

---

# High Performance Computing (HPC) Crash Course

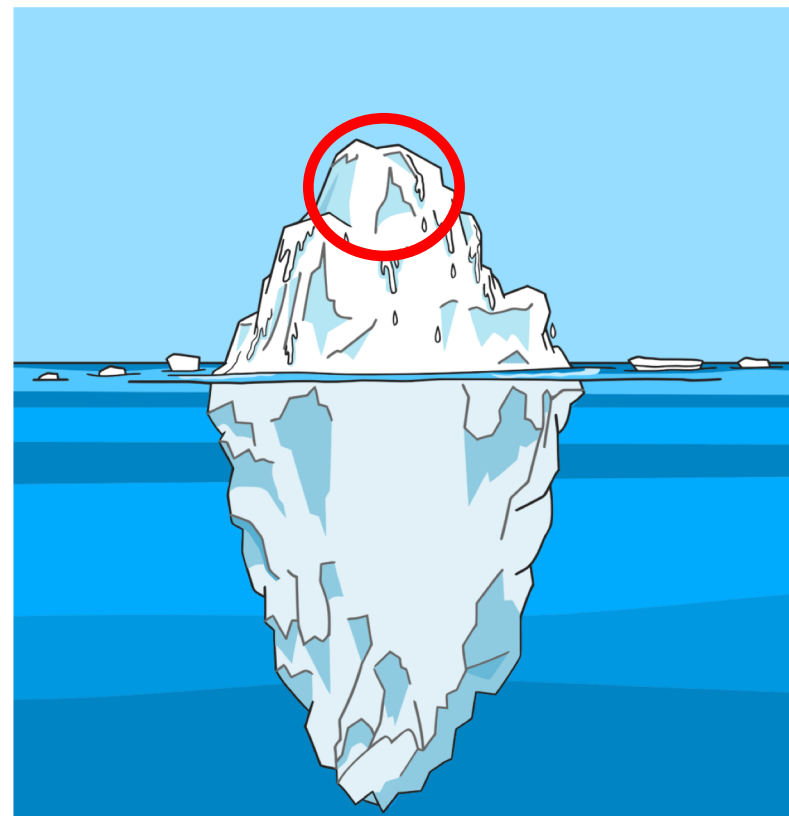
Brandon Reyes

For help email: [rc-help@colorado.edu](mailto:rc-help@colorado.edu)



