



HPC Job Submission

HPC Job Submission

Gerardo Hidalgo-Cuellar

gehi0941@colorado.edu

www.rc.colorado.edu

rc-help@colorado.edu

Slides available at:

https://github.com/ResearchComputing/Supercomputing_Spinup_Spring_2022

Tutorial at:

https://github.com/ResearchComputing/Supercomputing_Spinup_Spring_2022/tree/master/job_submission

Survey at: <http://tinyurl.com/curc-survey18>

Adapted from presentations by RC members Andrew Monaghan, Aaron Holt, John Blaas, and Mea Trehan: [1](#), [2](#), [3](#), [4](#).

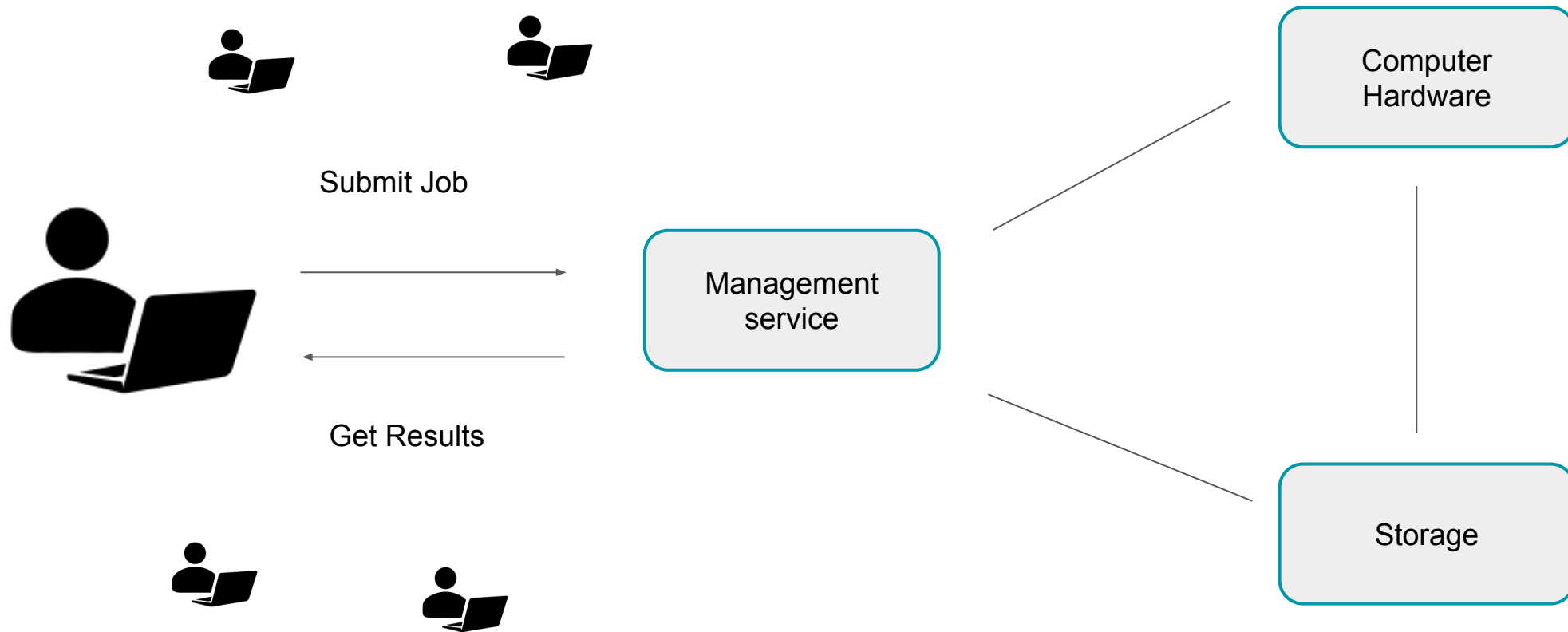
Outline

- General Information
 - Cluster resources
- Examples of submitting jobs to the supercomputer!
 - Traditional job submission (terminal)
 - Simple batch jobs: hello world, running programs
 - Advanced batch jobs: mpi, serial-parallel
 - Interactive jobs
 - Non traditional (gateways)
 - Interactive applications
 - Check jobs

