Batch System

For **performance** and **stability** reasons, we recommend using **qsub** and **xqsub** commands for submitting batch jobs.
Moreover, when submitting multiple jobs, add a sleep delay between jobs or use **job arrays** for submitting identical jobs.
Be aware the there is a maximum size limit of 64KB for scripts submitted to the queue system. Scripts submitted to the queuing system should mainly consist of parameters for the queuing system and executions of the "real" job.

Batch system

The batch job queuing system on Computerome is based on TORQUE Resource Manager (generally **qsub** and **q...** type commands) and Moab Workload Manager (generally **msub** and **m...** type commands).Additionally, we have **xqsub and xmsub**, perl wrapper scripts to qsub and msub respectiv ely, which build a job submission script for you.Extensive documentation is available here:

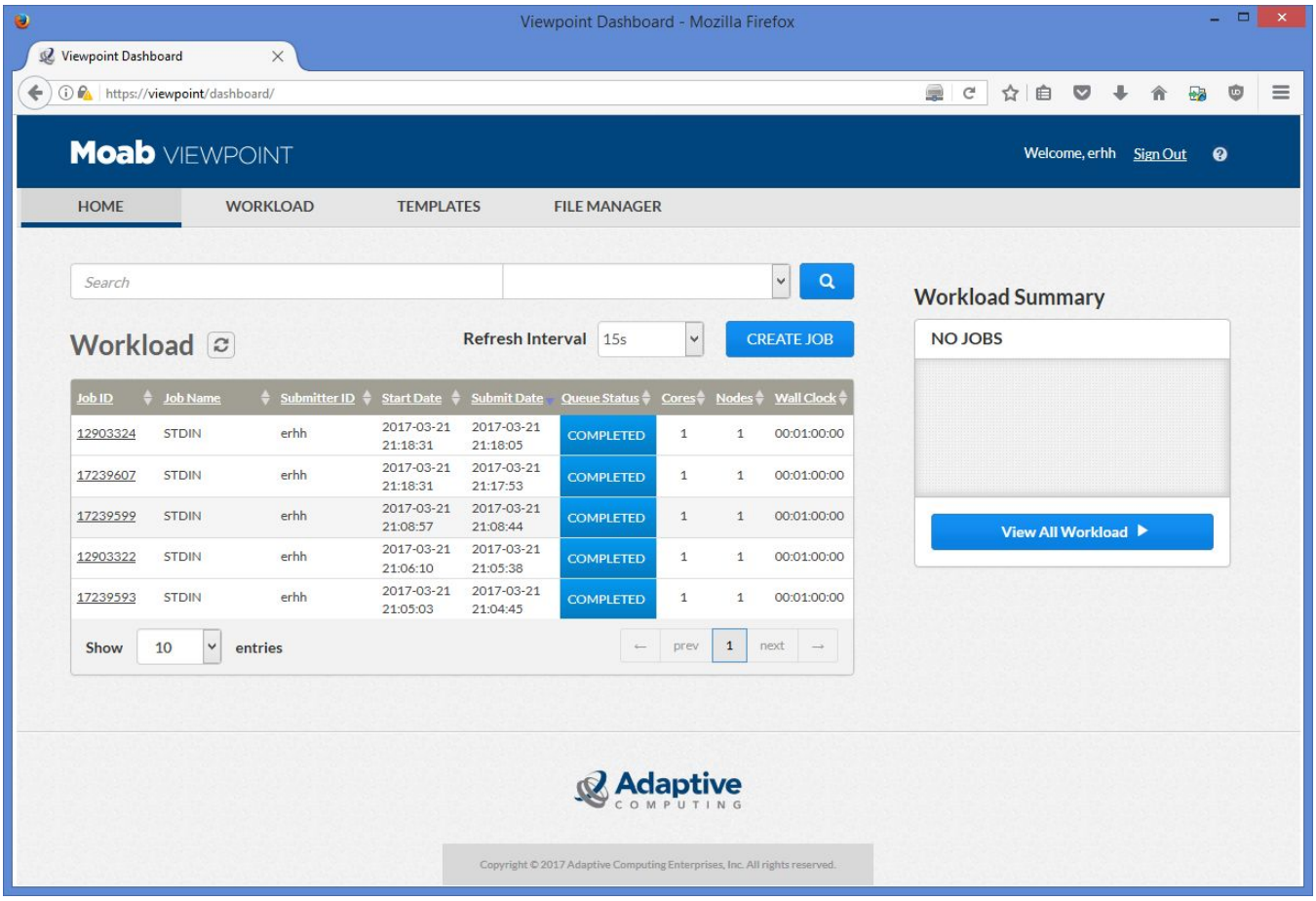
- TORQUE Resource Manager Administrator Guide
- Moab Workload Manager Administrator Guide

