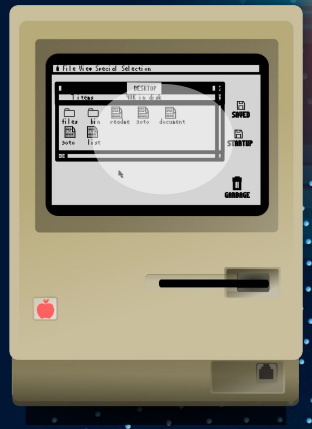
Managing and Optimizing Your Jobs on HPC

Tips to improve your workflow and usage of computational resources

Provided by the University of Arizona HPC Center

Outline

1. **Prerequisites and Overview**
2. **Job Management**
   a. Open OnDemand
   b. Command Line
3. **Job Optimization**
   a. Overview
   b. Resource Requests
   c. Hardware
   d. Testing
   e. Configuration



Prerequisites

familiarity with HPC system

- system **access** (both SSH and OnDemand)
- system **layout** (node types)
- **batch jobs**
- simple **bash commands**

computer literacy

- terminology like CPU, RAM, storage, etc

Prerequisites

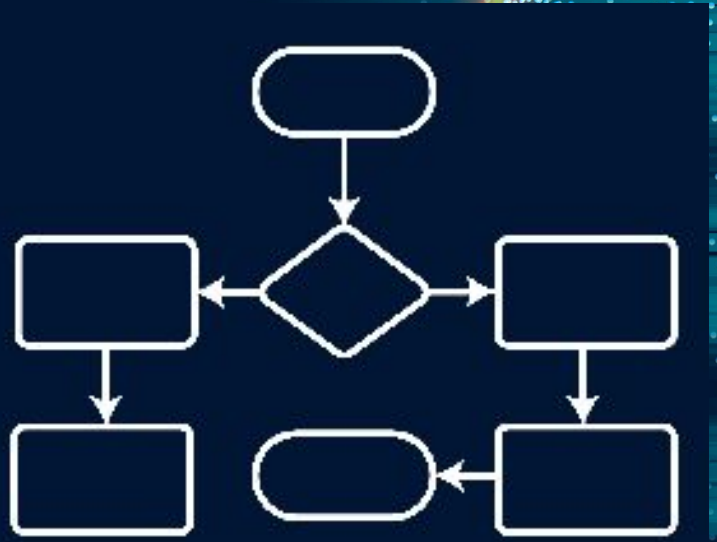
Check out our Intro to HPC course if you are a new user!

Overview

Managing your **workflows** on HPC is a crucial component of the research process

It's quite common for users to have dozens to hundreds or even ***thousands*** of jobs running simultaneously

Alternatively, you may have a small number of *highly resource-intensive* jobs



Overview

In this workshop, you will learn:

- available **tools** and **best practices** for keeping track of your work
- tailoring *resource requests* for all job types
- system commands for tracking job performance
- general practices for configuring software to run more **efficiently**

What is job management?

- keeping track of your active jobs and how many resources they are utilizing
- keeping track of how previous jobs have performed
- awareness of and adherence to best practices

Why is job management important?

- help keep your work organized
- have a clearer understanding of how HPC works
- improve your research throughput

What is job optimization?

- requesting the *optimal* amount of compute resources
  - not to much, not too little
  - requires understanding your compute needs
- configuring and/or executing your software to use those resources most effectively

Why is job optimization important?

