# HPC at the Smithsonian

## Introduction to Hydra

# Overview

- Why HPC?

- Submitting Jobs

- SI Computing Cluster

- Best Practices

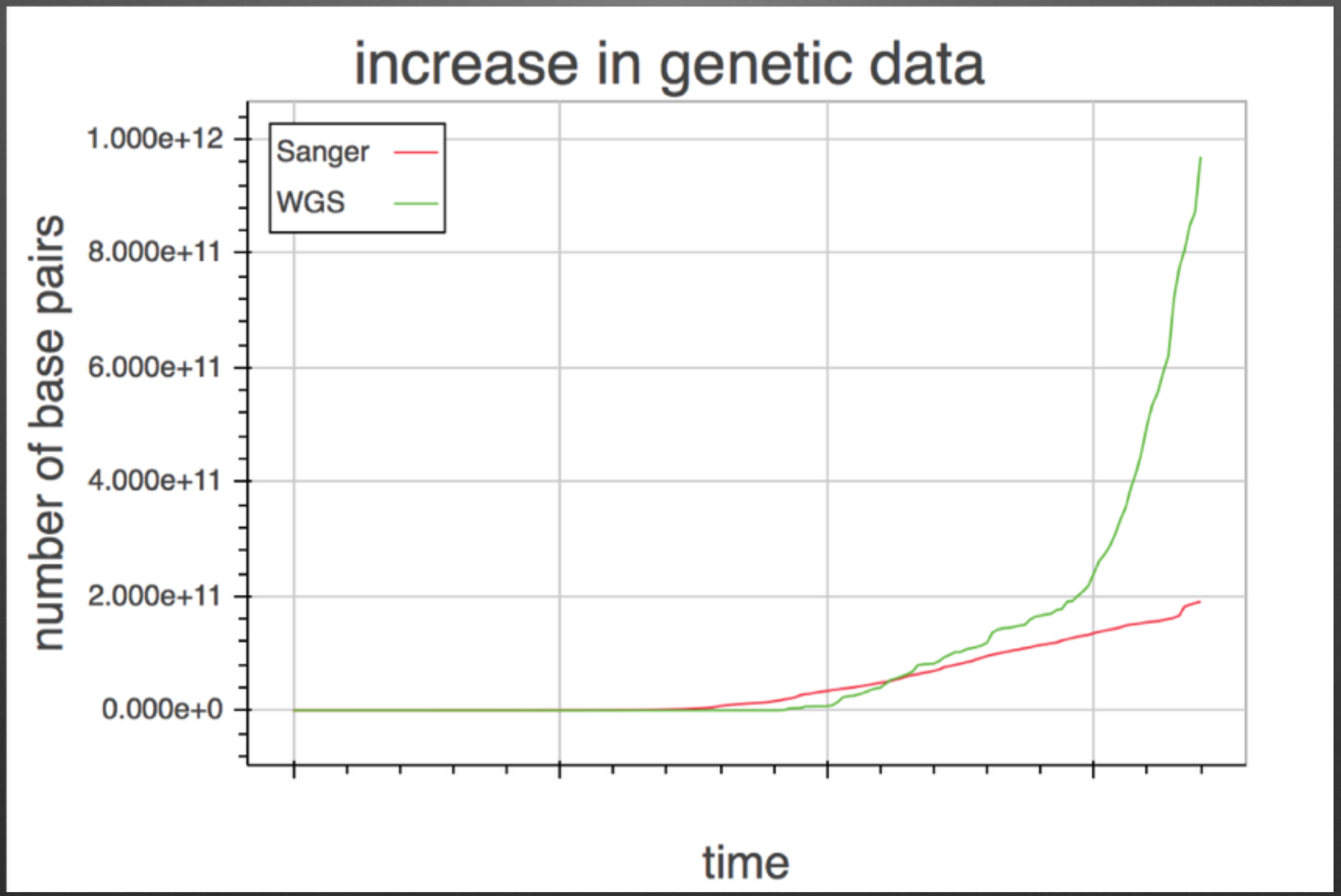
- Support

- Tips

- Cluster Architecture

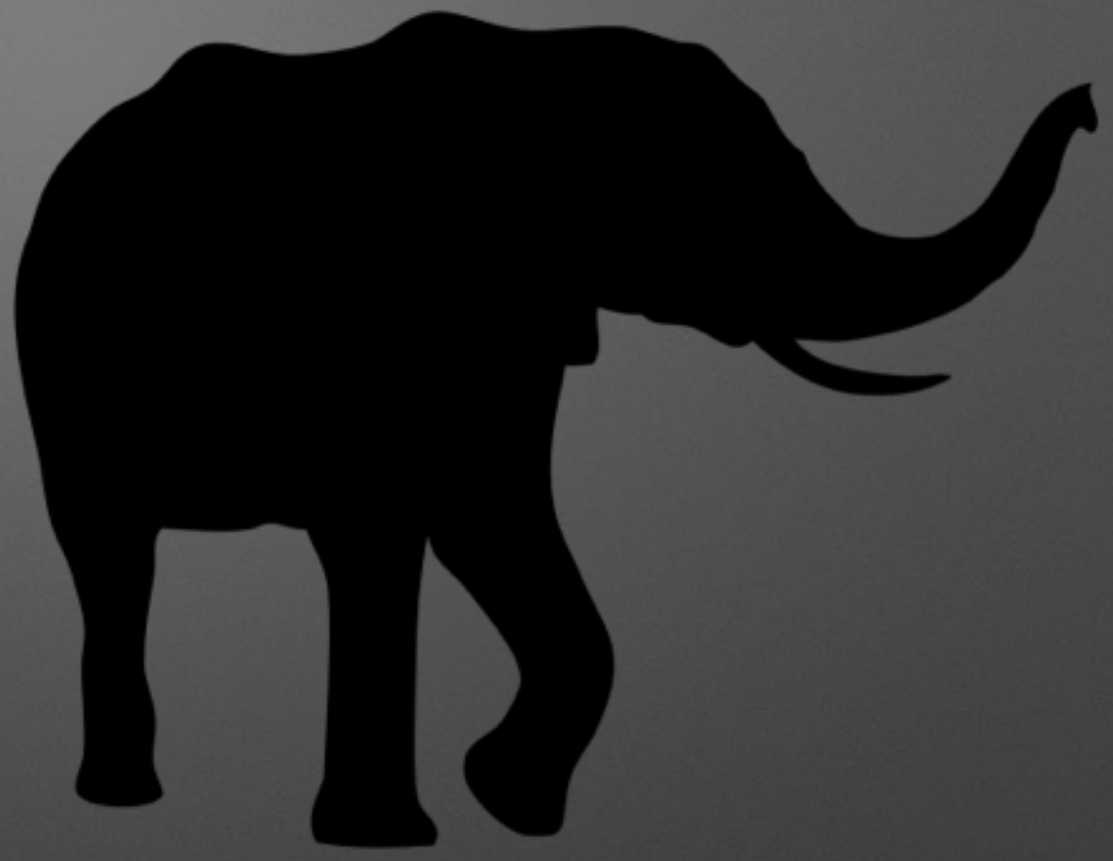
# Why High Performance Computing (HPC)?