Batch queues

The queueing system on Computerome works almost like on earlier CBS systems, but simpler, because we have a more uniform infrastructure.Because of the simplified setup it is no longer necessary to use partition or queue options when submitting jobs - however, as described in Working in projects is is required to supply a **<group_NAME>** corresponding to the **project** you are currently working in.

Moab Viewpoint

Since 2017.03.24 we have **Moab Viewpoint**, a new, web-based way for users to interact with the queueing system.To access it, you open a Web browser **f rom inside Computerome** (i.e. either an **interactive session** or through the **Virtual desktop solution**) and go to **https://viewpoint/**.After login with your regular userid, you will get the following screen:



Submitting batch jobs

On Computerome, there are two types of machines:

- 27 systems each with 32 CPU cores and 1TB of memory
- Approximately 500 systems each with 28 CPU cores and 128 GB of memory.

We have a new environment which makes use of the modules environment command. So in order to get `qsub` and `msub` in you PATH variable execute:

```
$ module load moab torque
```

Now you can submit jobs via the command `qsub` and/or `msub`.We strongly encourage you to take advantage of modules in your pipelines as it gives you better control of your environment.In order to submit jobs that will run on one node only you will only have to specify the following resources:

1. How long time you expect the job to run **'-l walltime=<time>'**
2. How much memory your job requires **'-l mem=xxxgb'**
3. How many CPUs **'-l nodes=1:ppn=<number of CPUs>'** ;for the 1TB nodes number of CPUs can be from 1 to 32, for the other nodes it will be from 1 to 28.
4. The **<group_NAME>** for your current project **'-W group_list=<group_NAME> -A <group_NAME>'** .

To run a job with 23 CPUs, 100GB memory lasting an hour you can use the command:

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=1:ppn=23,mem=100gb,walltime=3600 <your script>
```

or using msub:

```
$ msub -W group_list=<group_NAME> -A <group_NAME> -l nodes=1:ppn=23,mem=100gb,walltime=3600 <your script>
```

The parameters nodes, ppn, mem is just an example and you should be change to suit your specific job

Interactive jobs

When you want to test something in the batch system, it is strongly recommended to run in an **interactive job**, by using the following:

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -X -I
```

This will give you access to a single compute node, where you can perform your testing without affecting other users.

iqsub

Computerome is now offering an even more straightforward way to work interactively, the way you do on your own computer or a local linux server, instead of having to submit everything through the queuing system.Just login and type **`iqsub`** and the system will ask you 3 simple questions, after which you'll be redirected to a full, private node.

```
$ iqsub

[ Interactive job ]

  => [ Select group ]

    => [ Select time needed (non extendable) ]

      => [ Select number of Processors needed (1-28/32)
```

Under **no** circumstances should you **ever** run scripts on the Computerome login node.

Script file example

A script for a file to be submitted with qsub might begin with lines like:

```
#!/bin/sh
### Note: No commands may be executed until after the #PBS lines
### Account information
#PBS -W group_list=pr_12345 -A pr_12345
### Job name (comment out the next line to get the name of the script used as the job name)
#PBS -N test
### Output files (comment out the next 2 lines to get the job name used instead)
#PBS -e test.err
#PBS -o test.log
### Only send mail when job is aborted or terminates abnormally
#PBS -m n
### Number of nodes
#PBS -l nodes=1:ppn=8
### Memory
#PBS -l mem=120gb
### Requesting time - format is <days>:<hours>:<minutes>:<seconds> (here, 12 hours)
#PBS -l walltime=12:00:00
### Forward X11 connection (comment out if not needed)
#PBS -X

# Go to the directory from where the job was submitted (initial directory is $HOME)
echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR

### Here follows the user commands:
# Define number of processors
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS nodes

# Load all required modules for the job
module load tools
module load perl/5.20.2
module load <other stuff>

# This is where the work is done
# Make sure that this script is not bigger than 64kb ~ 150 lines, otherwise put in seperat script and execute
from here
<your script>
```

The **$PBS...** variables are set for the batch job by Torque.