- get your jobs to run faster, or larger
- utilize your CPU-time allocation more efficiently
- free up idle resources for other HPC users

Job Management

how to **keep track of** and
get information on your jobs
- both *current* and *previous*

Job Management

2 main modalities for HPC interface:

- *you may use just one, or perhaps both, depending on your needs*

graphical web browser

command line interface

```
Last login: Mon Apr 22 14:30:18 2024 from gatekeeper.hpc.arizona.edu
***
The default cluster for job submission is Puma
***
Shortcut commands change the target cluster
----------------------------------------
Puma:
$ puma
(puma) $
Ocelote:
$ ocelote
(ocelote) $
ElGato:
$ elgato
(elgato) $
----------------------------------------
```

Job Management - Job Types

graphical web browser

- Interactive Desktop
- Graphical Apps (e.g. Ansys, Matlab)
- Servers (e.g. Jupyter, R Studio)

command line interface

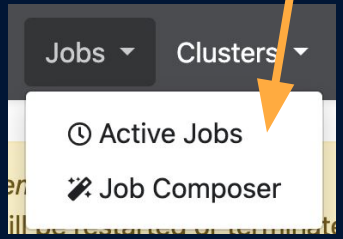
```
Last login: Mon Apr 22 14:30:18 2024 from gatekeeper.hpc.arizona.edu
***
The default cluster for job submission is Puma
***
Shortcut commands change the target cluster
----------------------------------------
Puma:
$ puma
(puma) $
Ocelote:
$ ocelote
(ocelote) $
ElGato:
$ elgato
(elgato) $
----------------------------------------
```

- Interactive terminal sessions
- Batch jobs

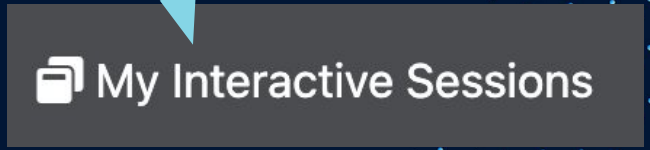
Available Tools

There are a few helpful options for job management in OOD, but they are somewhat limited

See all **active jobs**



See your **interactive desktops**



OPEN OnDemand

Limitations

These are great tools for viewing and keeping track of jobs that are ***currently running*** *or* ***pending***

If we want to...

- view our *previous jobs*
- get more *detailed information*

...we have to hop on the `command line`

Detailed Job Management

To get more details about your jobs, access the system via **SSH**

This is a way of accessing the HPC from a terminal window

- **Mac & Linux:** built-in command
- **Windows:** download SSH client like PuTTY

```
Last login: Mon Apr 22 14:30:18 2024 from gatekeeper.hpc.arizona.edu
***
The default cluster for job submission is Puma
***
Shortcut commands change the target cluster
----------------------------------------
Puma:
$ puma
(puma) $
Ocelote:
$ ocelote
(ocelote) $
ElGato:
$ elgato
(elgato) $
----------------------------------------
```

Let's Try It! 🤝🎉

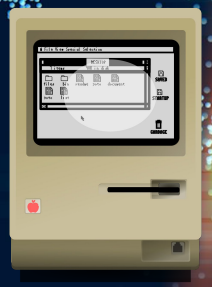
If you have never used SSH to access the system, try it now.

Pause the video, and follow these steps:

1. Go to docs.hpc.arizona.edu, and find the page on **System Access**
2. Follow the instructions for your system type
3. Make sure to access the ***login node!*** You should see "wentletrap" or "junonia" before your username:

```
(puma) [ejahn@wentletrap ~]$
```

review our ***system layout*** page if you don't know what that is!



Batch Job Recap

In case you are unfamiliar with batch jobs, let's recap:

Batch jobs are a way to **run code automatically**, without user oversight or even an active connection to HPC

Just like all other job types, batch jobs are **allocated compute resources** through the **task scheduler**, called Slurm

Unlike other job types, batch jobs require a **batch script** that determines **everything the job will do** *in advance*

The batch script is submitted using `sbatch <myscript.slurm>`, and will proceed through the queue *automatically*

Batch Job Recap

Example Batch Script:

```
#!/bin/bash

# --------------------
### Directives Section
# --------------------
#SBATCH --job-name=hello_world
#SBATCH --account=your_group
#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=00:01:00

# --------------------
### Code Section
# --------------------
module load python/3.9
cd ~/hello_world
python3 -c "print('hello world')"
### sleep is used for demonstration purposes
sleep 30
```

Sections:

“shebang”

directives request compute resources

code tells the computer what to run

Detailed Job Management

Slurm is the **task scheduler** used by the UArizona HPC (and many others)

There are ***two*** main commands that Slurm provides to get more details about your jobs:

- `squeue` : current (running, pending) jobs
- `sacct` : finished (completed, canceled, failed) jobs

Both have many options to get details on your jobs

See **slurm.schedmd.com/squeue.html** and **slurm.schedmd.com/sacct.html** for full list of options

squeue simple example

command:

```
squeue
```

output:

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

for which jobs?