RC account check

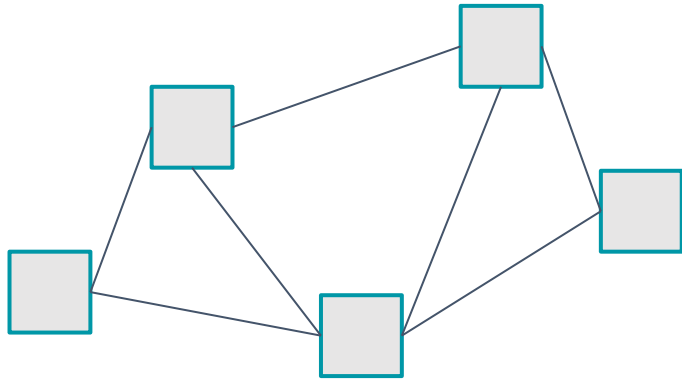
- Does anyone need a temporary RC account?

HPC - High Performance Computing



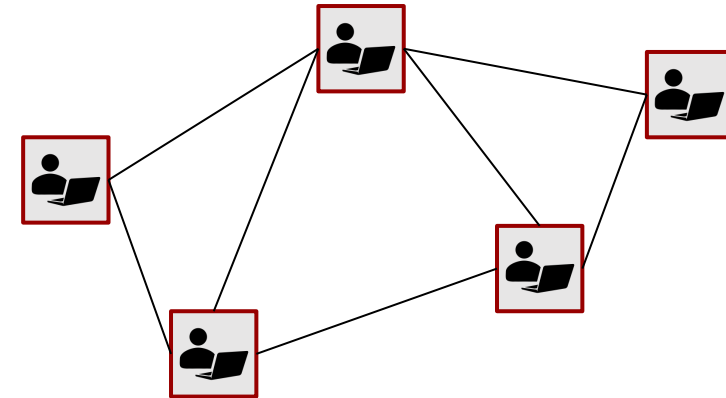
HPC Clusters at RC

Summit



- NSF-Funded
- Shared
- 450+ Nodes

Blanca



- Buy-in Cluster
- High priority use

RMACC Summit Supercomputer

- 450 General Compute nodes (Intel Xeon Haswell)
 - 24 cores per node
 - GPU, High Memory, Phi Nodes
 - **11,400 total cores**
 - Omni-Path network
 - 1.2 PB scratch storage
-
- 67% CU, 23% CSU, 10% RMACC



Additional Node Types on Summit

- 10 GPU Nodes
 - NVIDIA Tesla K80 (2 accelerators/node)
- 5 High Memory Nodes
 - 2 TB of memory/node, 48 cores/node
- 20 Phi Nodes
 - Intel Xeon Phi
 - 68 cores/node, 4x threads/core

Blanca Supercomputer



- **223 Compute nodes** (heterogeneous)
- **10 GPU nodes** (heterogeneous)
- Hardware owned by individual contributors
 - Preemptable access to all other nodes. i.e., you can use them when the node owners are not
- When you log in, the Summit queue is loaded in *by default*

```
$ module load slurm/blanca  
# to load in Blanca queue
```

Submitting Jobs via Terminal

RC Access: Logging in

- If you have an RC account already, login as follows from a terminal:

```
$ ssh <username>@login.rc.colorado.edu  
# Where username is your identikey
```

- If you do not have an RC account use one of our temporary accounts:

```
$ ssh user<XXXX>@tlogin1.rc.colorado.edu  
# Where user<XXXX> is your temporary username
```

Working on RC Resources

- When you first log in, you will be on a login node. Your prompt:

```
[user@loginNN ~]$
```

