



# HPC at the Smithsonian

Introduction to Hydra

# Overview



- Why HPC?



- SI Computing Cluster



- Support



- Cluster Architecture



- Submitting Jobs



- Best Practices



- Tips

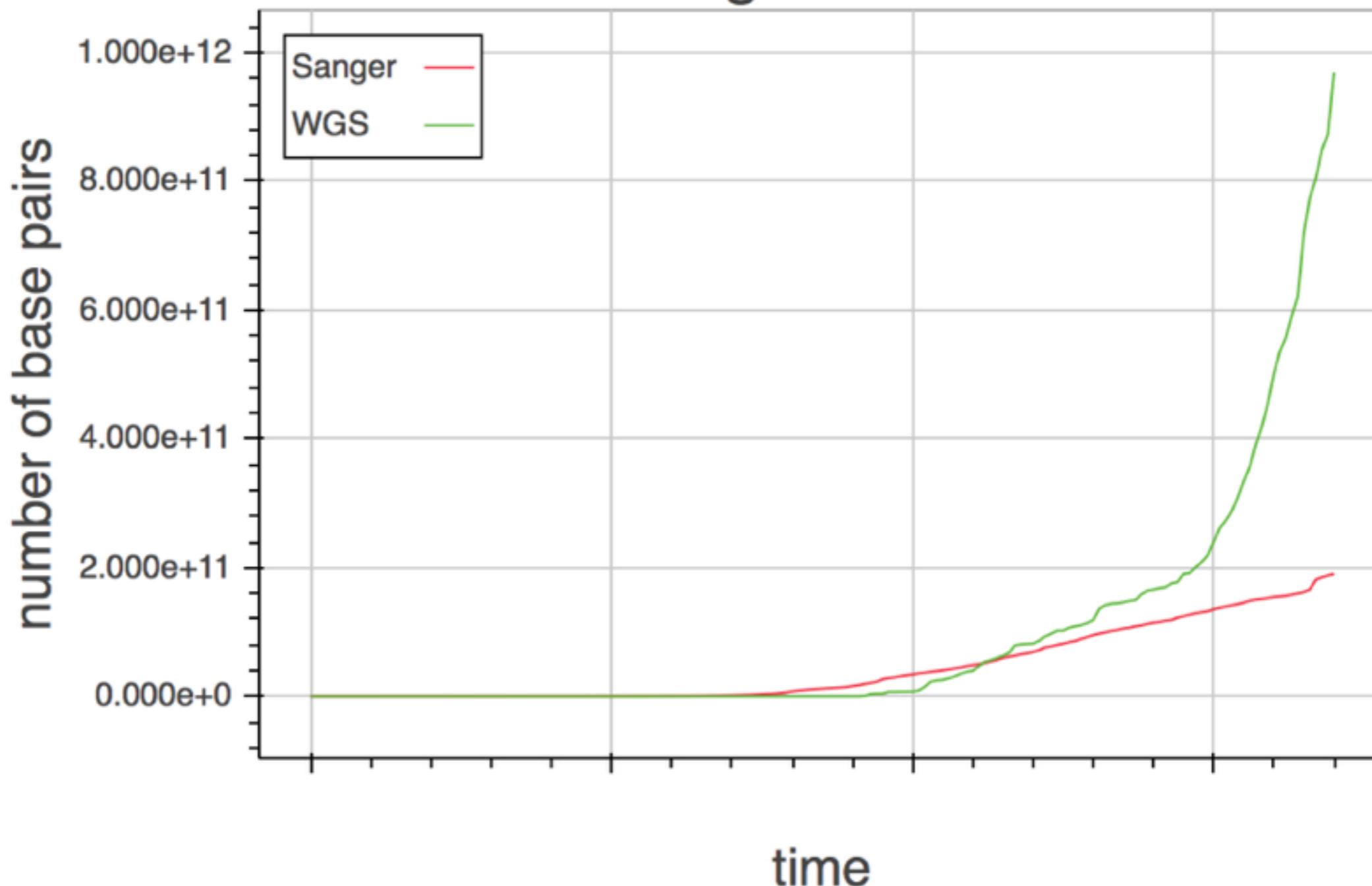
# Why High Performance Computing (HPC)?



