



HPC at the Smithsonian

Introduction to Hydra

Thank you!

Funds support from:

**Under Secretary for Science
Global Genome Initiative (GGI)**

Laboratories of Analytical Biology (LAB)

Office the Chief Information Officer (OCIO)

Smithsonian Institute for Biodiversity Genomics (SIBG)

Overview



- Why HPC?



- SI Computing Cluster



- Support



- Cluster Architecture



- Submitting Jobs



- Best Practices



- Tips

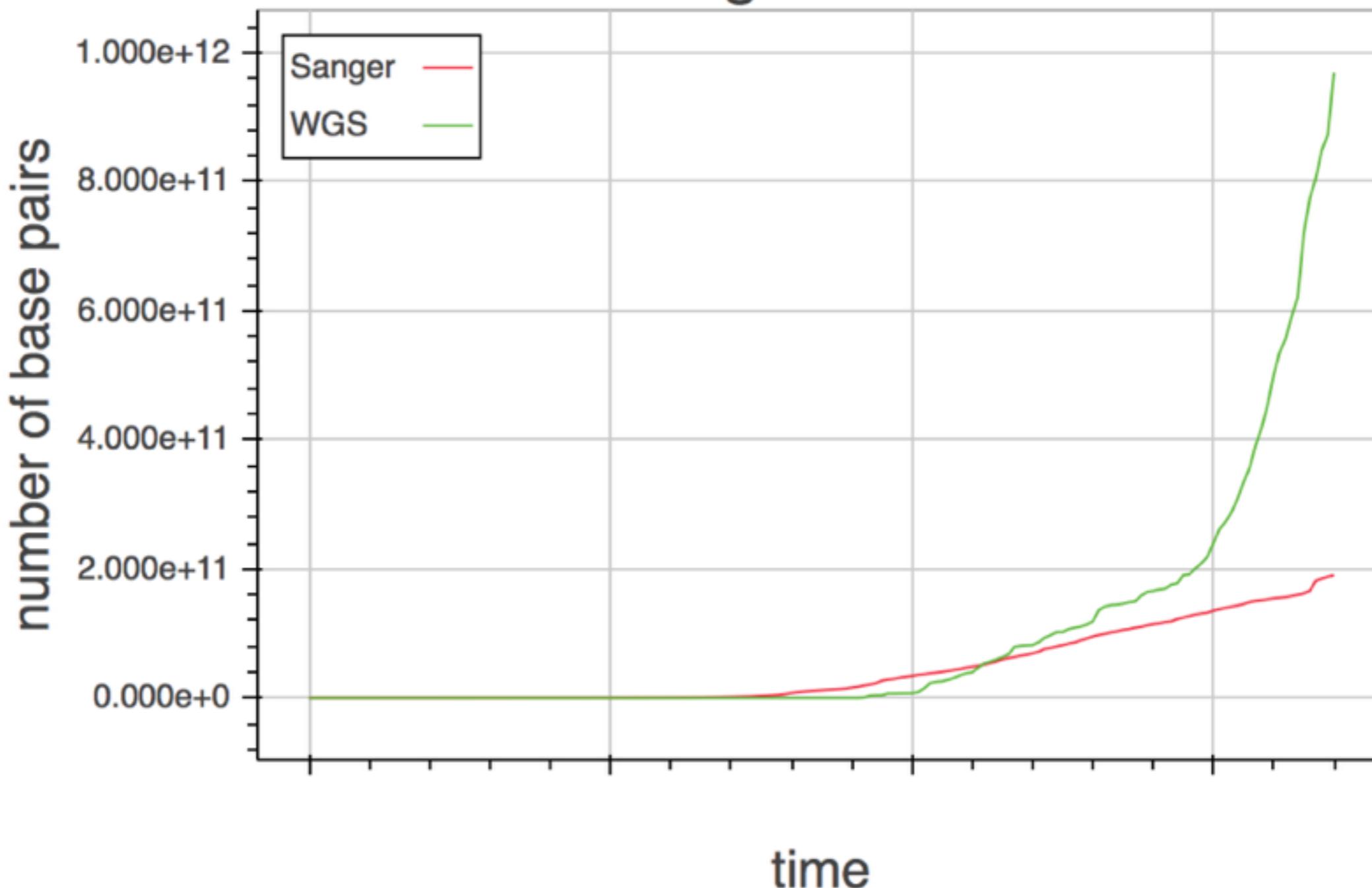
Why High Performance Computing (HPC)?



Why HPC?