- The login nodes are lightweight virtual machines primarily intended to serve as 'gateways' to RC resources. In order to get a better view of the software available on Summit we will go to a compile node.

```
[user@loginNN ~]$ ssh scompile
```

- Now go to a working directory (I'm using scratch here) and download the material for this workshop:

```
[user@shas0137 ~]$ cd /scratch/summit/$USER  
[user@shas0137 ~]$ git clone  
https://github.com/ResearchComputing/Supercomputing\_Spinup\_Spring\_2022.git
```

Working Directory

- Navigate to the “job_submission” directory

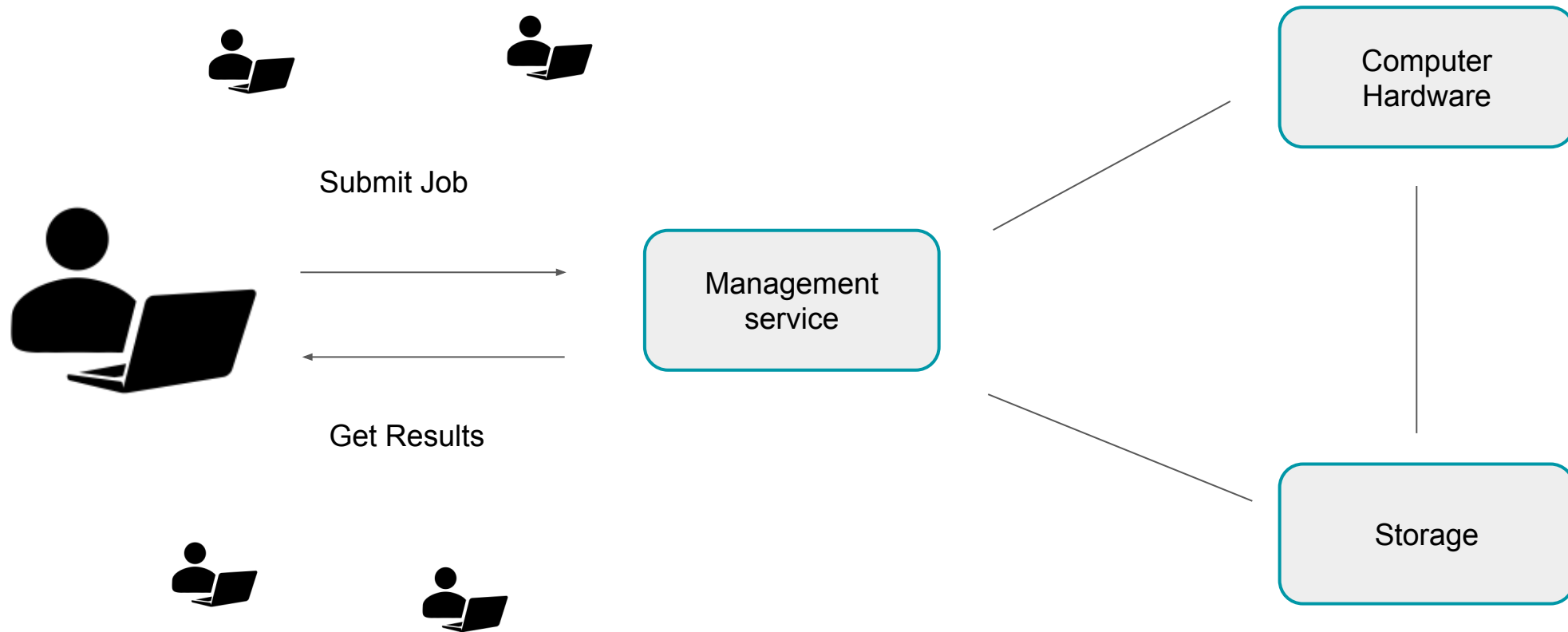
```
[user@loginNN ~]$ cd /path/to/job_submission
```

- This is the “working directory” we will be working with in this course/tutorial, keep in mind as we submit/create jobs