# Why HPC?



increase in genetic data



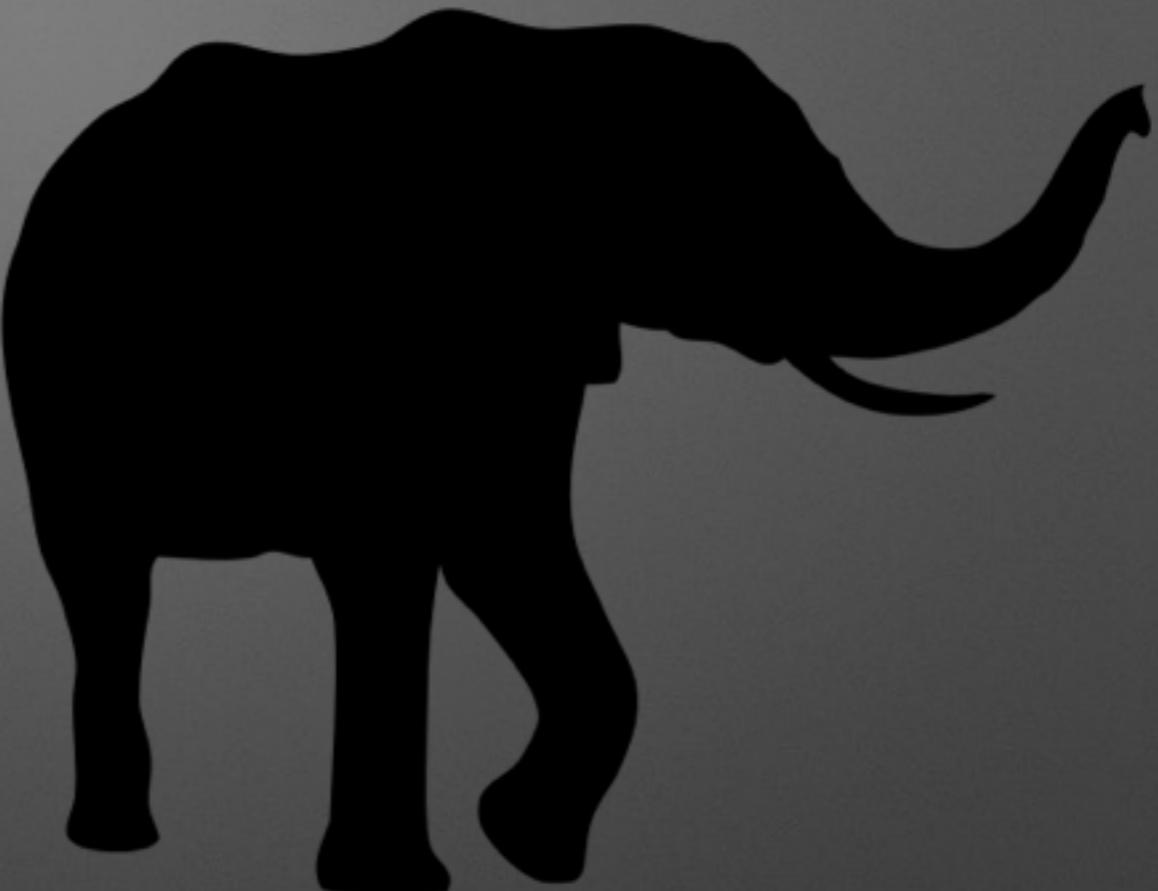
# Why HPC?