increase in genetic data

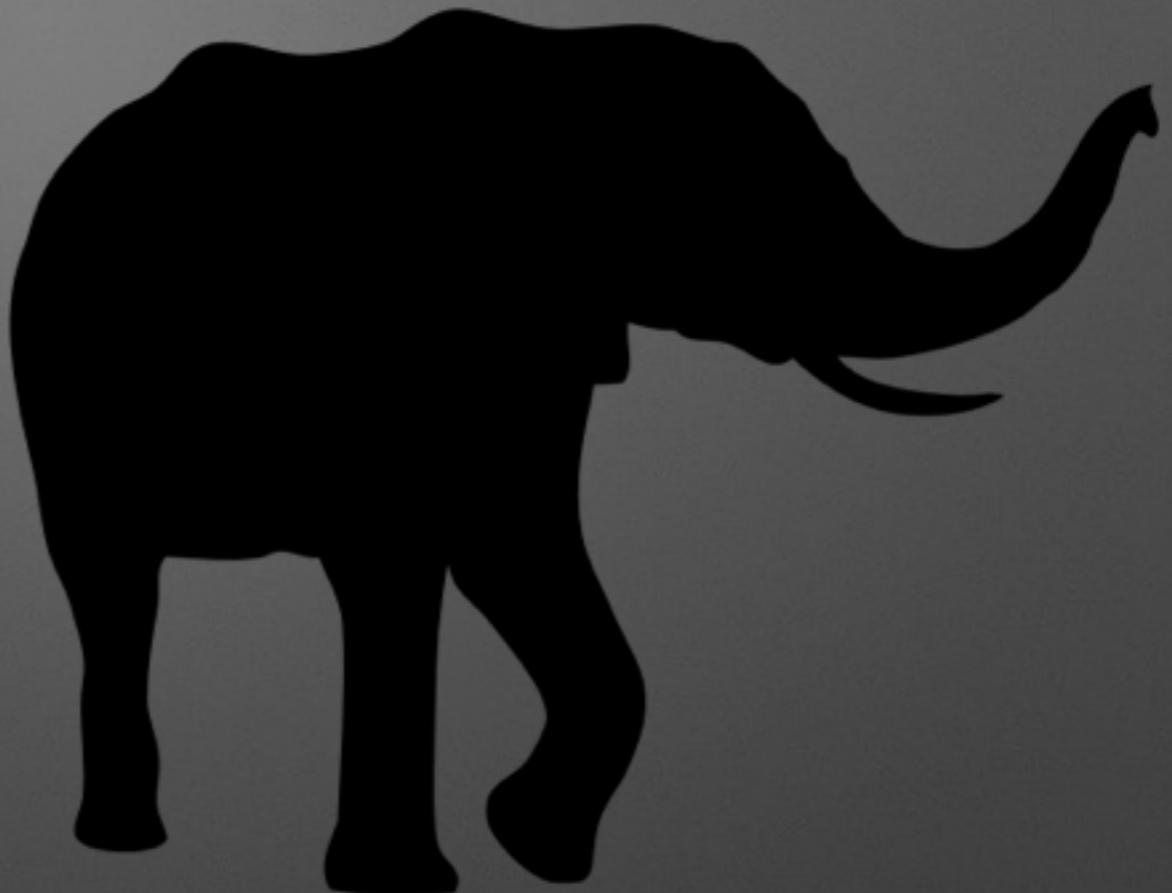


Why HPC?