# Why HPC?


increase in genetic data
Sanger
WGS
1.000e+12
8.000e+11
6.000e+11
4.000e+11
2.000e+11
0.000e+0
number of base pairs
time

# Why HPC?

# Size

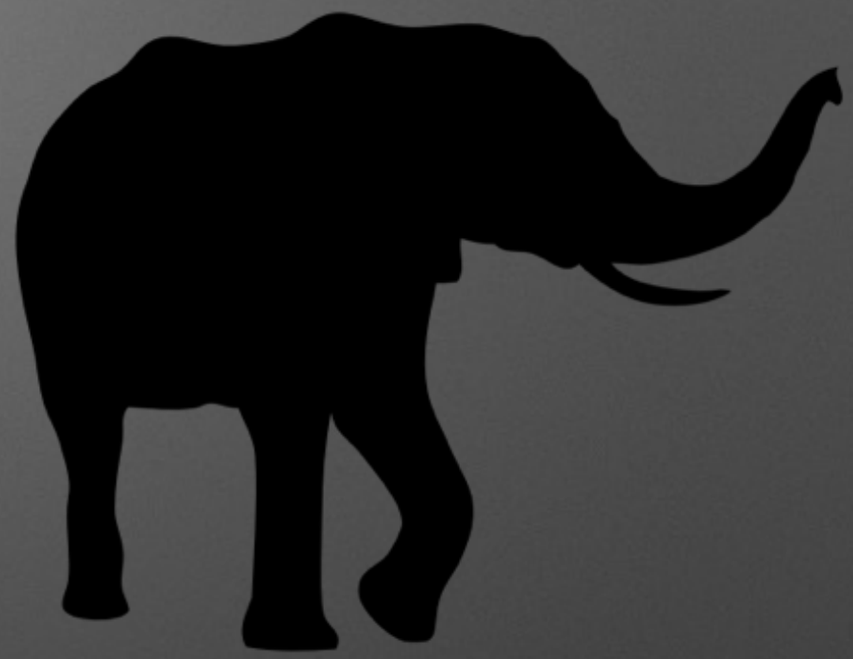


&

# *Speed*

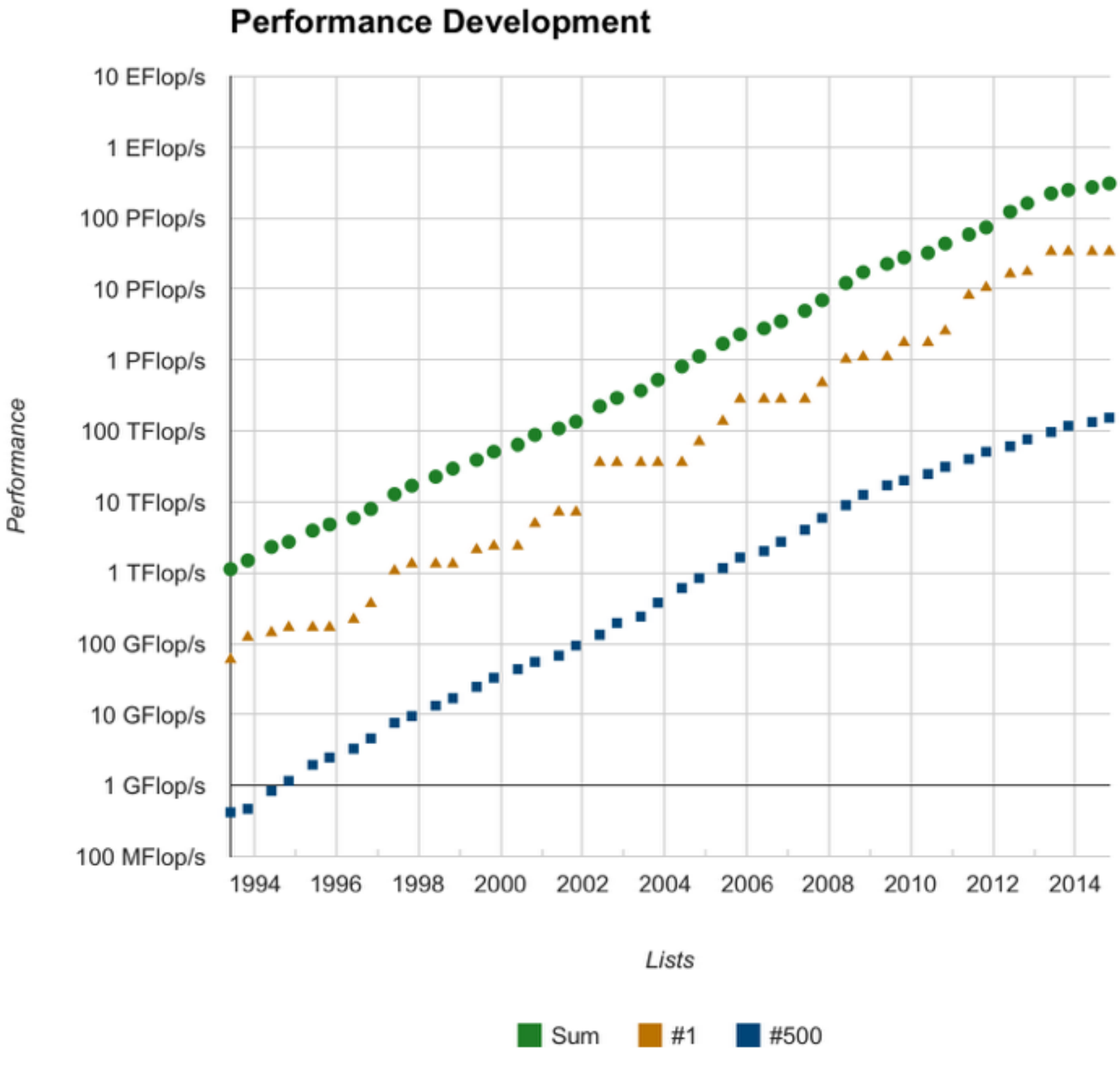
# Why HPC?