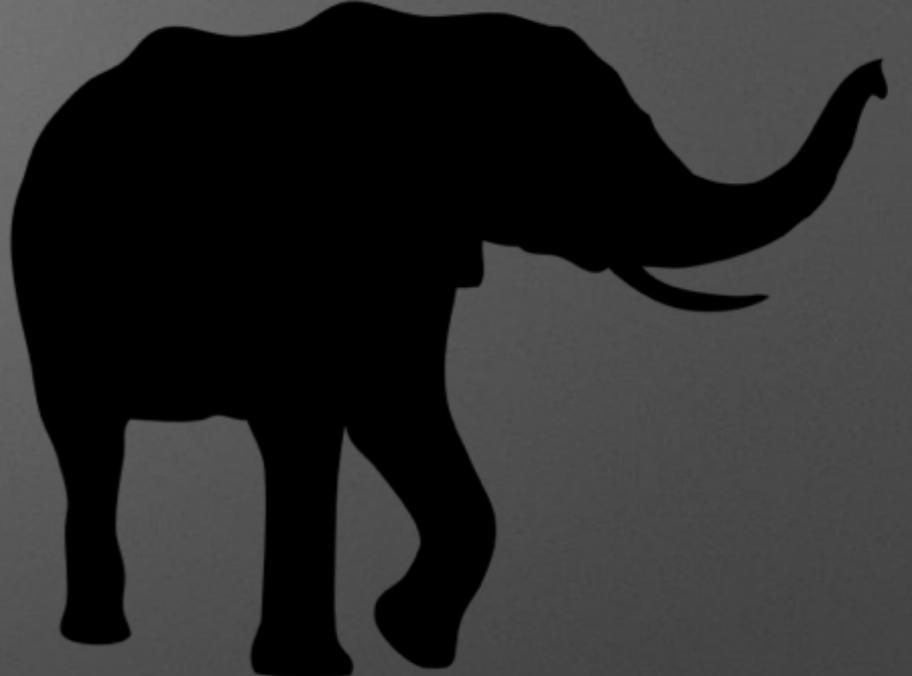
Size

&

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

# 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

The screenshot shows a web browser window with the URL [confluence.si.edu](https://confluence.si.edu) in the address bar. The page title is "High Performance Computing". The left sidebar contains a "PAGE TREE" with links to "Smithsonian Institution High Performance Computing Policies", "What's New", "Hydra Quick Start Guide", "Hydra Reference Pages", "GPUs on Hydra and the GPU Test Queue", "FAQ", and "Cluster Upgrade and Merger Status & M...". The main content area includes a brief introduction, a list of cluster details, a "What's New" section with a list of August 2016 updates, and a "Quick Links" section at the bottom.

confluence.si.edu

Smithsonian Spaces People Create ...

Pages Edit Save for later Watching Share ...

## High Performance Computing

Created by Davis, Daniel, last modified by Korzenik, Sylvain on Sep 07, 2016

Welcome to the Smithsonian Institution High Performance Computing Wiki.

This Wiki holds information for the use of HPC resources at the Smithsonian.

Central to the SI HPC resources is the Smithsonian Institution High Performance Cluster (SI/HPC), named Hydra.

- High performance computing is administered by the Office of Research Information Services ([ORIS](#)).
- The OCIO data center in Herndon, VA houses the high performance computing cluster, Hydra.
- The documentation is organized as follow:
  1. [A quick start guide](#);
  2. [Reference pages](#) (technical information about the cluster);
  3. [Cluster Upgrade documentation](#) (past upgrades);
  4. [Cookbooks pages](#) (didactic information on how to run specific tools); (NYA: not yet available)
  5. [FAQs](#), or frequently asked questions

### What's New

- August 2016
  1. We are about to implement a clear and simple [usage policy for Hydra](#) and associated resources.
  2. We have finalized the purchase of new hardware with FY16 funds.  
It's installation will require an OS release upgrade and some partial downtime.
  3. We will implement a [scrubber on the public disks](#).
  4. We have added a [hybrid parallel environment \(PE\)](#) to some queues.
  5. We have added a [~/.forward](#) file for every user of Hydra. We used your canonical email address.

We will update this page, the login nodes MOTD and send emails when these are implemented.
- Past news can be found at the [What's New page](#).

