

# Introduction to Computing on Raad2

Research Computing Core Group (RCCG)  
Office of the Vice President for Research (OVPR)  
March, 2026



عضو في مؤسسة قطر  
Member of Qatar Foundation

# 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 course.

# How & Where to Seek Assistance

---

- Research Computing Core Group Website
  - <https://rccg.hbku.edu.qa/>
  - Provides all information/documentation about our services, resources, and user support
  - [https://rccg.hbku.edu.qa/wiki/User\\_accounts](https://rccg.hbku.edu.qa/wiki/User_accounts)
  - Use this to apply for accounts on the system(s)
- Service desk (email queries)
  - [support@hbku.edu.qa](mailto:support@hbku.edu.qa)
  - 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.
    - Provide a clear description of the problem
- Walk-in assistance
  - We are in the Manufactory (TAMUQ building), suite 125
  - Will assist immediately if available, or schedule an appointment with you if busy

---

# **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, integrated computing resource
  - **A supercomputer is not a general-purpose computer, and should not be expected to be one**

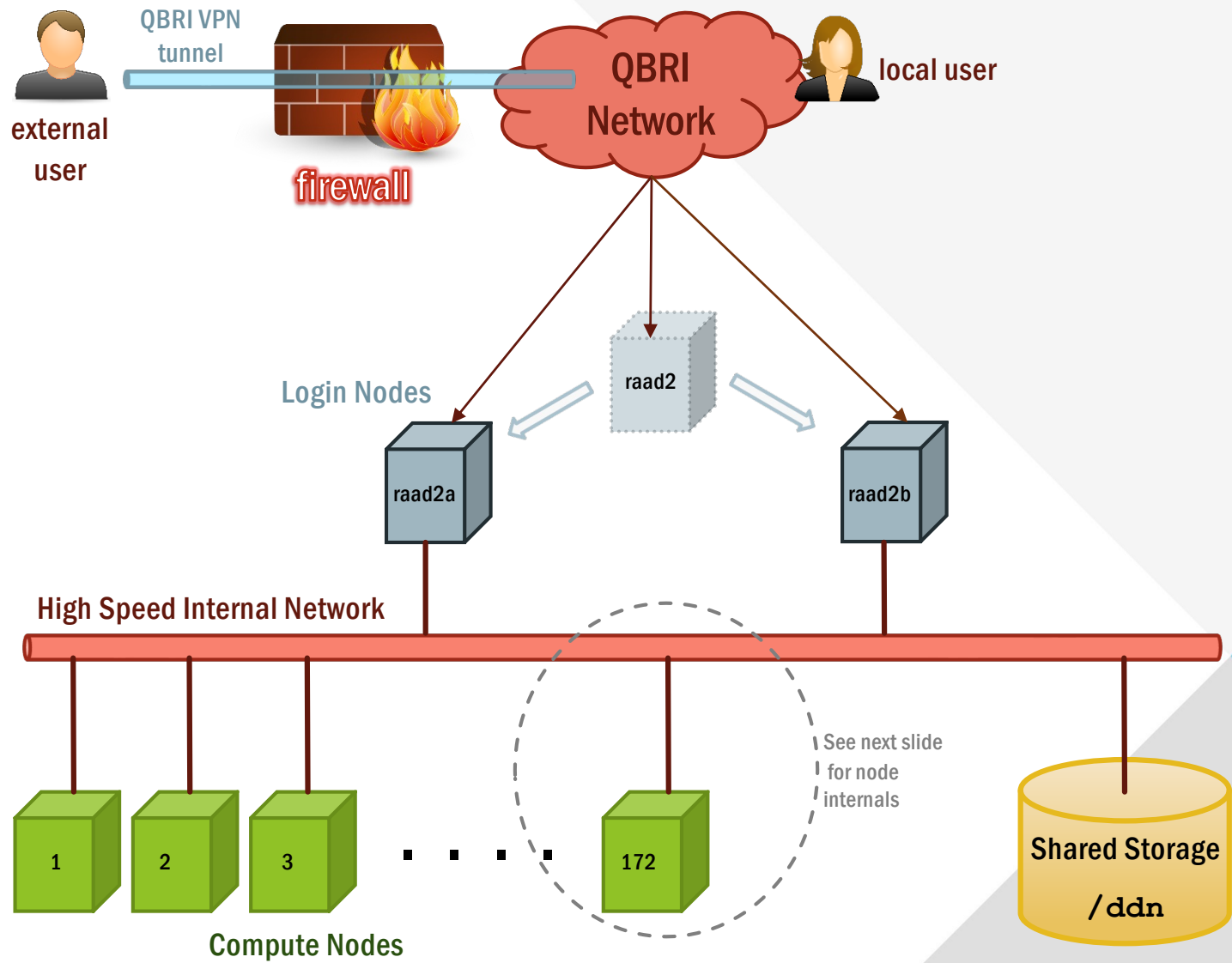


# What is a Supercomputer?

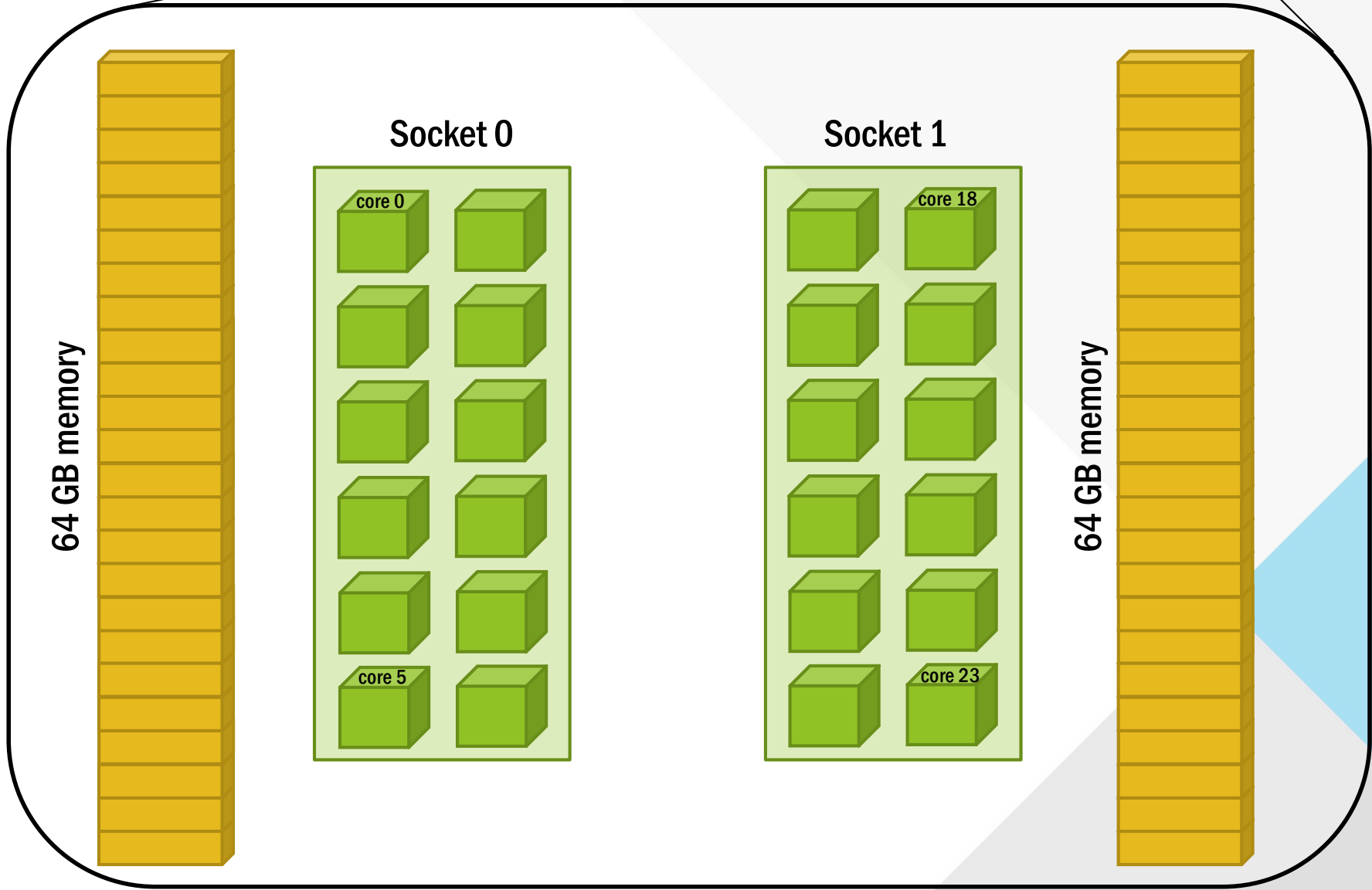
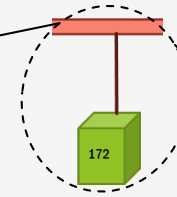
- Generic “big picture” description
  - A type of parallel or distributed processing system, which consists of a collection of interconnected computing elements working together as a single, integrated computing resource
    - A supercomputer is not a general purpose computer, and should not be expected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