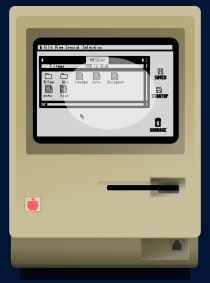
every single currently running or pending job on the system

Let's Try It! 🤝🎉

In your terminal session where you SSH'ed into the HPC, type the command `squeue` and hit enter.

What do you see?

Is this output useful?



more squeue options

command:	explanation:
`squeue | head -n 15`	print top 15 lines
`squeue | tail -n 15`	print bottom 15 lines
`squeue | wc -l`	count the number of lines
`squeue --user=$USER`	print only my jobs, or substitute for another username to get their jobs

squeue simple example

```
(puma) [ejahn@wentletrap ~]$ squeue | head -n 15
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9534978 high_prio blat_001 [redacted] CG      36:36      1 r4u39n1
           9469546 high_prio Job_DM_s [redacted] PD       0:00      1 (QOSGrpCPUMinutesLimit)
           9469547 high_prio Job_DM_s [redacted] PD       0:00      1 (QOSGrpCPUMinutesLimit)
9622767_[14-17,19- high_prio phytoora [redacted] PD       0:00      1 (QOSGrpCPUMinutesLimit)
     9662093_[191] high_prio basilmea [redacted] PD       0:00      1 (Dependency)
     9662092_[191] high_prio basilmea [redacted] PD       0:00      1 (Dependency)
     9662091_[191] high_prio basilmea [redacted] PD       0:00      1 (Dependency)
     9662090_[191] high_prio basilmea [redacted] PD       0:00      1 (Dependency)
     9662089_[191] high_prio preproc_ [redacted] PD       0:00      1 (Dependency)
           9664998 high_prio LT_hourl [redacted]  R    2:01:10     16 r3u06n2,r3u37n2,r4u38n2,r6u
08n1,r6u09n2,r6u10n[1-2],r6u11n[1-2],r[redacted],r6u21n1,r6u23n2
           9633918 high_prio    F1rep [redacted]  R 3-21:16:32      1 r3u40n2
           9614706 high_prio 30_chlor [redacted]  R 6-01:42:48      1 r2u29n2
           9664833 high_prio fs1.1-te [redacted]  R    2:36:33      1 r2u40n2
           9614708 high_prio 30_chlor [redacted]  R 6-01:42:48      1 r2u35n2
```

squeue advanced example

command:

```
squeue -o "      %10i %12P %10j %10u %8T %8M %6D %5C %10q %R" -u $USER
```

specify jobs run by current user

output:

```
JOBID       PARTITION     NAME       USER       STATE     TIME     NODES  CPUS  QOS        NODELIST(REASON)
```

other options:

`--state=R,PD`	specify running or pending jobs
`--partition={option}`	specific name of partition to look at

sacct

very similar syntax and options as squeue (though not exactly the same)

for brevity, we'll skip a detailed example

check the official slurm documentation, or numerous online tutorials, for detailed instructions

One More Slurm Command: scontrol

scontrol can be used to – you guessed it – control your jobs

First, get your job ID from squeue. Then, use a command:

```
scontrol <command> <job-id>
```

options for <command>

suspend	- pauses a job that is currently running
resume	- resumes a job paused with **suspend**
hold	- prevent a queued job from starting
release	- releases the held job
show job	- provides detailed information on a job