Space tools Quick Links

- New wiki: <https://confluence.si.edu/display/HPC/High+Performance+Computing>

# Hydra support:



**Paul Frandsen, ORIS**



**Sylvain Korzennik, SAO**



**DJ Ding- OCIO**

# Application support:



**Rebecca Dikow, SIBG**



**Vanessa González, GGI**



**Matt Kweskin, LAB**

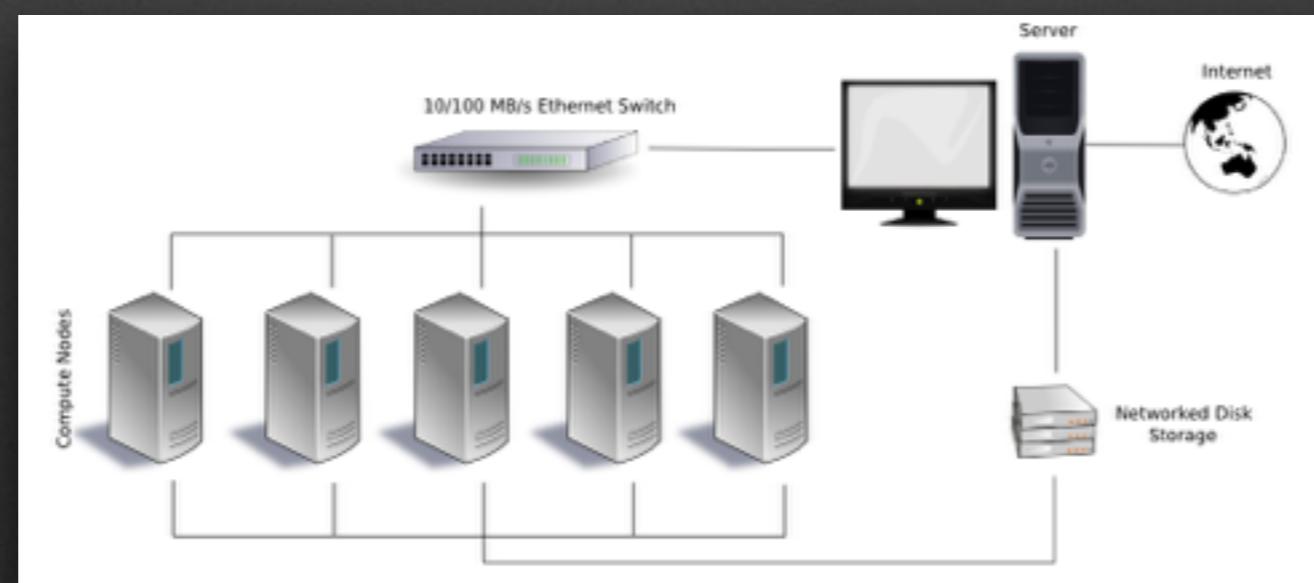
# Cluster Architecture





# 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

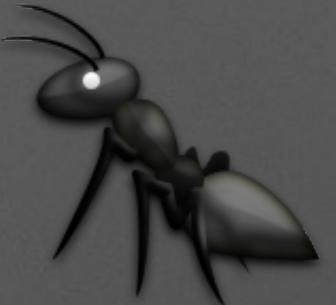


- Software that schedules the jobs and submits them to the compute nodes.
- Hydra uses the “(Sun) Grid Engine” queuing system.
- To submit jobs, you use the command "qsub".
- All jobs run in batch mode, not interactive mode.

# Submitting Jobs

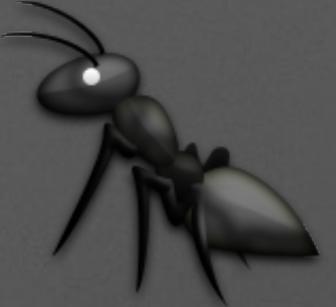


# Job Submission: You need to know...

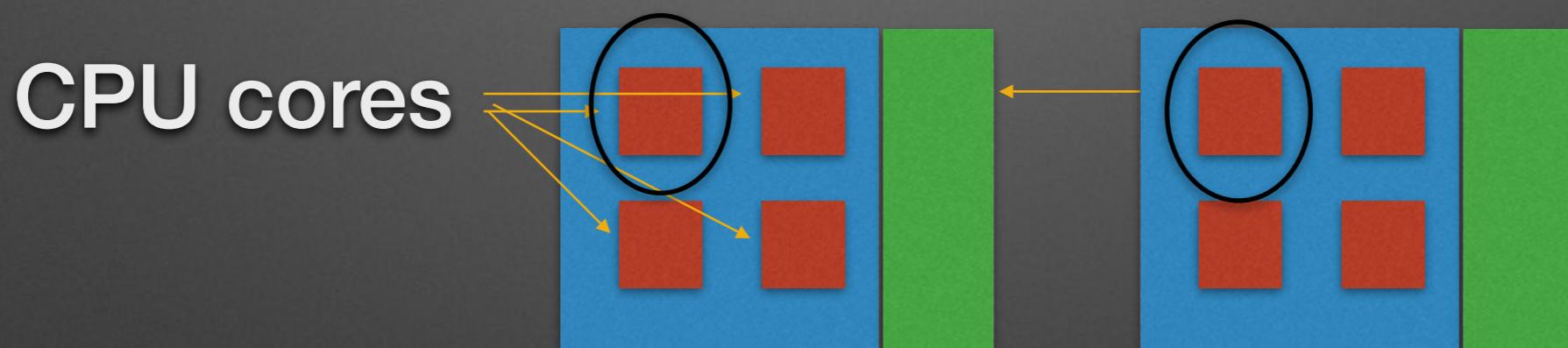


- Number of CPUs/type of parallel environment
- Time
- Memory use
- Job specific information (module required, job specific commands)

# Number of CPUs: Parallelization

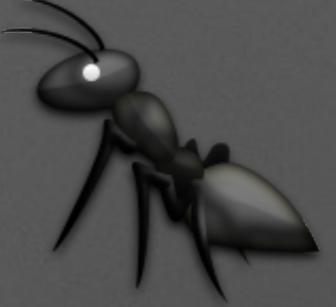


- Naive parallelization (embarrassingly parallel)- one large job easily split into separate jobs.