# Compute Node Internals

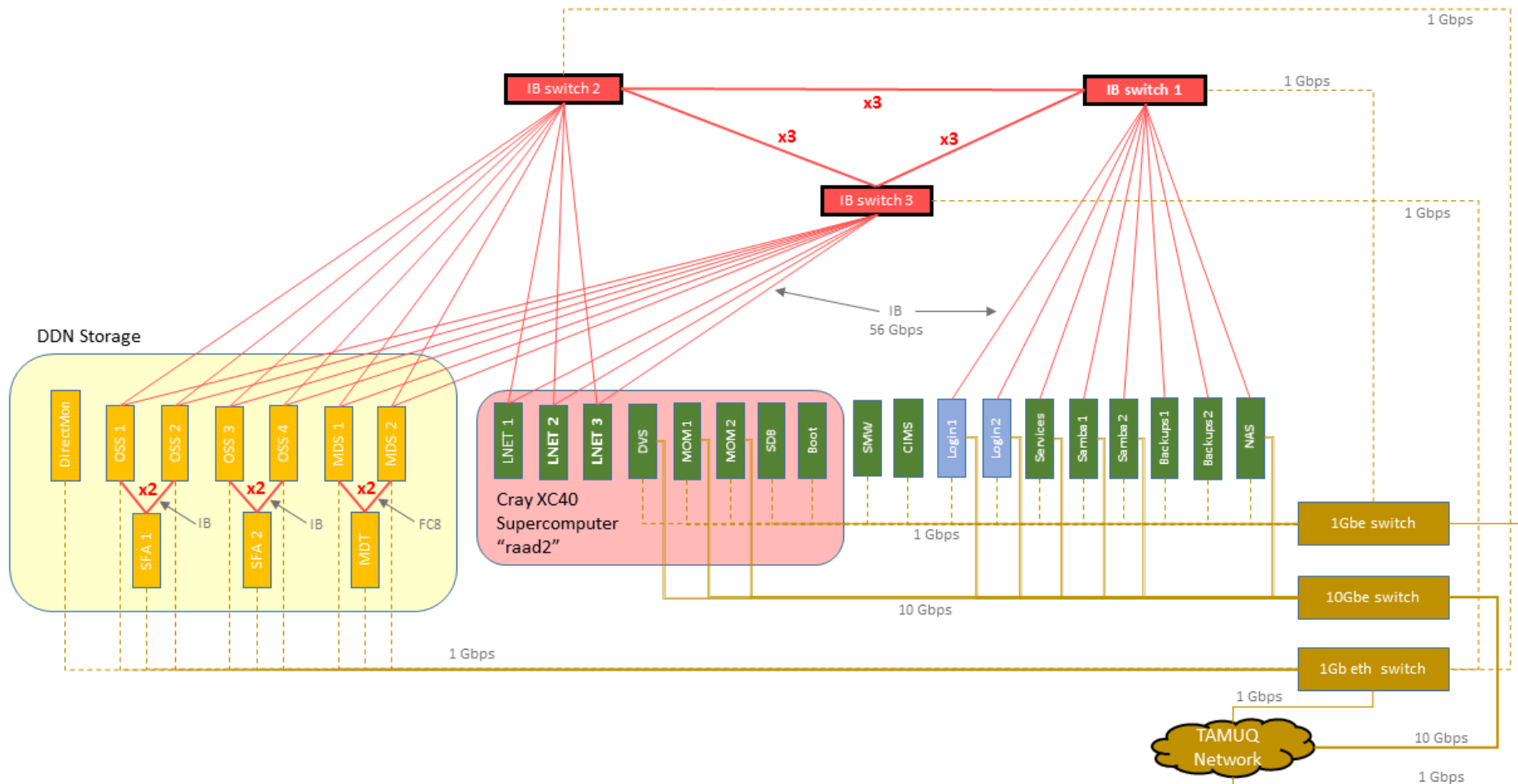


---

# **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, VASP, etc.)
  - Open source packages (LAMMPS, Gromacs, OpenFOAM, etc.)

# Raad2 Details

## Technical Specs

<b>Host Name</b>	raad2.biolab.local
<b>Batch/Resource Scheduler</b>	SLURM 20.02.7
<b>Operating System</b>	Cray Linux Environment (CLE) 7.0 – SLES 15 SP3
<b>Number of Nodes</b>	172
<b>Aggregate Number of CPU Cores</b>	4,128
<b>Interconnect</b>	Aries
<b>Aggregate System Memory</b>	22 TB
<b>Peak Performance</b>	120+ TFLOPS
<b>Local Compute Node Disks</b>	none
<b>Shared Storage System Capacity</b>	800 TB (usable)
<b>Parallel Filesystem</b>	Lustre 2.15

# Raad2 Compute Nodes

---

- Compute nodes are named **nid00xyz**
  - xyz is a contiguous set of 3-digit integers (e.g. 008, 009, 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 `/ddn/home/username`
  - e.g. `/ddn/home/irtambo19`
- Your default disk quota is 500GB
  - Quota extensions shall NOT be granted on the home directories
- 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 incremental remain available
- A scratch directory for use by batch jobs is available under `/ddn/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 `/ddn/scratch` directories are NOT backed up.

# Warehouse & Project Directories

---

- Additional space can be granted in a so-called “warehouse” directory in `/ddn/home/username`
  - e.g. `/ddn/warehouse/irtambo19`
- Your default disk quota here is 500GB
  - Quota extensions can be considered here with strong justification
- Warehouse directory backups are performed *less* frequently
  - Full backups after every 8-week interval, on a Friday
  - Incremental backups after every 1-week interval
  - At any given time, the 2 most recent full backups & associated incrementals remain available
- A “project” directory for use by research groups can be created under `/ddn/projects/foldername`
  - This is suitable as a data sharing space among group members
- The `/ddn/projects` directories may be backed up on a case-by-case basis

# 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://rccg.hbku.edu.qa/wiki/Application\\_Guides](https://rccg.hbku.edu.qa/wiki/Application_Guides)

# Remote Access to Raad2

---

- VPN required for access
  - First “log in” to the QBRI network, then log in to the Raad2 system
  - VPN enhances protection against network-based attacks
  - Detailed instructions on establishing a VPN connection
  - <https://rccg.hbku.edu.qa/wiki/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 with built-in
  - The recommended remote access utility for RC users
- Raad2 runs the Linux OS. For Linux training, see:
  - [https://rccg.hbku.edu.qa/wiki/Intro\\_linux](https://rccg.hbku.edu.qa/wiki/Intro_linux)

# EXERCISE: Log in to Raad2

---

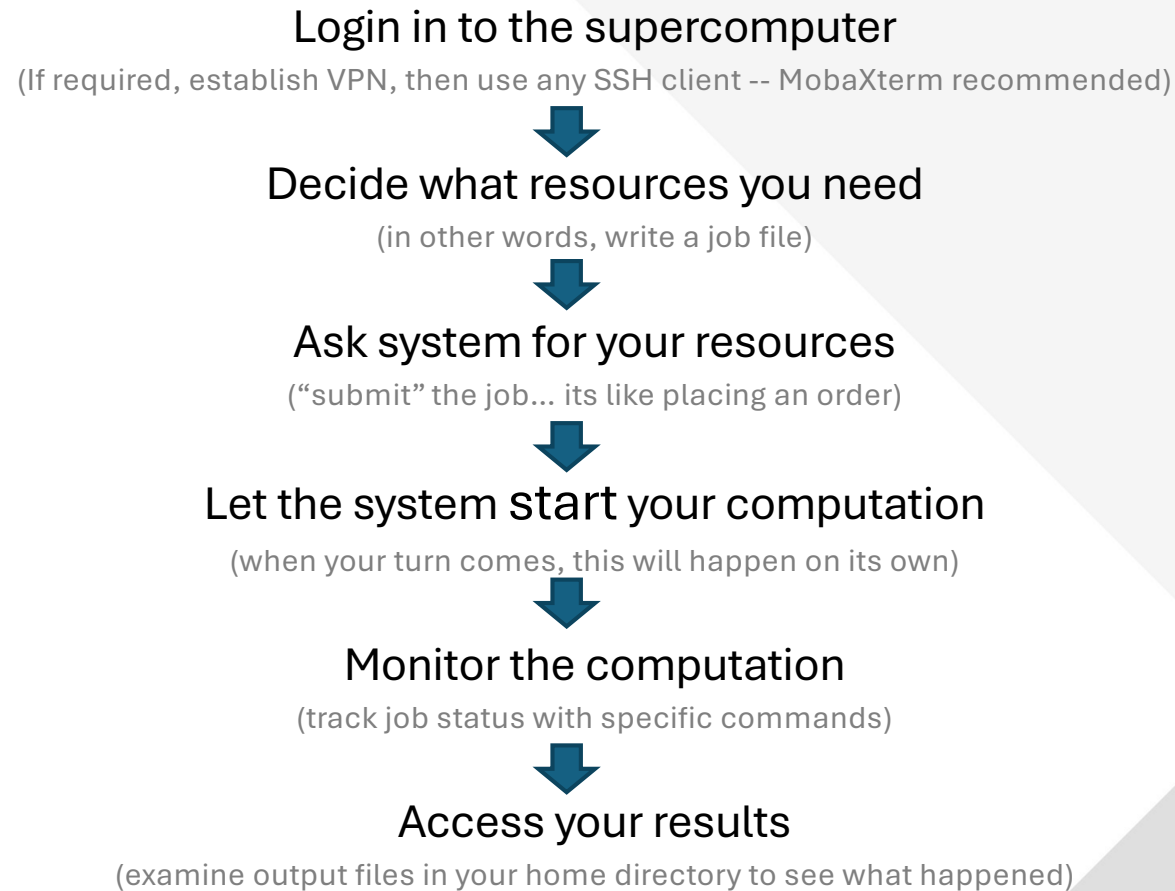
- Following the instructor, set up your session to raad2 using MobaXterm
  - Free software to remotely access linux systems from windows PCs
  - <https://mobaxterm.mobatek.net/download-home-edition.html>
  - [https://rc-docs.qatar.tamu.edu/wiki/Soft\\_mobaxterm#Installation](https://rc-docs.qatar.tamu.edu/wiki/Soft_mobaxterm#Installation)
- Login in to raad2 using the session created earlier
  - Supercomputer username will have a format like **irtambo19**
  - Credentials for supercomputer account are **independent** of credentials for HBKU domain account (the ones used for VPN access)
- 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>

