Using the Sapelo2 Cluster at the GACRC

Cluster New User Training Hands-out

Georgia Advanced Computing Resource Center (GACRC)

EITS/University of Georgia

Zhuofei Hou zhuofei@uga.edu

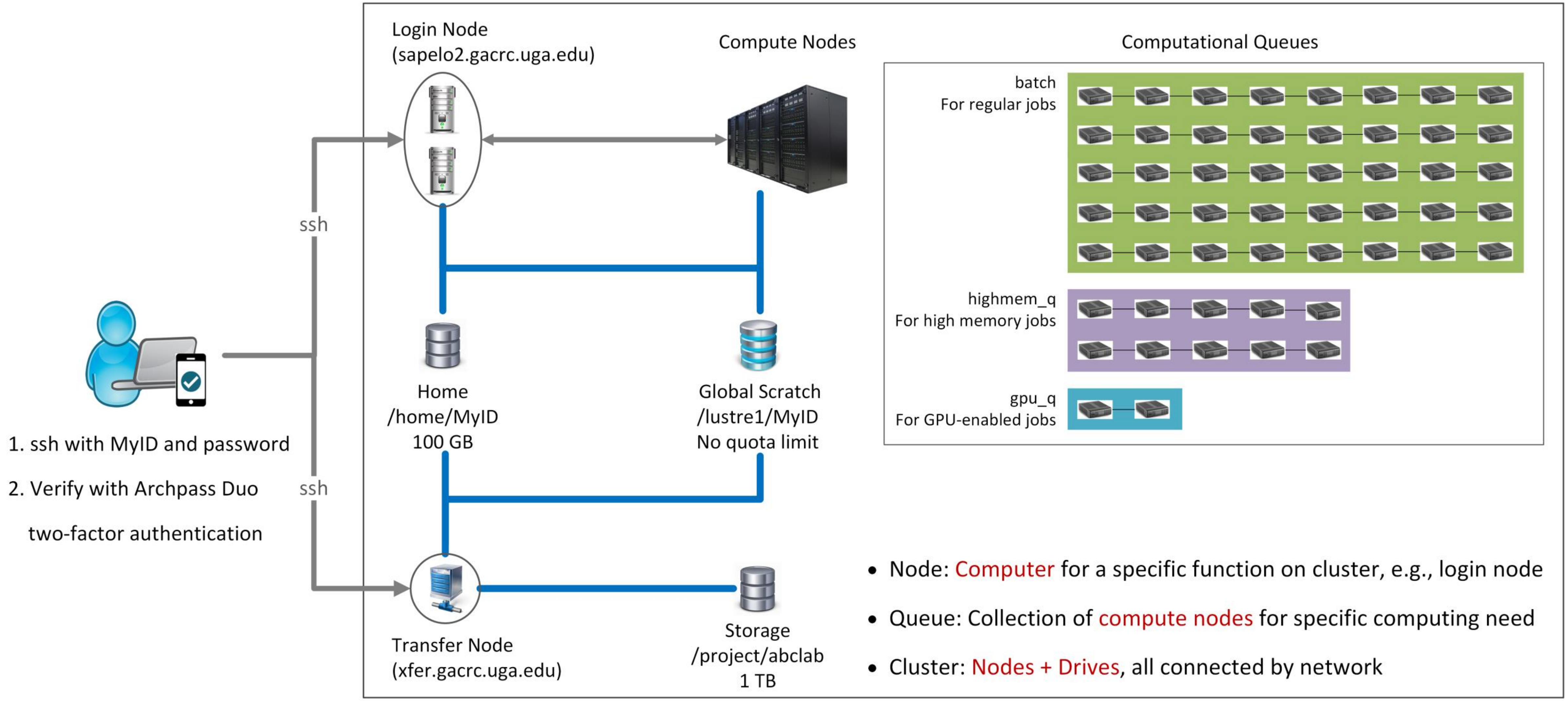
Outline

- What You Need to Know about Sapelo2 Cluster
- Work on Sapelo2 Cluster
- Appendix
- Guideline and Practical Tips

What You Need to Know about Sapelo2 Cluster

1. Cluster Overview
2. Storage Environment
3. Computing Resources
4. Software Environment

Sapelo2 Cluster



- Node: Computer for a specific function on cluster, e.g., login node
- Queue: Collection of compute nodes for specific computing need
- Cluster: Nodes + Drives, all connected by network