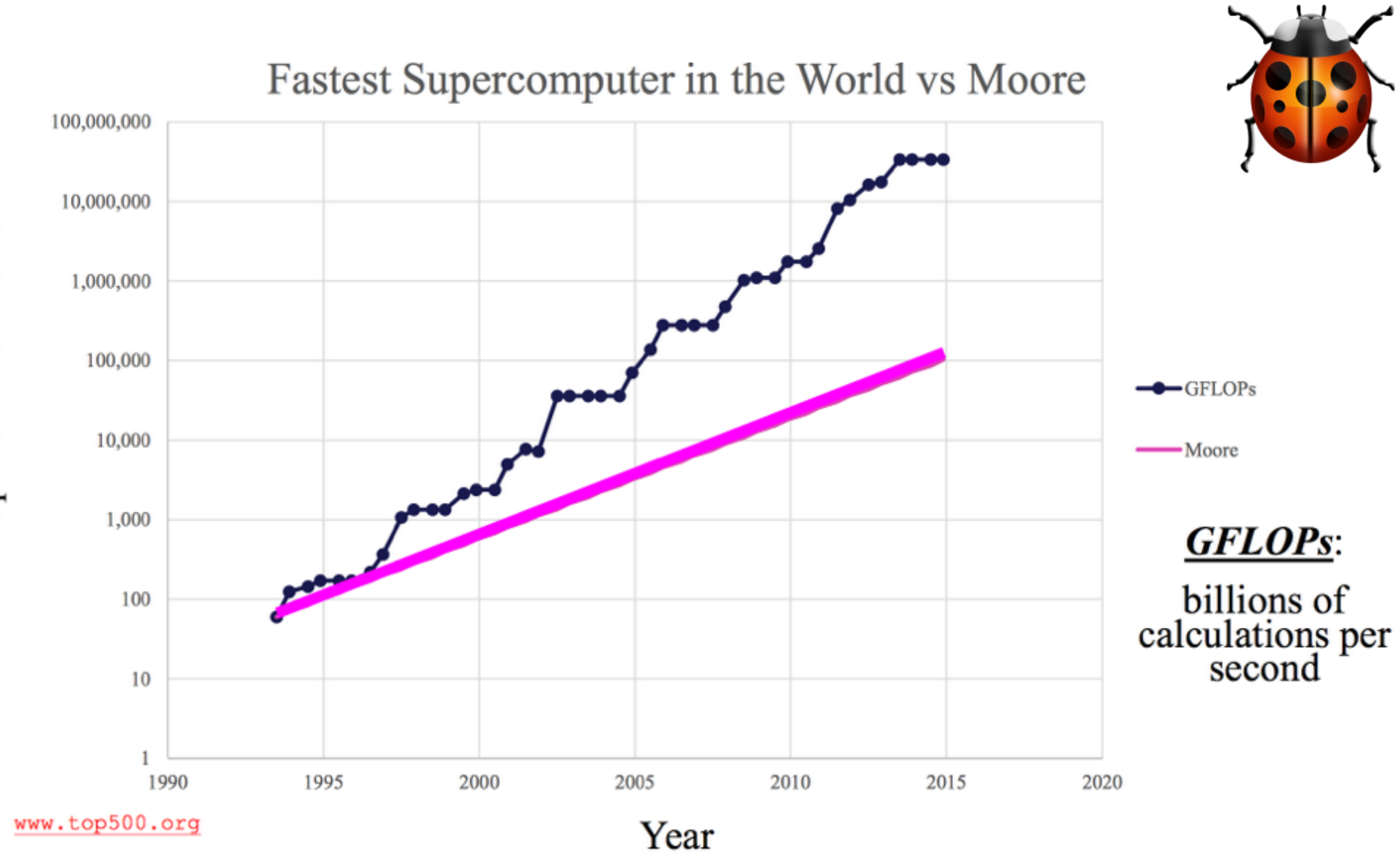
- Size: some analyses use too much memory to be completed on a standard desktop/laptop, e.g. *de novo* genome assembly



- Speed: all about parallel computing! A problem that might take months on a desktop can take hours on a HPC, e.g. BLAST searches

# top500.org


Performance Development
10 EFlop/s
1 EFlop/s
100 PFlop/s
10 PFlop/s
1 PFlop/s
100 TFlop/s
10 TFlop/s
1 TFlop/s
100 GFlop/s
10 GFlop/s
1 GFlop/s
100 MFlop/s
Performance
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
Lists
Sum
#1
#500






