Minerva Scientific Computing Environment

An Introduction to Minerva

https://hpc.mssm.edu

Mount Sinai

Outline

- Logging in and Storage Layout
- Modules: Software Environment Management
- LSF: Load Sharing Facility, i.e, Batch Queue
- How to Submit and Monitor Jobs
- Job Submission Strategies

Logging in - General

- Minerva is a Linux machine, with centos 6.2
  - Linux is command line based, not GUI
  - Linux was developed using TTY devices. Commands are short and many times cryptic, but there is usually a good reason.

- Logging in requires a username, a memorized password, and a code obtained from a physical token.
  - https://hpc.mssm.edu/access/loggingin

- Assume I have already opened a terminal window on my workstation

  username: fludee01

  password: MyPaS$w0rd123456 < hidden, of course

  (Red is memorized password; Blue is generated token)

  <Hello messages>

  prompt>

Logging in - General

```
triumph:~ gene$ ssh fludee01@minerva.hpc.mssm.edu
**********************************************************************
*****
* WARNING: UNAUTHORIZED USE, POSSESSION, DUPLICATION, OR
TAMPERING WITH    *
* MOUNT SINAI HOSPITAL COMPUTERS, DATA, INFORMATION, PROGRAMS
OR SERVICES  *
* IS A VIOLATION OF POLICY AND A CRIMINAL OFFENSE.  VIOLATORS ARE
SUBJECT  *
* TO DISMISSAL AND/OR PROSECUTION.                              *
**********************************************************************
*****

Please enter your password followed by your security token:
Password:
Last login: Tue Sep 26 08:38:41 2017 from triumph.1425mad.mssm.edu
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~
        The /sc/orga/ file system is optimized for performance and
        capacity. It provides minimal redundancy and no backups.
```