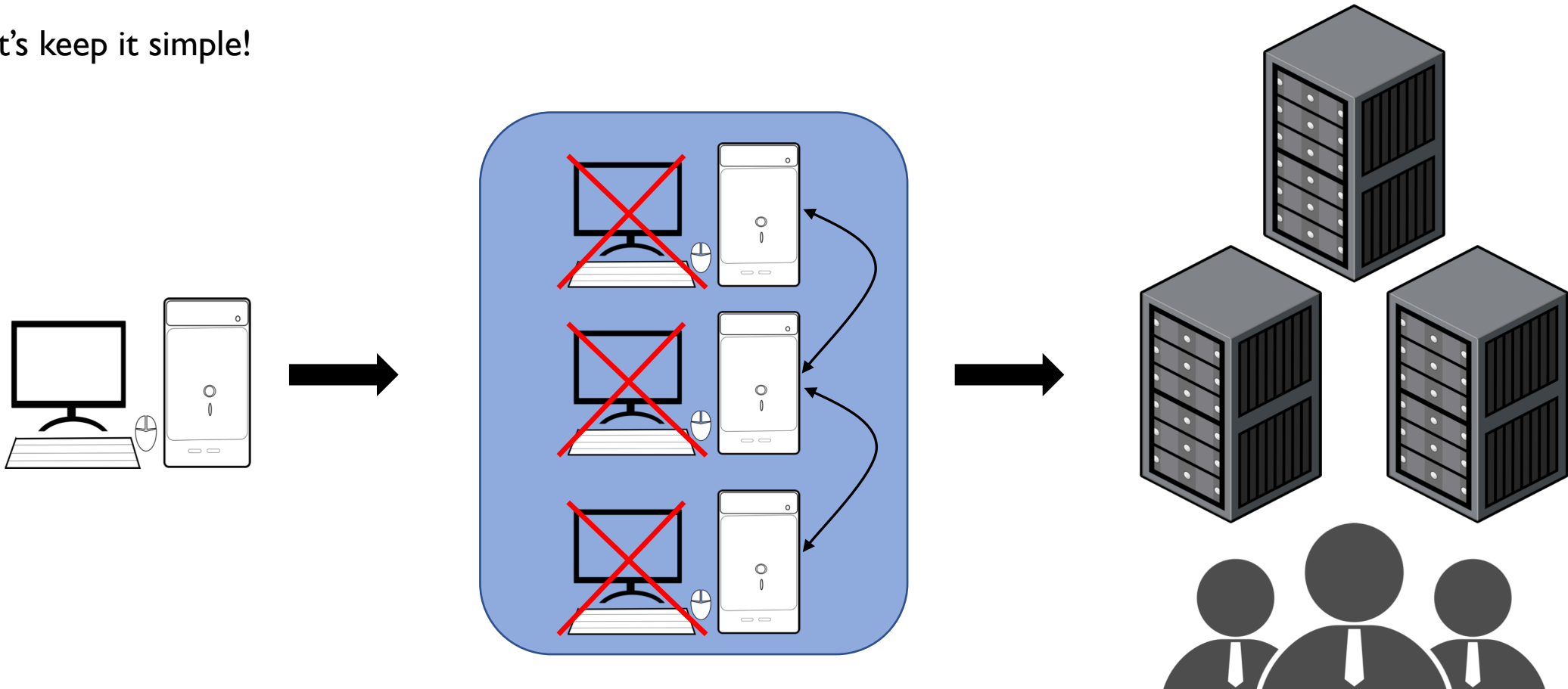
# WHAT DO WE WANT TO ACHIEVE?

- **Goal:** Obtain an overview of a common workflow on an HPC system
- **Audience:** Individuals who are new to HPC systems



# WHAT IS AN HPC SYSTEM?

Let's keep it simple!



## SOME PERKS OF AN HPC SYSTEM

Perks	HPC System
Common software and programming languages are easily available (modules)	✓
Large storage, RAM, and number of CPUs or GPUs (+ multiple nodes)	✓
Independent tasks or parallel applications can be completed significantly faster	✓

# ALPINE STATS

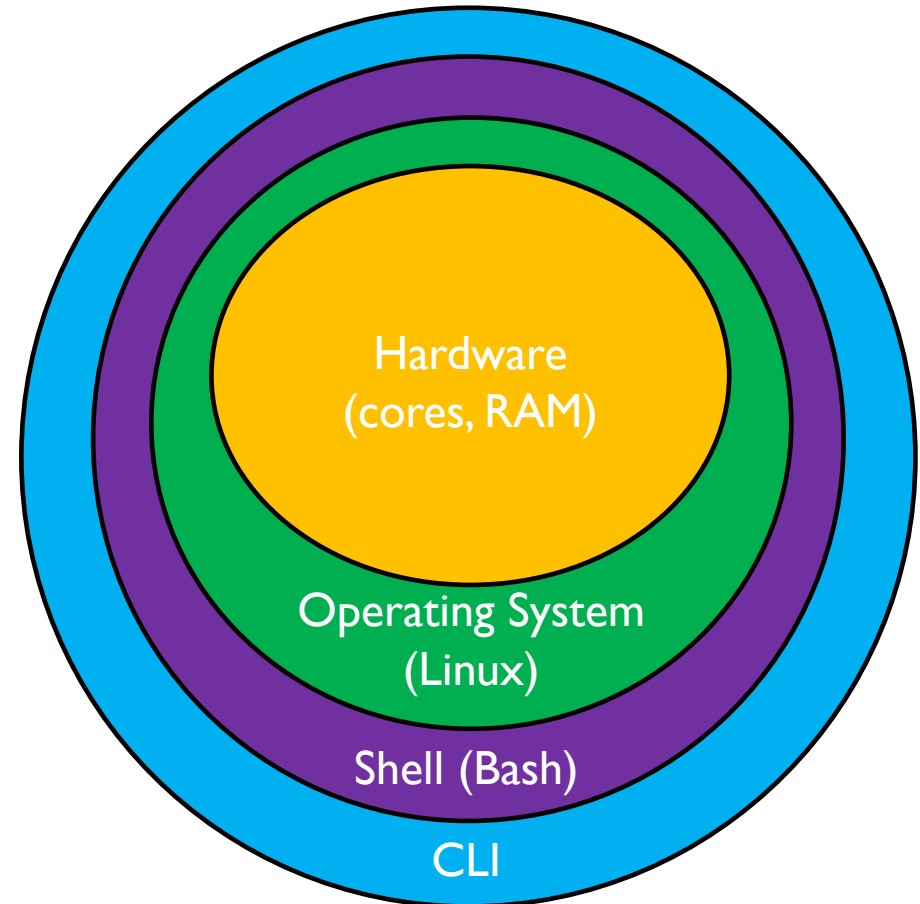
Partitions available on Alpine:

