 2,675 MB per core otherwise					

The values above are subject to change. To display the actual configuration at any given time:

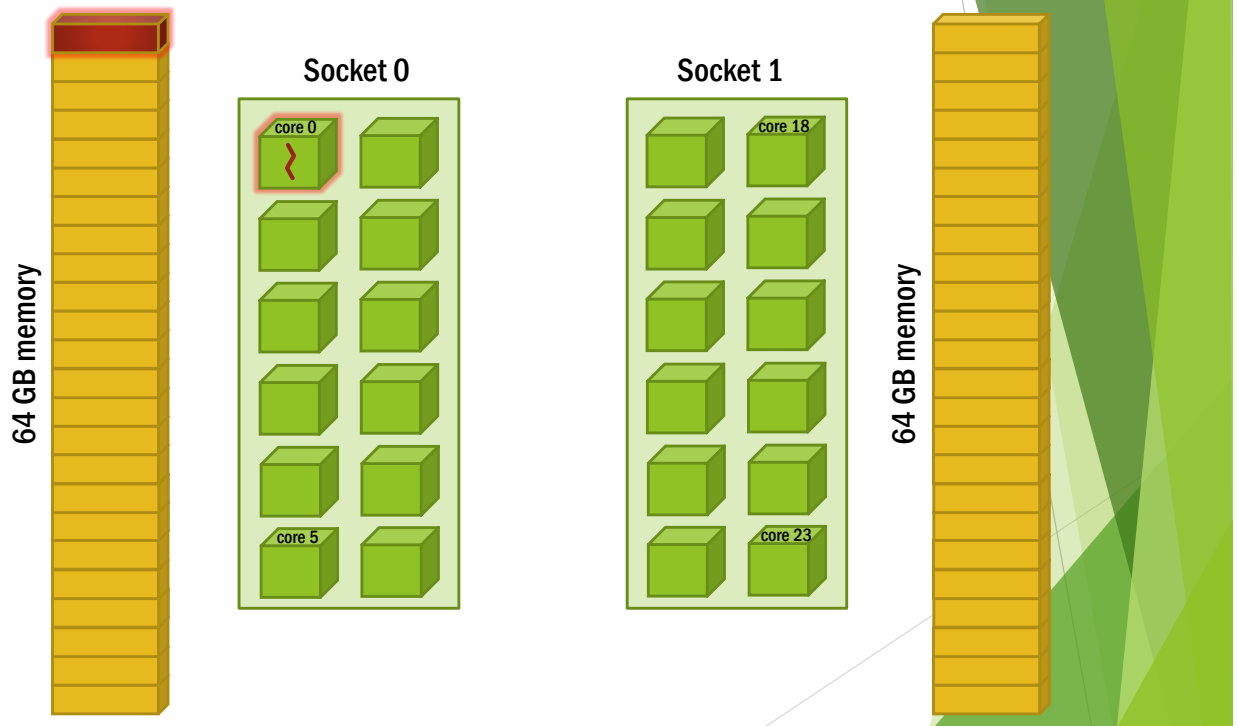
```
scontrol show partition
```

Terminology

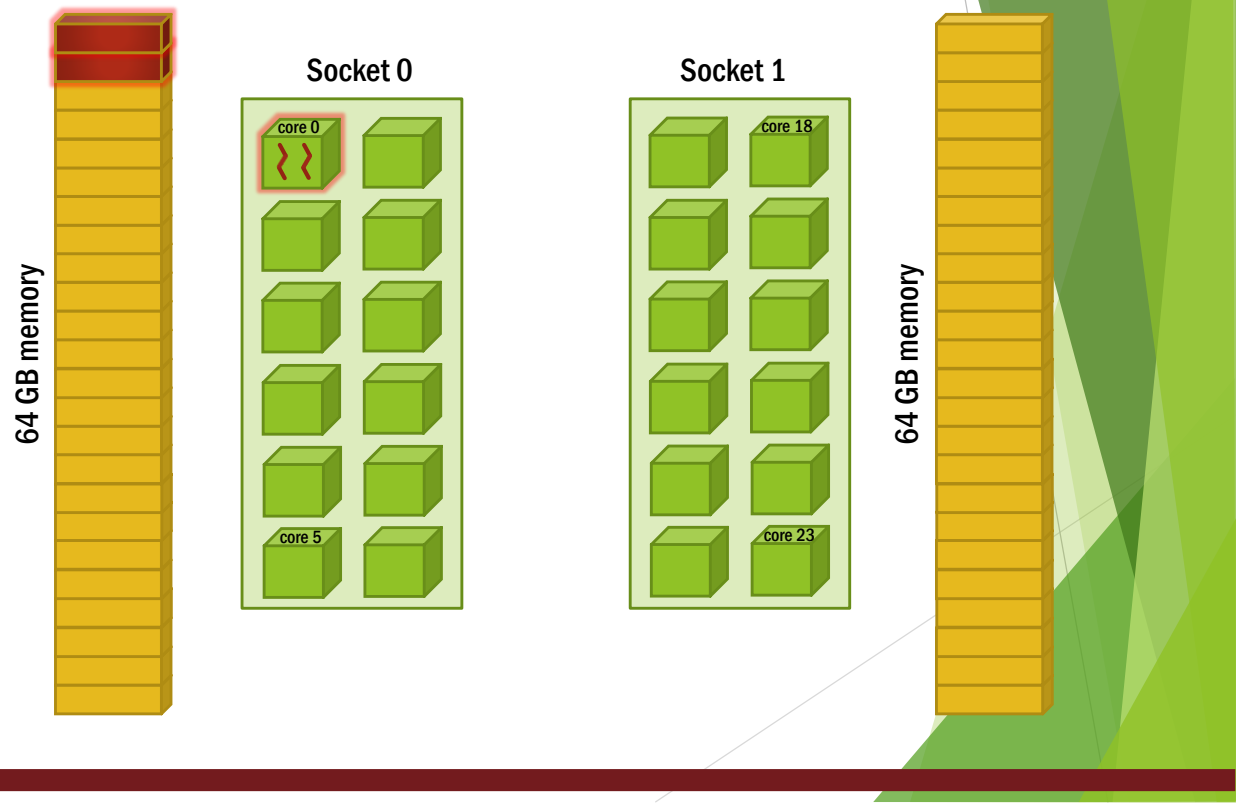
- "Will the real CPU please stand up?"