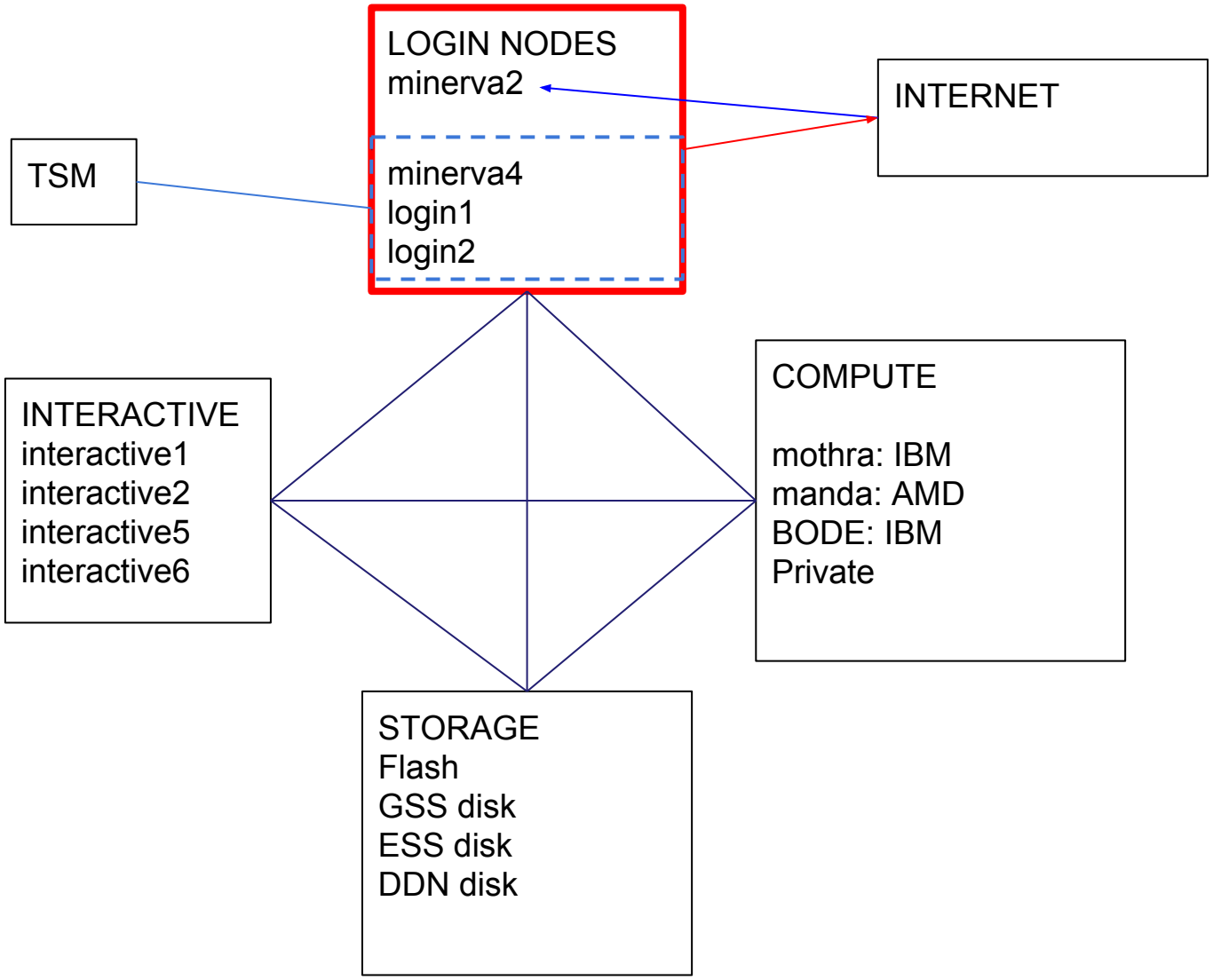
Logging in - Windows

- Install PuTTY from www.putty.org
  - Google it. It will be the first hit.
  - https://www.youtube.com/watch?v=ma6Ln30iP08
- If you are going to be using GUI's
  - In Putty: Connection > SSH > X11
    - Ensure "Enable X11 forwarding" is selected
  - On Windows box install Xming
    - Google; Download; Follow bouncing ball
  - Test by logging in and run the command: xclock
    - Should see a clock
- More ssh client: MobaXterm
  - Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more
  - https://mobaxterm.mobatek.net/

Logging in - Mac

- Open terminal window
  - I prefer Xterm2 download rather than supplied terminal emulator
- If using GUI (aka, X-windows), download and install Xquartz
- Login by:
  - “ssh <userid>@minerva.hpc.mssm.edu”
  - If using a GUI:
    - “ssh -X <userid>@minerva.hpc.mssm.edu”

Minerva


LOGIN NODES
minerva2
minerva4
login1
login2
INTERNET
TSM
INTERACTIVE
interactive1
interactive2
interactive5
interactive6
COMPUTE
mothra: IBM
manda: AMD
BODE: IBM
Private
STORAGE
Flash
GSS disk
ESS disk
DDN disk

Nodes

- Login nodes - Setup jobs; very quick tests; transferring data into/out of minerva
  - Address: minerva.hpc.mssm.edu or mothra.hpc.mssm.edu
  - 4 in number: minerva2; minerva4; login1; login2
  - minerva2 is reachable from outside Mount Sinai
    - Use minerva2 if you are transferring data to/from Mount Sinai. It is connected directly to the internet so you don't have to drag the data through the internal networks.
- Interactive nodes - The wild west. Free for all but be nice
  - ssh interactive1 or ssh interactive2
  - ssh interactive5 or ssh interactive6 (BODE only)
- Compute nodes - this is where batch submitted jobs run
  - manda: AMD; 64cores; 256GB
  - mothra/BODE: IBM; 12 cores; 64GB
  - Others: 2 himem nodes, GPU nodes
- Private- private nodes purchased by various groups

Storage

- Storage is in folders and subfolders. In linux, subfolders are separated by “/”
- 4 folders you can have