Partition	Description	# of nodes	cores/node	RAM/core (GB)
amilan	AMD Milan (default)	184	64	3.74
ami100	GPU-enabled (3x AMD MI100)	8	64	3.74
aa100	GPU-enabled (3x NVIDIA A100)	8	64	3.74
amem <sup>1</sup>	High-memory	4	48	21.486
csu	Nodes contributed by CSU	77	32 or 48	3.74
amc	Nodes contributed by AMC	20	64	3.74

~ 239 GB per node!

# COMMAND-LINE INTERFACE (CLI)

- Text-based interaction with operation system
- A simple HPC workflow will use the CLI to:
  - Access HPC
  - Work with folders (directories) and files
    - View or edit files (text editor)
  - Submit our code to the HPC system
  - Help us monitor the HPC jobs



# WHAT DOES A CLI LOOK LIKE?

## Real world CLI

```
root@abe5c26dbf5d:/home# ls
HOMEWORK      data_analysis.py
Presentations hello_world.cpp
README.txt    slurm_script.sh
```

## Documentation CLI

```
$ ls
```

# ACCESSING AN HPC SYSTEM

- Create a [Research Computing account](#)
- Set up [Duo 2-factor Authentication](#)
- Open a terminal



```
$ ssh username@login.rc.colorado.edu
```

- ssh is a secure way to access a computer over a network



```
$ module load slurm/alpine
```

- Allows one to submit jobs to Alpine (more on this later)



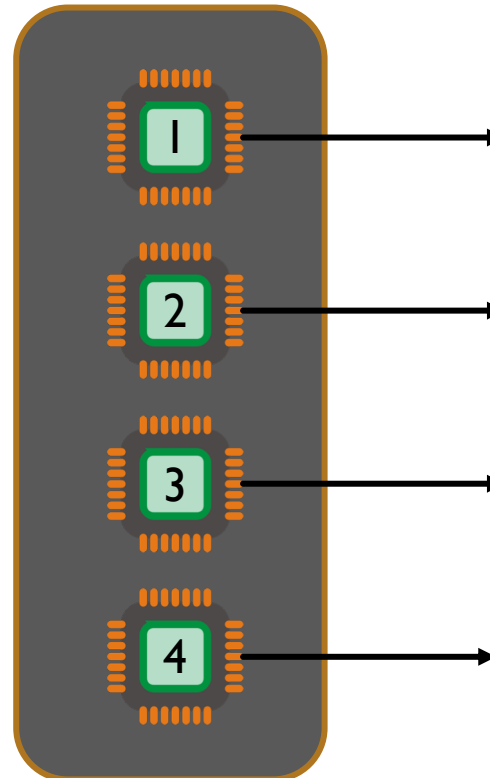
# PARALLEL APPLICATION

## Parallel Application Pseudocode

```
1 # assign work to core
2 if core number is even:
3     print("Even")
4 else:
5     print("Odd")
```