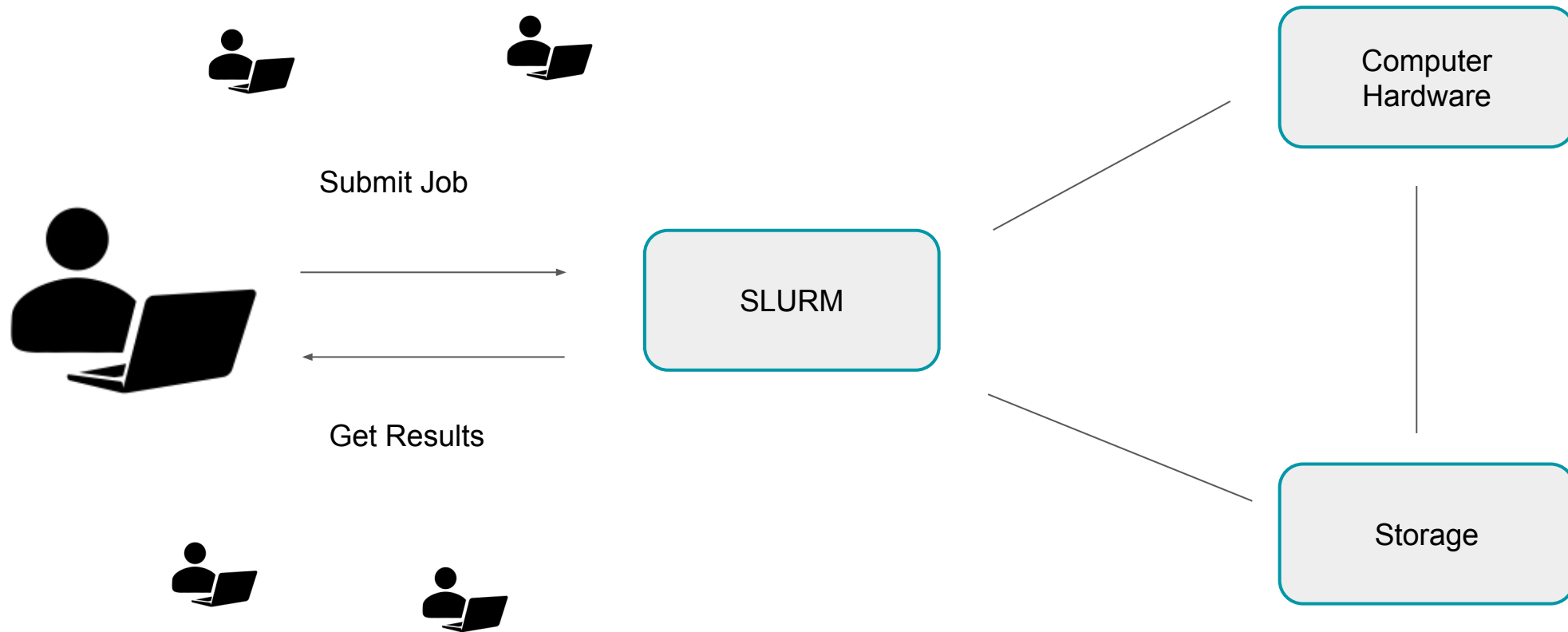
Jobs

- Because our clusters are shared resources with many users trying to utilize available compute with their applications, we need a system to divide compute in a simple and fair system.
- SLURM
 - **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Through SLURM, users can grab allotments of compute resources called Jobs
- 2 Types of Jobs
 - **Batch Jobs**
 - **Interactive Jobs**

HPC - High Performance Computing



HPC - High Performance Computing



Batch Jobs

- Batch Jobs are jobs you submit to the scheduler where they are run later without supervision.
 - By far the most common job on Summit
 - Requires a job script
 - "batch of cookies"
- A job script is simply a script that includes **SLURM directives** (resource specifics) ahead of any commands.