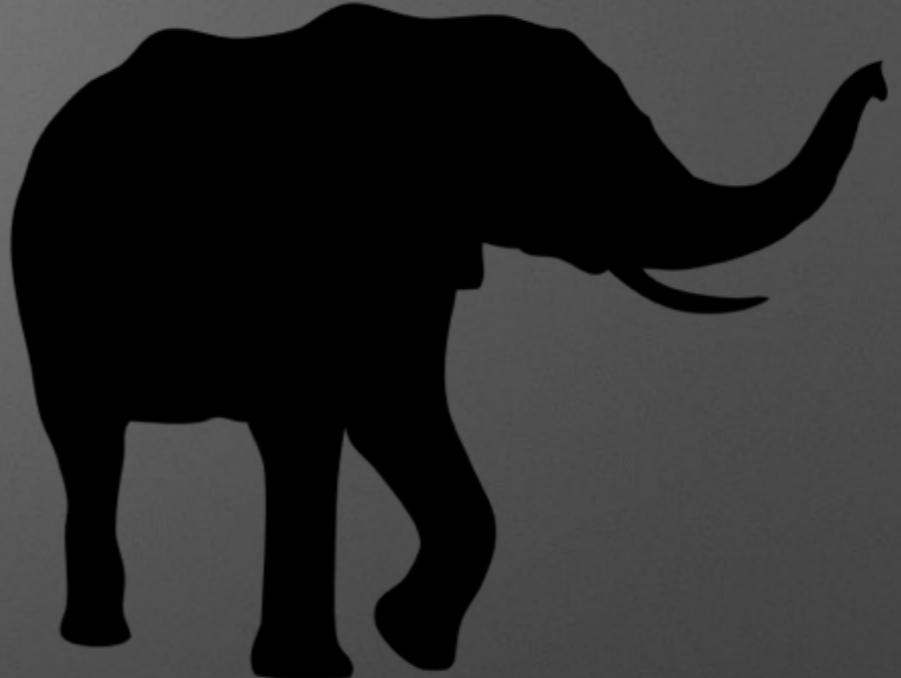
Size

&



Speed

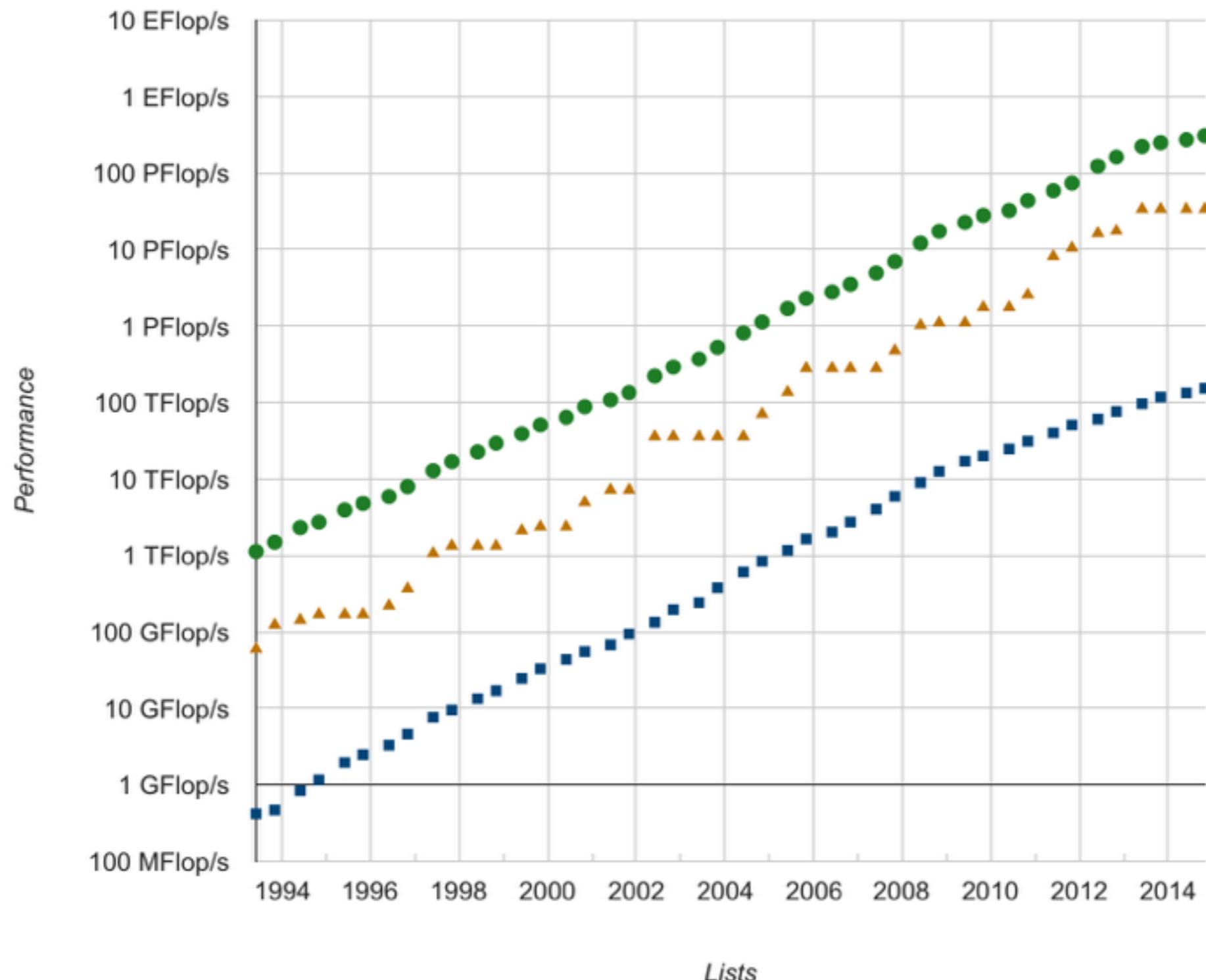
Why HPC?