## Node



## Output

Odd

Even

Odd

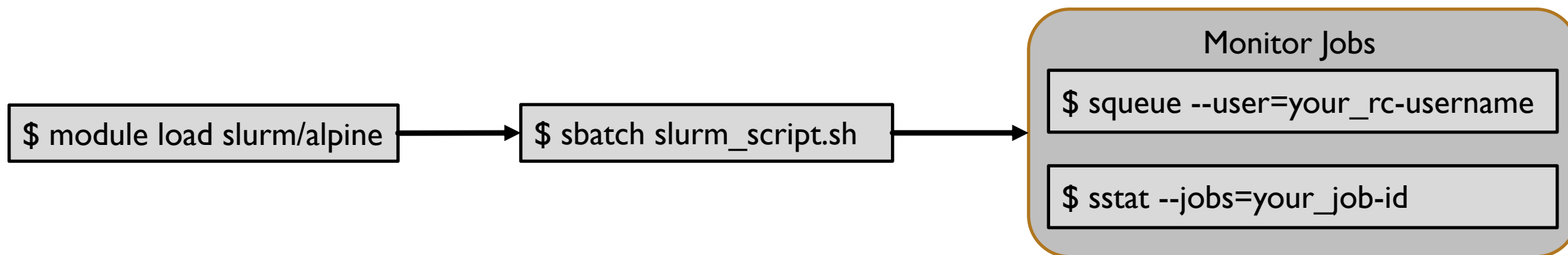
Even

# SLURM JOB SCRIPTS

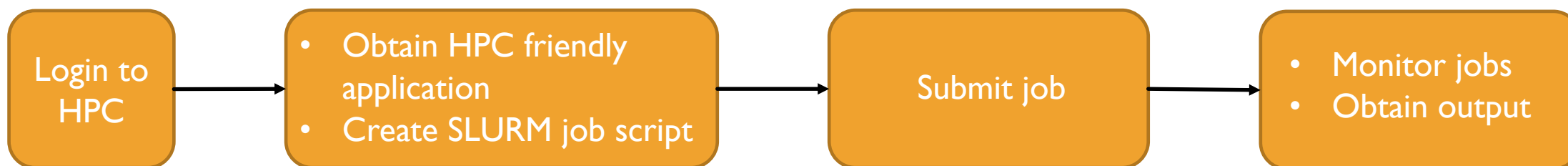
slurm\_script.sh

<input type="radio"/>	Account to be used
	Hardware to be used
	Runtime of job
<input type="radio"/>	Job name
	Output location
	Modules needed
<input type="radio"/>	Program to be run

# SUBMITTING A SLURM SCRIPT



# OVERVIEW OF HPC WORKFLOW



# FURTHER HELP

- Documentation
  - [curc.readthedocs.io](https://curc.readthedocs.io)
- [rc-help@colorado.edu](mailto:rc-help@colorado.edu)

Research Computing University of Colorado Boulder  
latest

Search docs

Frequently Asked Questions

[ACCESSING RC RESOURCES](#)

Logging In  
Duo 2-factor Authentication  
RMACC Access to Summit

[THE COMPUTE ENVIRONMENT](#)

Node types  
Filesystems  
The Modules System  
Data Transfer  
Compiling and Linking  
Monitoring Resources

[CLUSTERS](#)

Alpine  
Blanca  
Summit

[RUNNING JOBS](#)

Running applications with Jobs  
Batch Jobs and Job Scripting  
Interactive jobs  
Useful Slurm commands  
Slurm Flags, Partitions, and QoS  
squeue status and reason codes  
Using the Summit ssky Condo Partition

[STORAGE](#)

Read the Docs v: latest

Docs » Research Computing User Guide

[Edit on GitHub](#)

## Research Computing User Guide

Documentation covering the use of Research Computing resources.