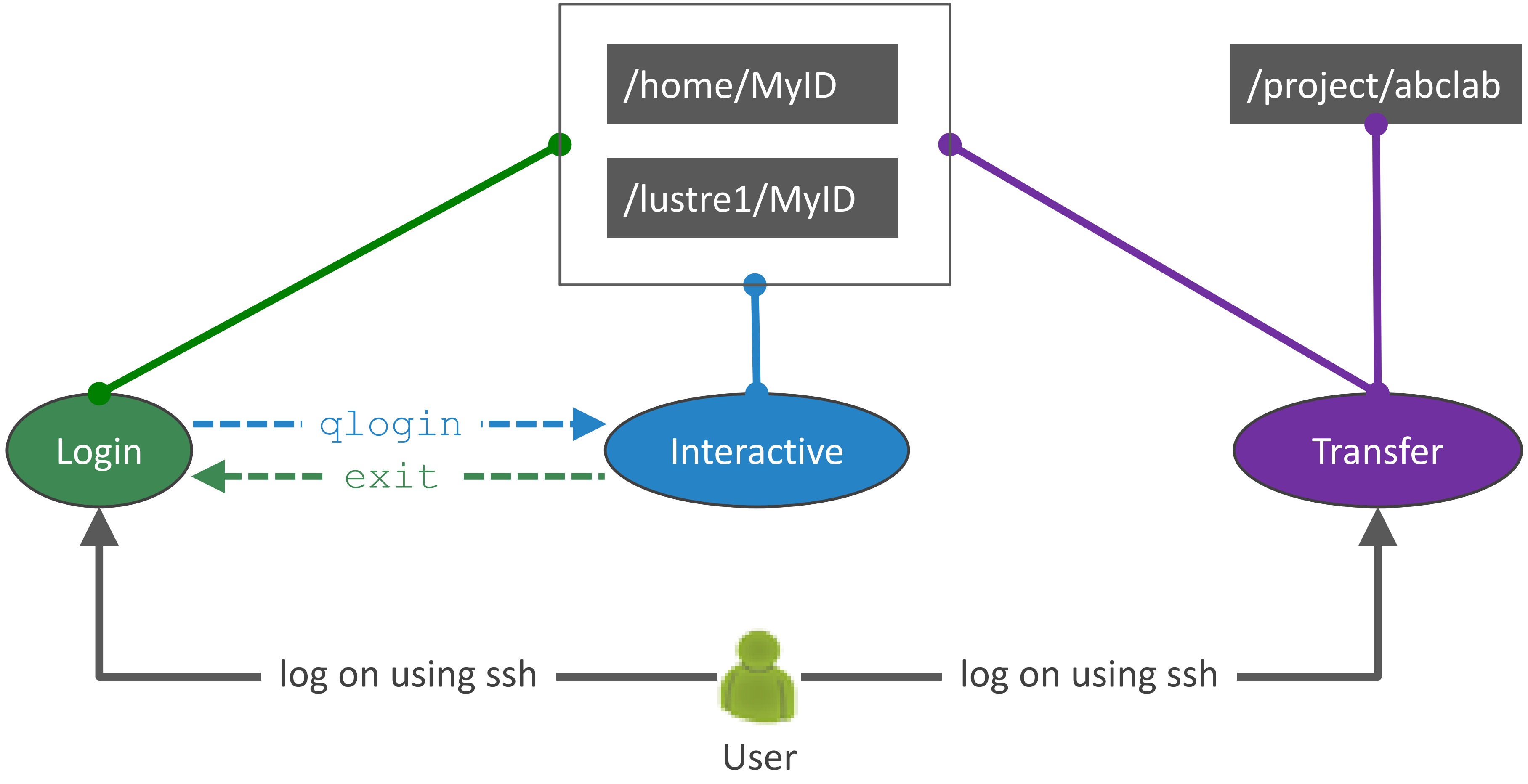
Cluster Overview

1. Sapelo2 cluster is a Linux (64-bit Centos 7) high performance computing (HPC) cluster
2. You can log on to 2 nodes: Login node (sapelo2.gacrc.uga.edu) and Transfer node (xfer.gacrc.uga.edu)
3. From Login node, you can open Interactive node using qlogin command
4. You have 4 directories: Home, Global Scratch, Storage and Local Scratch
5. You can submit jobs to 4 computational queues: batch, highmem_q, gpu_q, grpBuyin_q
6. You can use more than 400 modules installed on cluster (as of 05/17/2018)

Storage Environment – 4 Directories

4 Directories	Role	Quota	Accessible from	Intended Use	Backuped	Notes
/home/MyID	Home	100GB	Login Transfer Interactive	Static data: 1. Scripts, source codes 2. Local software	Yes	
/lustre1/MyID	Global Scratch	No Limit		Current job data: data being read/written by running jobs	No	User to clean up! *Subject to deletion in 30 days
/project/abclab	Storage	1TB (initial)	Transfer	Temporary data parking: non-current active data	Yes	Group sharing possible
/lscratch	Local Scratch	200GB (regular)	Compute	Jobs with heavy disk I/O	No	1. User to clean up when job exists from the node! 2. Persistent data