- 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



Performance Development

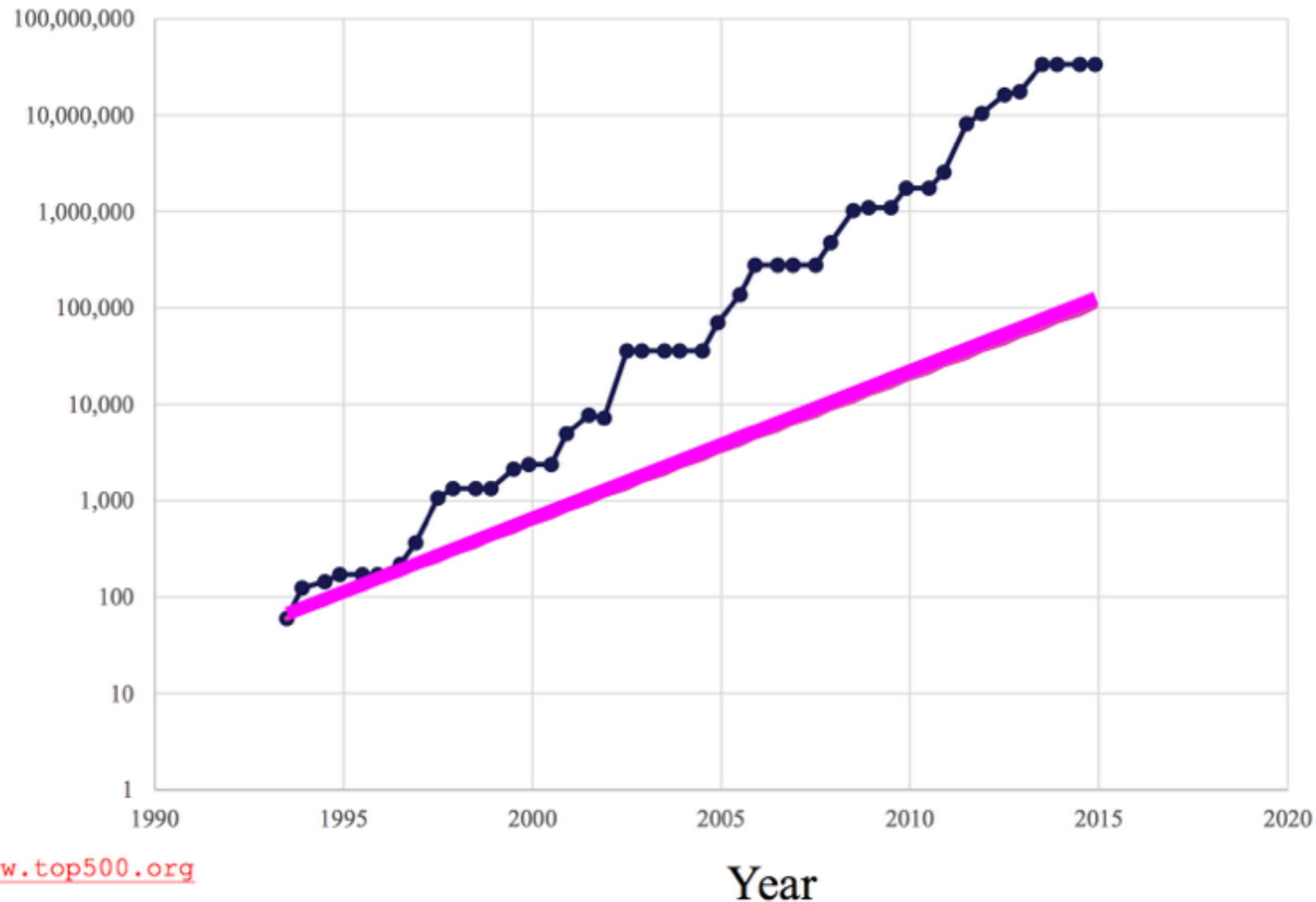


Lists

■ Sum ■ #1 ■ #500



Fastest Supercomputer in the World vs Moore



● GFLOPs
— Moore

GFLOPs:
billions of
calculations per
second

www.top500.org

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





Hydra

- Note: For a variety of hardware/software reasons, hydra currently consists of two separate clusters. They will be merged July 20-24. During that week, the cluster will be turned off to be reorganized and re-racked.



Support



Wiki



- Current wiki <https://www.cfa.harvard.edu/twiki5/view/HPC/WebHome>
- New wiki in development: <http://confluence.si.edu>

Screenshot of a Confluence page titled "Hydra Hardware". The page includes a sidebar with links to "Pages", "Blog", "SPACE SHORTCUTS", "Shared Links", and "How-To Articles". The main content area shows a table of server specifications for the Hydra cluster.

Server type	RAM per server	CPUs per server	CPU type	RAM per CPU	Number in cluster	Infiniband
Sun Fire X2200	16 GB	8	AMD Opteron	2 GB	175	no
HP DL165-G7	32 GB	16	AMD Opteron	2 GB	24	yes
Dell R415	32 GB	12	AMD Opteron	2.7 GB	24	yes
Dell R715	64 GB	24	AMD Opteron	2.7 GB	10	yes
Dell R815 (high CPU)	256 GB	64	AMD Opteron	4 GB	16	yes ^a

^aFour R815 servers in rack 0 are not configured with infiniband

Total number of CPUs: 3,336
Total RAM: 9,072 GB
Note: these numbers do not include the memory and CPUs on login nodes.

Lattice

Lattice is a separate (for now) cluster from Hydra that consists of both high CPU servers and high memory servers. The high memory servers are ideal for tasks such as *de novo* transcriptome or genome assembly, while the high CPU servers are well suited to analyses that can be run massively parallel such as reciprocal blast searches.

Server type	RAM per server	CPUs per server	CPU type	RAM per CPU	Number in cluster	Infiniband
Dell R815 (high CPU)	256 GB	64	AMD Opteron	4 GB	3	no
Dell R820 (high memory)	512 GB	24	Intel Xeon	20 GB	2	yes, software not configured

Total number CPUs: 240
Total RAM: 1,792 GB
Note: these numbers do not include the memory and CPUs on login nodes.