Home	/hpc/users/<userid>	• 10GB quota. • Slow. Use for “config” files, executables…NOT DATA • NOT purged and is backed up
Work	/sc/orga/work/<userid>	• 100GB quota • Fast, keep your personal data here • NOT purged but is NOT backed up
Scratch	/sc/orga/scratch/<userid>	• Free for all, shared wild west • Current size is about 100TB • Purge every 14 days
Project	/sc/orga/projects/<projectid>	• Groups can request project storage. • Need to submit an allocation request and get approval from allocation committee https://hpc.mssm.edu/hpc-admin/forms/allocation-request • NOT Purged; NOT backed up

Modules (Software Environment Management)

- $module avail (ca. 1,400 total packages and growing)

```
------------ /hpc/minerva-common//packages/modulefiles/ -------------------------------
EMAN2/2.04          gatk/1.5-21-g979a84a     plinkseq/0.08
MACS/1.4.2          gcc/4.1.2                protobuf/2.4.1-gcc
R/2.15.0            gcc/4.6.3(default)       python/2.6.7
Xmipp/2.4           gcc/4.7.0                python/2.7.2(default)
```

- $ module load *module_file*
- $ module load *[module_file1] [module-file2]*
- $ module switch [module-file_old] [module-file_new]
- $ module unload module-file
- $ module purge
- $ module list

```
Currently Loaded Modulefiles:
  1) tclpath/1.0                5) mpc/1.0.2                  9) hdf5/1.8.12-serial          13) allocations/setup
  2) projectPI/0.0              6) gcc/4.8.2                 10) zlib/1.2.8
  3) gmp/5.1.3                  7) python/2.7.6(default)          11) qt/5.3.2(default)
  4) mpfr/3.1.2                 8) intel/parallel_studio_xe_2015  12) py_packages/2.7(default)
```

Load Sharing Facility(LSF)

A Distributed Resource Management System

What is a Distributed Resource Management System, aka, Queuing System

- Control usage of hard resources
  - CPU cycles
  - Memory
  - Disk Space
  - I/O capacity

- Goal of DRMS is to achieve best utilization of resources and maximize system throughput.

- Can be decomposed into subsystems:
  - Job management
  - Physical resource management
  - Scheduling and queuing

Common/Useful LSF Commands

- `lsid`
  - A good choice of LSF command to start with is the **`lsid`** command
- `lshosts/bhosts`
  - shows all of the nodes that the LSF system is aware of
- `bsub`
  - submits a job interactively or in batch using LSF batch scheduling and queue layer of the LSF suite
- `bjobs`
  - displays information about a recently run job. You can use the **-l** option to view a more detailed accounting
- `bqueues`
  - displays information about the batch queues. Again, the **-l** option will display a more thorough description
- `bkill <job ID# >`
  - kill the job with job ID number of #
- `bhist -l <job ID# >`
  - displays historical information about jobs. A "-a" flag can displays information about both finished and unfinished jobs
- `bpeek -f <job ID#>`
  - displays the stdout and stderr output of an unfinished job with a job ID of #.
- bacct –l <job ID#>
  - Accounting statistics for finished job
- bmod [options] <job ID#>
  - modifies bsub options but only before job starts.

Submit and Monitor Jobs via LSF on Minerva

Two types of jobs:

Interactive jobs

- This will set up an interactive environment on compute nodes for users
- Useful for testing and debugging jobs particularly those that need special resources
- $ bsub -XF -P acc_hpcstaff -q premium -n 5 -W 1:10 -Ip /bin/bash

Batch jobs

- For production run
- The user submits a batch script to the batch system
  - Job parameters cannot be changed after job starts!!!

Submit Batch Jobs via LSF on Minerva - bsub

bsub options can be entered on command line and/or by placing #BSUB “cookies” in the submitted script

bsub [options] my_batch_job

- This will submit the script “my_batch_job” using the options on the command line. This will NOT interpret the #BSUB cookies in the script.

bsub [options] < my_batch_job, if the job script contains #BSUB cookies:

- This will interpret the #BSUB cookies in the script. Options on the command line override what is in the script.