Storage Environment (Cont.) – Accessing Directories from Nodes



Storage Environment (Cont.) - Intended Use of Directories

I/O speed

Fast

Slow

/lustre1/MyID ← Current Job Data being used by current running jobs on cluster

/home/MyID ← Static Data being used frequently and not being modified often, e.g., scripts or local software

/project/abclab ← Non-current Active Data to be analyzed in the future, e.g., 2 months

User's Local Computer ← Final Data, e.g., final outputs and results

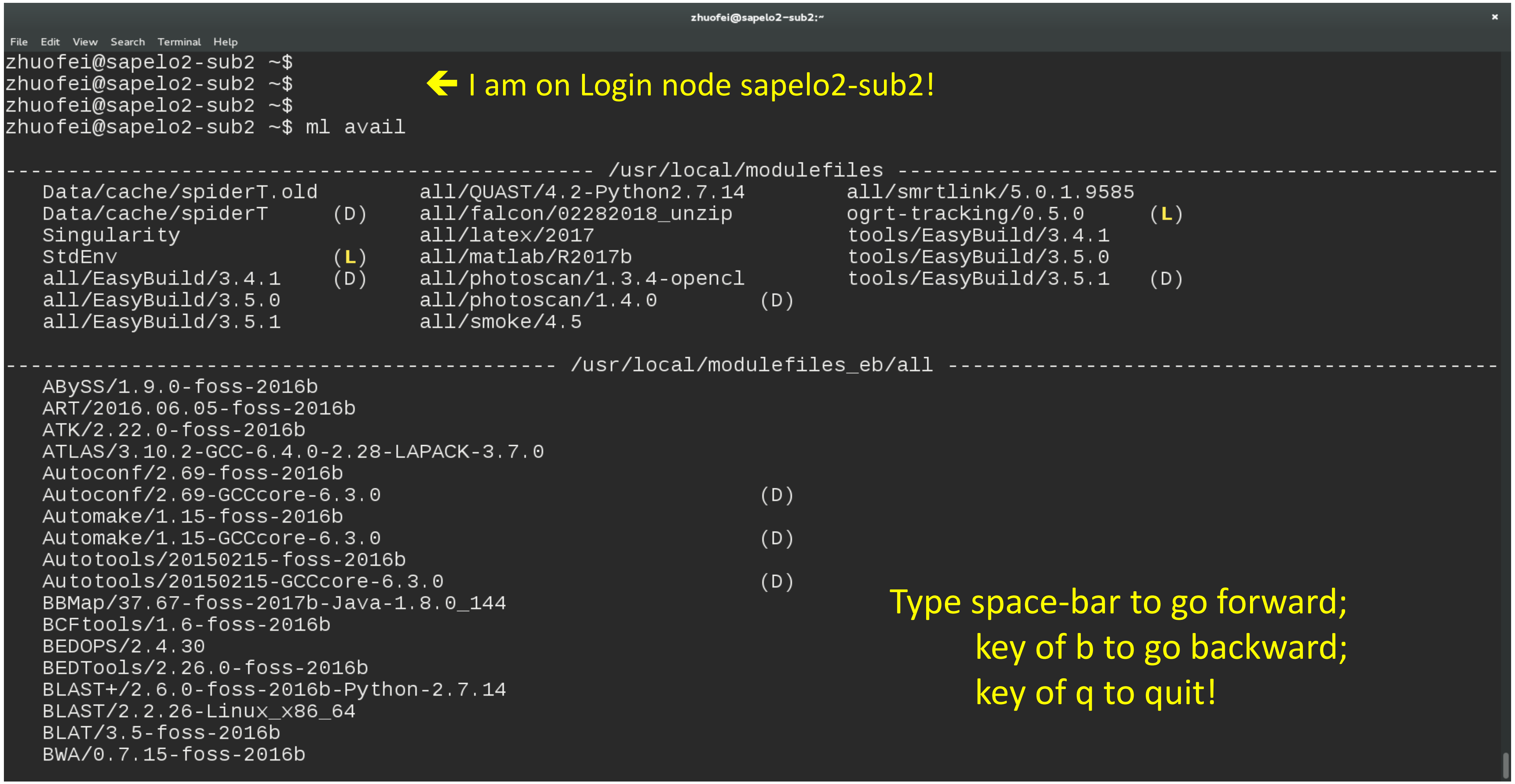
Computing Resources – 4 Computational Queues

Queue	Node Feature	Total Nodes	RAM(GB) /Node	Max RAM(GB) /Single-node Job	Cores /Node	Processor Type	GPU Cards /Node	InfiniBand
batch	Intel	30	64	62	28	Intel Xeon	N/A	Yes
	AMD	40	128	125	48	AMD Opteron		
highmem_q	Intel	1	1024	997	28	Intel Xeon		
	AMD	2	512	503	48	AMD Opteron		
gpu_q	GPU	2	96	94	12	Intel Xeon	7 NVIDIA K20Xm	
grpBuyin_q	variable							

Software Environment

1. Software names are long and have a Easybuild toolchain name associated to it, e.g., foss-2016b
2. Software names are case-sensitive!