Like Be the first to like this No labels



Paul Frandsen, OCIO/ORIS



Rebecca Dikow, SIBG



Matt Kweskin, LAB



DJ Ding- OCIO/ORIS



Vanessa González, GGI



Sylvain Korzennik- SAO

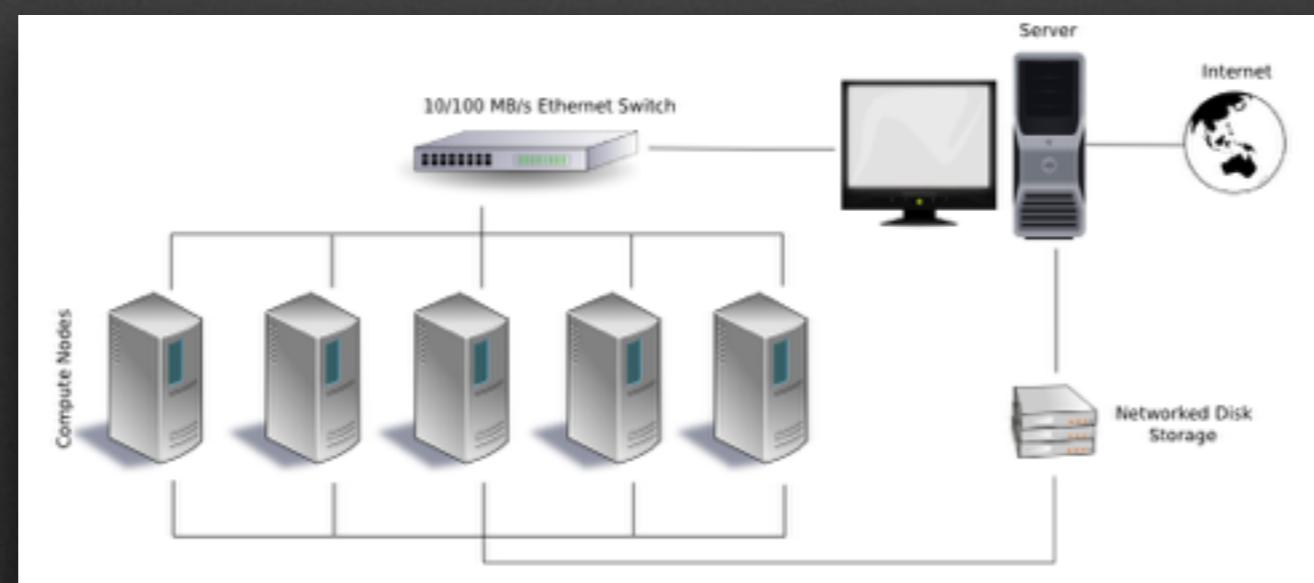
Cluster Architecture





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

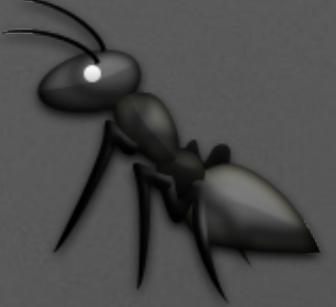


Job Submission: You need to know...

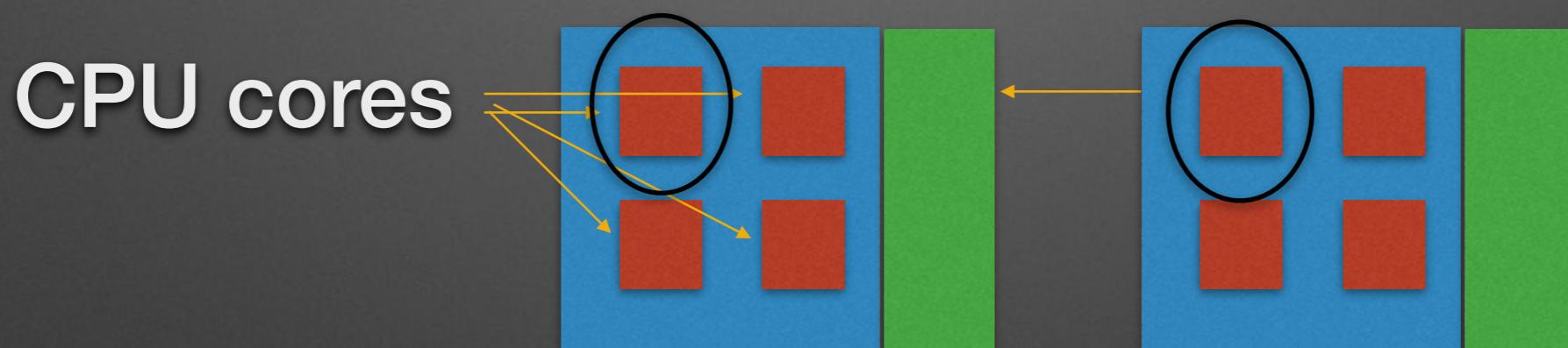


- Number of CPUs/type of parallel environment
- Time
- Memory use
- Job specific information (module required, job specific commands)

Number of CPUs: Parallelization

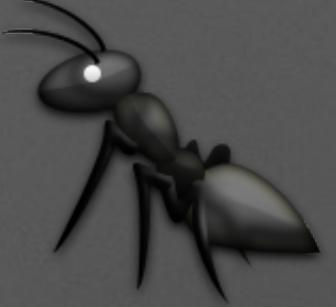


- Naive parallelization (embarrassingly parallel)- one large job easily split into separate jobs.