Here are some quick links into the documentation to get you started.

- [Logging In](#)
- [Research Computing Filesystems](#)
- [Compiling Software](#)
- [Batch Jobs](#)
- [The Module System](#)
- [Frequently Asked Questions \(FAQ\)](#)

Can't find what you need? [Provide feedback on the CURC docs!](#)

More information is available at <https://www.colorado.edu/rc>.

If you have any questions, please contact [rc-help@colorado.edu](mailto:rc-help@colorado.edu).

## Courses using RC Resources

Students are welcome to use RC resources on their own for class projects and can request access as a regular UCB affiliate via the link off the RC homepage at: <https://www.colorado.edu/rc>. When requesting help please indicate that the work is for a class project and any deadlines. If students are to be required to use RC resources for a class, see below.

Instructors who wish to lead a class using RC resources must contact us at [rc-help@colorado.edu](mailto:rc-help@colorado.edu) before the class begins. This is to ensure that our resources can meet your needs and if adequate resources and support are available. Early in the process we will need to know details about the proposed class usage such as:

- Number of students
- Software needed, and if it will be installed by instructor/TA
- Typical computational work (number of jobs or sessions, length, number of CPUs)
- Date of 1st usage in class/lab
- Class roster including TAs and auditors.



Thank You!



## WHEN SHOULD YOU USE AN HPC SYSTEM?

- Your calculation consumes more memory (RAM) than you have
- You need to run a large number of tasks that are independent
  - Your application can be run in parallel (more on this later)



# WHAT DOES A CLI LOOK LIKE?

## Real world CLI

```
root@abe5c26dbf5d:/home# ls
HOMEWORK      data_analysis.py
Presentations  hello_world.cpp
README.txt     slurm_script.sh
```

## Documentation CLI

```
/home $ ls
HOMEWORK      data_analysis.py
Presentations  hello_world.cpp
README.txt     slurm_script.sh
```

## Access to a CLI:

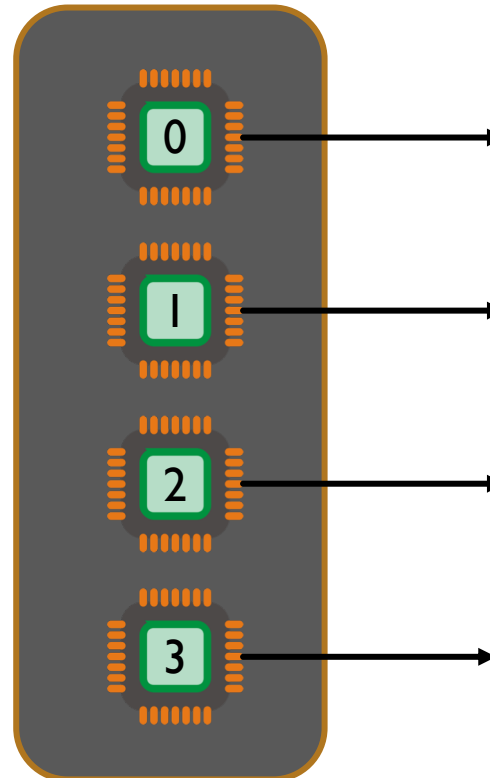
- macOS and Ubuntu (Linux)
  - Start up a terminal
- Windows
  - PuTTY (terminal emulator)

# PARALLEL APPLICATION

## Parallel Application Pseudocode

```
1  # get parallel library
2  Load_MPI(...)
3
4  # get core number
5  core = MPI_get_core_number(...)
6
7  # print core number
8  print("core number: " core)
```