***GFLOPs***:

billions of calculations per second

slide from “Supercomputing in Plain English” from University of Oklahoma

# SI Computing Cluster

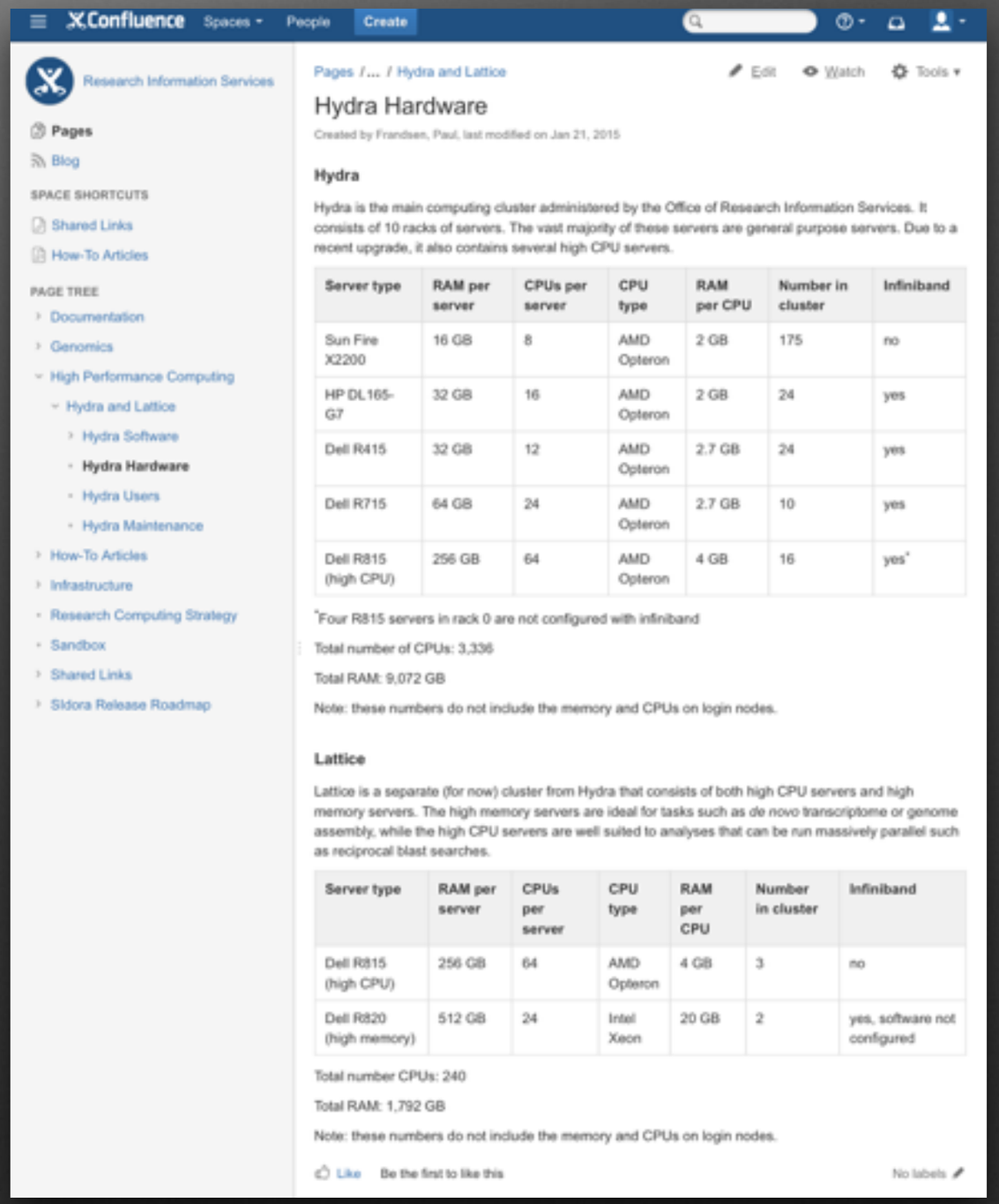
# Hydra

- Smithsonian HPC cluster
- > 3,000 CPUs and > 16 TB of RAM
- 16 high memory nodes (14 512GB; 2 1TB RAM)
- Many high CPU nodes with 64 CPUs

# Support

# Wiki

- New wiki: https://confluence.si.edu



# Hydra support:

**Paul Frandsen, ORIS**

**Sylvain Korzennik, SAO**

**DJ Ding- OCIO**

# Application support:

**Rebecca Dikow, SIBG**

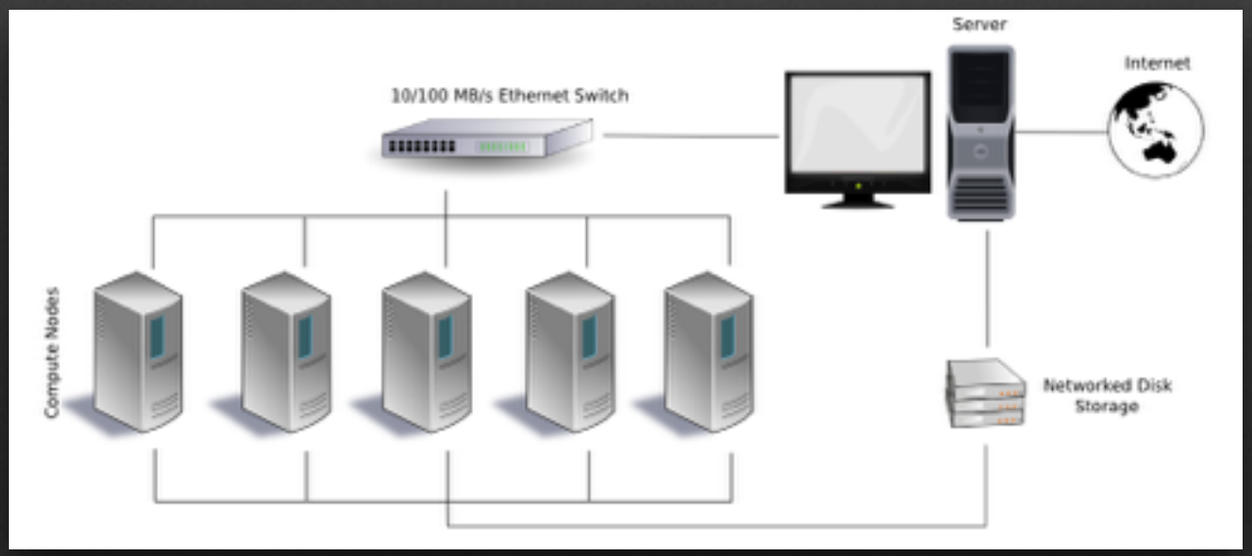
**Vanessa González, GGI**

**Matt Kweskin, LAB**

# Cluster Architecture

# Cluster Architecture

- Many computers connected to each other via an interconnection network (interconnect) with software that controls where jobs are run
- Each computer in the cluster is called a node
  - Login nodes: users log into these to access the cluster and launch jobs, they perform no real computation (hydra-login01.si.edu, hydra-login02.si.edu)
- Compute nodes: actually run your program
- Head node: runs the software that controls the cluster and the jobs



# Queuing System/Job Scheduler

- Software that schedules the jobs and submits them to the compute nodes.
- Hydra uses the “(Sun) Grid Engine” queuing system.
- To submit jobs, you use the command "`qsub`".
- All jobs run in batch mode, not interactive mode.