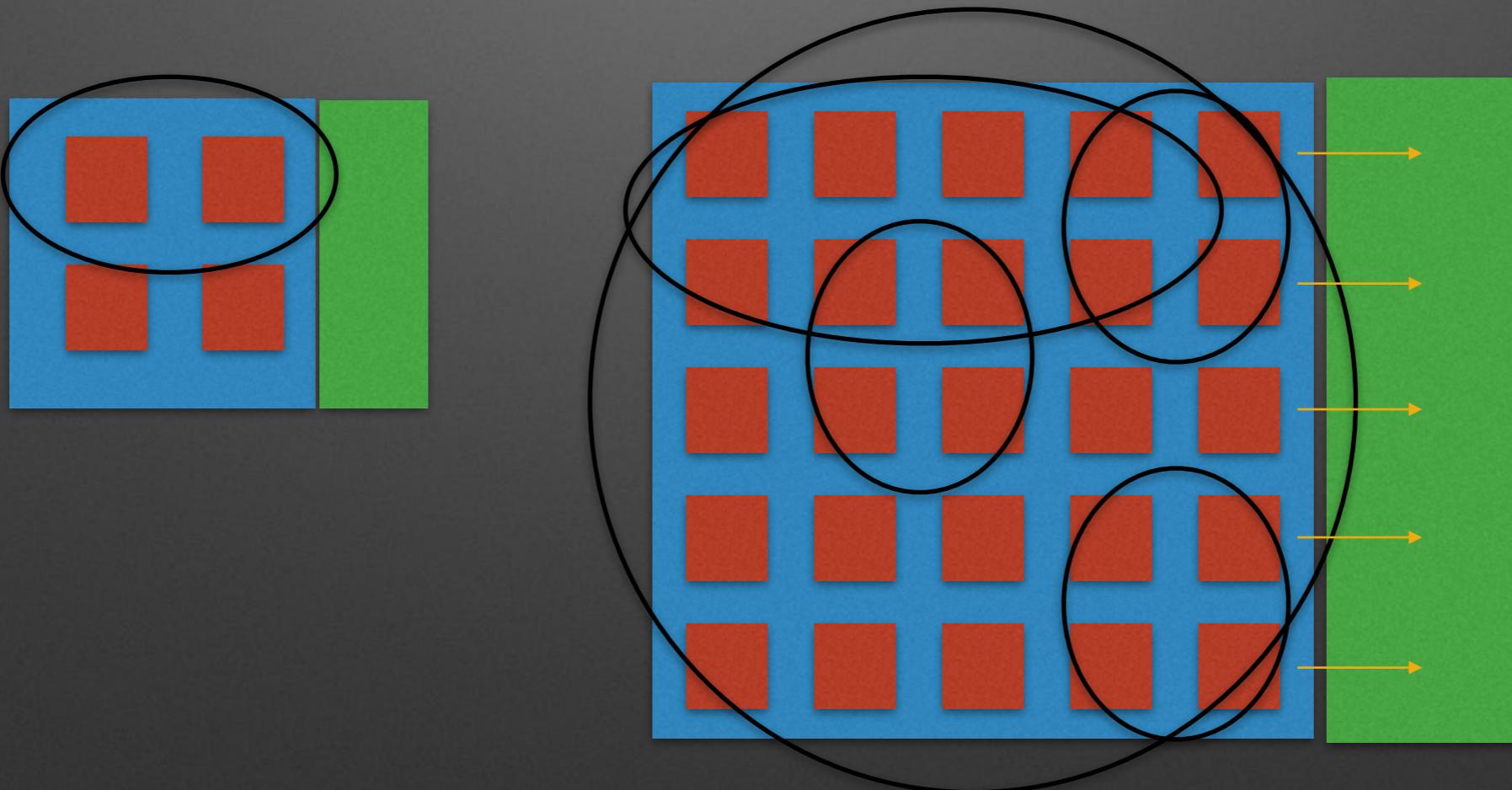


- E.g. many sequential blast searches at once, splitting bootstrap reps.

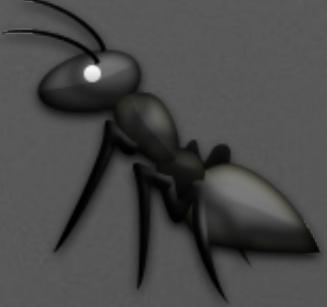
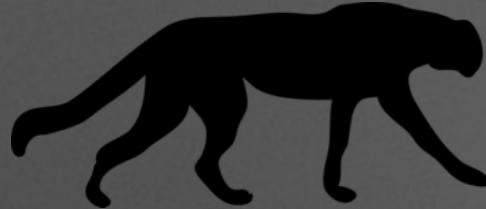
Number of CPUs: Parallelization