Quick example job: testit.lsf

```
#!/bin/bash
#BSUB -J myjob
#BSUB -P YourAllocationAccount
#BSUB -q alloc
#BSUB -n 1
#BSUB -W  02:00
#BSUB -o t.out

echo “salve munde!”
```

Some bsub options

Option	Use
-q *qname*	Specify queue
-n *min[,max]*	Specify number of job slots(cores). This is total number of slots. They can be allocated anywhere. By default system will ***try*** to fill a node first, cf. –R and –app options
-I	Run job interactively.
-Ip	Interactive with pseudo-terminal (vi needs one)
-W *walltime*	Wall time in HH:MM NO SECONDS!
-o *path*	Append output to specified file. By default output is mailed. This option specifies output should be concatenated to specified file. Can use %J in path to specifiy job id Can use %I in path to specify job array index

Some bsub options (Cont.)

Option	Use
-oo *path*	Overwrite output file if it exists
-e *path*	Append stderr to specified file. Will be emailed by default. If not specified, stderr gets merged with stdout
-oe *path*	Overwrite error file if it exists
-J "*job-description*"	"Jobname[index | start-end:increment]" Enclosed in quotes. Optional index specifications signify this is a job array. Job index starts at 1. LSB_JOBINDEX is the index of the job

Some bsub options (Cont.)

Option	Use
-x	Specifies exclusive use of the node If –n 1 : get all cores/all memory/no affinity If –n x ≠ 1: get x cores/all memory/no affinity
-app app-script	Specify application profile. Preset bsub parameters. E.g. mpi switch configuration, checkpointing
-L <login shell>	Initializes environment using specified login shell. Needs full path.
-r	Re-run job if node goes down on another node. Re-queued if entire system goes down
-M	per slot memory limit.

Queues on Minerva

bsub -q [queue_name]

Queue	Description	Maximum Wall Time
alloc	Jobs that will be charged against an allocation.	144h (6d)
expressalloc	High throughput for jobs that will be charged to an allocation	2h
premium	High priority. Charged at 1.5 normal rate	144h (6d)
low	For jobs that are not to be charged against an allocation	24h

bsub

If Script is NOT executable:

bsub test.lsf

ERROR: Project must be 'acc_*'.
Request aborted by esub. Job not submitted.

bsub -P acc_hpcstaff -q premium -W 1 –o t.out test.lsf

Job <764676> is submitted to queue <premium>.
t.out:
/tmp/1395692735.764676: line 8: test.lsf: command not found

bsub (Cont.)

Script is NOT executable:

bsub < t.lsf

Job <764687> is submitted to queue <alloc>.

t.out -> Salve Munde!

Script is executable:

bsub –o t.out ./t.lsf

Job <764689> is submitted to queue <alloc>.

t.out -> Salve Munde!

bsub (Cont.)

With LSF, you can even bsub a shell command:

bsub -P acc_hpcstaff -q premium -W 2 –o ls.out ls

$tail ls.out #(The output (if any) follows:)

45.tar

759125.out

acc_7.txt

Aligned.out.sam

a.otf

arjun.rd_isa

audit

How to monitor jobs - bjobs

- Check your own jobs: **$bjobs**

```
JOBID      USER    JOB_NAME   STAT     QUEUE  FROM_HOST
EXEC_HOST   SUBMIT_TIME TIME_LEF
764943   fludee01 *txt ttt.out    RUN   gpualloc     login1 24*node25-23  Mar
25 12:08  95:28 L
```

- Check all jobs: **$bjobs –u all**

```
764905   zhangj21 *minimac.log   PEND     alloc    login1     -       Mar 25 11:41    -
764906   zhangj21 *minimac.log   PEND     alloc    login1     -       Mar 25 11:41    -
764907   zhangj21 *minimac.log   PEND     alloc    login1     -       Mar 25 11:41    -
764908   zhangj21 *minimac.log   PEND     alloc    login1     -       Mar 25 11:41    -
764909   zhangj21 *minimac.log   PEND     alloc    login1     -       Mar 25 11:41    -
764910   zhangj21 *minimac.log   PEND     alloc    login1     -       Mar 25 11:41    -
```