More Job Management Tools

System Usage (use these to determine how busy the system currently is)

- `nodes-busy` - details for current cluster only (graphical)
- `cluster-busy` - overview of whole system (graphical)
- `system-busy` - overview of whole system (text only)

Your Jobs

- `past-jobs -d <N>` - list my jobs from the last N days on the current cluster
- `seff <job-id>` - efficiency report
- `job-history <job-id>` - detailed job info

Job Management - Recap

We just learned:

- how to see *current* and *pending* jobs in the **Open OnDemand** browser
- how to **SSH** into the system
- how to use **Slurm** commands `squeue`, `sacct`, and `scontrol` to view details of *running*, *pending*, and *completed* jobs
- other system commands to help with job management

What's next...?

Job Optimization

how to get the most out of your jobs, and out of your standard CPU-time allocation

Job Optimization - Why?

With so many CPUs available on the clusters, why should I bother optimizing my requests?

- **get more out of your allocation**
  - **150,000** CPU hours per month on **Puma**
  - **100,000** CPU hours per month on **Ocelote**
  - If your allocation is shared between **many users**, or if you tend to *run out of CPU hours* by the end of the month, optimization practices can help **improve your research throughput**
- **improve overall HPC efficiency**
  - If your jobs are allocated resources that go unused, those are unavailable to other users, causing ***longer wait times***

Job Optimization - Overview

The majority of job optimization can be broken down into two categories:

1. Making appropriate **resource requests**
2. Calling the appropriate **options** when running your software

We will spend most of our time on category 1, as it is the almost generally applicable.

We will discuss runtime options for certain common software, but we cannot cover every software used on HPC, nor every use case for those softwares

Note: It is also possible to further optimize your jobs by developing more efficient code, but that falls more under the umbrella of a research software *engineer/developer* than a typical research software *user*. Optimizing code under the hood can be incredibly detail-oriented, case-specific, and time consuming, so we will not cover it in this tutorial.

Resource Requests - Overview

The goal of making an appropriate resource request is to as closely as possible – match the allocated compute resources with the minimum amount needed by your job to successfully complete.

This means answering:

1. What are the available resources?
2. How many do you need to run your job?

Resource Requests - Overview

1. What are the available resources?

- This is an easy question to answer. We will cover our hardware soon

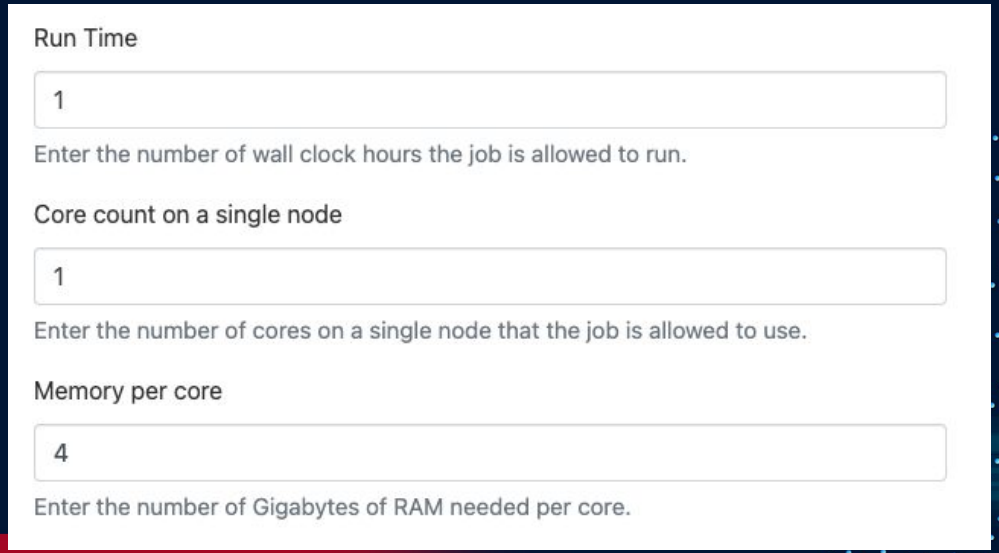
2. How many do you need to run your job?