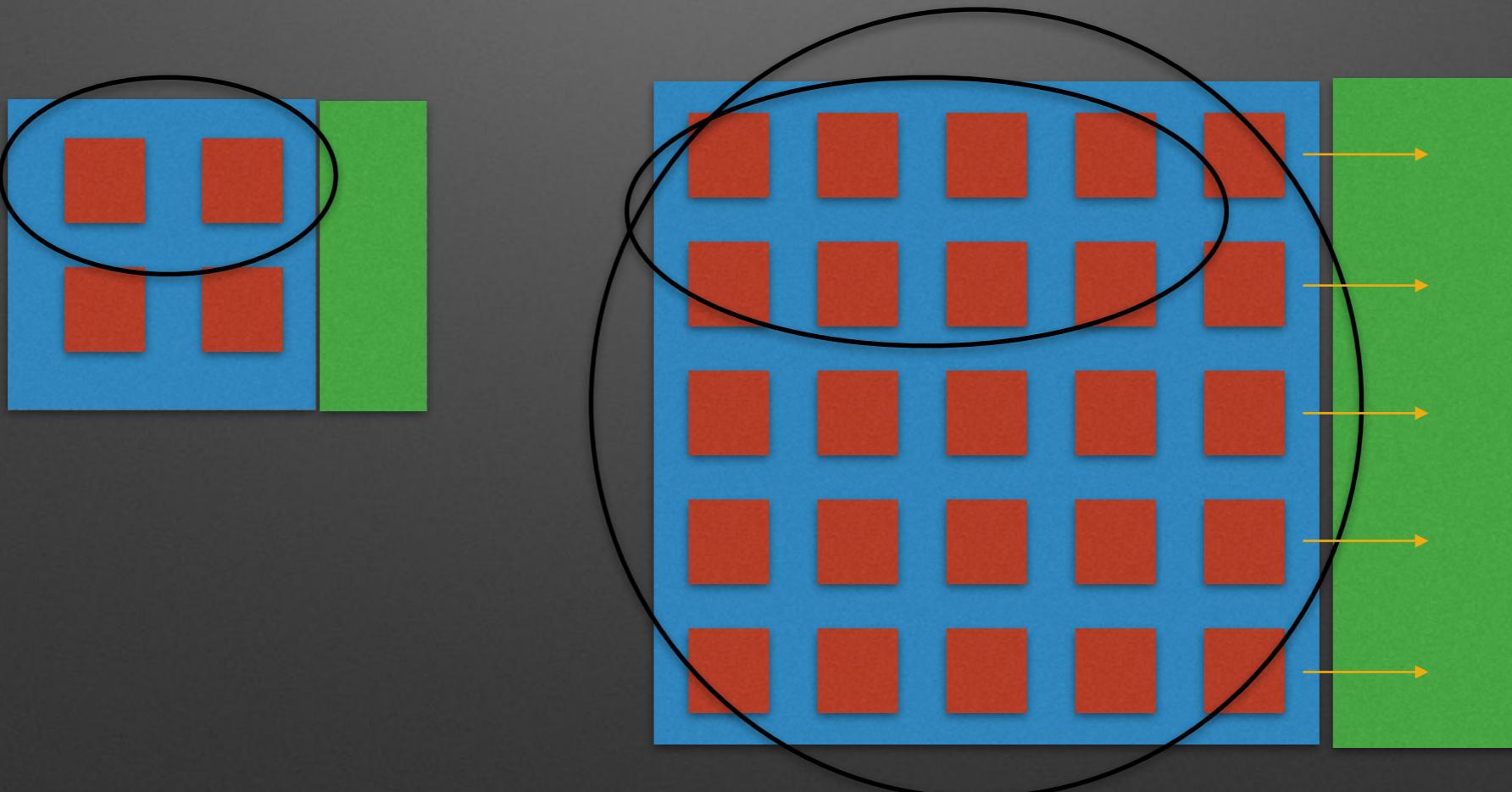


- E.g. many sequential blast searches at once, splitting bootstrap reps.

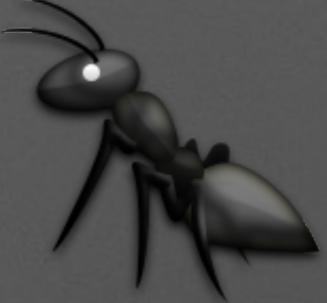
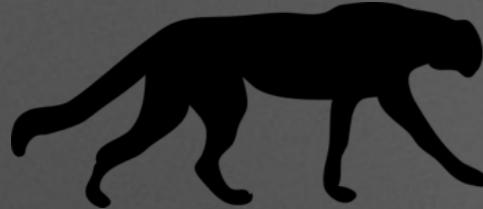
# Number of CPUs: Parallelization