Submit your first batch job

- `sbatch`: command to submit a batch job
- Submit your first job! :

```
$ cd Supercomputing_Spin_Up_Fall_2021/job_submission  
$ sbatch scripts/submit_test.sh
```

- Script contains most of the parameters needed to define a job
- Additional flags can be used to temporarily replace any set parameters.

```
$ sbatch --reservation=scs submit_test.sh
```

<http://slurm.schedmd.com/sbatch.html>

Anatomy of a job script

- open `scripts/submit_test.sh` (nano or vim)

```
#!/bin/bash

## Directives
#SBATCH --ntasks=1           # Number of requested tasks
#SBATCH --time=0:01:00       # Max wall time
#SBATCH --partition=shas-testing # Specify Summit Haswell nodes
#SBATCH --output=test_%j.out  # Rename standard output file

## Software
module purge                  # Purge all existing modules

## User commands
echo "This is a test of user $USER"
```

Anatomy of a job script

```
#!/bin/bash

## Directives (HPC Resources)
#SBATCH --<resource>=<amount>

## Software
module load <software>

## User scripting
<command>
```

Job Options

Specified at command line or in job script as...

#SBATCH <options> ...where options include:

- **Partition:** `--partition=<partition_name>`
- Sending emails: `--mail-type=<type>`
- Email address: `--mail-user=<user>`
- Number of nodes: `--nodes=<nodes>`
- Number of cores: `--ntasks=<number-of-tasks>`
- **Quality of service:** `--qos=<qos>`
- Allocation: `--account=<account_name>`
- Wall time: `--time=<wall time>`
- Job Name: `--job-name=<jobname>`
- Output: `--output=<name>`

More on slurm commands: <https://slurm.schedmd.com/quickstart.html>

FYI: You do NOT actually type <> above – this designates something specific you as a user must enter about your job

Partitions

- Partitions specify the type of compute node that you wish to use
 - Specify with the `--partition` flag

```
#SBATCH --partition=shas
```

Partition	Description	# of nodes	RAM/core (GB)	cores/node	GPUs/node
shas	General Compute	~450	4.84	24	0
sgpu	GPU-enabled nodes	10	4.84	24	effectively 4
smem	High-memory nodes	5	42.7	48	0
sknl	Phi (Knights Landing) nodes	20	5.25	68	0

Sub-Partitions

- In addition to normal compute partitions, Summit Users also have access to several testing and interactive partitions
 - Quick access to get your applications functional!

Partition	Description	Max wall time	Max jobs/user	Max nodes/user
shas-testing sgpu-testing sknl-testing	For quick turnaround when testing	30 M	1	2 12 cores/node
shas-interactive	For interactive jobs (command or GUI)	4 H	1	1 core

Quality of Service (--qos)

- Quality of Service specifies additional constraints Job
 - On Summit, this means if your job needs to run longer than 1 day
 - only `shas` and `skn1`
 - Specify with the `--qos` flag
 - Doesn't need to be set otherwise

```
#SBATCH --qos=long
```

QoS	Description	Max wall time	Max jobs/user	Max nodes/user
normal	Default QoS	Derived from partition	n/a	256
long	For jobs needing longer wall times	7 D	n/a	20

Writing your first job script

Your turn!

- Create a job script and submit it as a batch job with the following instructions:
 1. Navigate to the 'job_submission' directory
 2. Create file 'scripts/submit_sleep.sh'
 3. The job should contain the following commands:

```
echo "Running on host" `hostname`  
echo "Starting Sleep"  
sleep 30  
echo "Ending Sleep. Exiting Job!"
```

Details on job script parameters are in the next slide

Job details of `submit_sleep.sh`

1. The job will run on 1 core on 1 node
2. We will request a 1 minute wall time
3. Run on the shas-testing partition
4. Set the output file to be named “./output/sleep.%j.out”
5. Name your job “sleep”
6. Bonus: Email yourself when the job ends
7. Contains the following commands ->

```
echo "Running on host" `hostname`  
echo "Starting Sleep"  
sleep 30  
echo "Ending Sleep. Exiting Job!"
```

```
$ sbatch --reservation=spinup scripts/submit_sleep.sh
```