- This is the harder question. The main way to find out is trial and error, or scaling tests.

How Do I Submit a Resource Request?

OPEN OnDemand

Job Composer will ask you for the necessary information



How Do I Submit a Resource Request?

Command Line → Batch Directives

```
Last l
***
The default cluster for job submission is Puma
***
Shortcut commands change the target cluster
----------------------------------------
Puma:
$ puma
(puma) $
Ocelote:
$ ocelote
(ocelote) $
ElGato:
$ elgato
(elgato) $
----------------------------------------
```

```
# --------------------
### Directives Section
# --------------------
#SBATCH --job-name=hello_world
#SBATCH --account=your_group
#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --time=00:01:00
```

Fill out the **batch directives** section with the relevant information

Memory-per-CPU will be set to default value (standard node) if not specified

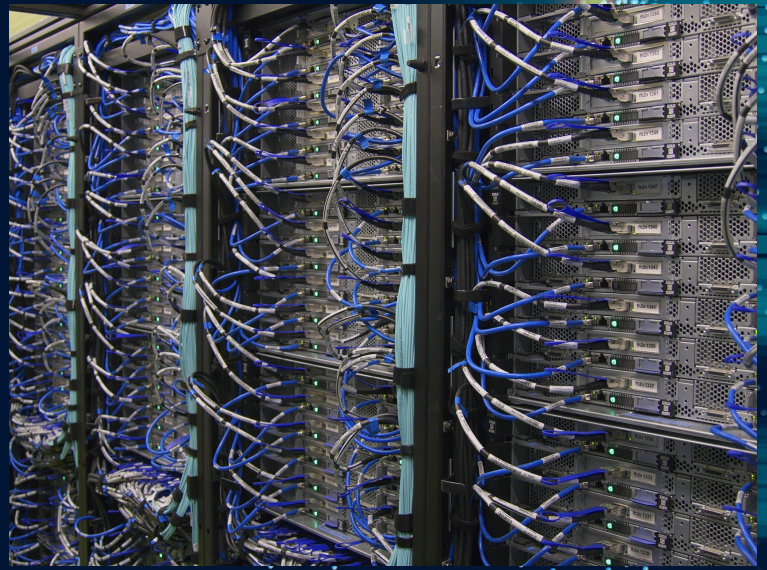
See docs.hpc.arizona.edu for full list of batch directives in the Running Jobs section!

Hardware - Overview

UArizona's HPC Center has three clusters

- Puma
- Ocelote
- El Gato

Each cluster is built from different hardware, so resource requests in Slurm will need to reflect this!



Lots of numbers incoming, no need to memorize!

Hardware - Puma

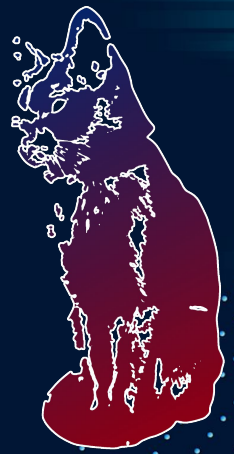
Node Type	Standard	High Memory	GPU
Number of Nodes	192 standard 108 buy-in	3 standard 2 buy-in	8 standard 7 buy-in
CPUs/Node	94	94	94
RAM/CPU	5 GB	32 GB	5 GB
CPU RAM/Node	470 GB	3008 GB	470 GB
GPUs/Node			4
RAM/GPU			32 GB (v100s) 20 GB (MIGs)
GPU RAM/Node			128 GB
Total GPUs			32 standard 28 buy-in