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

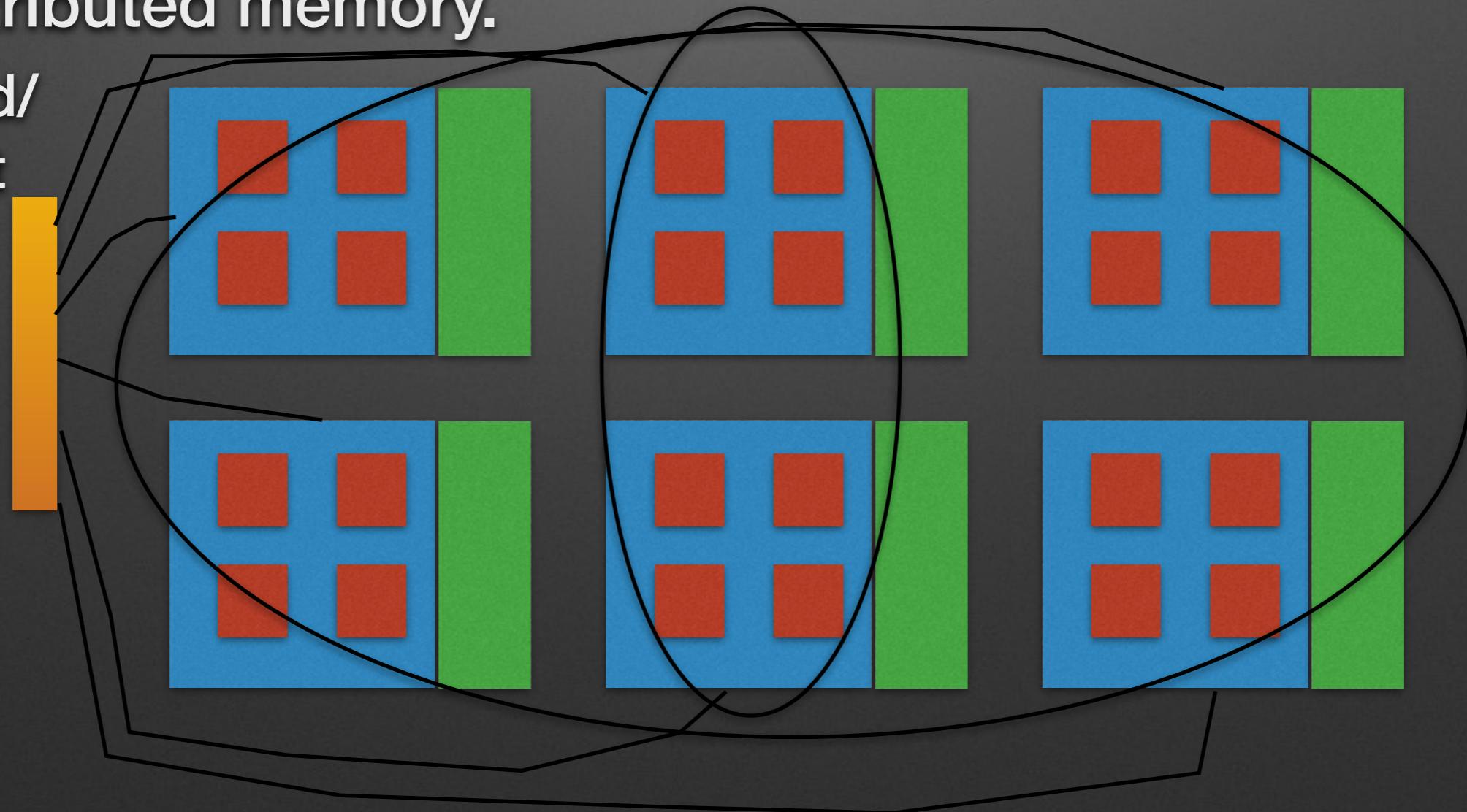


# Number of CPUs: Parallelization

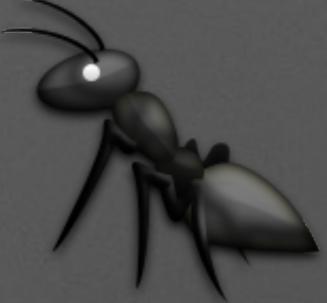


- MPI (Message Passing Interface)-multiple CPUs across multiple nodes. Messages are passed between nodes via the interconnect. This method uses distributed memory.