Solution are prefixed with 'answer'

Job Output

- Once a job completes its execution, the standard output of the script will be redirected to an output file.
 - Great for debugging!
 - Could be different from output generated by your application
 - File is created in directory job was run unless specified in your `--output` directive.
 - If the directive `--output` is not provided then a generic file name will be used (slurm_XXXXXX.out).

```
$ cat output/sleep.aaaaaa.out # where aaaaaa is your Job Id
```

Solution can be found in “./solutions” subdirectory

Checking your jobs (1)

- **squeue**: Monitor your jobs status in queue and while running:
 - By Default shows all jobs in queue
 - Narrow this down with:

```
$ squeue -u <username>  
$ squeue -p <partition>
```

- **sacct**: Check back on usage statistics of previous Jobs
 - By default only checks all jobs from the start of the current day.
 - Narrow this down with:

```
$ sacct -u <username>  
$ sacct --start=MM/DD/YY -u <username>  
$ sacct -j <job-id>
```

More on slurm commands:
<https://slurm.schedmd.com/quickstart.html>

Checking your jobs (2)

- Another method of checking details of your job is with `scontrol`
- Advanced command usually used by system administrators, but you can use it too!

```
$ scontrol show job <job number>
```

- `seff`: Utility to check efficiency post-job

```
$ module load slurmtools  
$ seff <job number>
```

More on slurm commands:
<https://slurm.schedmd.com/quickstart.html>

Software and Jobs

- "Okay so running a job is easy, but how do I run a job with my software?"
- LMOD
 - Module system on CURC systems
 - Modifies your environment to make your desired software visible to your terminal.

```
$ module load matlab  
$ ml matlab #shorthand version!
```

More on slurm commands:
<https://slurm.schedmd.com/quickstart.html>

Software and Jobs (2)

- More LMOD commands:

```
$ module purge           #Unloads all current modules
$ module unload matlab   #Unloads matlab
$ module spider matlab    #Searches for matlab in module tree
```

- What if my software isn't available through LMOD?
 - Software must be installed locally if not available through LMOD
 - RC User support is happy to assist, but *installs are best effort*
 - For more assistance contact rc-help@colorado.edu

More on slurm commands:
<https://slurm.schedmd.com/quickstart.html>

Example 1: Serial R Code

Running an external program

- Let's run R on an R script
- Batch script calls and runs *programs/R_program.R*
 - Let's take a look at the R program
- Let's examine the batch script *scripts/submit_R.sh*
 - Note how R is loaded
 - R program can be run with “Rscript <script>”
- Go ahead and submit the batch script *scripts/submit_R.sh*

Example 2: Serial Matlab Code

Launch Matlab!

- Create a job script and submit it as a batch job with the following instructions:

1. Name it 'scripts/submit_matlab.sh'
2. Load the 'matlab' module
3. The job should contain the following commands:

```
cd programs  
matlab -nodisplay -nodesktop -r "matlab_tic;"
```

Details on job script parameters are in the next slide

Job details of `submit_matlab.sh`

1. The job will run on 1 core of 1 node
2. We will request a 2 minute wall time
3. Run on the shas-testing partition
4. Set the output file to be named “./output/matlab.%j.out”
5. Name your job “matlab”

Bonus: Email yourself when the job ends

6. Contains the following commands ☐

```
cd programs  
matlab -nodisplay -nodesktop -r "matlab_tic;"
```

```
$ sbatch --reservation=spinup scripts/submit_matlab.sh
```

Solution are prefixed with 'answer'

Advanced Job Scripts

Running an mpi job

- For cases where you have a code that is parallelized, meaning it can run across multiple cores.
- Number of tasks always > 1 . E.g.,

```
#SBATCH --ntasks=4
```

- Will always need to load a compiler and mpi. E.g.,

```
module load intel impi
```