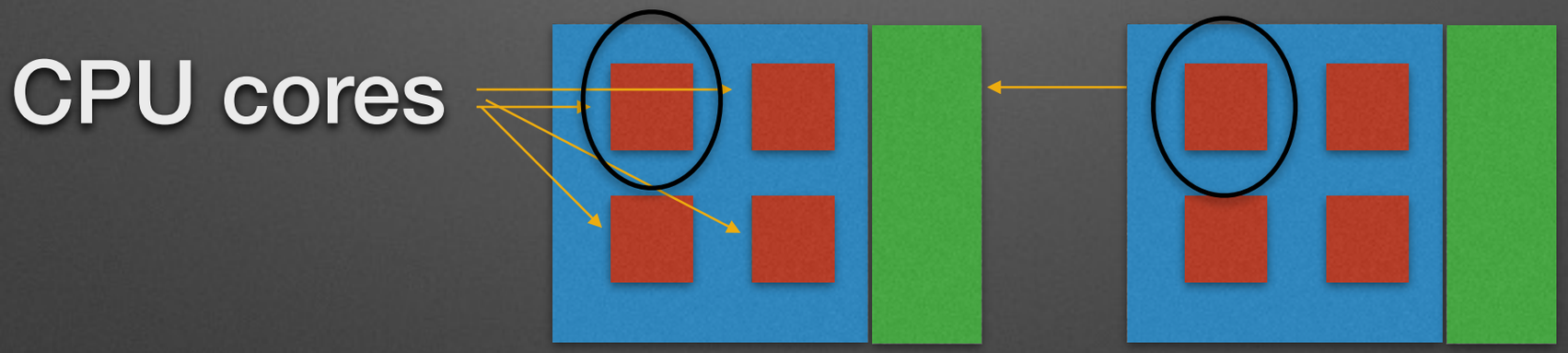
# Submitting Jobs

# Job Submission: You need to know…

- Number of CPUs/type of parallel environment
- Time
- Memory use
- Job specific information (module required, job specific commands)

# Number of CPUs: Parallelization

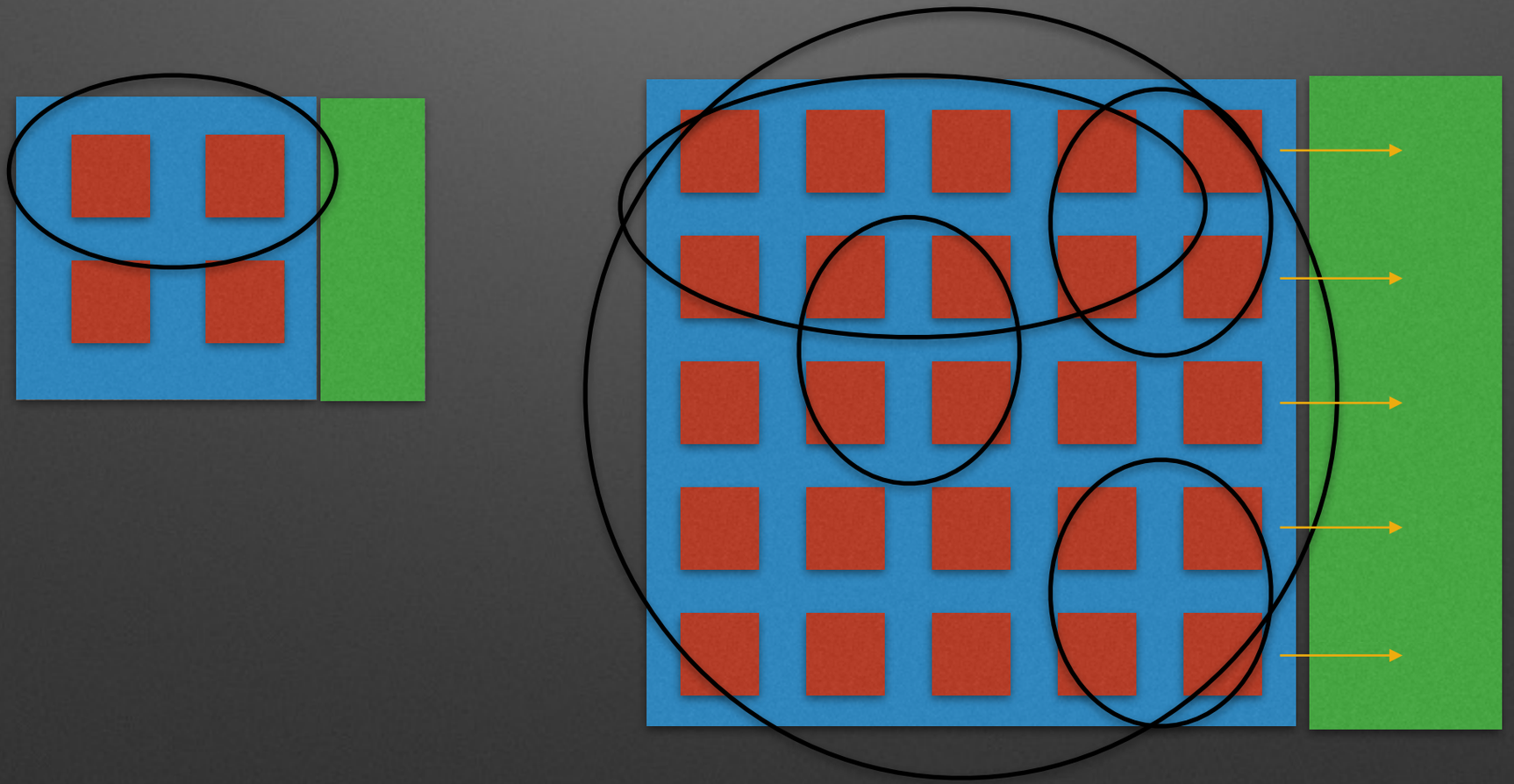
- Naive parallelization (embarrassingly parallel)- one large job easily split into separate jobs.




- E.g. many sequential blast searches at once, splitting bootstrap reps.