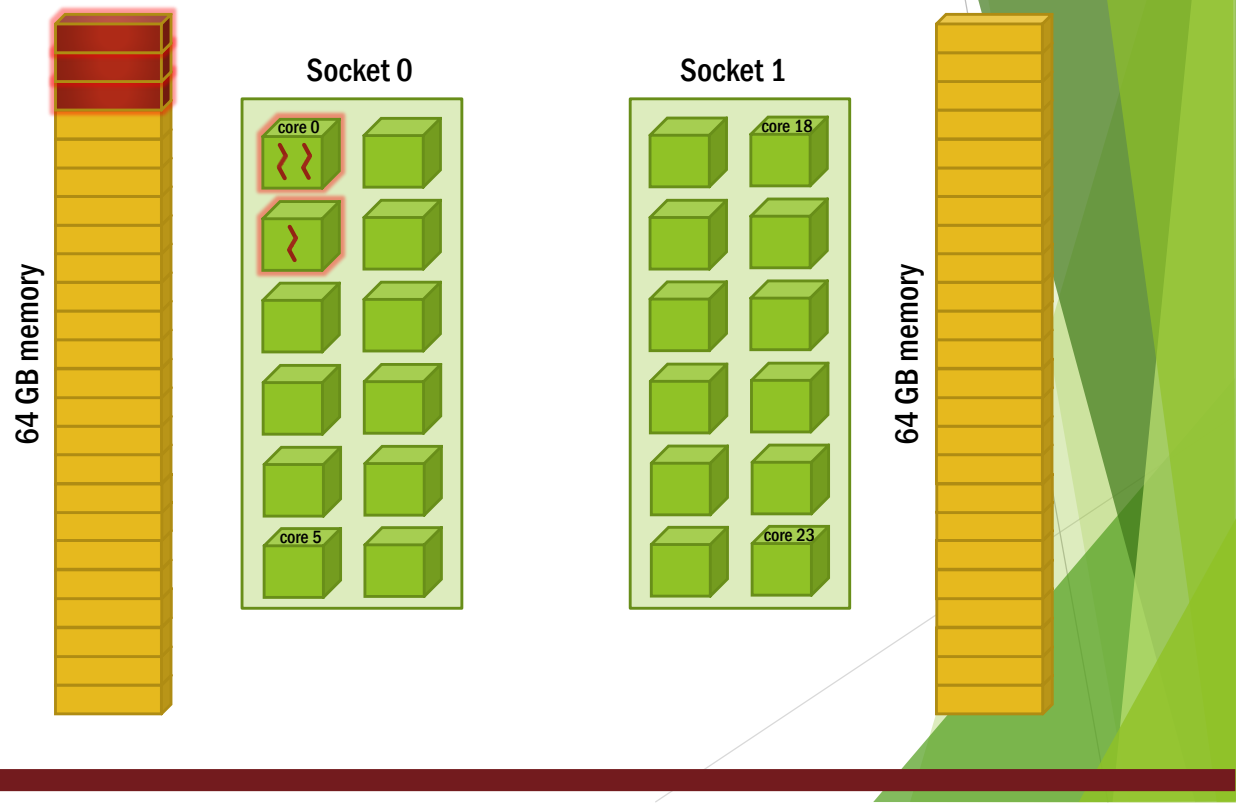
CPU Resource Allocation



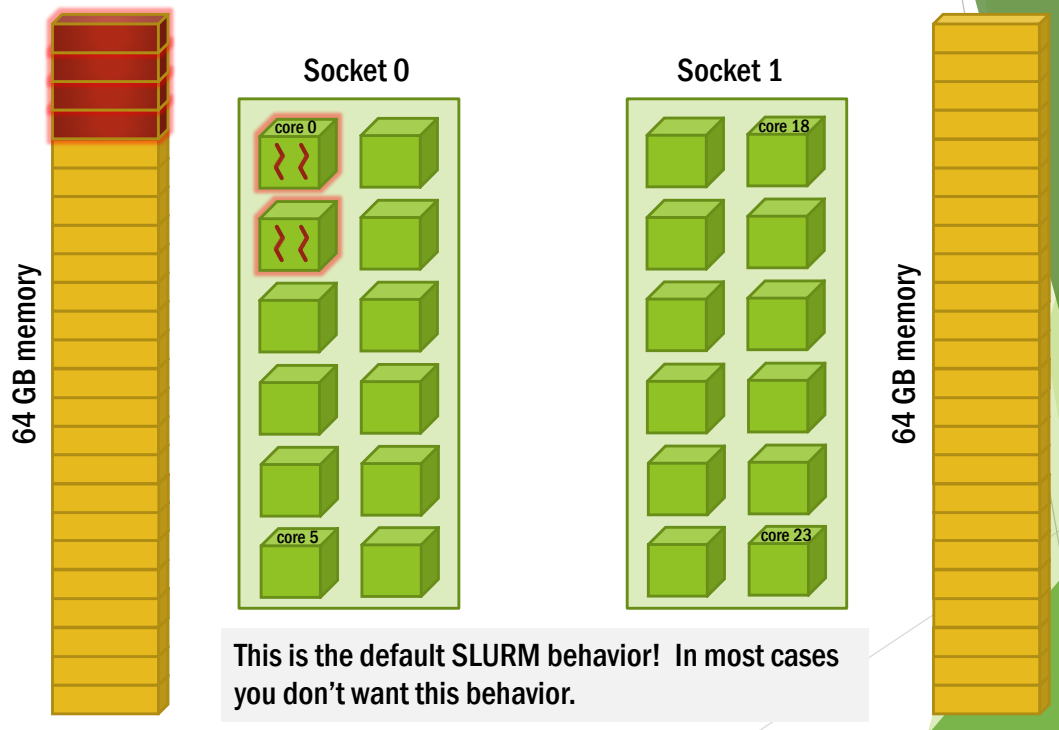
CPU Resource Allocation



CPU Resource Allocation

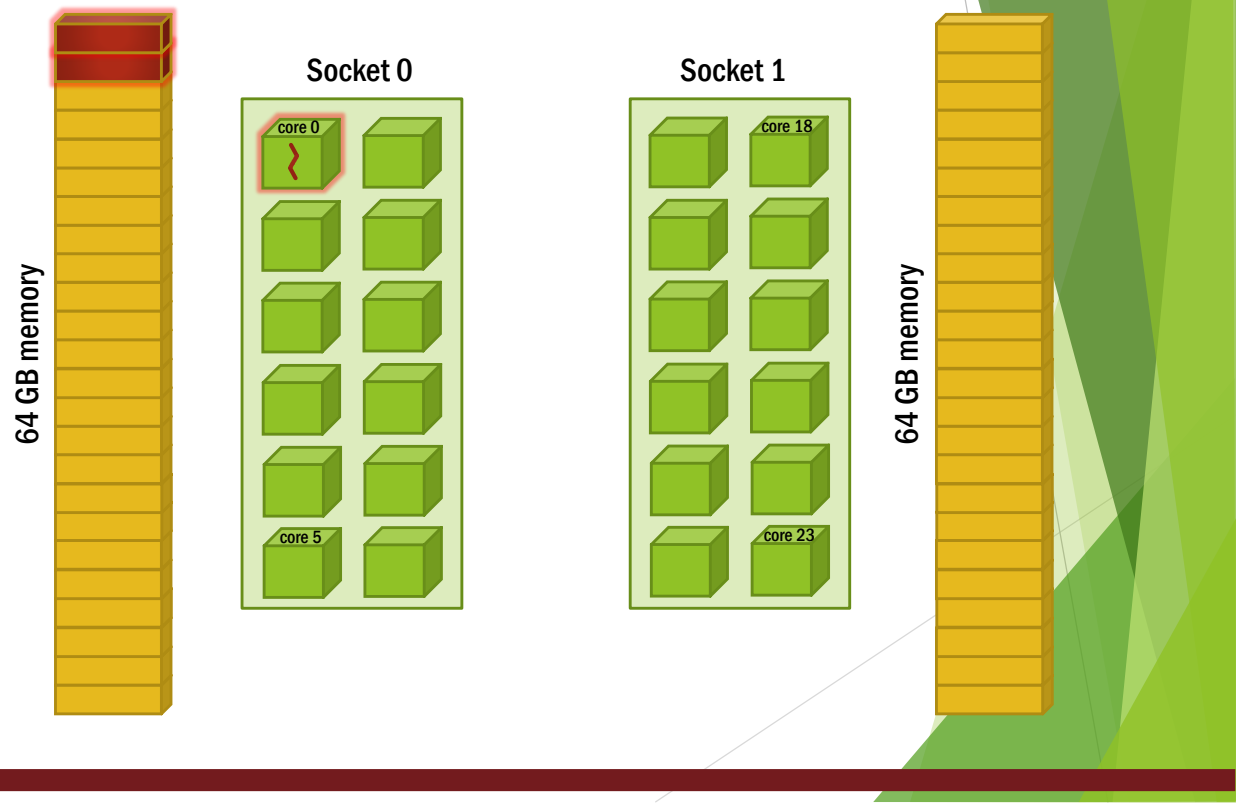


CPU Resource Allocation