- `ml avail`: List all available software modules installed on cluster
- `ml` *`moduleName`*: Load a module into your working environment
- `ml`: List modules currently loaded
- `ml unload` *`moduleName`*: Remove a module from your working environment
- `ml spider` *`pattern`*: Search module names matching a pattern (case-insensitive)

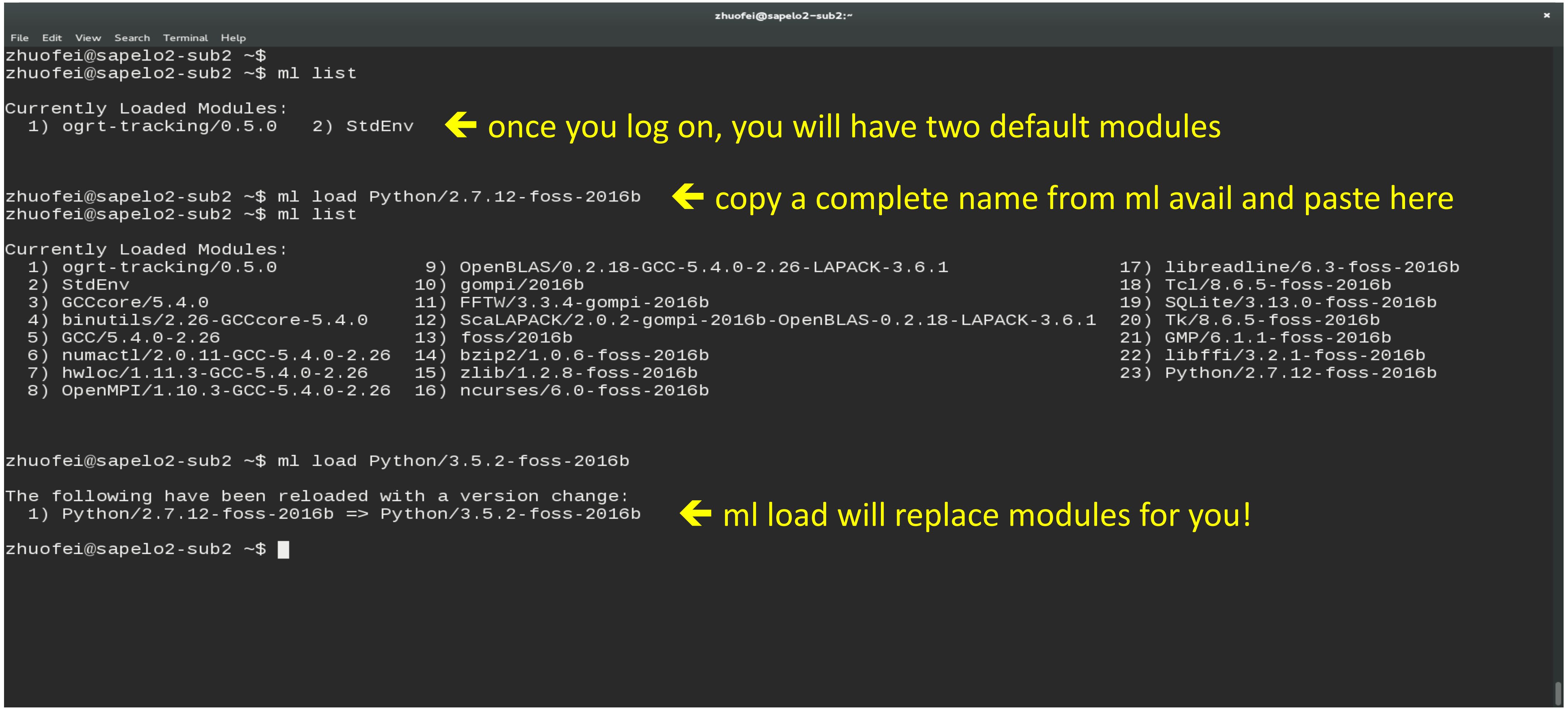


```
zhuofei@sapelo2-sub2 ~$
zhuofei@sapelo2-sub2 ~$
zhuofei@sapelo2-sub2 ~$
zhuofei@sapelo2-sub2 ~$ ml avail

------------------------------------------ /usr/local/modulefiles -------------------------------------------
   Data/cache/spiderT.old          all/QUAST/4.2-Python2.7.14          all/smrtlink/5.0.1.9585
   Data/cache/spiderT      (D)     all/falcon/02282018_unzip           ogrt-tracking/0.5.0      (L)
   Singularity                     all/latex/2017                      tools/EasyBuild/3.4.1
   StdEnv                  (L)     all/matlab/R2017b                   tools/EasyBuild/3.5.0
   all/EasyBuild/3.4.1     (D)     all/photoscan/1.3.4-opencl          tools/EasyBuild/3.5.1    (D)
   all/EasyBuild/3.5.0             all/photoscan/1.4.0         (D)
   all/EasyBuild/3.5.1             all/smoke/4.5

--------------------------------------- /usr/local/modulefiles_eb/all ---------------------------------------
   ABySS/1.9.0-foss-2016b
   ART/2016.06.05-foss-2016b
   ATK/2.22.0-foss-2016b
   ATLAS/3.10.2-GCC-6.4.0-2.28-LAPACK-3.7.0
   Autoconf/2.69-foss-2016b
   Autoconf/2.69-GCCcore-6.3.0                                 (D)
   Automake/1.15-foss-2016b
   Automake/1.15-GCCcore-6.3.0                                 (D)
   Autotools/20150215-foss-2016b
   Autotools/20150215-GCCcore-6.3.0                            (D)
   BBMap/37.67-foss-2017b-Java-1.8.0_144
   BCFtools/1.6-foss-2016b
   BEDOPS/2.4.30
   BEDTools/2.26.0-foss-2016b
   BLAST+/2.6.0-foss-2016b-Python-2.7.14
   BLAST/2.2.26-Linux_x86_64
   BLAT/3.5-foss-2016b
   BWA/0.7.15-foss-2016b
```