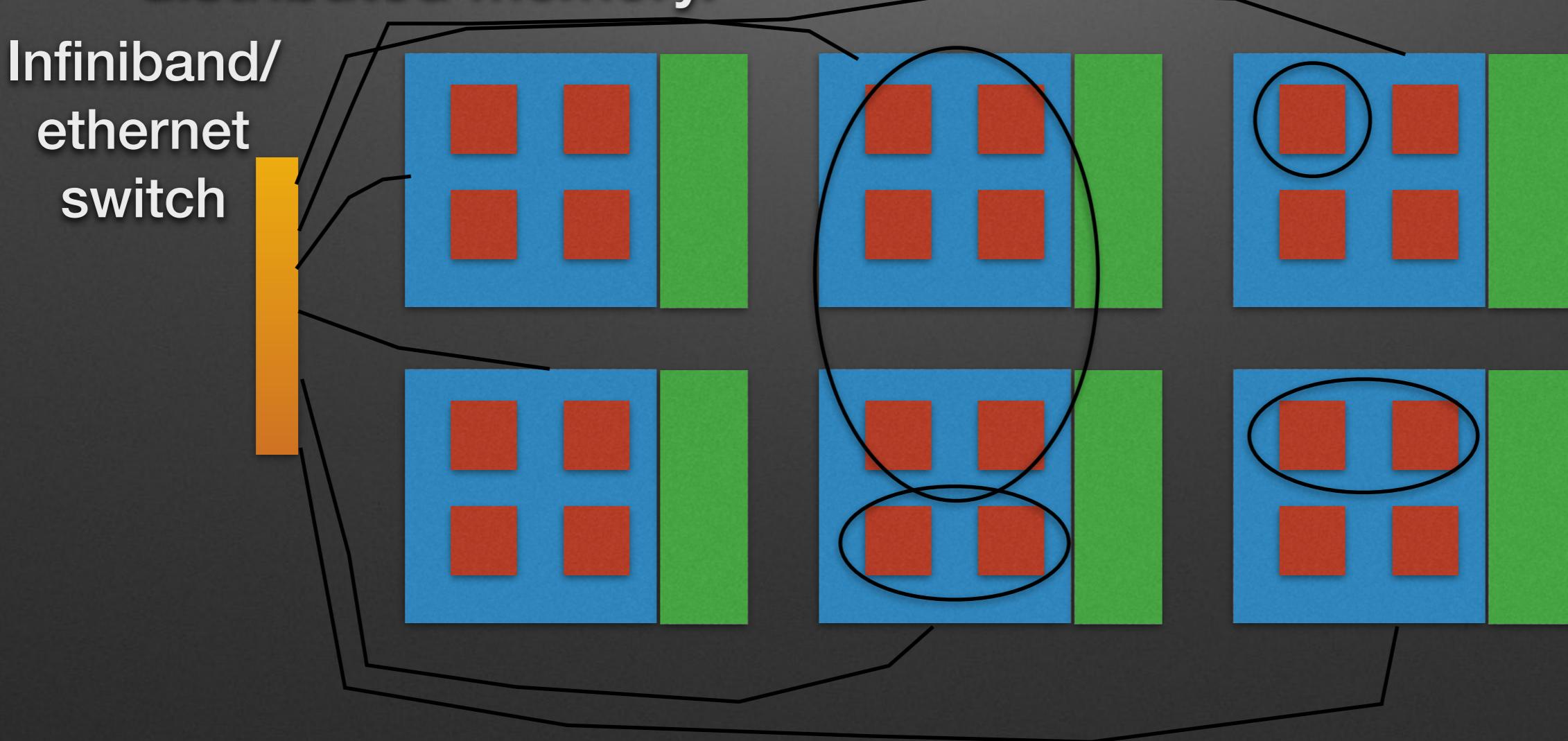
Infiniband/  
ethernet  
switch



# Number of CPUs: Parallelization

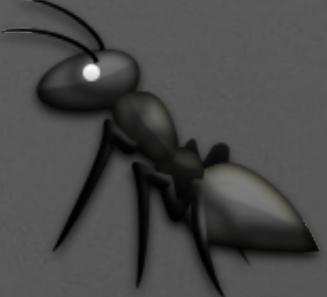


- MPI (Message Passing Interface)-multiple CPUs across multiple nodes. Messages are passed between nodes via the interconnect. This method uses distributed memory.



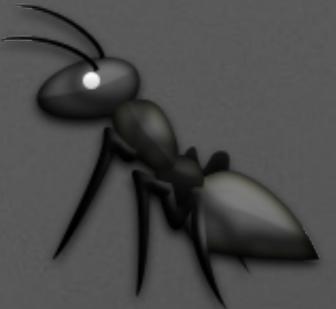


# Number of CPUs: Parallelization

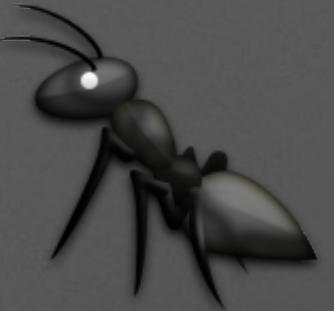
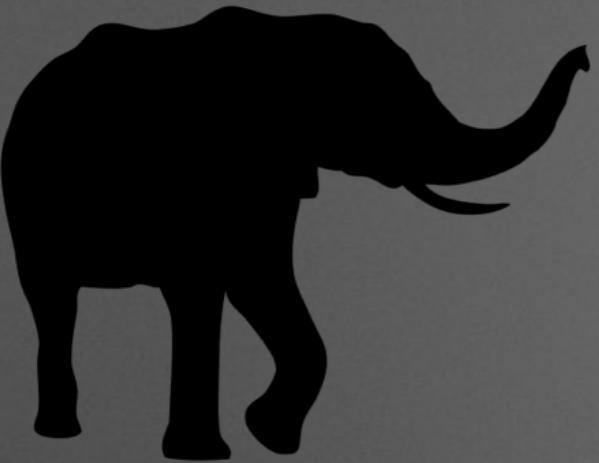