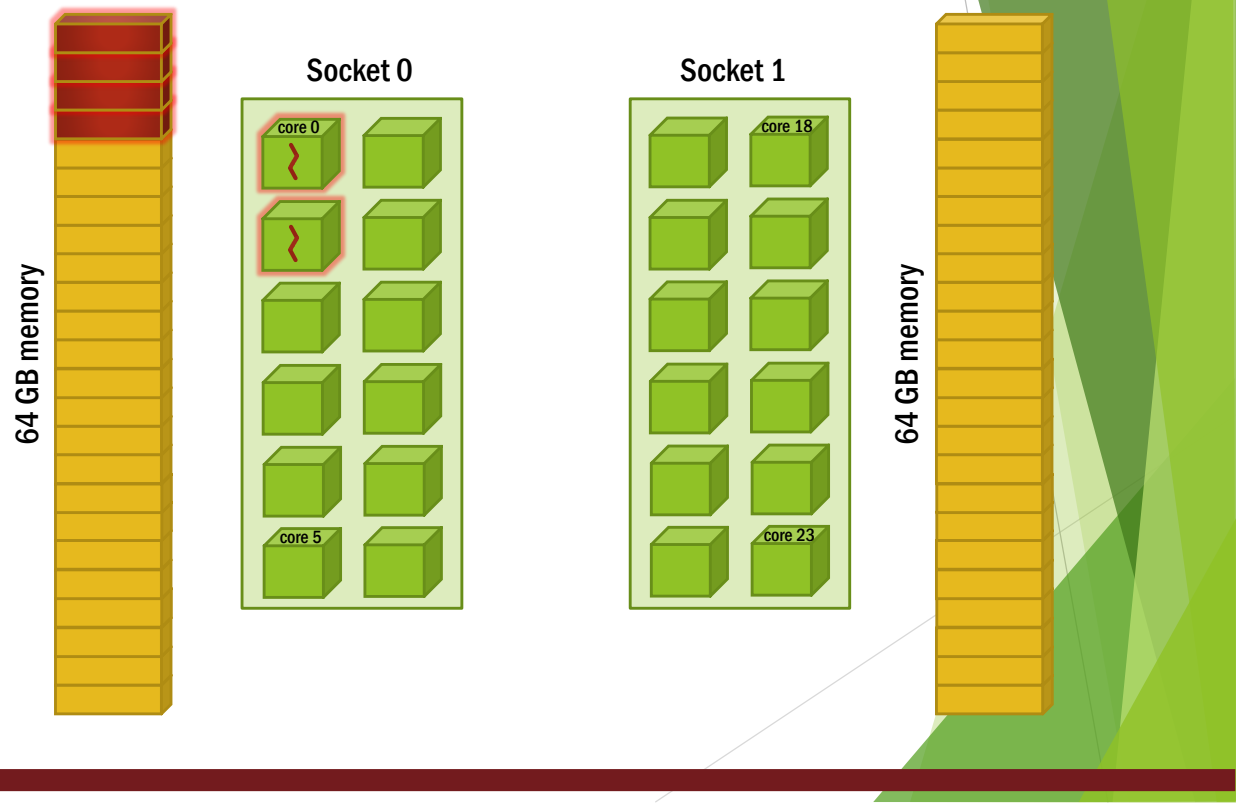


This is the default SLURM behavior! In most cases you don't want this behavior.

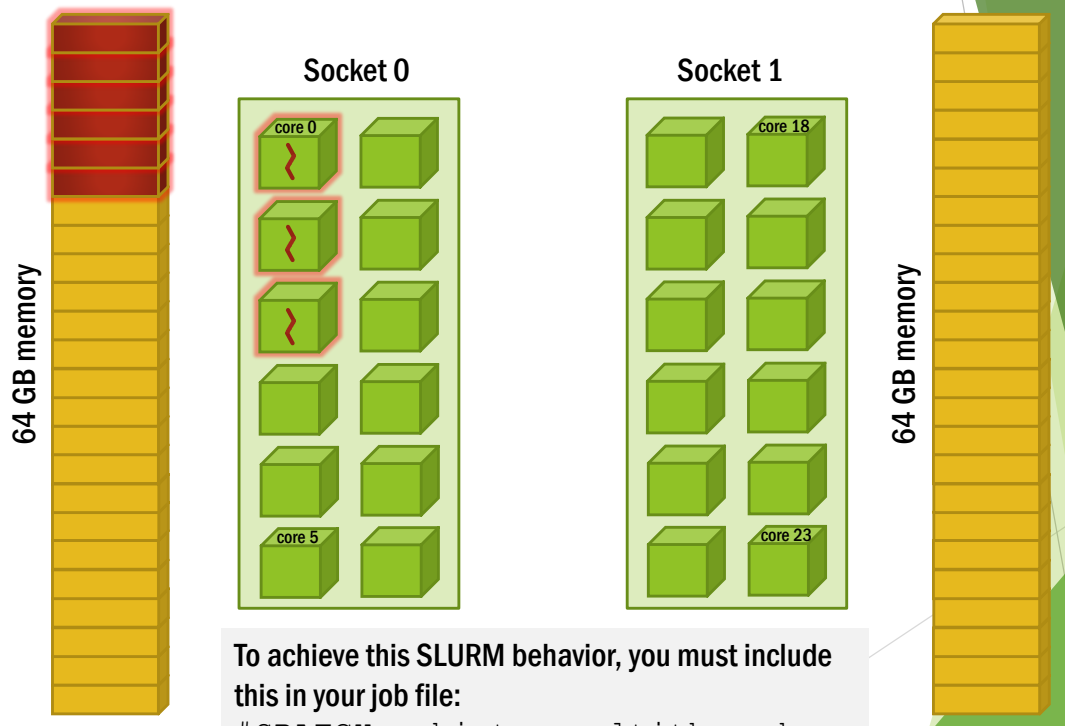
Resource Allocation with `--hint=nomultithread`