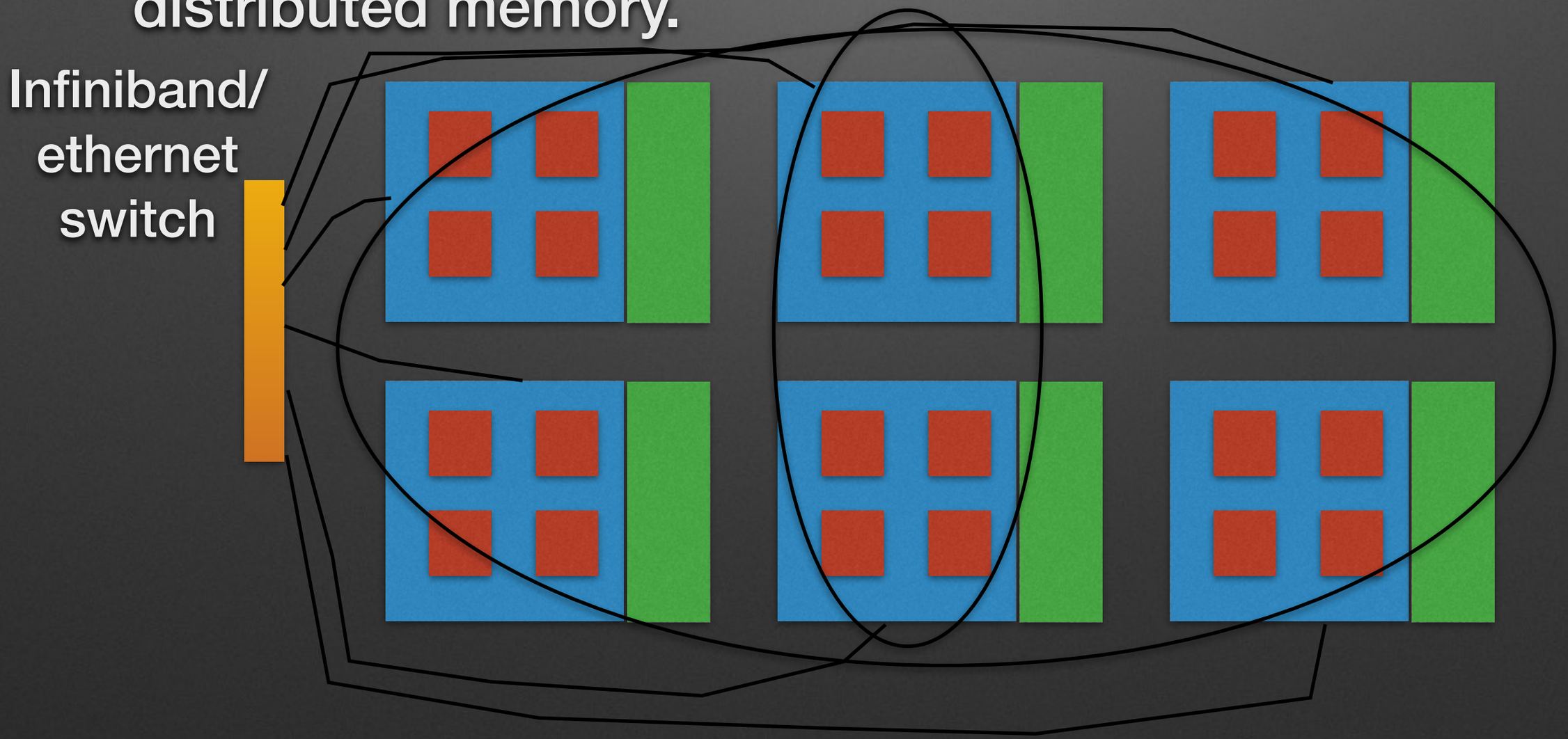
# Number of CPUs: Parallelization

- Multi-threading (pthreads, OpenMP)-multiple CPUs on a single node. This method uses shared memory.



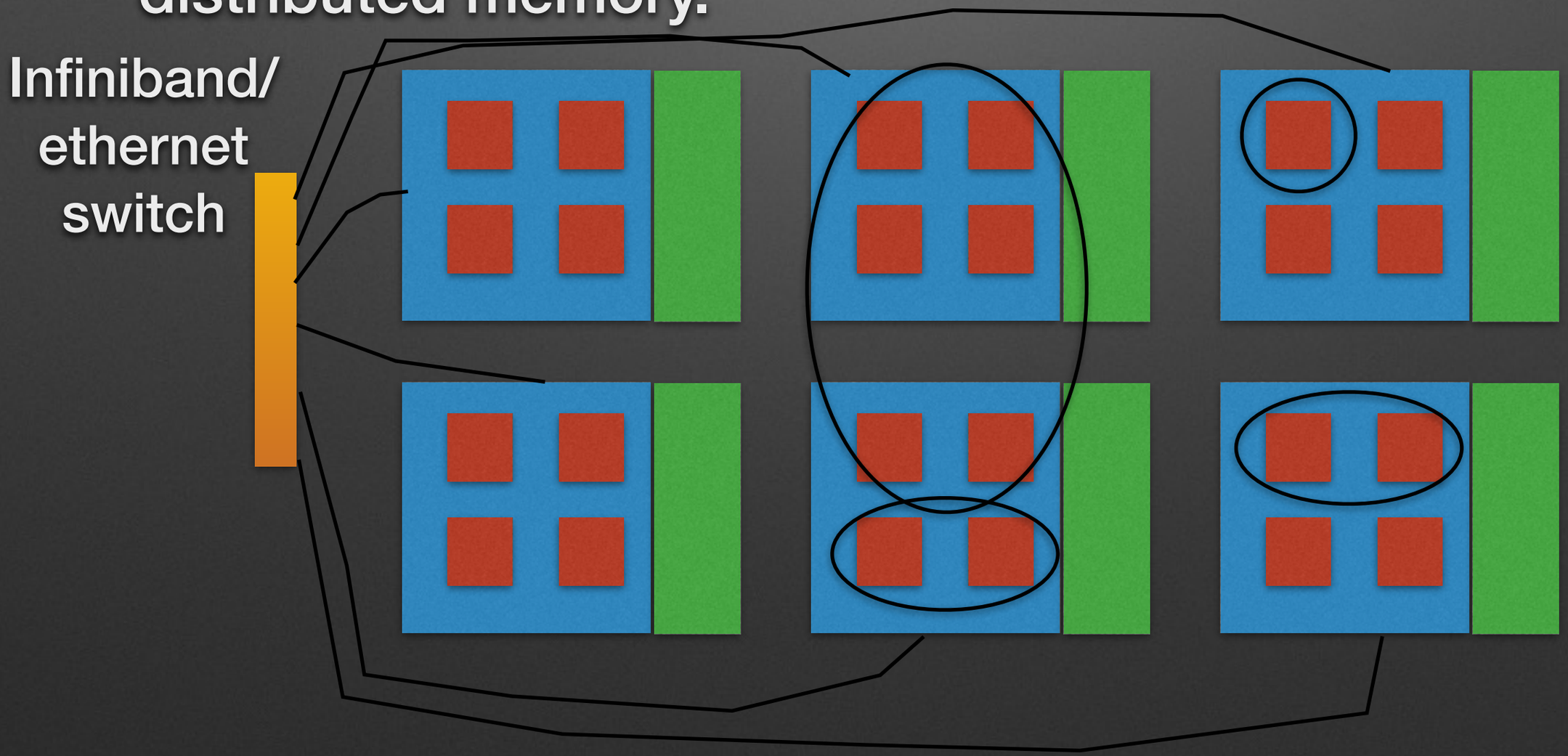
# Number of CPUs: Parallelization

- MPI (Message Passing Interface)-multiple CPUs across multiple nodes. Messages are passed between nodes via the interconnect. This method uses distributed memory.