← I am on Login node sapelo2-sub2!

Type space-bar to go forward;
key of b to go backward;
key of q to quit!

Software Environment (Cont.)



```
zhuofei@sapelo2-sub2 ~$
zhuofei@sapelo2-sub2 ~$ ml list

Currently Loaded Modules:
  1) ogrt-tracking/0.5.0   2) StdEnv


zhuofei@sapelo2-sub2 ~$ ml load Python/2.7.12-foss-2016b
zhuofei@sapelo2-sub2 ~$ ml list

Currently Loaded Modules:
  1) ogrt-tracking/0.5.0              9) OpenBLAS/0.2.18-GCC-5.4.0-2.26-LAPACK-3.6.1             17) libreadline/6.3-foss-2016b
  2) StdEnv                          10) gompi/2016b                                               18) Tcl/8.6.5-foss-2016b
  3) GCCcore/5.4.0                   11) FFTW/3.3.4-gompi-2016b                                    19) SQLite/3.13.0-foss-2016b
  4) binutils/2.26-GCCcore-5.4.0     12) ScaLAPACK/2.0.2-gompi-2016b-OpenBLAS-0.2.18-LAPACK-3.6.1  20) Tk/8.6.5-foss-2016b
  5) GCC/5.4.0-2.26                  13) foss/2016b                                                21) GMP/6.1.1-foss-2016b
  6) numactl/2.0.11-GCC-5.4.0-2.26   14) bzip2/1.0.6-foss-2016b                                    22) libffi/3.2.1-foss-2016b
  7) hwloc/1.11.3-GCC-5.4.0-2.26     15) zlib/1.2.8-foss-2016b                                     23) Python/2.7.12-foss-2016b
  8) OpenMPI/1.10.3-GCC-5.4.0-2.26   16) ncurses/6.0-foss-2016b


zhuofei@sapelo2-sub2 ~$ ml load Python/3.5.2-foss-2016b

The following have been reloaded with a version change:
  1) Python/2.7.12-foss-2016b => Python/3.5.2-foss-2016b

zhuofei@sapelo2-sub2 ~$
```

← once you log on, you will have two default modules

← copy a complete name from ml avail and paste here

← ml load will replace modules for you!

Work on Sapelo2 Cluster

1. Job Submission Workflow
2. How to Know Job Details
3. How to Know Node Details
4. qlogin Commands: Open Interactive Node for Running Interactive Tasks

Job Submission Workflow (Refer to training workshop PDF for details)

1. Log on to Login node using MyID and password, and two-factor authentication with Archpass Duo:

   `ssh MyID@sapelo2.gacrc.uga.edu`
2. On Login node, change directory to global scratch : `cd /lustre1/MyID`
3. Create a working subdirectory for a job : `mkdir ./workDir`
4. Change directory to workDir : `cd ./workDir`
5. Transfer data from local computer to workDir : use `scp` or SSH File Transfer to connect Transfer node

   Transfer data on cluster to workDir : log on to Transfer node and then use `cp` or `mv`
6. Make a job submission script in workDir : `nano ./sub.sh`
7. Submit a job from workDir : `qsub ./sub.sh`
8. Check job status : `qstat_me` or Cancel a job : `qdel JobID`