---

# **Resource Management & Job Scheduling**

# Typical Workflow

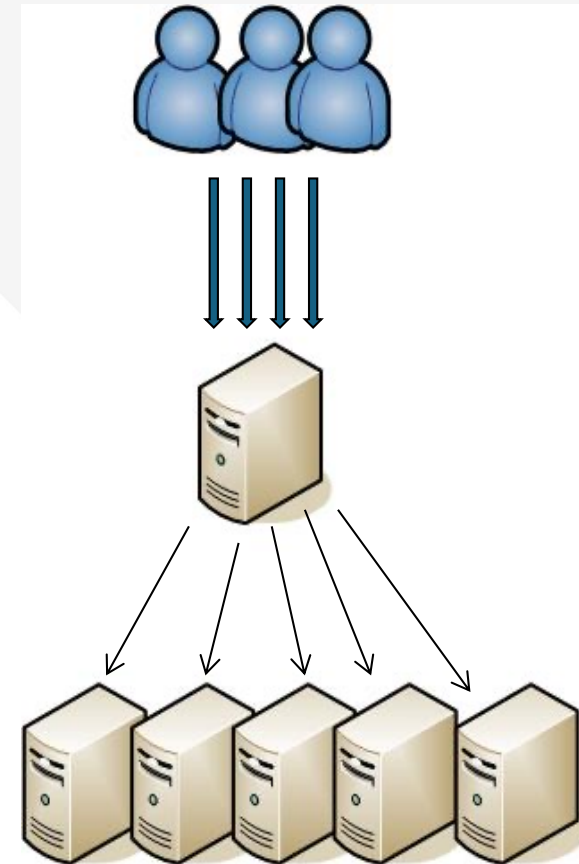
---



# 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”

Multiple users submit multiple “jobs”



Jobs assigned to compute nodes

# 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

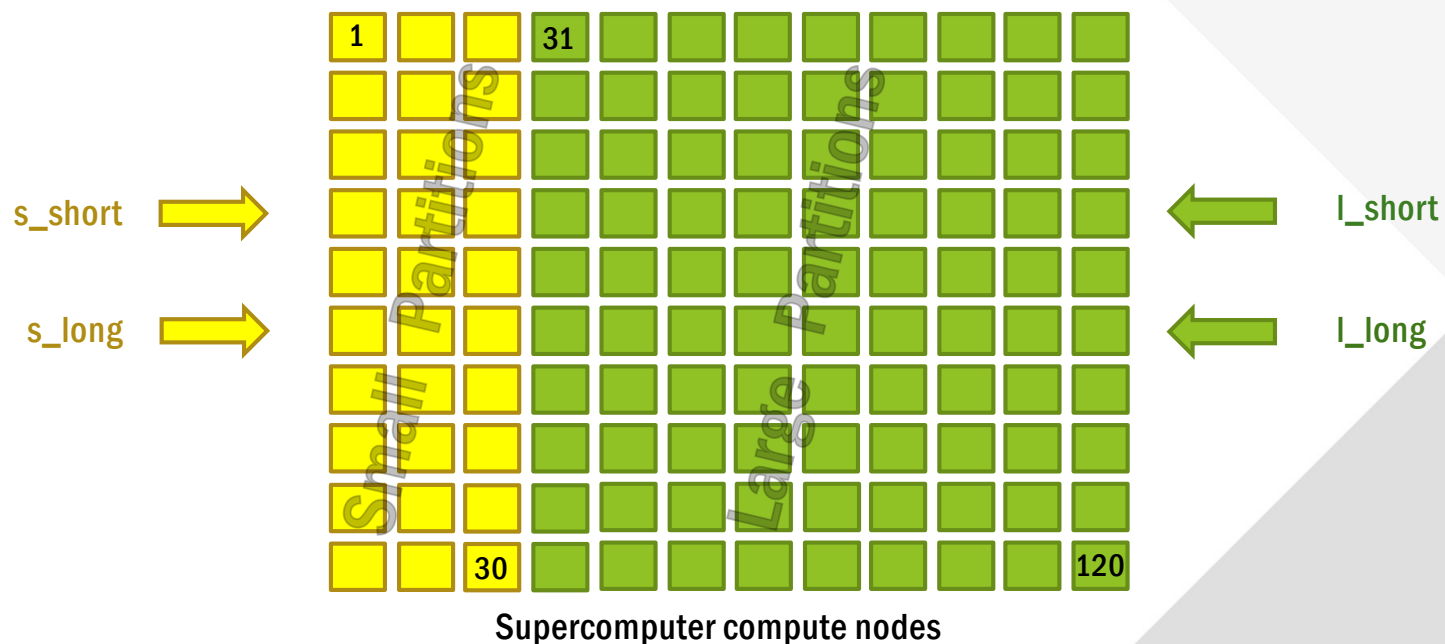
# What Resources are Managed by SLURM?

---

- 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 (QOS) to:
  - control the allocation of hardware resources to jobs
  - enforce resource limits on various categories of jobs

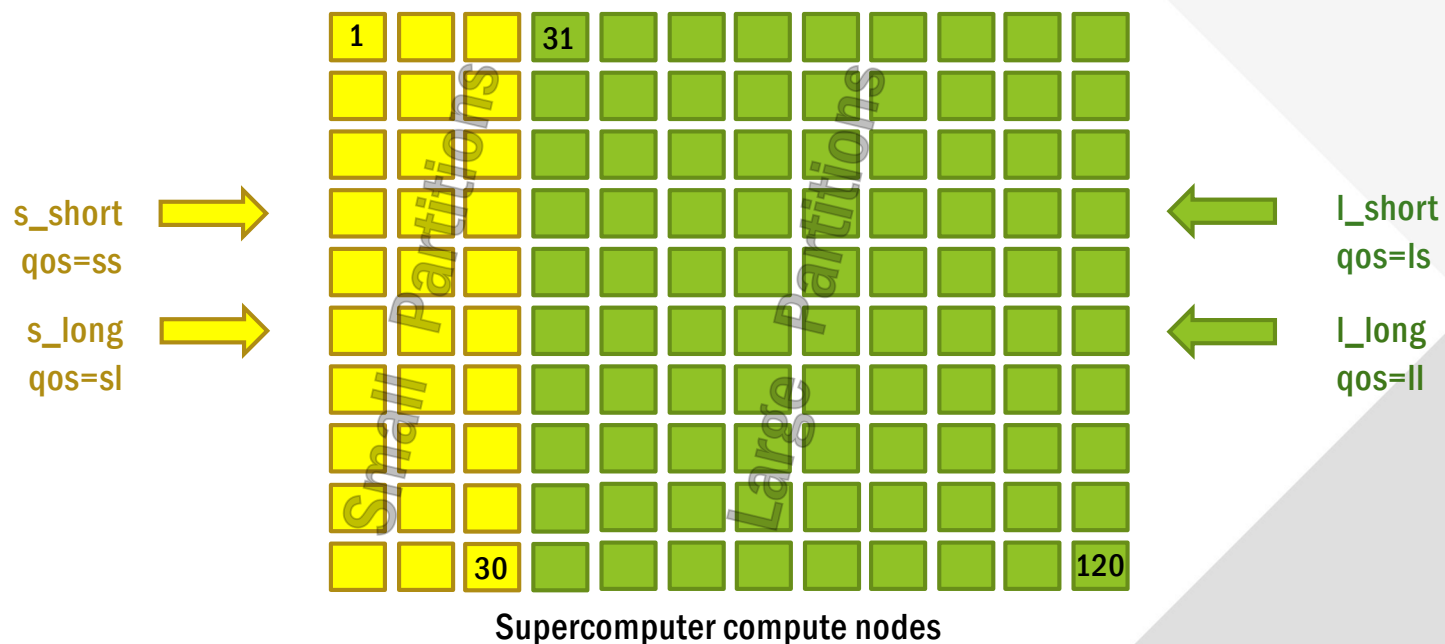
# Partitions

- A partition is simply a defined subset of the nodes in the supercomputer
- Every partition is assigned a name by the administrator
- Partitions can overlap; two partition names can map to the same set of nodes
  - On raad2, partitions **s\_short** & **s\_long** map to the same **set of nodes**
  - Partitions **l\_short** & **l\_long** map to a distinct second **set of nodes**



# Quality of Service (QOS)

- Quality of Service is a named, logical construct used to classify jobs for scheduling
- We use the QOS feature to set limits on:
  - the maximum # of cores a given partition may devote to jobs with that QOS
  - the max # of cores any given user may simultaneously consume for jobs with that QOS
- Every raad2 partition is configured to accept jobs requesting only one associated QOS
- Every job must specify both a partition as well as its associated QOS