# Number of CPUs: Parallelization

- MPI (Message Passing Interface)-multiple CPUs across multiple nodes. Messages are passed between nodes via the interconnect. This method uses distributed memory.



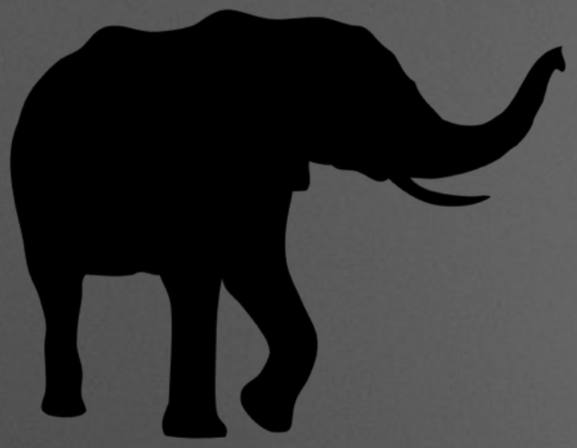
# Number of CPUs: Parallelization

- How do I know which one to use?

- *This is determined by the type of program you wish to run. Check the program documentation or check with us.*

# Time

- The cluster needs to be told estimates of run time
- Short jobs: 7 hours
- Medium: 3 days
- Long: 30 days
- Unlimited (there are very few of these…anything submitted to this queue should have some checkpointing capability)



# Memory use

- *Sylvain: “Memory is both an expensive and scarce resource”*

- Important to give a good estimate!

- How to estimate?

  - Check with others, program documentation

  - Monitor test runs, try smaller run and check how memory use scales up

# `qsub` flags

- How you tell the scheduler about your job
- `-cwd` runs job from current working directory
- `-j y` joins the stderr to the stdout
- `-o` sampleOut.log if you have chosen to join stderr and stdout, your job will write these to the sampleOut.log file
- `-N` The name of your job. Use something informative!
- `-m abe, -M example@si.edu` email with job status

# `qsub` flags

- Parallel enviromnent (`-pe`)
  - MPI (CPUs spread across nodes) `-pe orte 10` or `-pe mpich 10`
  - Threads (multiple CPUs, one node) `-pe mthread 16`
- Run Time `-q <q-name>` or `-l s_cpu <time>`

# qsub flags

- Memory use specification:
  - `-l memres=4G,h_data=4G,h_vmem=4G`
    - `memres`: “Memory Reserved” This is for the cluster to track memory use and prevent other from grabbing memory you will need. Corollary, it prevents other from using that memory. *This is a shared resource*.
    - `h_data,h_vmem`: Tells the system to terminate your job if it uses too much RAM
    - For large memory jobs, you must indicate that you wish to use the high memory queue by adding `-l himem` this can be added to the other variable using the `-l` flag.

🌻 ⚠️ MEMORY IS PER CPU NOT PER JOB ⚠️

- 16 threads with 4GB RAM each means 64GB total

# Modules

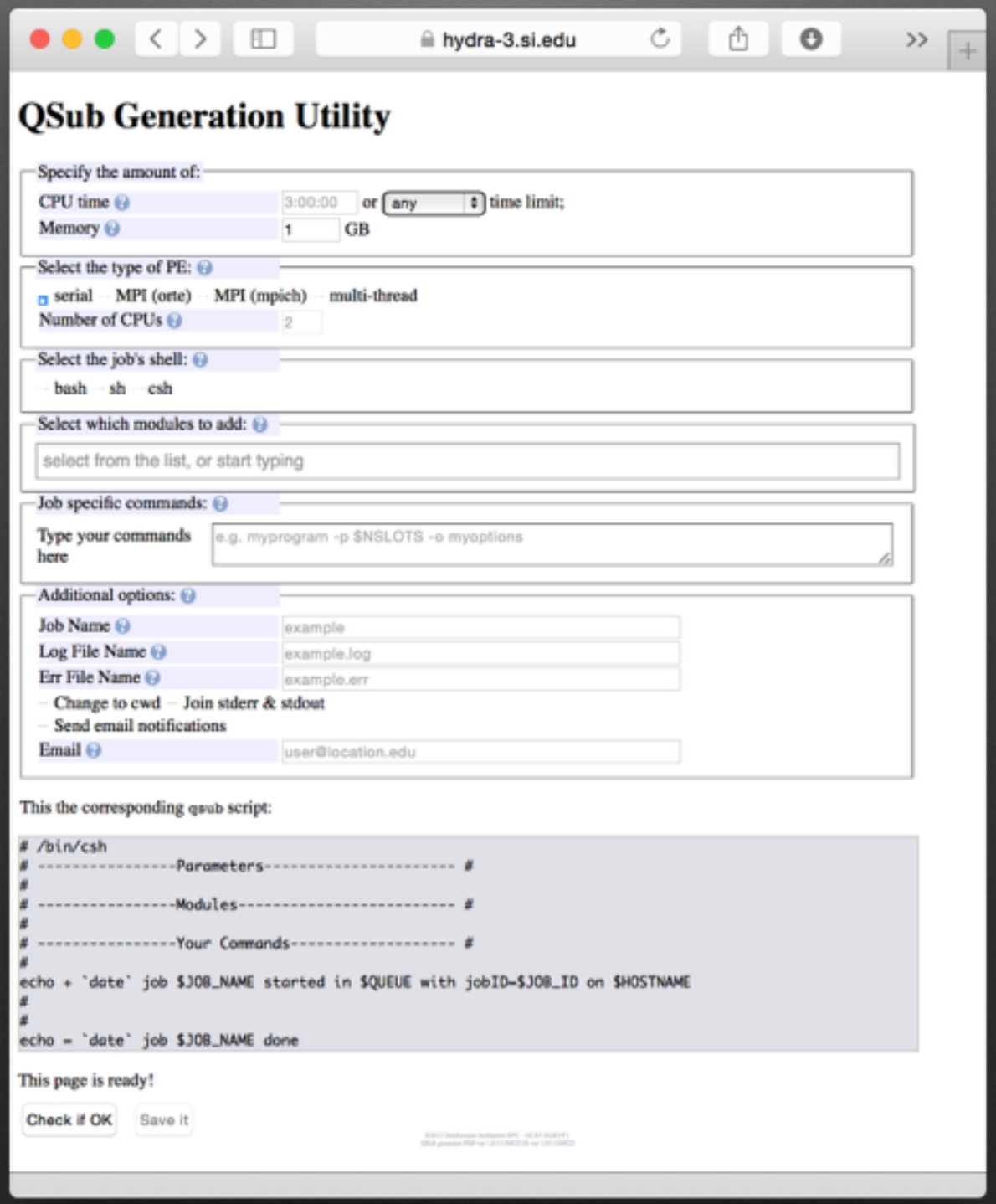
- Modules are a way to set your system paths and environmental variables for use of a particular program or pipeline.