Resource Allocation with `--hint=nomultithread`



Resource Allocation with `--hint=nomultithread`



To achieve this SLURM behavior, you must include this in your job file:

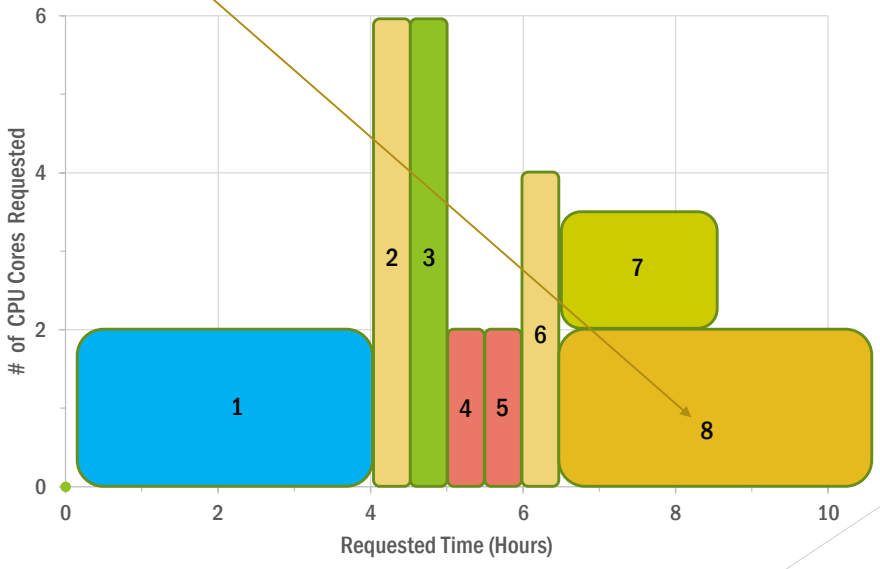
```
#SBATCH --hint=nomultithread
```

Resource Allocation vs. Resource Consumption

- With certain #SBATCH directives in your job file, you are simply booking resources.
- Once resources are assigned to you, how you use them or *whether* you use them is a different matter.
- In most cases, SLURM will do a good job of limiting over-utilization
  - Reserving 4 CPUs but attempting to use 5 or more instead
  - Booking 12 hours of time, but attempting to run for longer
  - Reserving 10 GB of memory but attempting to use more than that
- However, there is no straightforward solution to control deliberate under-utilization.
- In all circumstances, users must make efforts to avoid wasting resources.
- Deliberate wastage will be punished.

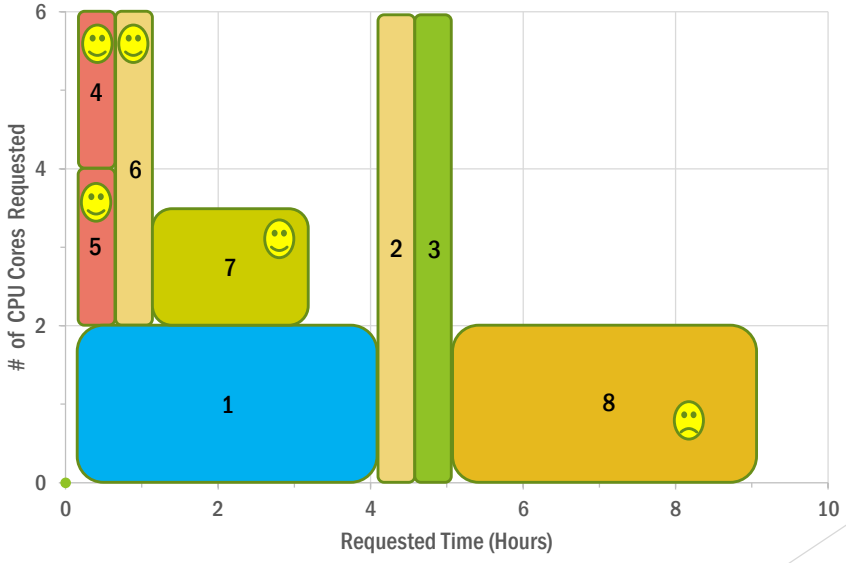
Excessive Resource Requests

- "My job will not take longer than 2 hours in the worst case ...but there's no harm in requesting 4 hours." Wrong!



Excessive Resource Requests

- Excessive resource requirements can prevent you from benefitting from scheduling optimizations that would otherwise allow your job to run earlier.



Using SLURM

What Resources Do I Need?

- Job's running time (walltime)
  - Aim for reasonably accurate estimate. Don't request excessive time without good reason.
- Partitions & QOS levels in the system
  - Which partition & QOS makes most sense for my job? Don't submit to the partition meant for workloads of a different type.
- Job's memory requirements
  - Aim for reasonably accurate estimate. Don't request excessive memory without good reason.
- Number of nodes or CPU cores required
  - Is my job serial, parallel? If parallel, how many CPUs or nodes should it be told to make use of?
- Job's environment
  - Do I need to configure any variables in my shell environment in preparation to run the job?

What's In a Job File?

- A SLURM batch job file is a text file with:
  - SLURM directives
  - Linux commands
- SLURM directives should always be at the beginning of the file and each directive line should start with `#SBATCH`
- These directives communicate your needs/requests to SLURM.
- SLURM also defines certain environment variables that may be accessed within a job file (e.g. working directory, shell, etc.).

Sample SLURM Job File

01_helloworld/run1.slurm

```
#!/bin/bash

#SBATCH --job-name=MyDemoJob
#SBATCH --partition=s_debug
#SBATCH --qos=sd
# This a comment. It is ignored.
#SBATCH --time=00:05:00
#SBATCH --ntasks=1

srun myprog.exe
```

01_helloworld/run2.slurm

```
#!/bin/bash

#SBATCH -J MyDemoJob
#SBATCH -p s_debug
#SBATCH --qos=sd
# This a comment. It is ignored.
#SBATCH -t 00:05:00
#SBATCH -n 1

srun myprog.exe
```

- A job file such as the one above can be created in any text editor
- A single # sign without the SBATCH keyword denotes a comment
- SBATCH directives usually have both a long and a short form (e.g. the 2 files above are functionally identical)

Job File Walk-Through

Line	Explanation
`#!/bin/bash`	The first line specifies the linux shell that is to interpret this file; 99% of the time there is no need to use anything other than "/bin/bash".
	Empty lines are ignored.
`#SBATCH -J MyDemoJob`	This simply helps identify running jobs using a name tag of sorts.
`#SBATCH -p s_debug`	This job will run in the s_debug partition.
`#SBATCH -qos=sd`	This job will run with the quality of service called "sd".
`# This is a comment. It is ignored.`	Anything to the right of a solitary # is a comment (except for "SBATCH" keyword).
`#SBATCH -t 00:05:00`	This job needs 5 minutes of walltime to do its thing.
`#SBATCH -n 1`	This job needs just a single CPU because it has only 1 task to launch.
`srun myprog.exe`	"myprog.exe" is the user program that needs to be run; it is launched using srun.