If you already have **loaded some modules in your login environment** , you do not need to specify them in the jobscript.
However, we **recommend** that you do it anyway, since it improves the portability of the jobscript and serves as a reminder of the requirements.

The complete list of variables is documented in Exported batch environment variables.Further examples of Torque batch job submission is documented in Job submission

Specifying a different project account

If you run jobs under different projects, for instance **pr_12345** and **pr_54321**, you must make sure that each project gets accounted for separately in the system's accounting statistics.You specify the relevant project account (for example, **pr_54321**) for each individual job by using these flags to the qsub command:

```
$ qsub -W group_list=pr_54321 -A pr_54321 ...
```

or in the job script file, add line like this near the top:

```
#PBS -W group_list=pr_54321 -A pr_54321
```

Please use project names only by agreement with your project owner.

Estimating job resource requirements

First time you run your script, you may not have a clear picture of what kind of resource requirements it has.To get a rough estimate, you could submit a job to a full node, with large walltime:Regular compute node (aka. 'thinnode'):

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=1:ppn=28:thinnode,walltime=99:00:00 -m n <script>
```

Fat node:

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=1:ppn=32:fatnode,walltime=99:00:00 -m n <script>
```

To see the actual resource usage, see output from command `tracejob`

As a result of recent performance and stability improvements to the queuing system, the 'tracejob' command is currently not available to regular users, but must be run as a privileged account on a particular set of servers.
We are working on providing a solution to this, but in the meantime, please contact Computerome support if you need to get results from 'tracejob'

Alternatively you can add theses linse to the bottom of your script

module load shared moab
checkjob -v $PBS_JOBID

They wil gennerate something like the following:

```
Total Requested Tasks: 20
Total Requested Nodes: 1
Req[0]  TaskCount: 20  Partition: torque
Dedicated Resources Per Task: PROCS: 1  MEM: 12G
Utilized Resources Per Task:  PROCS: 0.37  MEM: 12G  SWAP: 2020M
Avg Util Resources Per Task:  PROCS: 0.37
Max Util Resources Per Task:  PROCS: 0.80  MEM: 12G  SWAP: 2020M
Average Utilized Memory: 10761.74 MB
Average Utilized Procs: 8.47
```

To calculate what you should use for the "-l mem=" parameter you have to times the number of tasks with "Max Util Resource Per Task" "MEM:" Here it would bbe 20 * 12 gb = 240gb.

```
$ tracejob 5306250
/var/spool/torque/server_priv/accounting/20150212: Permission denied
/var/spool/torque/mom_logs/20150212: No such file or directory
/var/spool/torque/sched_logs/20150212: No such file or directory

Job: 5306250.risoe-r04-sn064.cm.cluster

02/12/2015 11:10:22  S    enqueuing into idle, state 1 hop 1
02/12/2015 11:10:37  S    Job Run at request of root@risoe-r04-sn064.cm.cluster
02/12/2015 11:10:37  S    child reported success for job after 0 seconds (dest=???), rc=0
02/12/2015 11:10:37  S    Not sending email: User does not want mail of this type.
02/12/2015 11:10:37  S    Not sending email: User does not want mail of this type.
02/12/2015 11:10:37  S    Exit_status=0 resources_used.cput=00:00:00 resources_used.mem=0kb resources_used.
vmem=0kb
                          resources_used.walltime=00:00:00
02/12/2015 11:10:37  S    on_job_exit valid pjob: 5306250.risoe-r04-sn064.cm.cluster (substate=50)
```

Look at **`resources_used.xyz`** for hints.