# Partitions & Associated QOS on Raad2

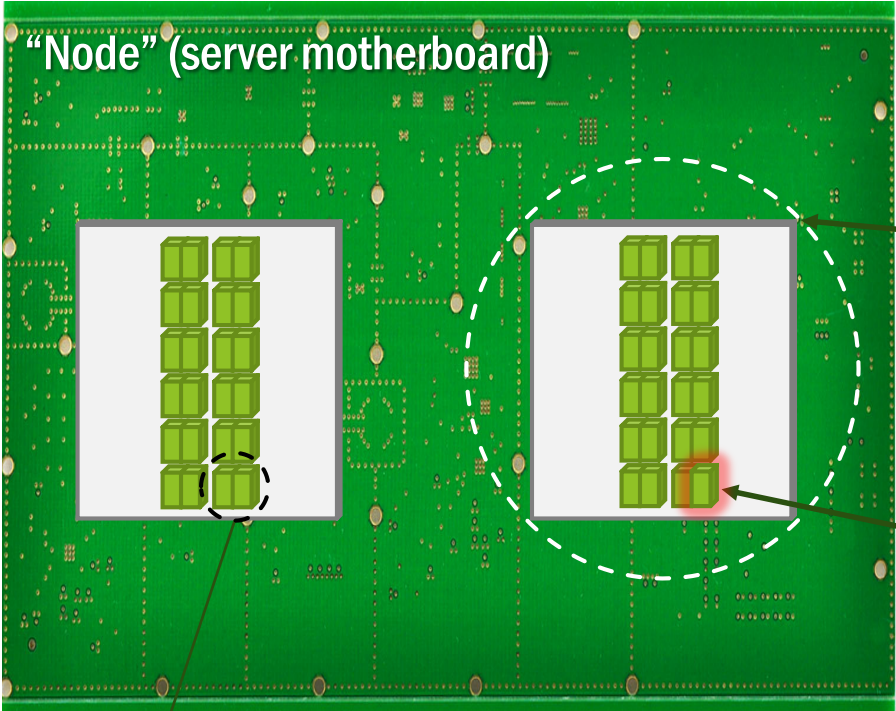
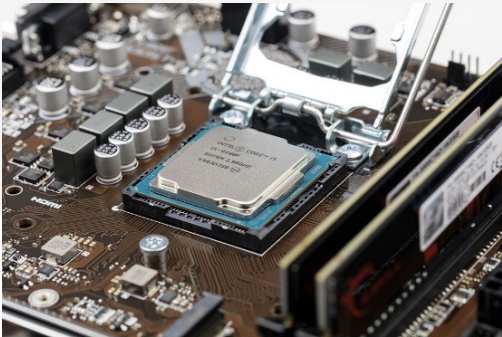
Partition Name (QOS)	express (ex)	overlapping partition		s_debug (sd)	overlapping partition	
		s_short (ss)	s_long (sl)		l_short (ls)	l_long (ll)
Max walltime	1 hour	8 hrs	168 hrs	4 hrs	8 hrs	168 hrs
Max CPUs per user	48	24	48	24	2 nodes	10 nodes
Max CPUs per QOS	96	144	456	48	6 nodes	113 nodes
Default walltime	30 mins	2 hrs	24 hrs	1 hrs	2 hrs	24 hrs
Default memory	5,350 MB per allocated core when using <code>#SBATCH --hint=nomultithread</code> 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?”

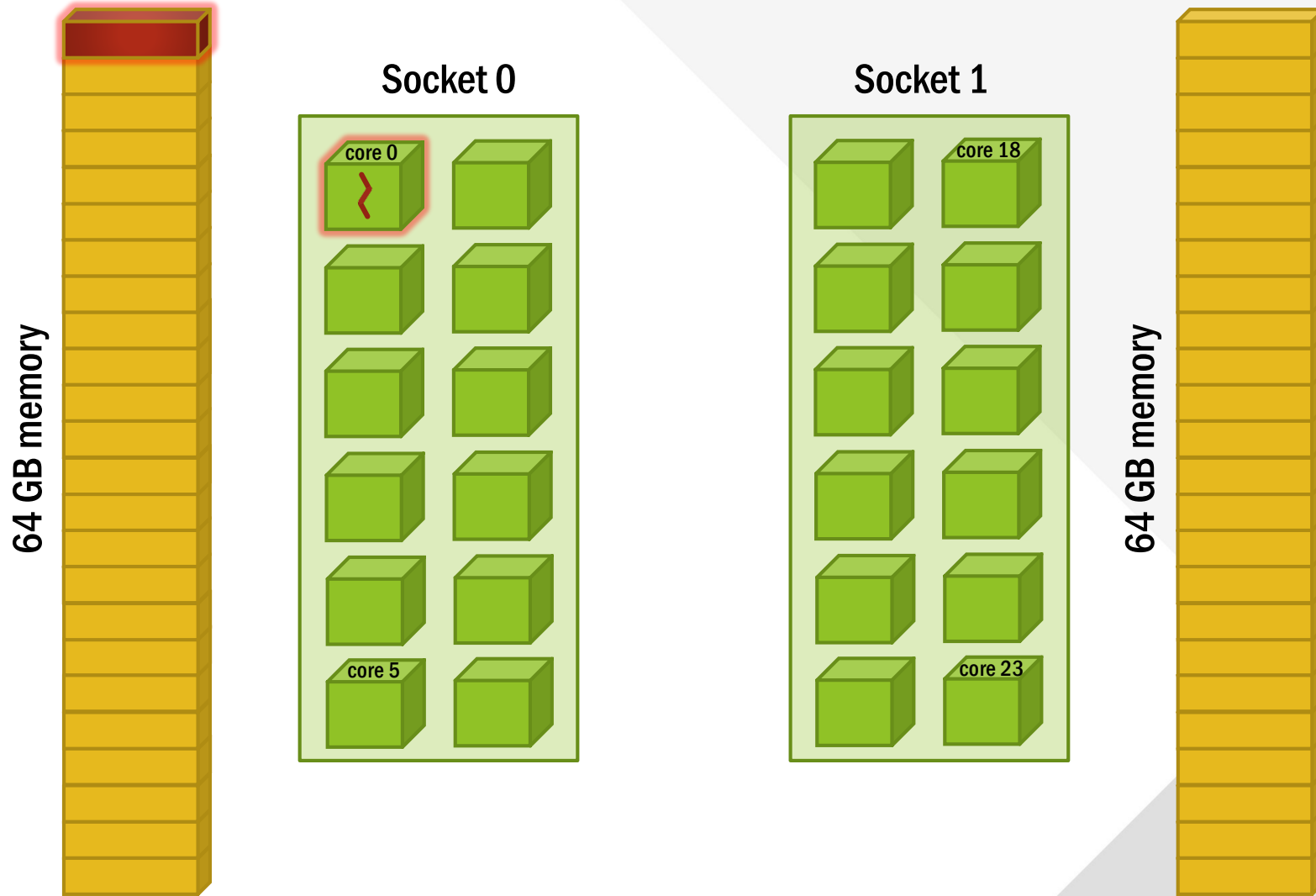


“Socket”  
(chip installed on the motherboard)

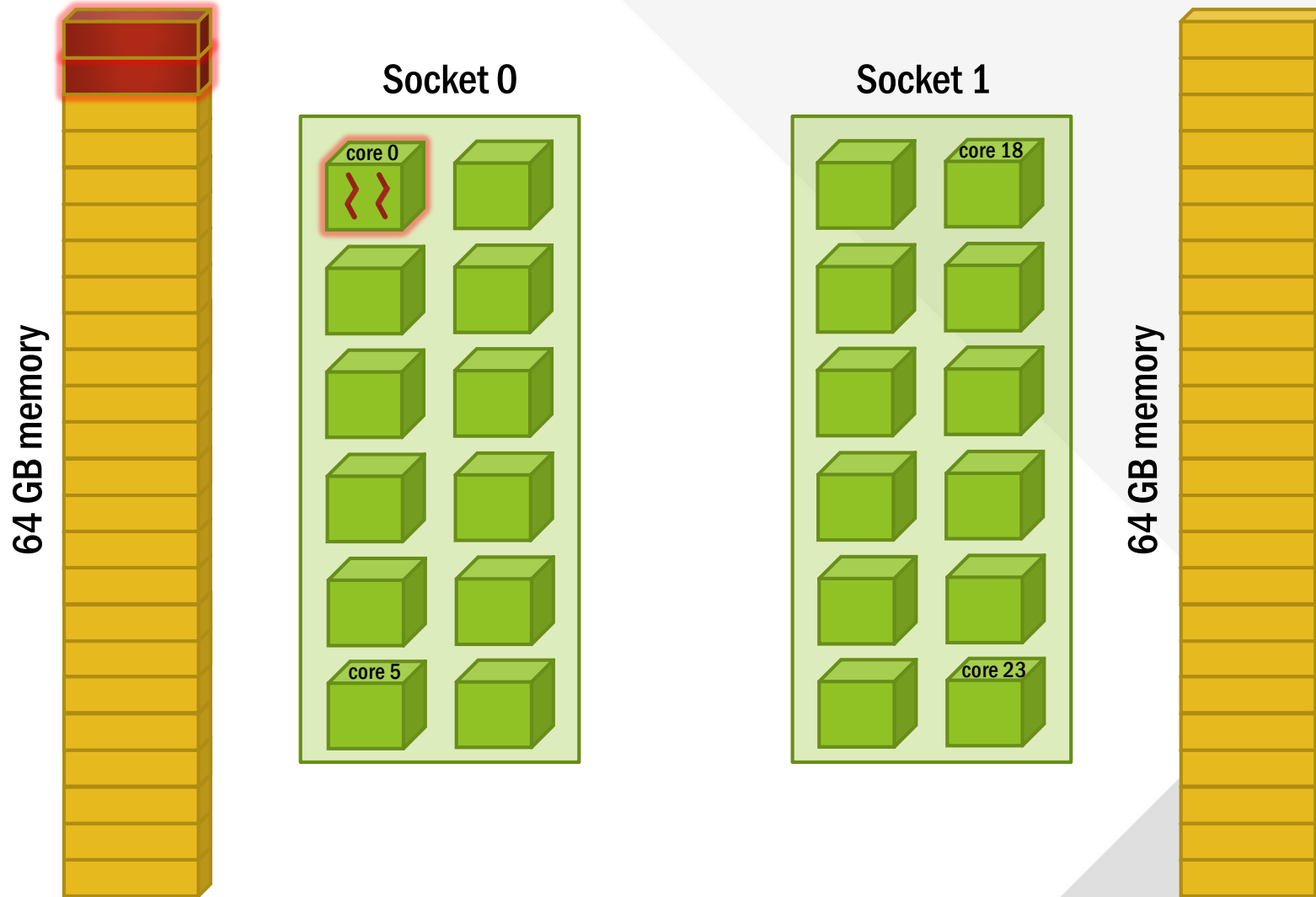
“Hyperthread”  
(a “logical” CPU inside the core)