A line beginning with `# SBATCH` is a comment. One beginning with `#SBATCH` is a directive.

no space allowed here for directives!

Common SLURM Directives

Directive (long & short)	Description
--job-name=*name* **-J** *name*	A string used to label a job with a name. Here, *name* can be up to 15 printable, non-whitespace characters, and it will appear along with the job ID number when querying running jobs on the system. If left unspecified, the default will be the name of the job file.
--nodes=*count* **-N** *count*	Used to allocate *count* complete nodes (24 CPUs per node) to a job.
--partition=*partition* **-p** *partition*	The partition from which your job will be assigned resources.
--qos=qos_name	The quality of service level requested for your job.
--cpus-per-task=*count* **-c** *count*	Commonly used by OpenMP jobs, where each task might want to spawn *count* threads, for which *count* CPUs should be requested.
--ntasks=*count* **-n** *count*	Used to request an allocation of *count* CPUs because the job intends to launch *count* tasks within the job.
--error=*filename* **-e** *filename*	Write error & warning messages (stderr) to *filename*.
--output=*filename* **-o** *filename*	Write normal output (stdout) to *filename*. By default both standard output and standard error are directed to a file called "slurm-%j.out", where "%j" is the job ID.

Common SLURM Directives

Directive (long & short)	Description
--hint=multithread	View each hyper-thread as an independent CPU resource for allocation to jobs. With this option, SLURM will "see" 48 CPU's in a node. Only communication intensive applications will typically benefit from this.
--hint=nomultithread	View each hyper-thread pair as an independent CPU resource for allocation to jobs. With this option, SLURM will only "see" 24 CPU's in a node. Most compute intensive applications will want to use this.
--mail-type=*type*	Notify user by email when certain event types occur. Common type values include NONE, BEGIN, END, FAIL, REQUEUE, and ALL.
--mail-user=*email*	The email address of the user to be notified.
--nodelist=*nodenamelist* **-w** *nodenamelist*	Request a specific list of compute nodes for the job. The list may be specified in a format like `nid00[008,012]`. This option is typically only useful for users who want to submit additional jobs to a node already exclusively reserved for them by an earlier job which did not use all the node resources.
--mem=*count*	Request *count* megabytes of real memory per node for the job. Using the suffixes K or G one may request kilobytes or gigabytes instead (e.g. --mem=12G). The --mem and --mem-per-cpu options are mutually exclusive.
--mem-per-cpu=*count*	Request memory (in megabytes) required per allocated CPU. The default mem per cpu value is 5,350 MB when the --hint=nomultithread option is in effect, or 2,675 MB otherwise.

Common SLURM Commands

Command	What it does…
`squeue`	Generate listing of all existing jobs in the system
`sinfo`	Display the current state of partitions and nodes on the system (how busy they are)
`scancel`	Kill one or more existing jobs
`scontrol`	Display detailed job information
`sbatch`	Submit a job for execution on the system; allocate resources for the job
`srun`	Launch a task or program using the resources allocated by sbatch
`sacct`	Display current and historical job information

Detailed Documentation

https://slurm.schedmd.com/archive/slurm-15.08-latest/man_index.html

The squeue Command

Command Example	What it does…
`squeue`	List all jobs currently present on the system
`squeue -l`	List all jobs currently present on the system (more verbose)
`squeue –u fachaud74`	List all jobs from user fachaud74
`squeue –u fachaud74 --state=pending`	List all pending (i.e. waiting) jobs from user fachaud74
`squeue --state=running`	List all jobs on the system that are in the "running" state

Job States

PD = Pending
R = Running
CA = Cancelled
CG* = Completing
F = Failed

* A state of CG lasting longer than a few minutes will require admin intervention to clear from the system. Alert us via email when this happens to your job.

Output Formatting

The output from squeue is highly customizable. See the `–o` option in the squeue man page. To experiment, try:

```
squeue -o "%.12i %.9u %.10P %.14j %.3t %22r %.11L %.5D %.4C %.16J %N"
```

The `sinfo` Command

Command Example	What it does…
`sinfo`	Display information about slurm nodes and partitions
`sinfo -s`	Display only a partition state summary (w/out node details)
`sinfo -N`	Display information in a node-oriented format
`sinfo -n nid00010,nid00400`	Display information only about nodes nid00010 and nid00400
`sinfo -p l_long`	Display information only about the l_long partition

The sinfo command often repeats information in its output due to the fact that at our site multiple partitions are defined over the same set of nodes. This can be confusing!

Output Formatting

The output from sinfo is highly customizable. See the `-o` option in the sinfo man page. To experiment, try these customizations:

```
sinfo -o '%.10P %.15F %.20C %.12T'
sinfo -o '%.10P %.15F %.20C %.12T %N'
sinfo -o '%.10P %.15F %.20C   %N'
```

The `scancel` Command

Command Example	What it does...
`scancel 1234`	Kill the job with job ID 1234
`scancel -u fachaud74`	Kill all jobs from user fachaud74
`scancel -u fachaud74 -state=pending`	Kill all pending jobs from user fachaud74
`scancel -u fachaud74 -state=running`	Kill all running jobs from user fachaud74

- Users are permitted to kill only their own jobs
- The system administrator may kill any job
- In some cases, the user is unable to kill her jobs and the sys admin must be requested to intervene to clear them from the system

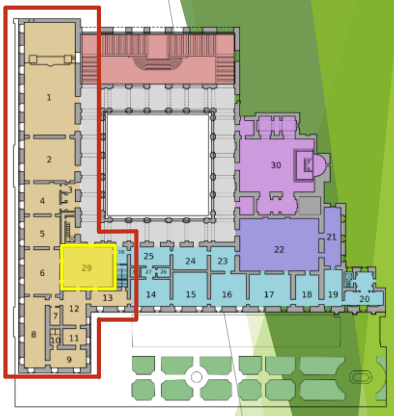
The `scontrol` Command

Command Example	What it does…
`scontrol show job 1234`	Display job information for job 1234
`scontrol show partition l_long`	Display configuration information about partition l_long

```
fachaud74@raad2b:~>  scontrol show job 1171482
JobId=1171482 JobName=myTest
   UserId=fachaud74(7001) GroupId=rc.users(6001)
   Priority=346 Nice=0 Account=default QOS=ll
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=6-15:39:31 TimeLimit=6-23:00:00 TimeMin=N/A
   SubmitTime=2017-11-20T15:03:09 EligibleTime=2017-11-20T15:03:09
   StartTime=2017-11-20T22:13:42 EndTime=2017-11-27T21:13:42
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=l_long AllocNode:Sid=raad2-int1:31795
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=nid00[051,228,247,405]
   BatchHost=nid00051
   NumNodes=4 NumCPUs=192 CPUs/Task=1 ReqB:S:C:T=0:0:*:1
   TRES=cpu=192,mem=513600,node=4
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=2675M MinTmpDiskNode=0
   Features=(null) Gres=craynetwork:1 Reservation=(null)
   Shared=USER Contiguous=0 Licenses=(null) Network=(null)
   Command=/lustre/home/fachaud74/myTest.job
   WorkDir=/lustre/home/fachaud74
   StdErr=/lustre/home/fachaud74/myTest.o1171482
   StdIn=/dev/null
   StdOut=/lustre/home/fachaud74/myTest.o1171482
   Power= SICP=0
```

sbatch & srun -- The difference?

