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
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

SI Computing Cluster

Hydra

- Smithsonian HPC cluster
- 3,960 CPUs, 26 TB of RAM
- 16 high memory nodes (14 512GB; 2 1TB RAM; 1 2TB RAM)
- Many high CPU nodes with 64 CPUs

Support

https://confluence.si.edu/HPC

Confluence Spaces People Create

Research Information Services

Pages
Blog

SPACE SHORTCUTS
Shared Links
How-To Articles

PAGE TREE
Documentation
Genomics
High Performance Computing
Hydra and Lattice
Hydra Software
Hydra Hardware
Hydra Users
Hydra Maintenance
How-To Articles
Infrastructure
Research Computing Strategy
Sandbox
Shared Links
SIdora Release Roadmap

Pages / ... / Hydra and Lattice Edit Watch Tools

Hydra Hardware

Created by Frandsen, Paul, last modified on Jan 21, 2015

Hydra

Hydra is the main computing cluster administered by the Office of Research Information Services. It consists of 10 racks of servers. The vast majority of these servers are general purpose servers. Due to a recent upgrade, it also contains several high CPU servers.

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*

*Four 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

Sysadmin support:

DJ Ding, OCIO

Sylvain Korzennik, SAO

Rebecca Dikow, OCIO

SI-HPC-admin@si.edu

Application support:

Rebecca Dikow, OCIO

Matt Kweskin, NMNH LAB

Vanessa González, GGI

Mirian Tsuchiya, OCIO

Mike Trizna, OCIO

SI-HPC@si.edu

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 (hydra-5.si.edu)

Queuing System/Job Scheduler

- Software that schedules the jobs and submits them to the compute nodes.
- Hydra uses the “Univa Grid Engine” queuing system.
- Most jobs should be run in batch mode. To submit jobs, you use the command "`qsub`"
- There is also an interactive mode for short jobs. To initiate, type “`qrsh`”

Submitting Jobs

Job Submission: You need to know…

- Number of CPUs/type of parallel environment
- Time you job will take
- Memory you would like to reserve
- 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

sample “job” file

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -pe mthread 4
#$ -q mThM.q
#$ -l mres=40G,h_data=10G,h_vmem=10G,himem
#$ -cwd
#$ -j y
#$ -N jellyfish_histo
#$ -o jellyfish_histo.log
#$ -m bea
#
# ----------------Modules------------------------- #
module load bioinformatics/jellyfish
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on
$HOSTNAME
echo + NSLOTS = $NSLOTS
#
jellyfish histo -t 10 reads.jf > reads.histo
#
echo = `date` job $JOB_NAME done
```

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.

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>
```

sample “job” file

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -pe mthread 4
#$ -q mThM.q
#$ -l mres=40G,h_data=10G,h_vmem=10G,himem
#$ -cwd
#$ -j y
#$ -N jellyfish_histo
#$ -o jellyfish_histo.log
#$ -m bea
#
# ----------------Modules------------------------- #
module load bioinformatics/jellyfish
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on
$HOSTNAME
echo + NSLOTS = $NSLOTS
#
jellyfish histo -t 10 reads.jf > reads.histo
#
echo = `date` job $JOB_NAME done
```

Putting it all together

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

©2015 Smithsonian Institution HPC - OCIO (SGK/PF)
QSub generator PHP ver 1.0/1/150522 JS ver 1.0/1/150522

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 (e.g. /pool/genomics/USER).

Tips

Helpful commands

- 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`

“Hello world”

- Visit https://hydra-adm01.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

©2015 Smithsonian Institution HPC - OCIO (SGK/PP)
QSub generator PHP ver 1.0/1/150522 JS ver 1.0/1/150522

- Change to the directory that houses your qsub file:
  `$ cd /pool/genomics/<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:
  `$ 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 `/data/genomics/workshops/hydra`
- 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

©2015 Smithsonian Institution HPC - OCIO (SGK/PF)
QSub generator PHP ver 1.0/1/150522 JS ver 1.0/1/150522

- This time, let's use multiple threads
- Select 'multi-thread' and 4 CPUs
- RAxML module file is `bioinformatics/raxml`
- 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
```

- Submit this job (you should now how after doing the 'hello world' example.
- Once your job is complete, download your output tree file.