“Core” (one of several stand-alone CPUs etched on silicon, packaged inside the chip)

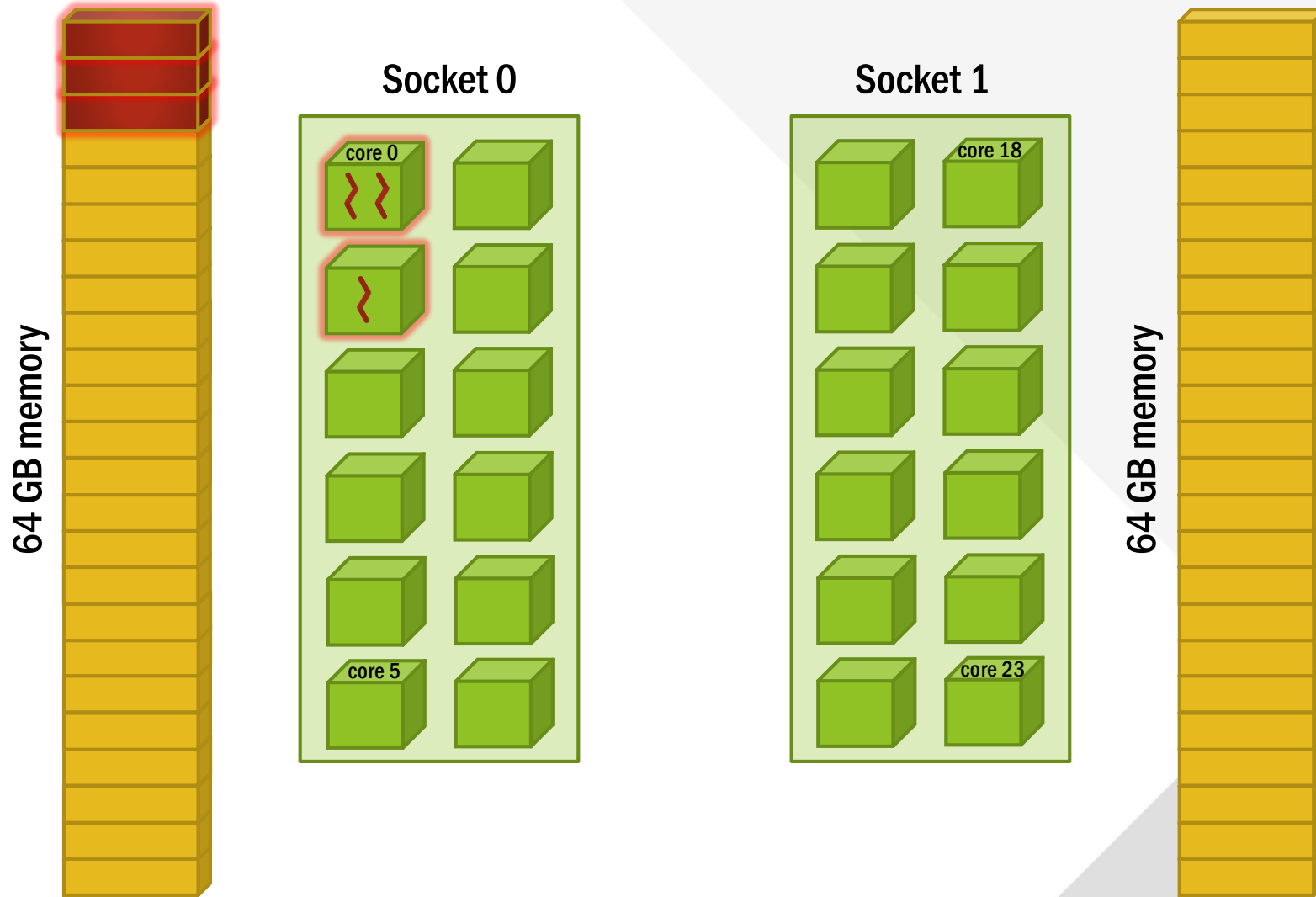
# CPU Resource Allocation



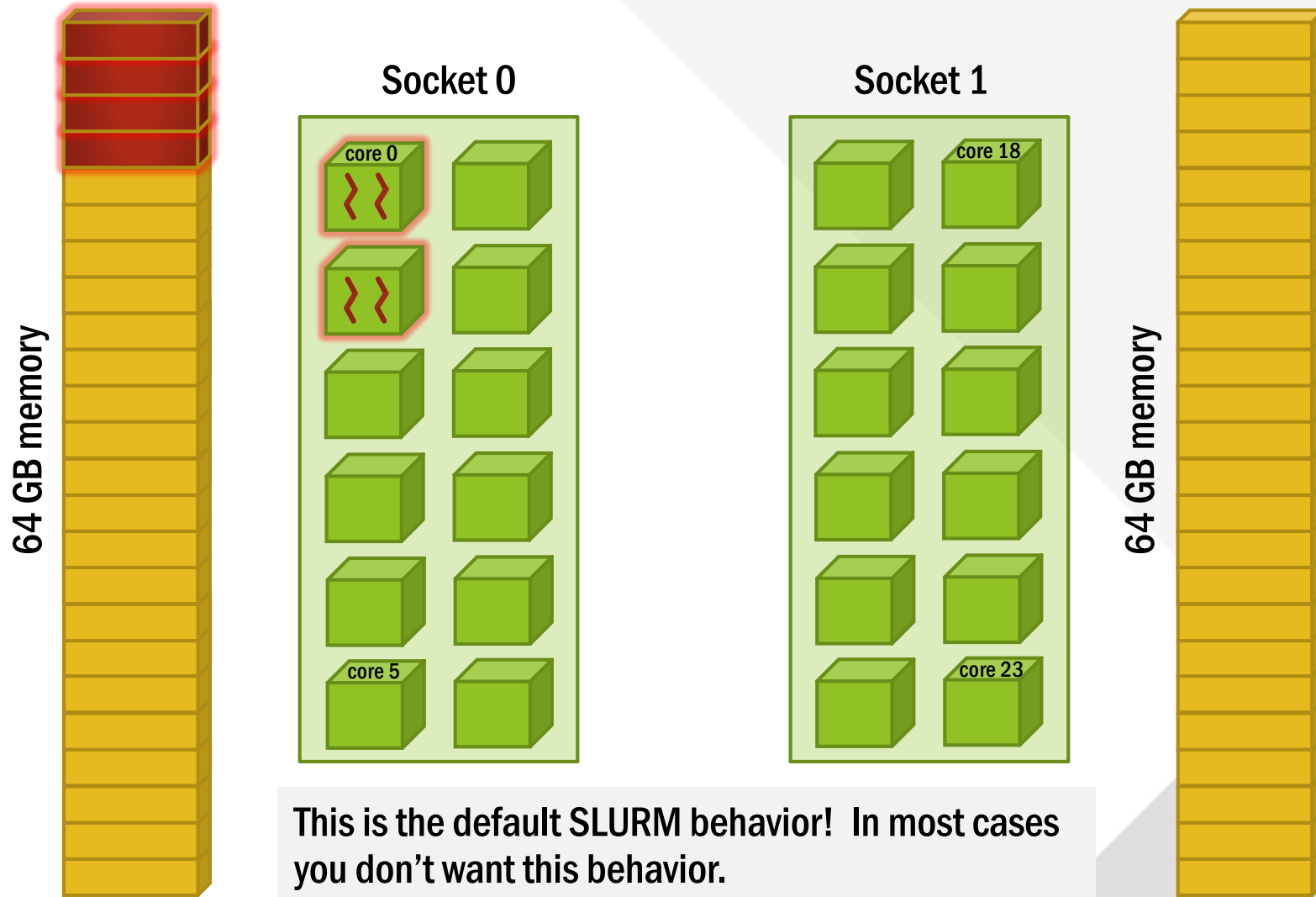