- Sbatch is like the travel agency that books an entire block of hotel rooms for a given holiday season, and then offers individual rooms to customers over that period.
- Srun is like the vacationer who books a room for some time from a particular travel agency.
- Normally, an sbatch command will reserve a collection of resources (e.g. 4 compute nodes) for a particular job.
  - One -- or multiple -- srun's can be issued within a job file to make use of subsets of the resources reserved by the sbatch
  - srun cannot request resources greater in quantity than what the corresponding sbatch has reserved for the job
- However, srun can independently launch a job (without sbatch)
  - In such cases, though, the system automatically constructs an appropriate sbatch transparently, without explicit user involvement



The `sbatch` Command

- Typically this command is issued with a batch file name as its only argument:
  `sbatch myBatchFile`
- However, it may also accept command line options (like those found in the #SBATCH directives inside the batch file itself).
- The sbatch command must be used to submit the job to the supercomputer
- Once accepted by SLURM, the job is assigned a unique job ID
- SLURM will begin job execution only when resources become available AND when the job's priority takes it to the head of the list of pending jobs.

```
[fachaud74@raad2 ~]$ sbatch barebones1.slurm
Submitted batch job 1147521
[fachaud74@raad2 ~]$
```

SLURM job file name

Job ID

The `srun` Command

- srun is a program launcher
  - typically used inside a job file to launch any programs you need to run as part of that job
- srun launches job "steps"
  - A job may consist of a sequence of steps, and each step could be launched by a different srun
  - A job may also consist of multiple parallel streams of execution, and srun can launch these parallel tasks as well, doing what programs like mpirun would otherwise do
  - Slurm accounting logs record information about job steps, and not just about the job as a whole
- Launching programs without srun will often work, but sometimes leads to tricky problems; as a rule of thumb, ALWAYS use srun
  - Jobs may not terminate properly and get stuck in CG state
  - Generally, instances of misbehavior are better managed by srun

The `srun` Command

Command Example	What it does...
`srun prog.exe`	Launch the program prog.exe
`srun --ntasks=12 prog.exe`	Launch prog.exe using 12 hyper-threads
`srun --hint=nomultithread -n 12 prog.exe`	Launch prog.exe with 12 cores
`srun --propagate=STACK,MEMLOCK prog.exe`	Launch prog.exe and propagate these two process limits

- An srun appearing within a job file will inherit certain default values from the #SBATCH directives of that job.
- For instance, an `srun prog.exe` inside a job file containing an `#SBATCH -n 24` directive will try to run prog.exe on 24 cores or threads
  - But if we issued an `srun -n 12 prog.exe` it will use only 12 cores or threads

Interactive Jobs

- Use of interactive jobs should be kept to a minimum, and used only for testing and development needs
- Interactive jobs should use the least amount of resources sufficient for successful development or testing
  - Do not request more than 2 nodes with an interactive job
  - Do not request more than an hour of walltime when using the production queues
  - Whenever possible, submit only to the s_debug or express partitions
  - These limitations are essentially rules of good citizenship, even though they are not enforced by system configuration at this time
- Use of interactive jobs to launch programs with graphical interfaces is **not supported** by RC
  - Technically you can do it, but we will not spend time fixing issues or problems with such use if and when you encounter problems

Interactive Jobs

- To launch an interactive job, you much first ssh into one of the "internal login nodes" (`raad2-int1` or `raad2-int2`). Here is a sample sequence of commands…

```
fachaud74@raad2b:~> ssh raad2-int1
Last login: Thu Nov 23 11:01:47 2017 from nid00002
fachaud74@raad2-int1:~> srun --pty --qos sd -p s_debug -t 00:30:00 -n 2 --hint=nomultithread /bin/bash
fachaud74@nid00388:~> echo hello world!
hello world!
fachaud74@nid00388:~> exit
exit
fachaud74@raad2-int1:~> exit
logout
Connection to raad2-int1 closed.
fachaud74@raad2b:~>
```

- Above, we request 2 cores in the s_debug partition for a 30 minute session

The `sacct` Command

Command Example	What it does…
`sacct –u fachaud74`	Produce a listing of user fachaud74's past and current jobs
`sacct -e`	Display a list of possible column headers usable with the "–o" option (which customizes the sacct report)
`sacct –u fachaud74 \` `  -o "JobID,JobName,CPUTime,Elapsed"`	Display the job ID, job name, CPU time, and walltime consumed by jobs from user fachaud74
`sacct –S 2017-07-01 –u fachaud74`	Display fachaud74's job report starting from Jul 1, 2017

- By default, sacct reports only on jobs beginning at 12am on the current day
  - Use `–S` and `–E` to control the time period for which a report is required
- Some of the values sacct can report on (using the `–o` option):
  - `AllocCPUS` `AllocGRES` `AllocNodes` `AllocTRES`
    `Account` `AssocID` `AveCPU` `CPUTime`
    `Elapsed` `Eligible` `End` `ExitCode`
    `GID` `Group` `JobID` `JobName`
    `NCPUS` `NNodes` `NodeList` `Ntasks`
    `Priority` `Partition` `QOS` `ReqCPUS`
    `ReqGRES` `ReqMem` `ReqNodes` `ReqTRES`
    `Reservation` `ReservationId` `Start` `State`
    `Submit` `Suspended` `SystemCPU` `Timelimit`
    `TotalCPU` `UID` `User` `UserCPU`

TEXAS A&M UNIVERSITY at QATAR

Managing the User Environment

Application Environment

- On raad2, the user environment for most commercial applications is managed by the “module” utility.
- Applications typically require environment variables to be set for proper operation. Users could do this manually with a series of relevant commands.
- However, with the `module` command, users need only to “load” or “unload” pre-defined scripts (called modules) to automatically perform (or undo) the required setup.

Common Module Commands

TEXAS A&M UNIVERSITY at QATAR

Command	Description
`module avail`	Show available modules
`module list`	Show currently loaded modules
`module load` *`modulename`*	Load a module
`module unload` *`modulename`*	Unload a module
`module switch` *`module1 module2`*	Unload *module1* and load *module2* in its place
`module help`	List the available module commands
`module help` *`modulename`*	Display the help information for a module
`module display` *`modulename`*	Show specific changes that would be made by the *modulename* script (without making them)