## Node



## Output

core number:0

core number : 1

core number : 2

core number : 3

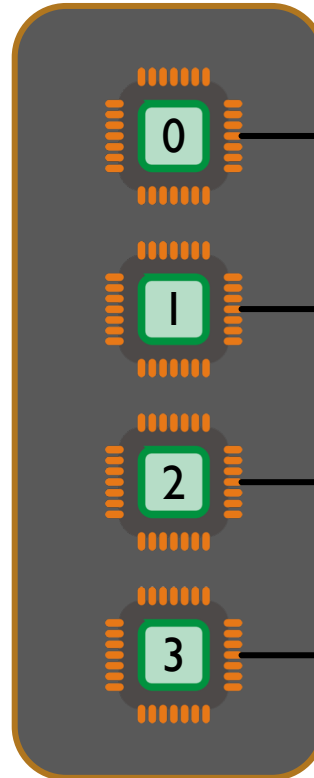
# PARALLEL CONCEPT

## Parallel Application

### MPI Hello World

```
1  #include <iostream>
2  #include <mpi.h>
3
4  int main(int argc, char **argv)
5  {
6      // initialize MPI
7      MPI_Init(&argc, &argv);
8
9      // Get the rank of the running process
10     int my_rank;
11     MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
12
13     // print the rank
14     std::cout << " my_rank: " << my_rank << std::endl;
15
16     //finalize MPI
17     MPI_Finalize();
18
19     return 0;
20 }
```

## Node



## Output

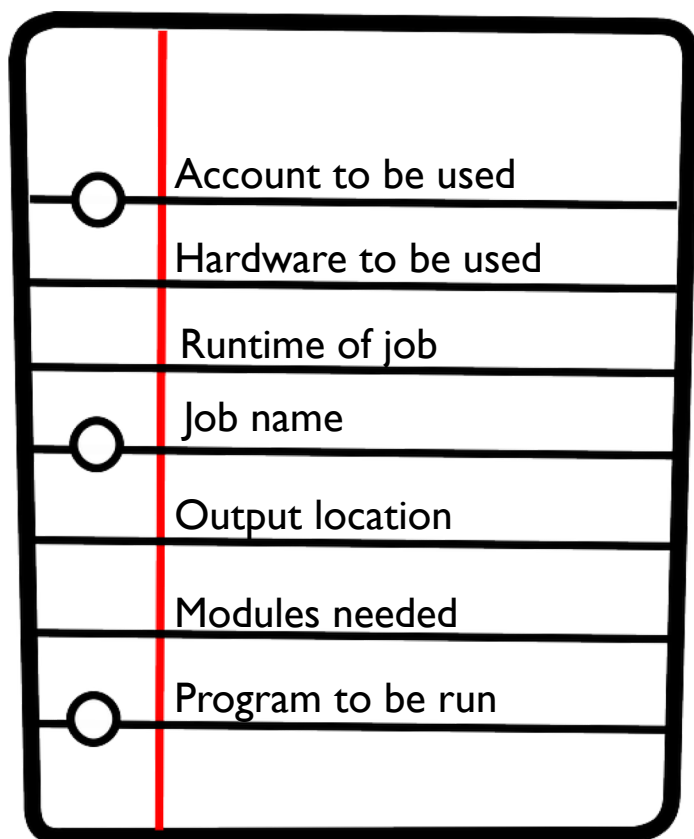
my\_rank: 0

my\_rank: 1

my\_rank: 2

my\_rank: 3

# SLURM JOB SCRIPTS



## slurm\_script.sh

```
1  #!/bin/bash
2
3  #SBATCH --account=...
4
5  #SBATCH --partition=amilan
6  #SBATCH --nodes=1
7  #SBATCH --ntasks=4
8
9  #SBATCH --time=01:00:00
10
11 #SBATCH --job-name=hello-world
12
13 #SBATCH --output=hello-world.%j.out
14
15 module purge
16
17 module load gcc
18 module load openmpi
19
20 mpirun -np 4 ./hello_world.exe
```

# SUBMITTING A SLURM SCRIPT