- You can view available modules with the command:
```
$ module avail
```

- To find out more about a module you can use:
```
$ module whatis <module_name>
$ module help <module_name>
$ module show <module_name>
```

- To load a module:
```
$ module load <module_name>
```

# Putting it all together



# Best Practices

# Best use practices

- Ramp up your use slowly to monitor the resources used by your jobs (CPU, memory, disk space)
- Run all analyses through `qsub`, i.e. NO analyses other than simple scripting/moving around should be done on the login nodes.
- Upload your files directly to the /pool drives, not your home directory.
- Check available disk space before you upload large files.
  - do this with:
    `$ df -h /pool/genomics` (`df`=disk free)
- Cleanup disk use following analysis
  - Remove files after transferring them back
  - Avoid leaving lots of small files (use tar!!)
- Understand the usage requirements of the programs that you are running and use reasonable `-l` flags for memory and compute time.

# Transferring files

- You can use `scp`, `sftp` or `rsync`
- Important!
- Do NOT store large files on your home directory.
- When given a user account, you will also be given a folder to place your files.
- For this workshop, we are using: `/pool/genomics/stri_workshop/<username>`

# Tips

# Helpful commands

- Display queue names: `$ qconf -sql`
- Display specifc queue property: `$ qconf -sq <q-name>`
- Display parallel environments: `$ qconf -spl`
- Display queue limits: `$ qconf -srqs`
- Show info for all of your running jobs:
  `$ qstat -u $USER -s r`
- Display info for a specific job: `$ qstat -j <job_number>`
- Check queue usage: `$ qstat -g c`
- Check job stats after complete: `$ qacct -j <job_number>`

# Helpful commands

- Delete a job: `$ qdel <job_number>`
- Delete all of your jobs: `$ qdel -u $USER`
- Start one job *only* after another has completed (–hold_jid <job>):

```
$ qsub -N FirstOne pre-process.job

Your job 12345678 ("FirstOne") has been submitted

$ qsub -hold_jid 12345678 -N SecondStage post-process.job

Your job 12345679 ("SecondStage") has been submitted
```

# Questions?

# Hands on

https://github.com/SmithsonianWorkshops/Hydra-workshop

# “Hello world”

- Visit https://hydra-3.si.edu/tools/QSubGen/

hydra-3.si.edu

# QSub Generation Utility

Specify the amount of:

CPU time 3:00:00 or any time limit;

Memory 1 GB

Select the type of PE:

serial MPI (orte) MPI (mpich) multi-thread

Number of CPUs 2

Select the job's shell:

bash sh csh

Select which modules to add:

select from the list, or start typing

Job specific commands:

Type your commands here

e.g. myprogram -p $NSLOTS -o myoptions

Additional options:

Job Name example

Log File Name example.log

Err File Name example.err

Change to cwd Join stderr & stdout

Send email notifications

Email user@location.edu

This the corresponding `qsub` script:

```
# /bin/csh
# ----------------Parameters---------------------- #
#
# ----------------Modules------------------------- #
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
#
echo = `date` job $JOB_NAME done
```

This page is ready!

Check if OK Save it

- Save file. Rename it to `<name>.job`
- upload it to hydra :

```
$ scp <name>.qsub <username>@hydra-
login01.si.edu:/pool/genomics/
stri_workshop/<username>
```

- Alternatively, you can copy and paste it into a new file using your favorite command line text editor.

- Change to the directory that houses your qsub file:
  ```
  $ cd /pool/genomics/stri_workshop/<username>
  ```

- Now submit your job using:
  ```
  $ qsub <name>.job
  ```

- You can check your job status at any time by typing:
  ```
  $ qstat -j <job_number>
  ```

- If the job has already completed, you can check how things went by typing (currently only available on head node):
  ```
  $ qacct -j <job_number>
  ```

- Check to ensure that your outfile has been written:
  ```
  $ ls
  ```

- If it is there, let’s see what it says:
  ```
  $ less outfile
  ```

- **There is a new alignment in** `/pool/genomics/stri_workshop/data` **called** `primates.dna.phy`
- Let’s set up a RAxML run with it.

hydra-3.si.edu

# QSub Generation Utility

Specify the amount of:

CPU time 3:00:00 or any time limit;

Memory 1 GB

Select the type of PE:

serial MPI (orte) MPI (mpich) multi-thread

Number of CPUs 2

Select the job's shell:

bash sh csh

Select which modules to add:

select from the list, or start typing

Job specific commands:

Type your commands here e.g. myprogram -p $NSLOTS -o myoptions

Additional options:

Job Name example

Log File Name example.log

Err File Name example.err

Change to cwd Join stderr & stdout

Send email notifications

Email user@location.edu

This the corresponding `qsub` script:

```
# /bin/csh
# ----------------Parameters---------------------- #
#
# ----------------Modules------------------------- #
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
#
echo - `date` job $JOB_NAME done
```

This page is ready!

Check if OK Save it

- This time, let's use multiple threads
- Select 'multi-thread' and 4 CPUs
- RAxML module file is `bioinformatics/raxml/8.2`
- Now give your RAxML command in the 'Job specific commands' field:

```
raxmlHPC-SSE3-PTHREADS -T $NSLOTS -n raxmltest
-s primates.dna.phy -m GTRCAT -f a -p 234532 -
x 345234 -N 100
```

- Now I want you each to submit this job (you should now how after doing the 'hello world' example.
- Once your job is complete, download your output tree file.

# Questions?