Requesting a minimum memory size

A number of node features can be requested, see the Torque Job Submission page. For example, you may require a minimum physical memory size by requesting:

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=2:ppn=16,mem=120gb <your script>
```

i.e.: 2 entire nodes, 16 CPU cores on each, the total memory of all nodes >= 120 GB RAM.

Do not request the maximum physical amount of RAM, since the RAM memory available to users is slightly less than the physical RAM memory.

To see the available RAM memory sizes on the different nodes types see the Hardware page.

Waiting for specific jobs

It is possible to specify that a job should only run after another job has completed succesfully, please see the **-W** flags in the **qsub** page.To run <your script> after job 12345 has completed succesfully::

```
$ qsub -W depend=afterok:12345 <your script>
```

Be sure that the exit status of job 12345 is meaningful: if it exits with status 0, you second job will run. If it exits with any other status, you second job will be cancelled.It is also possible to run a job if another job fails (``afternotok``) or after another job completes, regardless of status (``afterany``). Be aware that the keyword ``after`` (as in ``-W depend=after:12345``) means run after job 12345 has *started*.

Submitting jobs to 32-CPU nodes

The quad-processor, 8-core Intel *Sandy Bridge* Xeon E5-4610 v2 nodes (32 CPU cores total) we define to have a node property of **fatnode** (nodes f001-f027). You could submit a batch job like in these examples:: **2** entire fatnodes, **32** CPUs each, total 64 CPU cores

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=2:ppn=32:fatnode <your script>
```

Explicitly the **f015** node, **32** CPU cores:

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=f015:ppn=32 <your script>
```

2 entire fatnodes, **32** CPUs each, memory of all nodes => **1500 GB** RAM)

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=2:ppn=32:fatnode,mem=1500gb <your script>
```

Submitting jobs to 28-CPU nodes

The dual-processor, 14 core Intel E5-2683 v3 (28 CPU cores total) we define to have a node property of **thinnode** (nodes cn001-cn540).You could submit a batch job like in these examples::**2** entire **thinnodes**, **28** CPUs each, total 56 CPU cores)

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=2:ppn=28:thinnode <your script>
```

Explicitly the **cn038** node, **28** CPU cores

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=cn038:ppn=28 <your script>
```

Submitting 1-CPU jobs

You could submit a batch job like in this example:

```
$ qsub -W group_list=<group_NAME> -A <group_NAME> -l nodes=1:ppn=1 <your script>
```

Running parallel jobs using MPI

```
#!/bin/sh
### Note: No commands may be executed until after the #PBS lines
### Account information
#PBS -W group_list=pr_12345 -A pr_12345
### Job name (comment out the next line to get the name of the script used as the job name)
#PBS -N test
### Output files (comment out the next 2 lines to get the job name used instead)
#PBS -e test.err
#PBS -o test.log
### Only send mail when job is aborted or terminates abnormally
#PBS -m n
### Number of nodes, request 196 cores from 7 nodes
#PBS -l nodes=7:ppn=28
### Requesting time - 720 hours
#PBS -l walltime=720:00:00

### Here follows the user commands:
# Go to the directory from where the job was submitted (initial directory is $HOME)
echo Working directory is $PBS_O_WORKDIR
cd $PBS_O_WORKDIR
# NPROCS will be set to 196, not sure if it used here for anything.
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS nodes

module load moab torque openmpi/gcc/64/1.10.2 gromacs/5.1.2-plumed

export OMP_NUM_THREADS=1
# Using 192 cores for MPI threads leaving 4 cores for overhead, '--mca btl_tcp_if_include ib0' forces
InfiniBand interconnect for improved latency
mpirun -np 192 $mdrun -s gmx5_double.tpr -plumed plumed2_path_re.dat -deffnm md-DTU -dlb yes -cpi md-DTU -
append --mca btl_tcp_if_include ib0
```

In order to optimize performance, the queuing system is configured to place jobs on nodes connected to the same InfiniBand switch (30 nodes per switch) if possible.

To get nodes close to each other, use **`procs=<number_of_procs>`** and leave out **node=** and **ppn=**.To avoid interference with other jobs, **`procs=`** should be a multiple of cores per node (ie. 28 for **mpinode**).