- How do I know which one to use?
- *This is determined by the type of program you wish to run. Check the program documentation or check with us.*

# Time



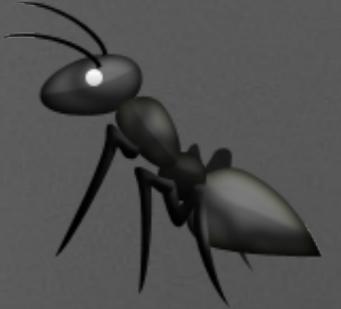
- The cluster needs to be told estimates of run time
- Short jobs: 7 hours
- Medium: 3 days
- Long: 30 days
- Unlimited (there are very few of these...anything submitted to this queue should have some checkpointing capability)



# 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

# 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



hydra-3.si.edu

## QSub Generation Utility

Specify the amount of:

CPU time  or  time limit;  
Memory  GB

Select the type of PE:

serial - MPI (orte) - MPI (mpich) - multi-thread  
Number of CPUs

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   
Log File Name   
Err File Name   
- Change to cwd - Join stderr & stdout  
- Send email notifications  
Email

This is 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!

©2017 Indiana Institute MPI - DCI01000000  
Last process PHP on 10/11/2017 at 10:11:00Z

# 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.

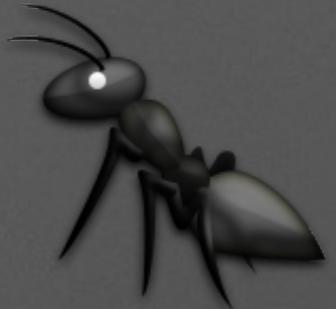
# Transferring files



- 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>`

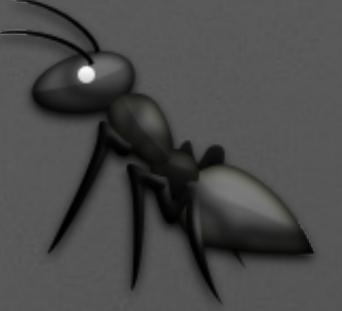
# Tips





# qsub flags

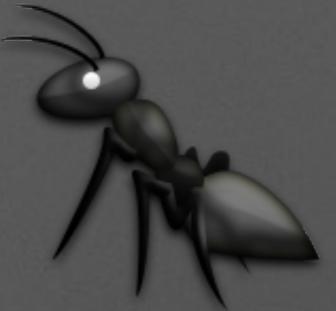
- How you tell the scheduler about your job
- -cwd runs job from current working directory
- -j y joins the stderr to the stdout
- -o sampleOut.log if you have chosen to join stderr and stdout, your job will write these to the sampleOut.log file
- -N The name of your job. Use something informative!
-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



# qsub flags

- Parallel environment (-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



# Helpful commands

- Display queue names: `$ qconf -sql`
- Display specific queue property: `$ qconf -sq <q-name>`
- Display parallel environments: `$ qconf -spl`
- Display queue limits: `$ qconf -srqsl`
- 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/  
Unix\\_Hydra\\_2hr](https://github.com/SmithsonianWorkshops/Unix_Hydra_2hr)

# “Hello world”

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

hydra-3.si.edu

## QSub Generation Utility

Specify the amount of:

CPU time  or  time limit;  
Memory  GB

Select the type of PE:  serial - MPI (orte) - MPI (mpich) - multi-thread  
Number of CPUs

Select the job's shell:  bash  sh  csh

Select which modules to add:

select from the list, or start typing

Job specific commands:   
e.g. myprogram -p \$NSLOTS -o myoptions  
here

Additional options:

Job Name   
Log File Name   
Err File Name   
- Change to cwd - Join stderr & stdout  
- Send email notifications  
Email

This is 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!

©2017 QSub Generation Utility (QGU) - QGU is open source software.  
QGU generation PHP ver 1.0.1 (150225) - ver 1.0.1 (150225)

- **Save file. Rename it to <name>.job**

- **upload it to hydra :**

```
$ scp <name>.qsub <username>@hydra-  
login01.si.edu:/pool/biology/<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/biology/<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/biology/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  or  time limit;  
Memory  GB

Select the type of PE:  serial - MPI (orte) - MPI (mpich) - multi-thread  
Number of CPUs

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   
Log File Name   
Err File Name   
- Change to cwd - Join stderr & stdout  
- Send email notifications  
Email

This is 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!

©2012 GridEngine® Application MPC - QSUB (QSUB.PHP)  
QSub generator PHP ver 1.01/150222.05 ver 1.01/150222

- 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?