- Executable preceded with mpirun, srun, or mpiexec. E.g.,

```
mpirun -np 4 python yourscrip.py
```

- Examine and run the example `'submit_python_mpi.sh'`

```
$ sbatch --reservation=scs submit_python_mpi.sh
```

Running serial jobs in parallel

- Not all code is designed to run with MPI (nor always makes sense to do so)
- RC has a couple different tools that lets users run serial programs in parallel
 - [RC LoadBalancer](#)
 - [GNU Parallel](#)
- Example in: [scripts/submit_run_hello.sh](#)

Interactive jobs

- Sometimes we want our job to run in the background
- Sometimes we want to work on program in real time
 - Great for testing, debugging
- We can get access to a compute node interactively with `sinteractive`
- For example, let's run the R job we previously ran as a batch job, but this time let's do it interactively...

Running an interactive job

- To work with R interactively, we request time from Summit
- When the resources become available the job starts
- Commands to run:

```
$ sinteractive --time=00:10:00 --reservation=spinup
```

- Once we receive a prompt, then:

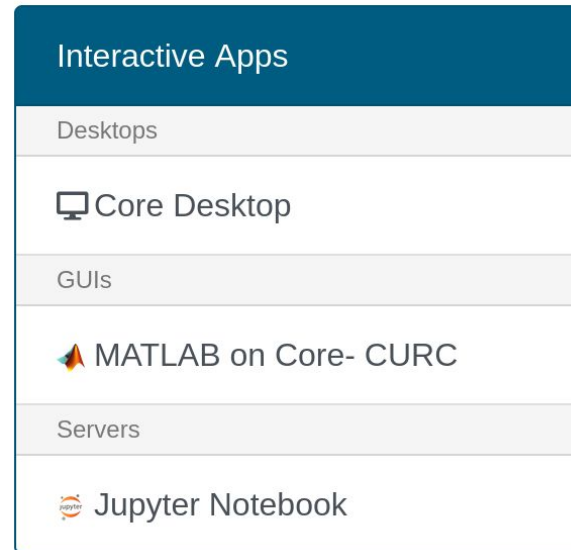
```
$ module load R  
$ cd programs  
$ Rscript R_program.R
```

- Once we finish we must exit! (job will time out eventually)

```
$ exit
```

Open OnDemand (demo)

- If you are new to linux, submitting jobs to an HPC system can be overwhelming
- We have browser-based applications that give you the power to connect to a compute node straight away
- CURC Open OnDemand
 - JupyterHub
 - MATLAB
 - Virtual Desktop



Thank you!

- Survey: <http://tinyurl.com/curc-survey18>
- Contact information: rc-help@Colorado.edu
- Slurm Commands: <https://slurm.schedmd.com/quickstart.html>

Minimal Batch Script

What is the *minimum* required by a batch script?

```
#!/bin/bash
```

```
## Directives (HPC Resources)
```

```
## Software
```

```
## User Scripting
```

Minimal Batch Script

```
sacct -j 9151881 --format=account,elapsed,ncpus,qos,reqmem
```

Account	Elapsed	NCPUS	QOS	ReqMem
ucb-general	00:00:03	1	normal	4848Mc
ucb-general	00:00:03	1		4848Mc
ucb-general	00:00:03	1		4848Mc

Running from Different Directory

- What happens if we try to run a script from a different directory?
 - either by giving full or relative address?
- Where does the output go?
- Potential benefits?

Running from Different Directory

```
#!/bin/bash

## Directives (HPC Resources)

## Software

## User Scripting
pwd
touch test.txt
```

Running to Different Directory

- What happens if we try to run a script and change into a different directory?
- Where does the output go?
- Potential benefits?

Running to Different Directory

```
#!/bin/bash

## Directives (HPC Resources)

## Software

## User Scripting
echo "starting dir: $(pwd)"
touch start.txt
cd ../test
touch end.txt
echo "ending dir: $(pwd)"
```