How to Know Job Details

Option 1: `qstat -f JobID` for ***running jobs*** or ***finished jobs in 24 hours***

Option 2: Email notification from ***finished jobs (completed, canceled, or crashed)***,

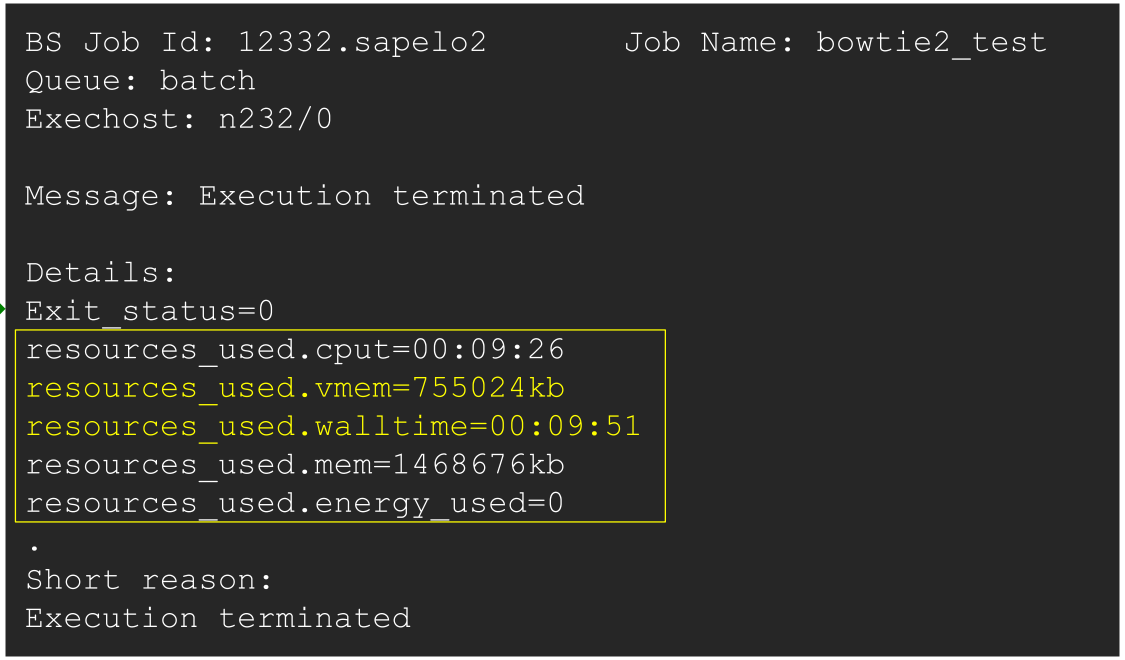
if using:

```
#PBS -M MyID@uga.edu
#PBS -m ae
```

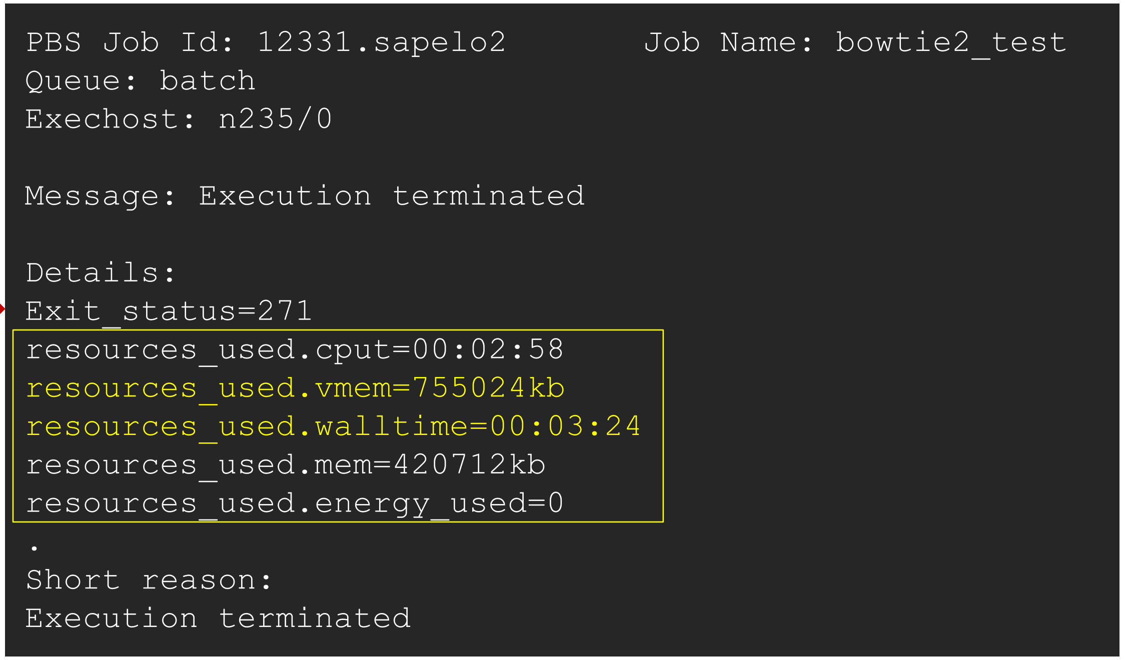
Option 1: qstat -f JobID (running jobs or finished jobs in 24 hour)

```
$ qstat -f 12222
Job Id: 12222.sapelo2
    Job_Name = testBlast
    Job_Owner = zhuofei@10.56.200.51
    resources_used.cput = 00:00:00
    resources_used.vmem = 316864kb
    resources_used.walltime = 00:15:01
    resources_used.mem = 26780kb
    resources_used.energy_used = 0
    job_state = C
    queue = batch
    .
    Error_Path = sapelo2-sub2.ecompute:/lustre1/zhuofei/examples/testBlast.e12222
    exec_host = n236/0-3
    Output_Path = sapelo2-sub2.ecompute:/lustre1/zhuofei/examples/testBlast.o12222
    .
    Resource_List.nodes = 1:ppn=4:Intel
    Resource_List.mem = 20gb
    Resource_List.walltime = 02:00:00
    Resource_List.nodect = 1
    .
    Variable_List = PBS_O_QUEUE=batch,PBS_O_HOME=/home/zhuofei,……
                     PBS_O_WORKDIR=/lustre1/zhuofei/workDir,
```