Hardware - Ocelote

Node Type	Standard	High Memory	GPU
Number of Nodes	400	1	46
CPUs/Node	28	48	28
RAM/CPU	6 GB	41 GB	8 GB
CPU RAM/Node	168 GB	1968 GB	224 GB
GPUs/Node			1
RAM/GPU			16 GB
GPU RAM/Node			16 GB
Total GPUs			46

Hardware - El Gato

Node Type	Standard
Number of Nodes	130
CPUs/Node	16
RAM/CPU	4 GB
CPU RAM/Node	64 GB

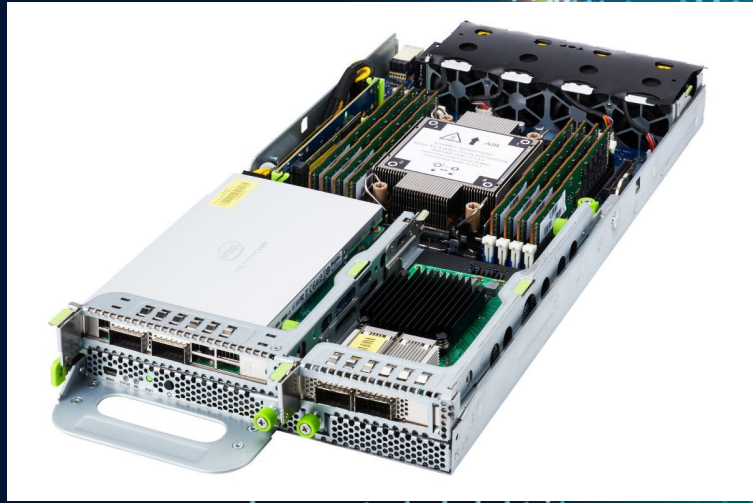


What does this mean?

Common fields in resource requests include:

- number of CPUs
- number of nodes
- memory (RAM)
- GPUs

We need to know what our options are when inputting these fields!



Note on CPUs and Memory

Each compute node has a certain number of processors *physically mounted* on it

Each of those processors has a memory chip *physically connected* to it

That means

Number of CPUs determines Memory*

***think RAM, not disk space!**

Total Mem = Mem/CPU x N(CPU)

Memory is <u>not</u> a continuous scale or arbitrary number!!

Memory Takeaways

- Know the value for RAM per CPU for your machine!
- If using the total memory flag, make sure this equals N_{CPUs} x mem-per-CPU
- Only input allowed values for mem-per-CPU

Optimizing Resource Requests

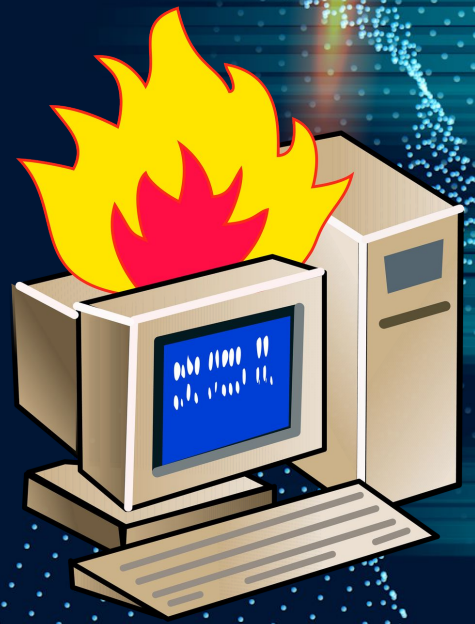
Now we know what our options are for requesting compute hardware

How do we know what the best configuration will be for our jobs?