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

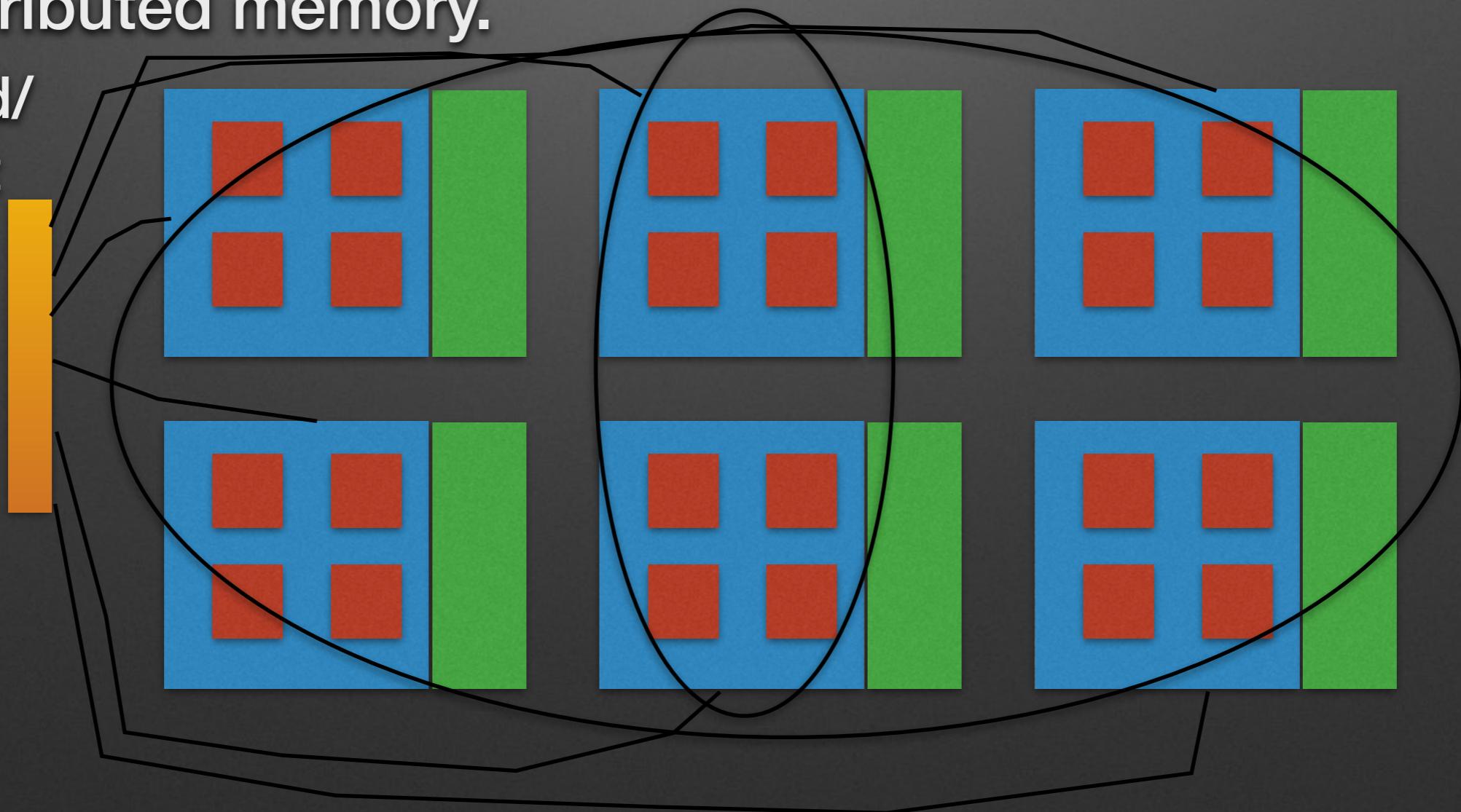


Number of CPUs: Parallelization



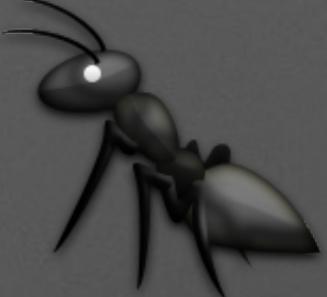
- MPI (Message Passing Interface)-multiple CPUs across multiple nodes. Messages are passed between nodes via the interconnect. This method uses distributed memory.

Infiniband/
ethernet
switch



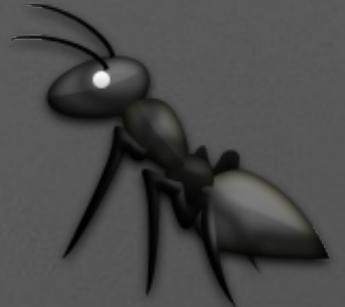


Number of CPUs: Parallelization

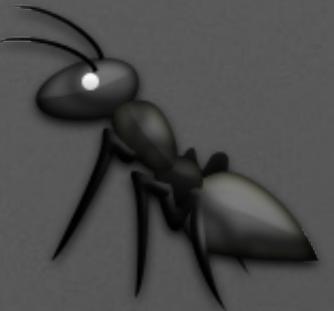
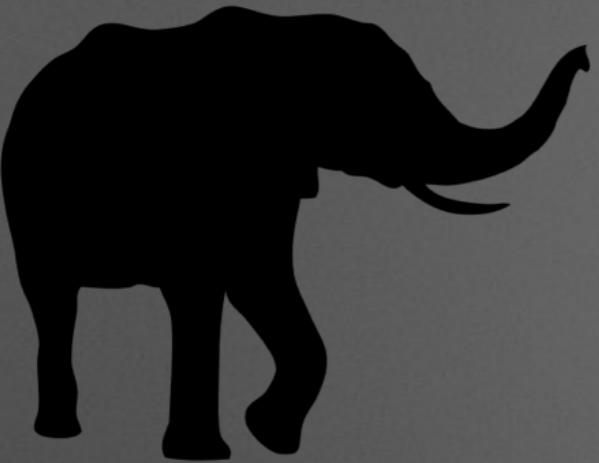


- How do I know which one to use?
- *This is determined by the type of program you wish to run. Check the program documentation or check with us.*

Time

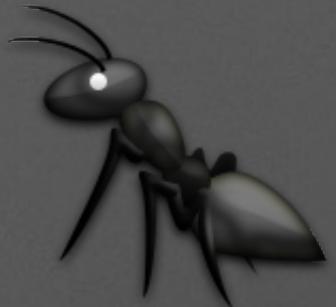


- The cluster needs to be told estimates of run time
- Short jobs: 7 hours
- Medium: 3 days
- Long: 30 days
- Unlimited (there are very few of these...anything submitted to this queue should have some checkpointing capability)



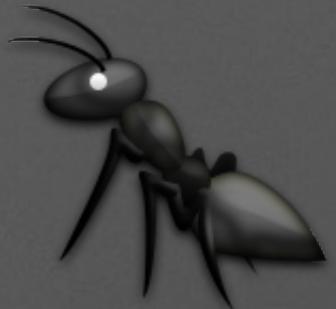
Memory use

- *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

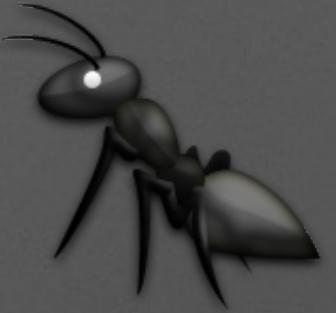
- How you tell the scheduler about your job
- -cwd runs job from current working directory
- -j y joins the stderr to the stdout
- -o sampleOut.log if you have chosen to join stderr and stdout, your job will write these to the sampleOut.log file
- -N The name of your job. Use something informative!
-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



qsub flags

- Parallel 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 mres=4G, h_data=4G, h_vmem=4G`
 - **mres:** “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.