# CPU Resource Allocation



# CPU Resource Allocation

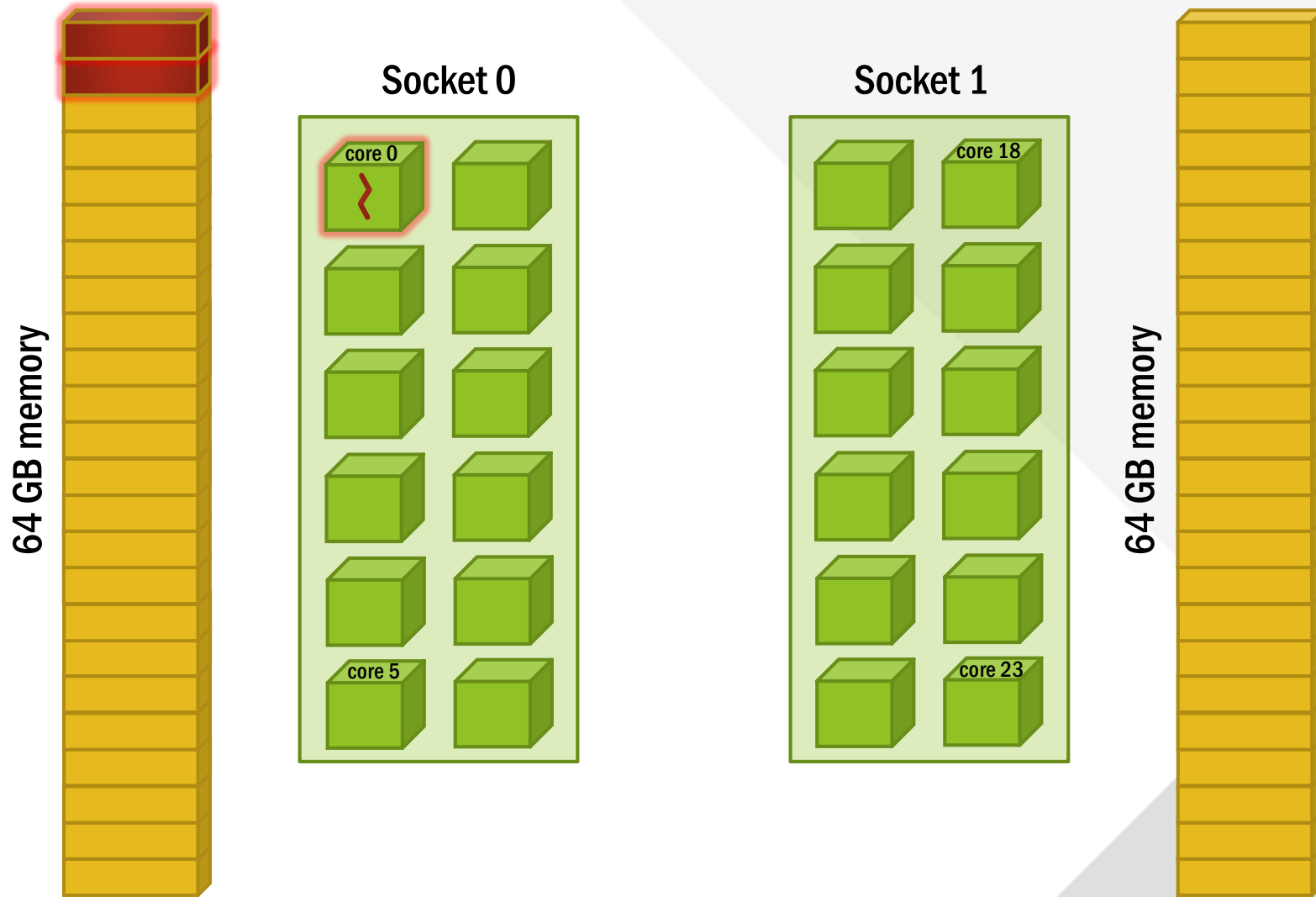


# CPU Resource Allocation

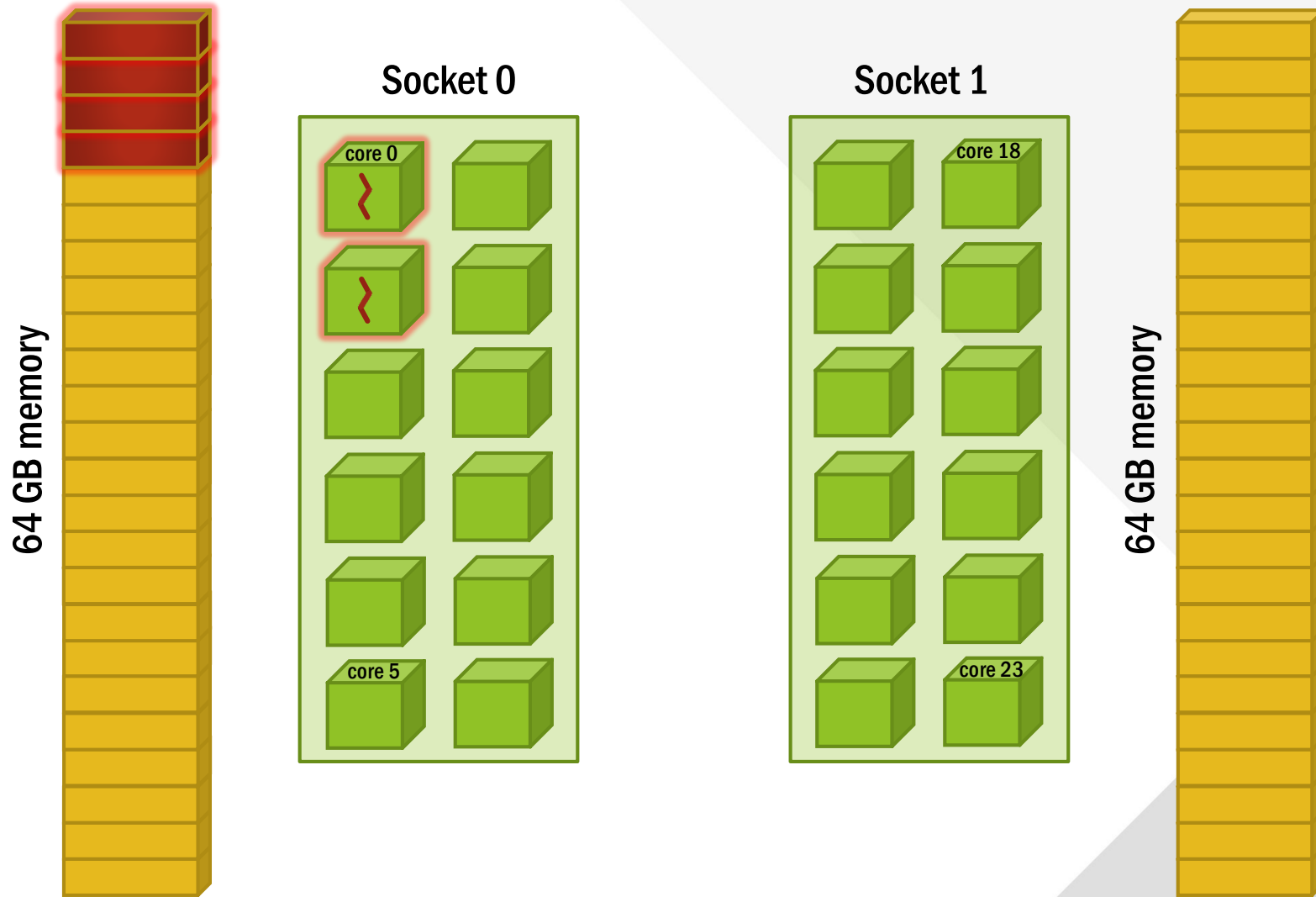


This is the default SLURM behavior! In most cases you don't want this behavior.