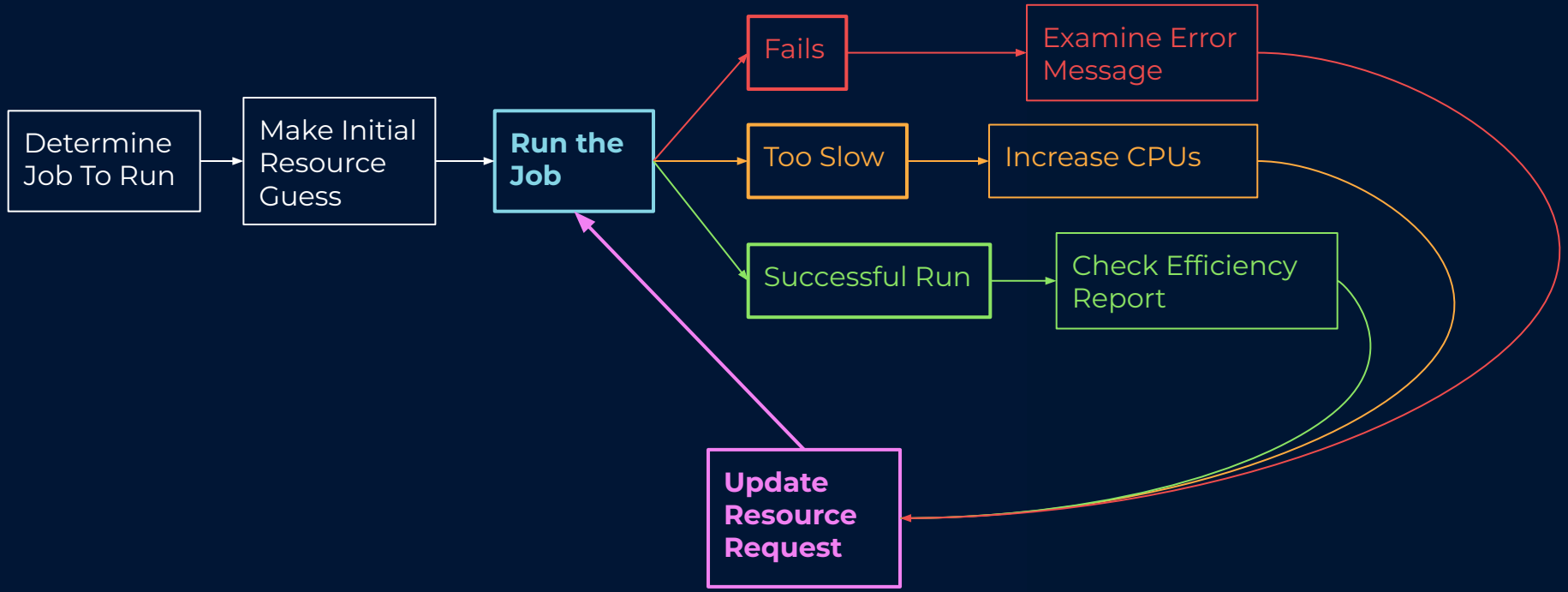
The answer...

- ***Testing***
- ***Trial and Error***
- ***Scaling Tests***

Unfortunately there is no precise formula to tell you exactly which hardware configuration will run you job best...



Testing Overview



Testing Overview - Additional Notes

Job failures can result from *insufficient memory*, or numerous other factors. **Check the logs!**

If **increasing** the number of **CPUs** does not **decrease** the **time** to completion, then your job is ***NOT*** *CPU-Limited*. *Reduce the number of requested CPUs!*

You may need to ***decrease*** the number of CPUs requested for a successful run to obtain an optimal value!

Efficiency Reporting Tools

Once you've run a test job, it's important to view the efficiency report to see how well it utilized the allocated resources

The best two system tools for efficiency are:

`seff <job-id>`	- gives CPU and memory efficiency
`job-history <job-id>`	- provides many job details

Additional online tool: XDMoD

metrics.hpc.arizona.edu

Efficiency Reporting Tools

→ Caveat with "`seff`"

it reports the *average* CPU and memory utilization for a job

These metrics can fluctuate up and down throughout the course of a particular job.