Option 2: Email notification from finished jobs



```
BS Job Id: 12332.sapelo2        Job Name: bowtie2_test
Queue: batch
Exechost: n232/0

Message: Execution terminated

Details:
Exit_status=0
resources_used.cput=00:09:26
resources_used.vmem=755024kb
resources_used.walltime=00:09:51
resources_used.mem=1468676kb
resources_used.energy_used=0
.
Short reason:
Execution terminated
```



```
PBS Job Id: 12331.sapelo2       Job Name: bowtie2_test
Queue: batch
Exechost: n235/0

Message: Execution terminated

Details:
Exit_status=271
resources_used.cput=00:02:58
resources_used.vmem=755024kb
resources_used.walltime=00:03:24
resources_used.mem=420712kb
resources_used.energy_used=0
.
Short reason:
Execution terminated
```

Sender: dispatch_root

How to Know Node Details

Option 1: mdiag -v –n | grep [pattern] | …

```
mdiag -v -n | grep batch | grep AMD
mdiag -v -n | grep batch | grep Intel
mdiag -v -n | grep highmem_q
mdiag -v -n | grep grpBuyin_q
```

Option 2: from login node, you can ssh to a compute node and run a command there!

```
ssh n72 'lscpu‘
ssh n222 'free -g‘
ssh n237 "ps aux | grep '^MyID'"
```

qlogin Commands

1. Type qlogin commands from Login node to open Interactive node:
   - `qlogin_intel`: Start an interactive session on an Intel node
   - `qlogin_amd`: Start an interactive session on an AMD node
   - `qlogin`: start an interactive job on either types of nodes
2. Type `exit` command to quit and back to Login node

qlogin Commands

Purpose1: Open interactive node for running interactive tasks of R, Python, Bash scripts, etc.

```
zhuofei@sapelo2-sub1 ~$ qlogin
qsub: waiting for job 12426.sapelo2 to start
qsub: job 12426.sapelo2 ready

zhuofei@n204 ~$ ml spider R
----------------------------------------------------------------------------
  R: R/3.4.1-foss-2016b-X11-20160819-GACRC
----------------------------------------------------------------------------
…
zhuofei@n204 ~$ ml R/3.4.1-foss-2016b-X11-20160819-GACRC
zhuofei@n204 ~$ R
R version 3.4.1 (2017-06-30) -- "Single Candle"
…
[Previously saved workspace restored]
> a<-1 ; b<-7
> a+b
[1] 8
>
```

qlogin Commands

Purpose2 : Open interactive node for compiling/testing source codes of Fortran, C/C++, Python, etc.

```
zhuofei@sapelo2-sub1 ~$ qlogin_intel
qsub: waiting for job 20912.sapelo2 to start
qsub: job 20912.sapelo2 ready

zhuofei@n206 ~$  ml spider iomkl
---------------------------------------
 iomkl:
---------------------------------------
   Description:
     Intel Cluster Toolchain Compiler Edition provides Intel C/C++
and Fortran compilers, Intel MKL & OpenMPI.

    Versions:
       iomkl/2018a
…
zhuofei@n206 ~$ ml iomkl/2018a
zhuofei@n206 ~$ icc mysource.c -o myexec.x
zhuofei@n206 ~$
```