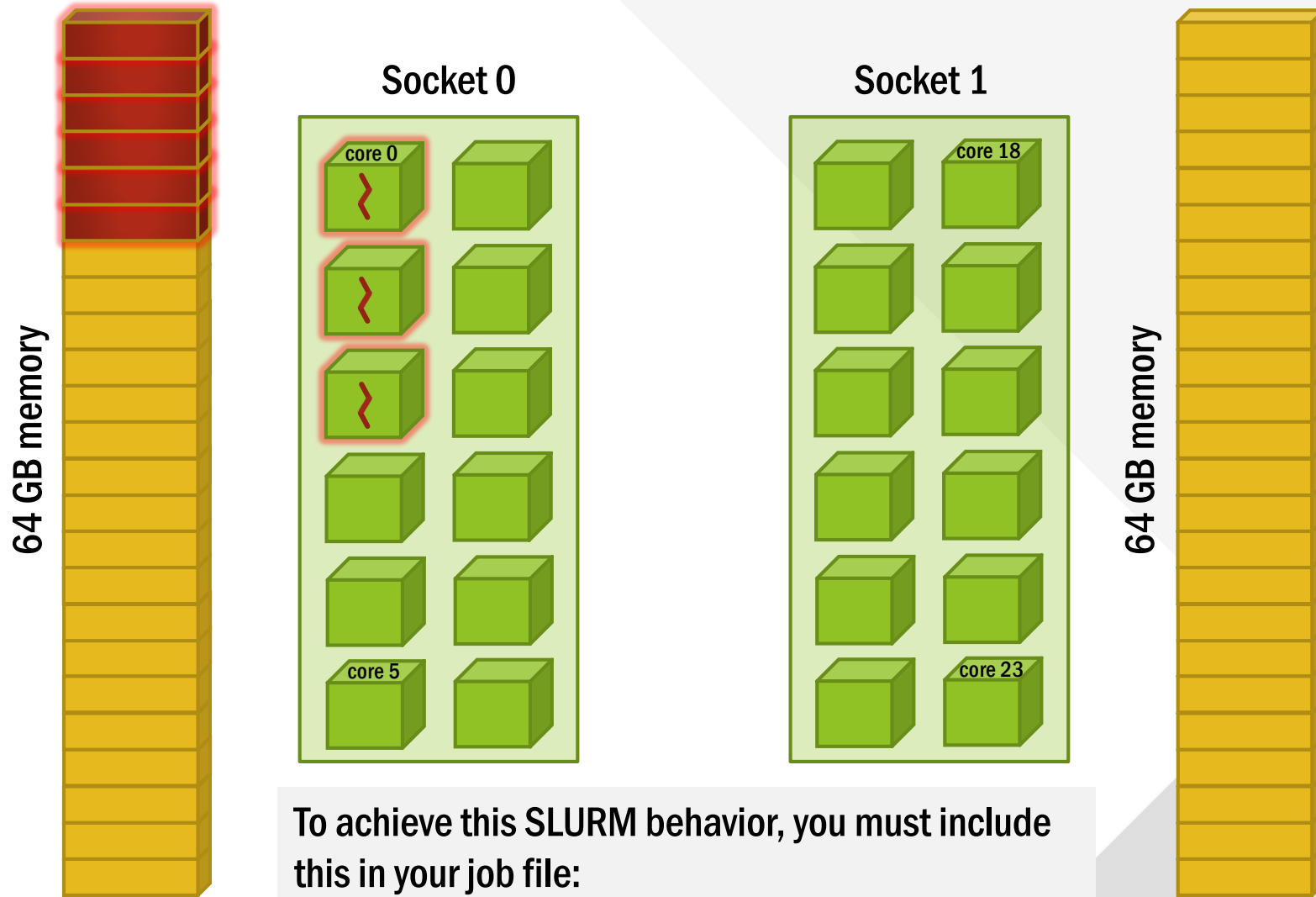
# Resource Allocation with `--hint=nomultithread`



# 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
```

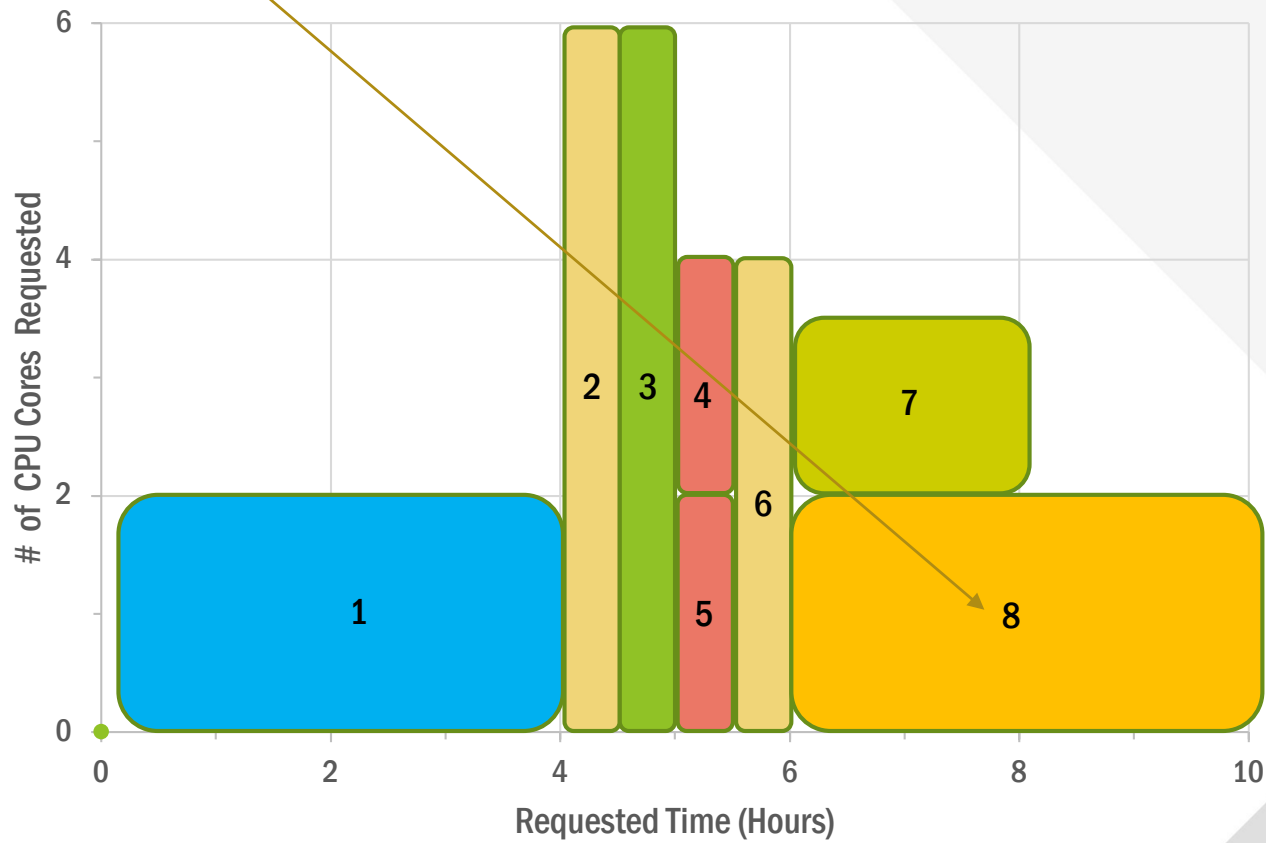
# Resource Allocation vs. Resource Consumption

---

- 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.

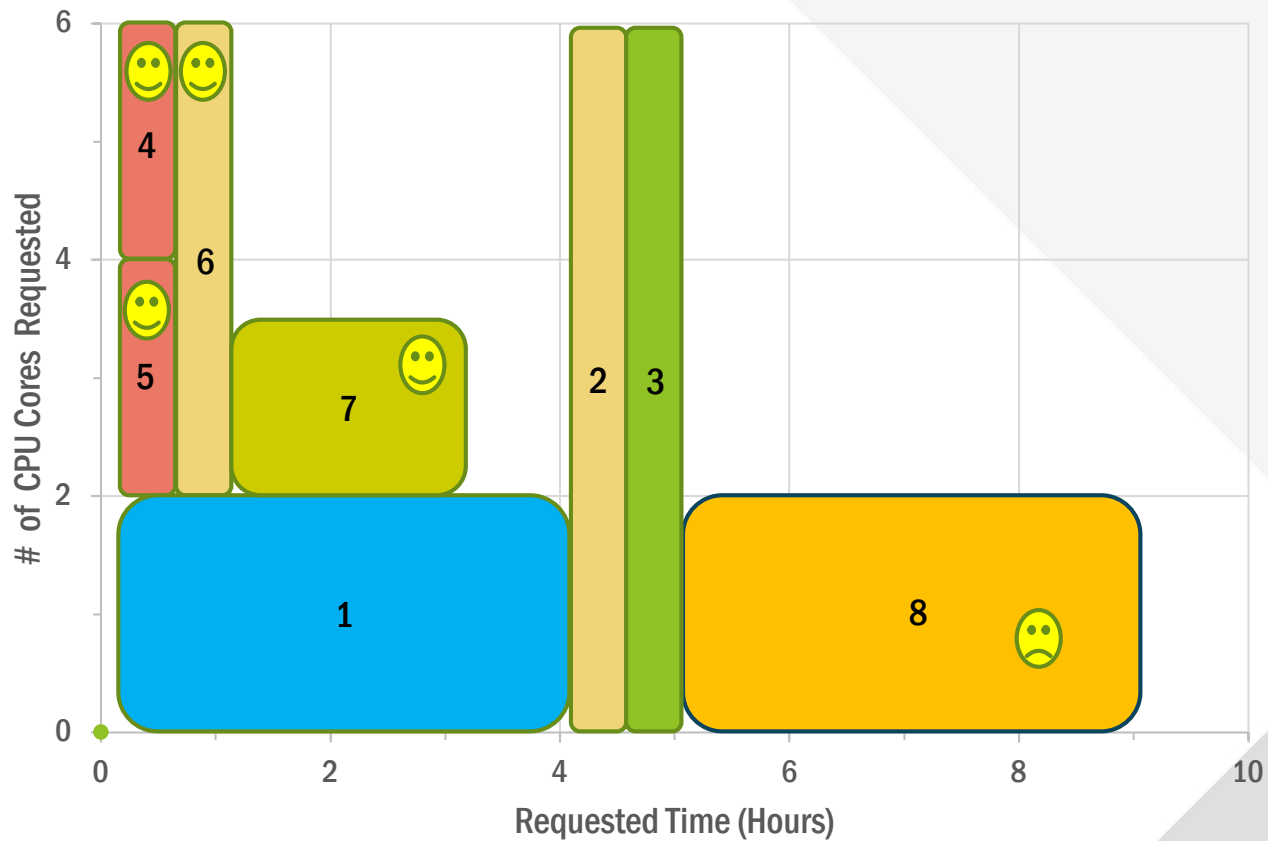
# Excessive Resource Requests

“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
```

- Job files such as the ones 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
  - Hence, the 2 files above are functionally identical

# Job File Walk-Through

Line	Explanation
<code>#!/bin/bash</code>	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.
<code>#SBATCH -J MyDemoJob</code>	This simply helps identify running jobs using a name tag of sorts.
<code>#SBATCH -p s_debug</code>	This job will run in the s_debug partition.
<code>#SBATCH -qos=sd</code>	This job will run with the quality of service called “sd”.
<code># This is a comment. It is ignored.</code>	Anything to the right of a solitary # is a comment (except for “SBATCH” keyword).
<code>#SBATCH -t 00:05:00</code>	This job needs 5 minutes of walltime to do its thing.
<code>#SBATCH -n 1</code>	This job needs just a single CPU because it has only 1 task to launch.
<code>sruntime myprog.exe</code>	“myprog.exe” is the user program that needs to be run; it is launched using sruntime.