Job Arrays

Submitting multiple identical jobs can be done using job arrays. Job arrays can be created by using the -t option in the qsub submission script. The -t option allows many copies of the same script to be submitted at once. Additional information about -t option can be found in the qsub command reference. Moreover, PBS_ARRAYID environmental variable allows to differentiate the different jobs in the array. The amount of resources required in the qsub submission script is the amount of resources that each job will get.
For instance adding the line:

```
  #PBS -t 0-14%5
```

in the qsub script will cause running the job 15 times with not more than 5 actives jobs at any given time.

Please, please, please, use the %# option for limiting the number of active jobs.

PBS_ARRAYID values will run from 0 to 14, as shown below:

```
 ( perl process.pl dataset${PBS_ARRAYID} )

  perl process.pl dataset0
  perl process.pl dataset1
  perl process.pl dataset2
  ….
  perl process.pl dataset14
```

Jobs in jobs array are run independently and not in any specific order.

Monitoring batch jobs

The Torque command qstat is used to inquire about the status of one or more jobs:

```
$ qstat -f <jobid>     (Inquire about a particular jobid)
$ qstat -r             (List all running jobs)
$ qstat -a             (List all jobs)
```

In addition, the Moab scheduler can be inquired using the showq command:

```
$ showq -r             (List all running jobs)
$ showq                (List all jobs)
```

If you want to check the status of a particular jobid use checkjob command:

```
$ checkjob <jobid>
```

`Adding -v flag(s) to this command will increase the verbosity.`

Badly behaving jobs

Another useful command for monitoring batch jobs is pestat, available as a module. Show status of badly behaving jobs, with bad fields marked by star (*)

```
$ module load tools pestat
$ pestat -f
Listing only nodes that are flagged by *
  node state  load    pmem ncpu   mem   resi usrs tasks  jobids/users
  risoe-r01-f002  free     2* 1034109  32 1046397   8017  1/1    1    103125 s147214
  risoe-r01-f010  free  0.53* 1034109  32 1046397   8451  0/0    0
  risoe-r01-f012  free  0.55* 1034109  32 1046397   8019  0/0    0
  risoe-r02-f019  offl* 0.27  1034107  64 1046395   6590  0/0    0
  risoe-r02-f024  free     1* 1034109  32 1046397   8730  0/0    0
  risoe-r03-cn001  excl    29* 128946  28 133042   8266  1/1    1    100096 qyli
...
```

One of the most common bad behaviors of batch jobs is exhausting of available RAM memory.

An example of usage of pestat:

```
$ pestat | grep -e node -e 263945
  node state  load    pmem ncpu   mem   resi usrs tasks  jobids/users
  q008  excl  4.08    7974   4  18628   1275  1/1    4    263945 user
  q037  excl  4.02    7974   4  18628   1285  1/1    4    263945 user
```

The example job above is behaving correctly. Please consult the script located at `which pestat` for the description of the fields. The most important fields are:**state = Torque state (second column)**node can be free (not all the cores used), excl (all cores used) or down.**load = CPU load average (third column)pmem = Physical memory (fourth column)**amount of physical RAM installed in the node**ncpu = total number of CPU cores (fifth column)resi = Resident (used) memory (seventh column)**total memory in use on the given node (the one reported under RES by the "top" command),If used memory exceeds physical RAM on the node, or CPU load is significantly lower than number of CPU cores, the job becomes a candidate to be killed.An example of a job exceeding physical memory:

```
$ pestat -f | grep 128081
m016  busy* 4.00    7990   4  23992   9937* 1/1    4    128081 user
m018  excl  4.00    7990   4  23992   9755* 1/1    4    128081 user
```

An example of a job with incorrect CPU load:

```
$ pestat -f| grep 129284
a014  excl  7.00*  24098   8  72097   2530  1/1    8    129284 user
```

Searching for free resources

Show what resources are available for immediate use (see `Batch_jobs#batch-job-node-properties`_ for more options):Fatnode:

```
$ showbf -f fatnode
```