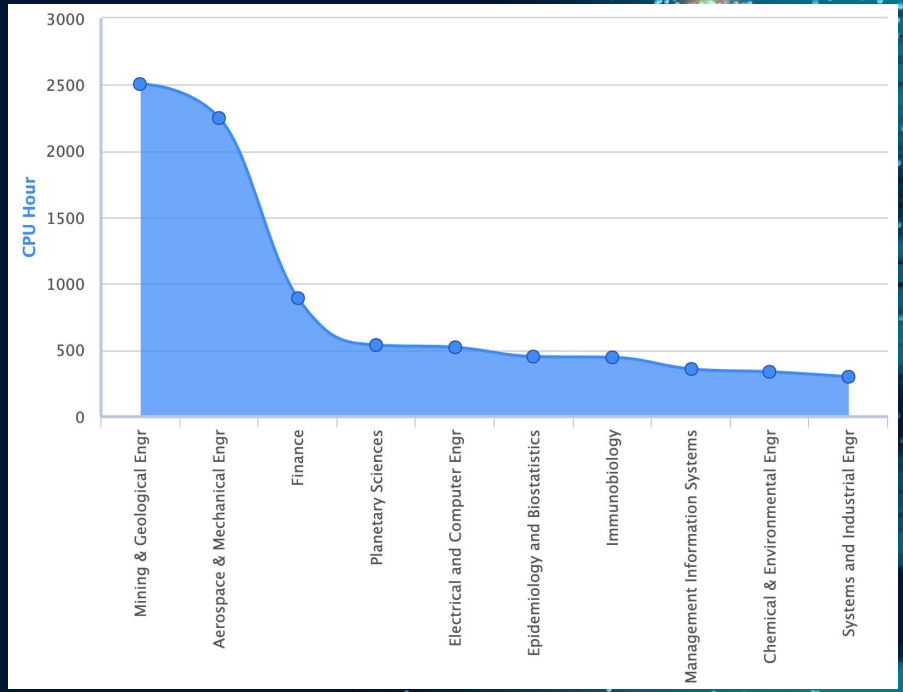
It is always important to check the logs to see what kind of errors are reported! This can indicate that a job may have crashed due to an out-of-memory error even if the seff report says it used less than 100%

Browser Tool: XDMoD

This is a browser-based job reporting interface available at metrics.hpc.arizona.edu

It can produce charts of job statistics, but can be a bit tricky to use.

But it is very powerful. Give it a try!



Configuring Your Software for Optimal Performance

With so many different softwares used on HPC, we can't cover all of the possible use cases.

Check with the developers of your software about their recommended practices for parallelization, or multi-CPU modes.

Important!

→ Your software *may not* automatically detect the number of CPUs you provided in your resource request!

→ You may have to *specifically configure your software to use those CPUs!*

Software Configuration Example

Let's say there is a python script that you found on github to perform your analysis. It has already been written using the *multiprocessing* library. You want to run the script using **16 processors on Puma**.

Try it yourself!

(1) what should the **batch directives** to request these processors look like?
(2) what should the line of **code to execute this program** using all of those processors look like?

Pause the video and write these in a text editor!

Software Configuration Example

(1) what should the **batch directives** to request these processors look like?

```
#SBATCH --ntasks=16
```

(2) what should the line of **code to execute this program** using all of those processors look like?

```
mpirun -np 16 python3 myscript.py
```

Note that we do not simply call “python3 myscript.py” – this will not run it in parallel, it will only use a single processor!

Job Optimization Recap

Why do we optimize our jobs?

- use our allocations more effectively
- allow unused resources to be *available* to other users

How do we optimize our jobs?

- **testing!** make an initial guess, then ***revise*** based on runtime and efficiency metrics
- **configuration!** make sure to call the right **batch directives** and *command line options* when running your code

Thanks for watching!

If you have any questions or would like to follow up about anything, you can contact the UArizona HPC Consult Team:

- ServiceNow (link)
- hpc-consult@list.arizona.edu
- Virtual Office Hours Wednesdays 2-4pm on GatherTown

Or go to docs.hpc.arizona.edu > Support > Consulting Services for more information on services we offer