Appendix: Examples of Batch Serial/Threaded/MPI Job Scripts

- Components you need to run a job:
  - ➢ Software already installed (cluster software or the one installed by yourself)
  - ➢ Job submission script to
    1. specify computing resources:
       - ✓ number of nodes and cores
       - ✓ amount of memory
       - ✓ maximum wallclock time
    2. load software using ml load (for cluster software)
    3. run any Linux commands you want to run, e.g., pwd, mkdir, cd, echo, etc.
    4. run the software
  - ➢ Input data for analysis, if have
- Common queueing commands you need:
  - ➢ qsub, qstat_me, qstat, qdel
  - ➢ qstat -f, showq

Example 1: Serial job script running NCBI Blast+ using 1 CPU

```
#PBS -S /bin/bash                  → Linux default shell (bash)
#PBS -q batch                      → Queue name (batch)
#PBS -N testBlast                  → Job name (testBlast)
#PBS -l nodes=1:ppn=1              → Number of nodes (1), number of cores (1),  node feature is NOT needed!
#PBS -l mem=20gb                   → Maximum amount of RAM memory (20 GB) is enforced by the cluster!
#PBS -l walltime=48:00:00          → Maximum wall-clock time (48 hours) for the job, default 6 minutes

cd $PBS_O_WORKDIR                  → Compute node will use the directory from which the job is submitted
                                     as the working directory, i.e., /lustre1/MyID/workDir

ml load BLAST+/2.6.0-foss-2016b-Python-2.7.14      → Load the module of ncbiblast+, version 2.6.0

time blastn [options] …            → Run blastn with 'time' command to measure the amount of time it
                                     takes to run the application
```

https://wiki.gacrc.uga.edu/wiki/BLAST%2B-Sapelo2

*Example 2: Threaded job script running NCBI Blast+ using 4 CPUS

```
#PBS -S /bin/bash
#PBS -q batch
#PBS -N testBlast
#PBS -l nodes=1:ppn=4                  ➔ Number of nodes (1), number of cores (4)
#PBS -l mem=20gb                          Number of cores requested (4) = Number of threads (4)
#PBS -l walltime=480:00:00

#PBS -M jsmith@uga.edu                 ➔ Email address to receive a notification for computing resources
#PBS -m ae                             ➔ Send email notification when job aborts (a) or terminates (e)
#PBS -j oe                             ➔ Standard error file (testBlast.e12345) will be merged into standard
                                         out file (testBlast.o12345)

cd $PBS_O_WORKDIR

ml load BLAST+/2.6.0-foss-2016b-Python-2.7.14

time blastn -num_threads 4 [options] …     ➔ Run blastn with 4 threads (-num_threads 4)
```

*Example 3: MPI job script running RAxML using 2 full nodes

```
#PBS -S /bin/bash
#PBS -q batch
#PBS -N testRAxML
#PBS -l nodes=2:ppn=28            ➔ Number of nodes (2), number of cores (28)
#PBS -l walltime=120:00:00          Total cores requested = 2 × 28 = 56
#PBS -l mem=100gb                   We suggest, Number of MPI Processes (50) ≤ Number of cores requested (56)

cd $PBS_O_WORKDIR

ml load RAxML/8.2.11-foss-2016b-mpi-avx        ➔ To run raxmlHPC-MPI-AVX, MPI version using OpenMPI


mpirun –np 50 raxmlHPC-MPI-AVX [options]       ➔ Run raxmlHPC-MPI-AVX with 50 MPI processes (–np 50),
                                                 default 56
```

Guideline Tips

- Do NOT use Login node to run CPU/memory intensive jobs directly ➔ submit jobs to queue!
- Do NOT use Login Node to upload/download large data to/from cluster ➔ use Transfer node!
- Do NOT use home dir for storing large job data ➔ use global scratch /lustre1/MyID
- Do NOT park data on global or local scratch ➔ clean up when job finished or exits from node
- NO large memory job running on batch or jlm_q ➔ use highmem_q
- NO small memory job running on highmem_q ➔ use batch or jlm_q
- As a general rule, threads # = cores # requested

Practical Tips

- Each directory should not have too many files inside! A rule of thumb would be to try to keep no more than a few tens of thousands of files (<10000 would be even better) in any single directory which is accessed frequently

All files are in ONE single dir! ✗

Files are organized in subdirs! ✓

Practical Tips

- Job name should have a specific computational meaning

  Good Examples: `#PBS -N blastn_dataSet1_trail2 ; #PBS -N M-10-1121`

  Bad Examples: `#PBS -N job1 ; #PBS -N bowtie ; #PBS -N 20160930`

- Monitor job progress from time to time, to catch if a job gets stuck