Module Help

```
fachaud74@raad2b:~> module help

  Modules Release 3.2.10.3 2012-12-21 (Copyright GNU GPL v2 1991):

  Usage: module [ switches ] [ subcommand ] [subcommand-args ]

Switches:
        -H|--help               this usage info
        -V|--version            modules version & configuration options
        -f|--force              force active dependency resolution
        -t|--terse              terse    format avail and list format
        -l|--long               long     format avail and list format
        -h|--human              readable format avail and list format
        -v|--verbose            enable  verbose messages
        -s|--silent             disable verbose messages
        -c|--create             create caches for avail and apropos
        -i|--icase              case insensitive
        -u|--userlvl <lvl>      set user level to (nov[ice],exp[ert],adv[anced])
  Available SubCommands and Args:
        + add|load              modulefile [modulefile ...]
        + rm|unload             modulefile [modulefile ...]
        + switch|swap           [modulefile1] modulefile2
        + display|show          modulefile [modulefile ...]
        + avail                 [modulefile [modulefile ...]]
        + use [-a|--append]     dir [dir ...]
        + unuse                 dir [dir ...]
        + update
        + refresh
        + purge
        + list
        + clear
        + help                  [modulefile [modulefile ...]]
        + whatis                [modulefile [modulefile ...]]
        + apropos|keyword       string
        + initadd               modulefile [modulefile ...]
        + initprepend           modulefile [modulefile ...]
        + initrm                modulefile [modulefile ...]
        + initswitch            modulefile1 modulefile2
        + initlist
        + initclear
```

`module help`

lists available module command options

Module Help

```
fachaud74@raad2b:~> module help gcc/5.2.0

----------- Module Specific Help for 'gcc/5.2.0' ------------------

gcc 5.2.0
=========

Release Date:
-------------
December 3, 2015

Purpose:
--------
The gcc 5.2.0 release.

Product and OS Dependencies:
----------------------------
.
.
.
[ output truncated for brevity ]
.
.
.
```

`module help` *`modulename`*

shows help information specific to a specific module

Listing Available Modules

`module avail` lists modules available on this system

```
fachaud74@raad2b:~> module avail

------------------------------------------------- /lustre/opt/modulefiles/tamuq -------------------------------------------------
PrgEnv-intel/17.1.132/64bit   gromacs/465-gnu              intel/compiler/17.1.132/64bit python/279
astrometry/067                gromacs/465-intel            intel/mkl/17.1.132/64bit      singularity/2.2
cmake/330                     gromacs/514                  intel/mpi/17.1.132/64bit      singularity/221

------------------------------------------------- /lustre/sw/xc40ac/modulefiles -------------------------------------------------
JDK/7u04        abaqus/2016     ansys/145       ansys/182       mathematica/901 matlab/r2016a
JDK/8u131       abaqus/613      ansys/162       lammps/17Nov16  matlab/r2014a   openfoam/1612+

---------------------------------------------- /opt/cray/craype/default/modulefiles ----------------------------------------------
craype-aarch64              craype-hugepages128K       craype-hugepages8M         craype-network-gemini
craype-abudhabi             craype-hugepages128M       craype-intel-knc           craype-network-gige
craype-abudhabi-cu          craype-hugepages16M        craype-interlagos          craype-network-infiniband
craype-accel-host           craype-hugepages256M       craype-interlagos-cu       craype-network-none
craype-accel-nvidia20       craype-hugepages2M         craype-istanbul            craype-sandybridge
craype-accel-nvidia35       craype-hugepages32M        craype-ivybridge           craype-shanghai
craype-arm-thunderx         craype-hugepages4M         craype-mc12                craype-target-compute_node
craype-barcelona            craype-hugepages512K       craype-mc8                 craype-target-local_host
craype-broadwell            craype-hugepages512M       craype-mic-knl             craype-target-native
craype-haswell              craype-hugepages64M        craype-network-aries

------------------------------------------------- /opt/cray/ari/modulefiles -------------------------------------------------
alps/5.2.4-2.0502.9774.31.11.ari(default)       pmi/5.0.10-1.0000.11050.0.0.ari(default)
configuration/1.0-1.0502.60535.1.2.ari(default) pmi/5.0.11
crash/7.1.0-1.0502.61934.1.1.ari                pmi-lib/5.0.11
crash_utility/1.0-1.0502.62977.3.1.ari          rca/1.0.0-2.0502.60530.1.62.ari(default)
dmapp/7.0.1-1.0502.11080.8.76.ari(default)      sdb/1.1-1.0502.63652.4.25.ari(default)
dvs/2.5_0.9.0-1.0502.2188.1.116.ari(default)    shared-root/1.0-1.0502.60523.1.31.ari(default)
gni-headers/4.0-1.0502.10859.7.8.ari(default)   switch/1.0-1.0502.60522.1.61.ari(default)
hosts/1.0-1.0502.60484.1.61.ari(default)        sysutils/1.0-1.0502.60492.1.1.ari(default)
krca/1.0.0-2.0502.63139.4.31.ari(default)       udreg/2.3.2-1.0502.10518.2.17.ari(default)
lbcd/2.1-1.0502.60476.1.1.ari(default)          ugni/6.0-1.0502.10863.8.29.ari(default)
logcb/1.0-1.0502.60472.1.1.ari(default)         wlm_detect/1.0-1.0502.64649.2.1.ari(default)
nodehealth/5.1-1.0502.64995.8.11.ari(default)   wlm_trans/1.0-1.0502.64650.3.9.ari(default)
nodestat/2.2-1.0502.60539.1.31.ari(default)     xpmem/0.1-2.0502.64982.5.3.ari(default)
pdsh/2.26-1.0502.60659.2.1.ari(default)
```

Listing Loaded Modules

TEXAS A&M UNIVERSITY at QATAR

`module list`

lists the currently loaded modules

```
fachaud74@raad2b:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.3                      8) udreg/2.3.2-1.0502.10518.2.17.ari     15) alps/5.2.4-2.0502.9774.31.11.ari
  2) eswrap/1.3.3-1.020200.1278.0          9) ugni/6.0-1.0502.10863.8.29.ari        16) rca/1.0.0-2.0502.60530.1.62.ari
  3) switch/1.0-1.0502.60522.1.61.ari     10) pmi/5.0.10-1.0000.11050.0.0.ari       17) atp/1.8.3
  4) craype-network-aries                 11) dmapp/7.0.1-1.0502.11080.8.76.ari     18) PrgEnv-cray/5.2.82
  5) cce/8.4.3                            12) gni-headers/4.0-1.0502.10859.7.8.ari  19) craype-haswell
  6) craype/2.5.1                         13) xpmem/0.1-2.0502.64982.5.3.ari        20) cray-mpich/7.3.1
  7) cray-libsci/13.3.0                   14) dvs/2.5_0.9.0-1.0502.2188.1.116.ari
fachaud74@raad2b:~>
```

Loading & Unloading Modules

```
fachaud74@raad2b:~> matlab -nojvm -nodisplay -nosplash
If 'matlab' is not a typo you can run the following command to lookup the package that contains the binary:
    command-not-found matlab
-bash: matlab: command not found
fachaud74@raad2b:~> module load matlab/r2014a
fachaud74@raad2b:~> matlab -nojvm -nodisplay -nosplash

                                                  < M A T L A B (R) >
                                        Copyright 1984-2014 The MathWorks, Inc.
                                          R2014a (8.3.0.532) 64-bit (glnxa64)
                                                   February 11, 2014


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> exit
fachaud74@raad2b:~> module unload matlab/r2014a
fachaud74@raad2b:~> module load matlab/r2016a
fachaud74@raad2b:~> matlab -nojvm -nodisplay -nosplash

                                                  < M A T L A B (R) >
                                        Copyright 1984-2016 The MathWorks, Inc.
                                        R2016a (9.0.0.341360) 64-bit (glnxa64)
                                                   February 11, 2016


For online documentation, see http://www.mathworks.com/support
For product information, visit www.mathworks.com.

>> exit
fachaud74@raad2b:~>
```

`module load` *`modulename`*

Applies changes needed for the system to be able to locate & launch the application referenced by *modulename*

`module unload` *`modulename`*

Reverses the changes applied by the "module load *modulename*" command

Note that the **`load`** and **`unload`** sub-commands to module will only produce output if an error is encountered.