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
<code>--job-name=name</code> <code>-J name</code>	A string used to label a job with a name. Here, <i>name</i> 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.
<code>--nodes=count</code> <code>-N count</code>	Used to allocate <i>count</i> complete nodes (24 CPUs per node) to a job.
<code>--partition=partition</code> <code>-p partition</code>	The partition from which your job will be assigned resources.
<code>--qos=qos_name</code>	The quality of service level requested for your job.
<code>--cpus-per-task=count</code> <code>-c count</code>	Commonly used by OpenMP jobs, where each task might want to spawn <i>count</i> threads, for which <i>count</i> CPUs should be requested.
<code>--ntasks=count</code> <code>-n count</code>	Used to request an allocation of <i>count</i> CPUs because the job intends to launch <i>count</i> tasks within the job.
<code>--error=filename</code> <code>-e filename</code>	Write error & warning messages (stderr) to <i>filename</i> .
<code>--output=filename</code> <code>-o filename</code>	Write normal output (stdout) to <i>filename</i> . 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
<code>--hint=multithread</code>	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.
<code>--hint=nomultithread</code>	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.
<code>--mail-type=type</code>	Notify user by email when certain event types occur. Common type values include NONE, BEGIN, END, FAIL, REQUEUE, and ALL.
<code>--mail-user=email</code>	The email address of the user to be notified.
<code>--nodelist=nodenamelist</code> <code>-w nodenamelist</code>	Request a specific list of compute nodes for the job. The list may be specified in a format like <code>nid00[008,012]</code> . 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.
<code>--mem=count</code>	Request <i>count</i> megabytes of real memory per node for the job. Using the suffixes K or G one may request kilobytes or gigabytes instead (e.g. <code>--mem=12G</code> ). The <code>--mem</code> and <code>--mem-per-cpu</code> options are mutually exclusive.
<code>--mem-per-cpu=count</code>	Request memory (in megabytes) required per allocated CPU. The default mem per cpu value is 5,350 MB when the <code>--hint=nomultithread</code> option is in effect, or 2,675 MB otherwise.

# Common SLURM Commands

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

## Detailed Documentation

[https://slurm.schedmd.com/archive/slurm-20.02-latest/man\\_index.html](https://slurm.schedmd.com/archive/slurm-20.02-latest/man_index.html)

# The `queue` Command

Command Example	What it does...
<code>queue</code>	List all jobs currently present on the system
<code>queue -l</code>	List all jobs currently present on the system (more verbose)
<code>queue -u fachaud74</code>	List all jobs from user fachaud74
<code>queue -u fachaud74 --state=pending</code>	List all pending (i.e. waiting) jobs from user fachaud74
<code>queue --state=running</code>	List all jobs on the system that are in the “running” state *

## Job States

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

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 `queue` is highly customizable. See the `-o` option in the `queue` man page. To experiment, try:

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

# The `sinfo` Command

Command Example	What it does...
<code>sinfo</code>	Display information about slurm nodes and partitions
<code>sinfo -s</code>	Display only a partition state summary (w/out node details)
<code>sinfo -N</code>	Display information in a node-oriented format
<code>sinfo -n nid00010,nid00400</code>	Display information only about nodes nid00010 and nid00400
<code>sinfo -p l_long</code>	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...
<code>scancel 1234</code>	Kill the job with job ID 1234
<code>scancel -u fachaud74</code>	Kill all jobs from user fachaud74
<code>scancel -u fachaud74 -state=pending</code>	Kill all pending jobs from user fachaud74
<code>scancel -u fachaud74 -state=running</code>	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

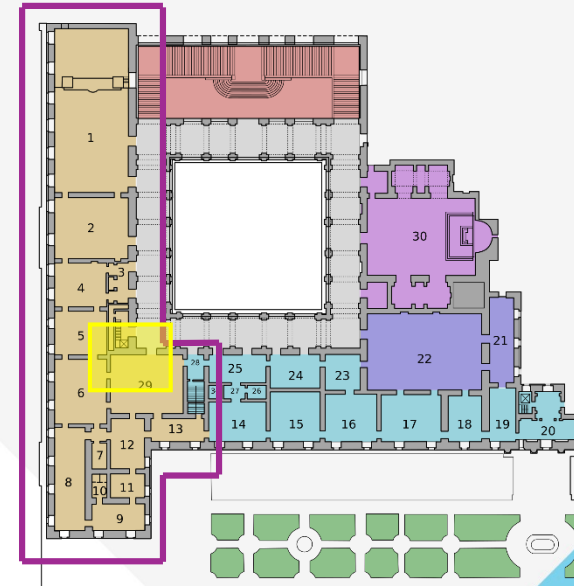
# The `scontrol` Command

Command Example	What it does...
<code>scontrol show job 1234</code>	Display job information for job 1234
<code>scontrol show partition l_long</code>	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=11
  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=/ddn/home/fachaud74/myTest.job
  WorkDir=/ddn/home/fachaud74
  StdErr=/ddn/home/fachaud74/myTest.o1171482
  StdIn=/dev/null
  StdOut=/ddn/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 ~]$
```

Job ID

SLURM job file name

# 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...
<code>srun prog.exe</code>	Launch the program <code>prog.exe</code>
<code>srun --ntasks=12 prog.exe</code>	Launch <code>prog.exe</code> using 12 hyper-threads
<code>srun --hint=nomultithread -n 12 prog.exe</code>	Launch <code>prog.exe</code> with 12 cores
<code>srun --propagate=STACK,MEMLOCK prog.exe</code>	Launch <code>prog.exe</code> 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 must first ssh into one of the “internal login nodes” (raad2-login1 or raad2-login2). Here is a sample sequence of commands...

```
fachaud74@raad2b:~> ssh raad2-login1
Last login: Thu Nov 23 11:01:47 2017 from nid00002
fachaud74@raad2-login1:~> 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-login1:~> exit
logout
Connection to raad2-login1 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...
<code>sacct -u fachaud74</code>	Produce a listing of user fachaud74's past and current jobs
<code>sacct -e</code>	Display a list of possible column headers usable with the <code>-o</code> option (which customizes the <code>sacct</code> report)
<code>sacct -u fachaud74 \</code> <code>-o "jobid,jobname,cputime,elapsed"</code>	Display the job ID, job name, CPU time, and walltime consumed by jobs from user fachaud74
<code>sacct -S 2017-07-01 -u fachaud74</code>	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

---

# **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. In principle, users could do this manually with a series of relevant commands.
  - See section “Variables” in our “Intro to Linux” short course
- 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

Command	Description
<code>module avail</code>	Show available modules
<code>module list</code>	Show currently loaded modules
<code>module load <i>modulename</i></code>	Load a module
<code>module unload <i>modulename</i></code>	Unload a module
<code>module switch <i>module1 module2</i></code>	Unload <i>module1</i> and load <i>module2</i> in its place
<code>module help</code>	List the available module commands
<code>module help <i>modulename</i></code>	Display the help information for a module
<code>module display <i>modulename</i></code>	Show specific changes that would be made by the <i>modulename</i> 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
```

```
----- /ddn/opt/modulefiles/tamug -----  
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
```

```
----- /ddn/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

**module list**

lists the currently loaded modules

```
fachaud74@raad2b:~> module list
```

```
Currently Loaded Modulefiles:
```

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

```
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
```

**module load *modulename***

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

```

                < M A T L A B (R) >
Copyright 1984-2014 The MathWorks, Inc.
R2014a (8.3.0.532) 64-bit (glnx64)
                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
```

**module unload *modulename***

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

```

                < M A T L A B (R) >
Copyright 1984-2016 The MathWorks, Inc.
R2016a (9.0.0.341360) 64-bit (glnx64)
                February 11, 2016
```

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

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

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:/ddn/home/fachaud74/bin:/usr/local/bin:/usr/bin:/bin:/us
r/bin/X11:/usr/X11R6/bin:/usr/games:/usr/lib/mit/bin:/usr/lib/mit/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:/ddn/home/fachaud74/bin:/usr/local/bin:/usr/bin:/bin:/us
r/bin/X11:/usr/X11R6/bin:/usr/games:/usr/lib/mit/bin:/usr/lib/mit/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 or library.

---



# **SLURM by Example**

# Examples Directory

---

- Found in: `/ddn/share/examples/slurm-tutorial-latest.tar`
  - 01\_helloworld
  - 02\_multi\_helloworld
  - 03\_mpi\_matmult
  - 04\_multi\_srums
  - 05\_interactive\_job
  - 06\_tidbits
  - 07\_openmp
  - 08\_matlab
  - 09\_ansys
  - 10\_datawarp

# SLURM Example Files

---

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