Thinnode:

```
$ showbf -f fatnode
```

pestat can also be used to check what resources are free:

```
$ pestat | grep free
  risoe-r01-f006  free    29* 1034109  32 1046397  13226  1/1    1    100074 qyli
  risoe-r01-f010  free   2.4* 1034109  32 1046397  79972  2/1    1*   20078 jogon 20079 jogon
  risoe-r01-f013  free  0.84  1034109  32 1046397   8395  0/0    1    102268 jensf
  risoe-r02-f015  free  0.81  1034109  32 1046397   8212  0/0    1    102268 jensf
  risoe-r02-f017  free  0.15* 1034109  32 1046397   8489  0/0    1    102268 jensf
  risoe-r02-f023  free  0.56  1034109  32 1046397   8313  0/0    1    102268 jensf
  risoe-r02-f024  free  0.08* 1034109  32 1046397   8101  0/0    1    102268 jensf
  risoe-r02-f025  free  0.02* 1034109  32 1046397   7984  0/0    1    102268 jensf
  risoe-r08-cn289  free   1.4  128946  28 133042   3117  1/1    1    102536 csabai
  risoe-r08-cn300  free   1.5  128943  56 133039   6995  3/2    1*   102406 jensf 102407 jensf 102533 cgar
  risoe-r12-cn527  free   1.3  128946  28 133042   2741  1/1    1    10047 sira
  risoe-r02-f018  free    29* 1034110  64 1046398  15376  1/1    1    99432 qyli
```

The node risoe-r01-f010 is occupied by 1 job (9th column) and two users (8th column) each requesting 1 core. The node risoe-r02-f024 is totally free.

Job control

Some commands only work for privileged accounts. Please contact Computerome support if you need to run these.

Canceling a given job:

Cancel job

```
$ mjobctl -c <jobid>
```

```
$ canceljob <jobid>
```

Force cancel job - try this if regular cancel fails

```
$ mjobctl -F <jobid>
```

Canceling all jobs of a given user (privileged command):

```
# mjobctl -c -w user=<someuser>
```

Re-queue a job (privileged command):

```
# mjobctl -R <jobex>
```

Change walltime (privileged command):

Changing the wallclock limit of a job by 10 hours 11 minutes and 12 seconds (request Computerome Support in good time to extend walltime for running job):

```
# mjobctl -m wclimit+=10:11:12 <jobex>
```

<jobex> is a regex(7) regular expression preceeded by x: e.g. "x:abc12[0-9]"

Get status if fair-share:

```
$ diagnose -f
```

Check resource usage of completed job (privileged command):

```
# tracejob -v <jobid>
```

Check job status:

```
$ checkjob -v <jobid>
```

Check when job will run:

```
$ showstart <jobid>
```

Special Solution for CPR users

Selected CPR data (jensenlab) are available on Computerome HPC nodes for selected users. In order to get access to these data, please send an aproved request to BIO-HPC@bio.dtu.dk for adding your user to the group jensenlab. Once we have confirmed that your user is member of jensenlab, every time you submit a job, jensenlab data will be mounted on the compute node on /mnt/$PBS_JOBID. PBS_JOBID is a variable that contains the JOB id while the job exists. Moreover you will be prompted with information on where to find the data. See the example below:

```
pazthy@computerome02 ~]$ qsub -I
qsub: waiting for job 17022503.risoe-r04-sn064.cm.cluster to start
qsub: job 17022503.risoe-r04-sn064.cm.cluster ready

*** INFO ***
jensenlab data are  mounted on /mnt/17022503.risoe-r04-sn064.cm.cluster
You can use /mnt/$PBS_JOBID as PATH to the data
*** INFO ***

Please use the following directories for your temp file
/scratch/$PBS_JOBID/ - Physical
/tmp/$PBS_JOBID/     - Memory (up to 50% of avalible memory)

[pazthy@risoe-r05-cn141 ~]$ ls -l /mnt/$PBS_JOBID
total 8
drwxr-xr-x  5 jensenlab jensenlab 4096 Jan 31 17:47 data
drwxr-xr-x 15 root      jensenlab 4096 Aug 30 14:34 home
```