Module Switching

`module switch module1 module2`

replaces loaded module *module1* with new module *module2*

```
fachaud74@raad2b:~> module load gcc/5.2.0
fachaud74@raad2b:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.3                     8) udreg/2.3.2-1.0502.10518.2.17.ari     15) alps/5.2.4-2.0502.9774.31.11.ari
  2) eswrap/1.3.3-1.020200.1278.0         9) ugni/6.0-1.0502.10863.8.29.ari        16) rca/1.0.0-2.0502.60530.1.62.ari
  3) switch/1.0-1.0502.60522.1.61.ari    10) pmi/5.0.10-1.0000.11050.0.0.ari       17) atp/1.8.3
  4) craype-network-aries                11) dmapp/7.0.1-1.0502.11080.8.76.ari     18) PrgEnv-cray/5.2.82
  5) cce/8.4.3                           12) gni-headers/4.0-1.0502.10859.7.8.ari  19) craype-haswell
  6) craype/2.5.1                        13) xpmem/0.1-2.0502.64982.5.3.ari        20) cray-mpich/7.3.1
  7) cray-libsci/13.3.0                  14) dvs/2.5_0.9.0-1.0502.2188.1.116.ari   21) gcc/5.2.0
fachaud74@raad2b:~> which gcc
/opt/gcc/5.2.0/bin/gcc
fachaud74@raad2b:~> module sw gcc/5.2.0 gcc/6.3.0
fachaud74@raad2b:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.3                     8) udreg/2.3.2-1.0502.10518.2.17.ari     15) alps/5.2.4-2.0502.9774.31.11.ari
  2) eswrap/1.3.3-1.020200.1278.0         9) ugni/6.0-1.0502.10863.8.29.ari        16) rca/1.0.0-2.0502.60530.1.62.ari
  3) switch/1.0-1.0502.60522.1.61.ari    10) pmi/5.0.10-1.0000.11050.0.0.ari       17) atp/1.8.3
  4) craype-network-aries                11) dmapp/7.0.1-1.0502.11080.8.76.ari     18) PrgEnv-cray/5.2.82
  5) cce/8.4.3                           12) gni-headers/4.0-1.0502.10859.7.8.ari  19) craype-haswell
  6) craype/2.5.1                        13) xpmem/0.1-2.0502.64982.5.3.ari        20) cray-mpich/7.3.1
  7) cray-libsci/13.3.0                  14) dvs/2.5_0.9.0-1.0502.2188.1.116.ari   21) gcc/6.3.0
fachaud74@raad2b:~> which gcc
/opt/gcc/6.3.0/bin/gcc
fachaud74@raad2b:~>
```

Module: How it Works

```
fachaud74@raad2b:~> echo $PATH
/opt/cray/mpt/7.3.1/gni/bin:/opt/cray/rca/1.0.0-2.0502.60530.1.62.ari/bin:/opt/cray/alps/5.2.4-
2.0502.9774.31.11.ari/sbin:/opt/cray/dvs/2.5_0.9.0-1.0502.2188.1.116.ari/bin:/opt/cray/xpmem/0.1-
2.0502.64982.5.3.ari/bin:/opt/cray/pmi/5.0.10-1.0000.11050.0.0.ari/bin:/opt/cray/ugni/6.0-
1.0502.10863.8.29.ari/bin:/opt/cray/udreg/2.3.2-
1.0502.10518.2.17.ari/bin:/opt/cray/craype/2.5.1/bin:/opt/cray/cce/8.4.3/cray-binutils/x86_64-unknown-linux-
gnu/bin:/opt/cray/cce/8.4.3/craylibs/x86-
64/bin:/opt/cray/cce/8.4.3/cftn/bin:/opt/cray/cce/8.4.3/CC/bin:/opt/cray/switch/1.0-
1.0502.60522.1.61.ari/bin:/opt/cray/eslogin/eswrap/1.3.3-
1.020200.1278.0/bin:/opt/modules/3.2.10.3/bin:/lustre/home/fachaud74/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:
/usr/X11R6/bin:/usr/games:/usr/lib/mit/bin:/usr/lib/mit/sbin:/sbin:/usr/sbin:.:/usr/lib/qt3/bin:/opt/cray/bin
fachaud74@raad2b:~> module load gcc/6.3.0
fachaud74@raad2b:~> echo $PATH
/opt/gcc/6.3.0/bin:/opt/cray/mpt/7.3.1/gni/bin:/opt/cray/rca/1.0.0-2.0502.60530.1.62.ari/bin:/opt/cray/alps/5.2.4-
2.0502.9774.31.11.ari/sbin:/opt/cray/dvs/2.5_0.9.0-1.0502.2188.1.116.ari/bin:/opt/cray/xpmem/0.1-
2.0502.64982.5.3.ari/bin:/opt/cray/pmi/5.0.10-1.0000.11050.0.0.ari/bin:/opt/cray/ugni/6.0-
1.0502.10863.8.29.ari/bin:/opt/cray/udreg/2.3.2-
1.0502.10518.2.17.ari/bin:/opt/cray/craype/2.5.1/bin:/opt/cray/cce/8.4.3/cray-binutils/x86_64-unknown-linux-
gnu/bin:/opt/cray/cce/8.4.3/craylibs/x86-
64/bin:/opt/cray/cce/8.4.3/cftn/bin:/opt/cray/cce/8.4.3/CC/bin:/opt/cray/switch/1.0-
1.0502.60522.1.61.ari/bin:/opt/cray/eslogin/eswrap/1.3.3-
1.020200.1278.0/bin:/opt/modules/3.2.10.3/bin:/lustre/home/fachaud74/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:
/usr/X11R6/bin:/usr/games:/usr/lib/mit/bin:/usr/lib/mit/sbin:/sbin:/usr/sbin:.:/usr/lib/qt3/bin:/opt/cray/bin
fachaud74@raad2b:~>
```

Among other things, the module command adds and removes values to certain environment variables such as PATH, MANPATH, and LD_LIBRARY_PATH to enable the shell to locate various components of an application.

TEXAS A&M UNIVERSITY at QATAR

SLURM by Example

Examples Directory

- /lustre/share/examples/slurm-tutorial
  - 01_helloworld
  - 02_multi_helloworld
  - 03_mpi_matmult
  - 04_multi_sruns
  - 05_interactive_job
  - 06_mpi_matmult_multinode
  - 07_tidbits
  - 08_openmp
  - 09_matlab
  - 10_ansys

SLURM Example Files

- **README** Text file with instructions relevant to the example
- ***.c** C source code files
- ***.f** Fortran source code files
- ***.exe** Executable files
- ***.slurm** SLURM job scripts