How to monitor jobs - bpeek

Check the output while job is running. –f option “tails” the output:

[fludee01@login1 ~]$ **bpeek 764943**

<< output from stdout >>

test size

****

Dynamic Bayesian Expert System based on Qualitative Hypotheses

****

The current working directory is:

- /sc/orga/scratch/fludee01/chang

10999 C-MYC 1

3280 OCT-4,SOX1,LEF1,FOXO1,SOX9,GATA2,ZFP64 0,0,1,1,1,1,1

1412 FOXA2 1

bkill

Kill jobs in the queue whether running or not

Lots of ways to get away with murder:

Kill by job id
bkill 765814

Kill by job name
bkill –J myjob_1

Kill a bunch of jobs
bkill –J myjob_*

Kill all your jobs
bkill 0

Job arrays get same job name and jobid…so:

Kill entire job array:
bkill 765815
bkill –J my_array

Kill one job in array:
bkill 765815[42]
bkill –J my_array[3]

Specifying a Resource - Memory

The –R option is used to specify resources;

-R rusage[*mem=mem_per_slot_in_MB*]

Specify how much memory per slot/core your program will require.

Default is 2000

bsub –n 6 –R rusage[mem=4000] …

This will allocate 6*4000MB or 24000MB to the job. (PER CORE, not per job)

Specifying a Resource - Job Slot

Span: define the shape of the slots you ask for:

-n 12 –R span[ptile=12] - all 12 slots/cores must be on 1 node

-n 24 –R span[ptile=12] - allocate 12 cores per node = 2 nodes

-n 24 –R span[hosts=1] - allocate all 24 cores to one host

bsub –n 12 –R span[hosts=1] < my_parallel_job

OMP_NUM_THREADS must be set in script:

export OMP_NUM_THREADS=12

export OMP_NUM_THREADS=$LSB_DJOB_NUMPROC – Dangerous

Better:

bsub –n 12 –R span[ptile=12] –a openmp < my_parallel_job

LSF sets it for you as number of procs per node

Specifying Resource - MPI

For MPI jobs, you want nodes allocated on one switch:

-R cu[type=switch:maxcus=1:pref=maxavail]

24 nodes per switch is maximum = 24*12 cores per switch maximum

bsub –n 20 –R cu[type=switch:maxcus=1:pref=maxavail] < my_mpi_job

But cores may not be on same node, so:

bsub –n 20 –R cu[type=switch:maxcus=1:pref=maxavail] -R span[ptile=12] …

Or

bsub –n 20 –R “cu[type=switch:maxcus=1:pref=maxavail] span[ptile=12]” …

Or

bsub –n 20 –app 1switch < my_mpi_job

Also have a 2switch

Recommendation for openmp parallel jobs

#BSUB -n 1 # One Job slot

#BSUB -R “affinity[core(5)]” # 5 cores to that job slot

#BSUB -R “rusage[mem=6000]” # 6000MB to that job slot...not per core

#BSUB -app openmp # Set OMP_NUM_THREADS for me please

Advantage: Can vary number of cores and/or memory without making any other changes or calculations.

A Bravura Submission - Mixing it all together

Suppose you want to run a combined MPI-openMP job. One mpi process per node, openMP in each MPI Rank:

bsub –n 160 –R span[ptile=8] –app 1switch –a openmp <my_awsome_job

1switch will insert resource requests for 1 switch and tile of 12/node

Command line span will override the app span so we will get 8 per node

The openmp esub script will start only 1 process per node and set OMP_NUM_THREADS on each node to 8

Final Friendly Reminder

- Never run jobs on login nodes
  - For file management, coding, compilation, etc., purposes only
- Never run jobs outside LSF
  - Fair sharing
  - Scratch disk not backed up, efficient use of limited resources
  - Old files will automatically be deleted without notification
- Logging onto compute nodes is no longer allowed.

Last but not Least

- Got a problem? Need a program installed? Send an email to:

hpchelp@hpc.mssm.edu