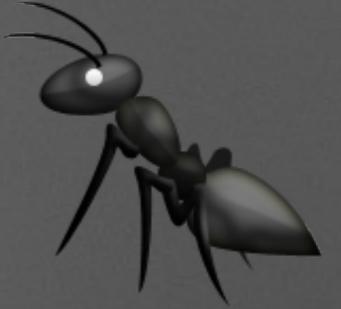


MEMORY IS PER CPU NOT PER JOB



- 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
```

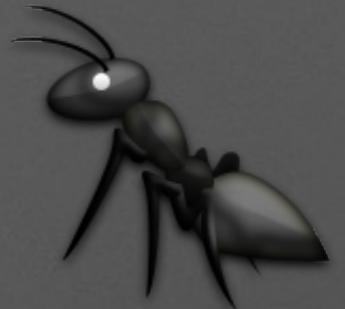
- To find out more about a module you can use:

```
$ module whatis <module_name>
$ module help <module_name>
$ module show <module_name>
```

- To load a module:

```
$ module load <module_name>
```

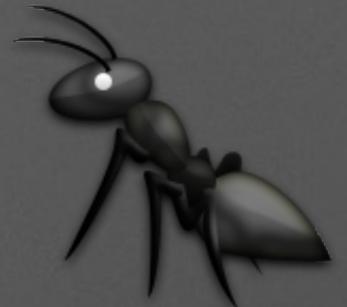
Putting it all together



sample qsub file: raxmltest.qsub:

```
qsub -cwd -j y \
-o raxmltest.log \
-N raxmltest \
-pe mthread 10 -q sThC.q \
-l mres=1G,h_data=1G,h_vmem=1G raxmltest.job
```

Putting it all together



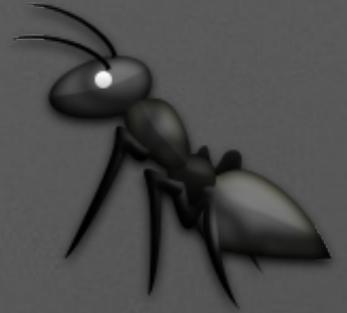
The file `raxmltest.job` holds the job instructions:

```
module load bioinformatics/raxml/8.1  
  
raxmlHPC-PTHREADS-SSE3 \  
  
-T $NSLOTS -n test \  
  
-f a -x 23434 -p 23423 \  
  
-N 100 -m GTRGAMMA -s primates.dna.phy
```

To run (aka to submit) the job:

```
$ source raxmltest.qsub
```

Putting it all together



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:

Type your commands 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!

Apache/2.4.18 (Ubuntu) PHP/7.2.1-1ubuntu3.10 x86_64 PHP/7.2.1-1ubuntu3.10
 QSub processor PHP ver: 1.0.1.0002.00 rev: 1.0.1.0002

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/cluster0/workshop/<username>`

Tips



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

Yay!

“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>.qsub**

- **upload it to hydra :**

```
$ scp <name>.qsub <username>@hydra-  
login01.si.edu:/pool/cluster0/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/cluster0/workshop/<username>
```

- Now submit your job using:

```
$ qsub <name>.qsub
```

- 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/cluster0/workshop/Hydra_workshop_2015/Day2/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/8.1
- 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.

- If you finish early, try to write the scripts without using the automatic qsub generator.
- Hint: this can be done with two scripts as mentioned earlier in the lecture.

Questions?



Intro to Hydra botany.key — root@hydra-3:/pool/cluster0/workshop/frandsenp — ssh — Solarized Dark xterm-256color — 58x19

```
root@hydra-3:/pool/cluster0/workshop/frandsenp frandsenp@hydra-2:~
```

-bash-4.1\$
-bash-4.1\$ ls
Day1 primates.dna.phy raxmltest.qsub
helloworld.qsub raxmltest.job
-bash-4.1\$ █

Intro to Hydra botany.key — root@hydra-3:/pool/cluster0/workshop/frandsenp — ssh — Solarized Dark xterm-256color — 58x19

root@hydra-3:/p...shop/frandsenp frandsenp@hydra-2:~

GNU nano 2.0.9 File: raxmltest.qsub

```
qsub -cwd -j y \
-o raxmltest.log \
-N raxmltest \
-pe mthread 10 -q sThC.q \
-l mres=1G,h_data=1G,h_vmem=1G raxmltest.job
```

[Read 6 lines]

^G Get Help ^O Write ^R Read F^Y Prev P^K Cut Te^C Cur Pos
^X Exit ^J Justif ^W Where ^V Next P^U UnCut ^T To Spell

Intro to Hydra botany.key — root@hydra-3:/pool/cluster0/workshop/frandsenp — ssh — Solarized Dark xterm-256color — 58x19

root@hydra-3:/p...shop/frandsenp frandsenp@hydra-2:~

GNU nano 2.0.9 File: raxmltest.job

```
module load bioinformatics/raxml/8.1
raxmlHPC-PTHREADS-SSE3 \
-T $NSLOTS -n test \
-f a -x 23434 -p 23423 \
-N 100 -m GTRGAMMA -s primates.dna.phy
```

[Read 6 lines]

^G Get Help ^O Write ^R Read F^Y Prev P^K Cut Te^C Cur Pos
^X Exit ^J Justif ^W Where ^V Next P^U UnCut ^T To Spell