Using the RHUL-Psychology scheduling system

Tibor Auer

RHUL Department of Psychology

Research Fellow in MRI

Courtesy to Russel Thompson (MRC-CBSU)

Overview

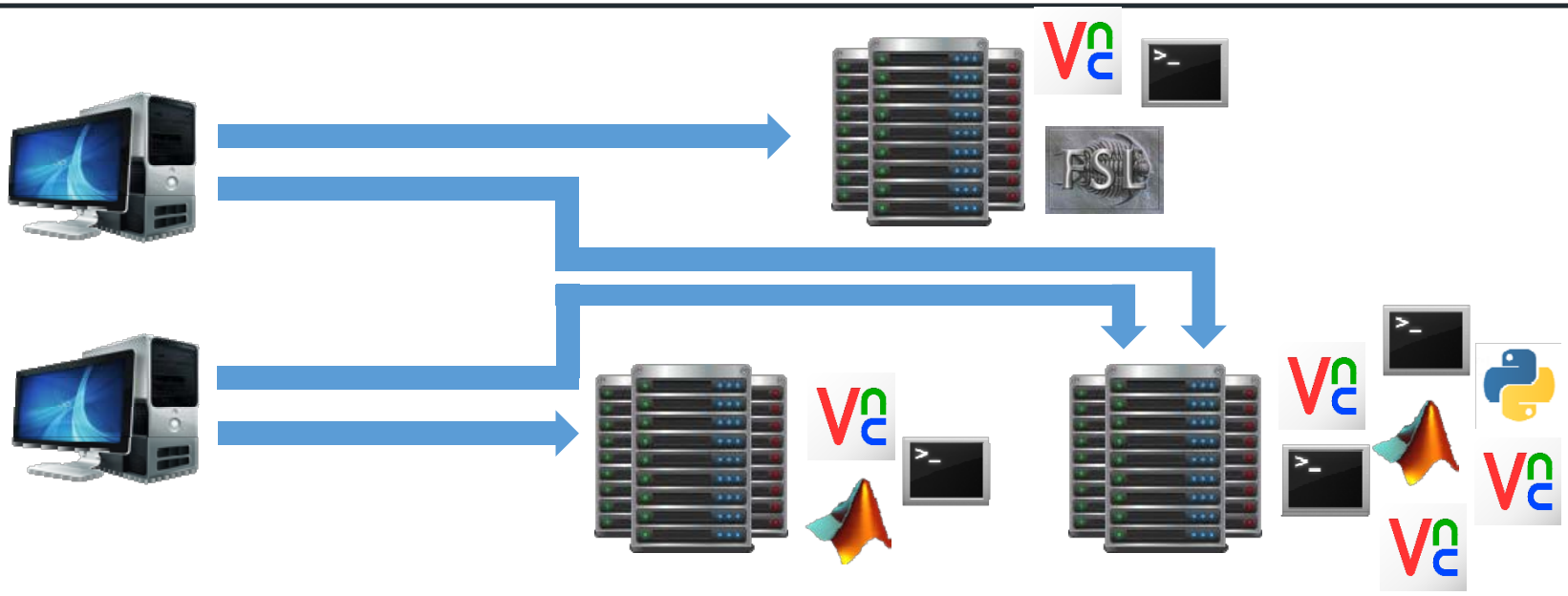
- Why use a scheduling system?
- Submitting jobs
- Monitoring and managing jobs
- MATLAB
- Best Practices

Overview

- **Why use a scheduling system?**
- Submitting jobs
- Monitoring and managing jobs
- MATLAB
- Best Practices

Why use a scheduling system?

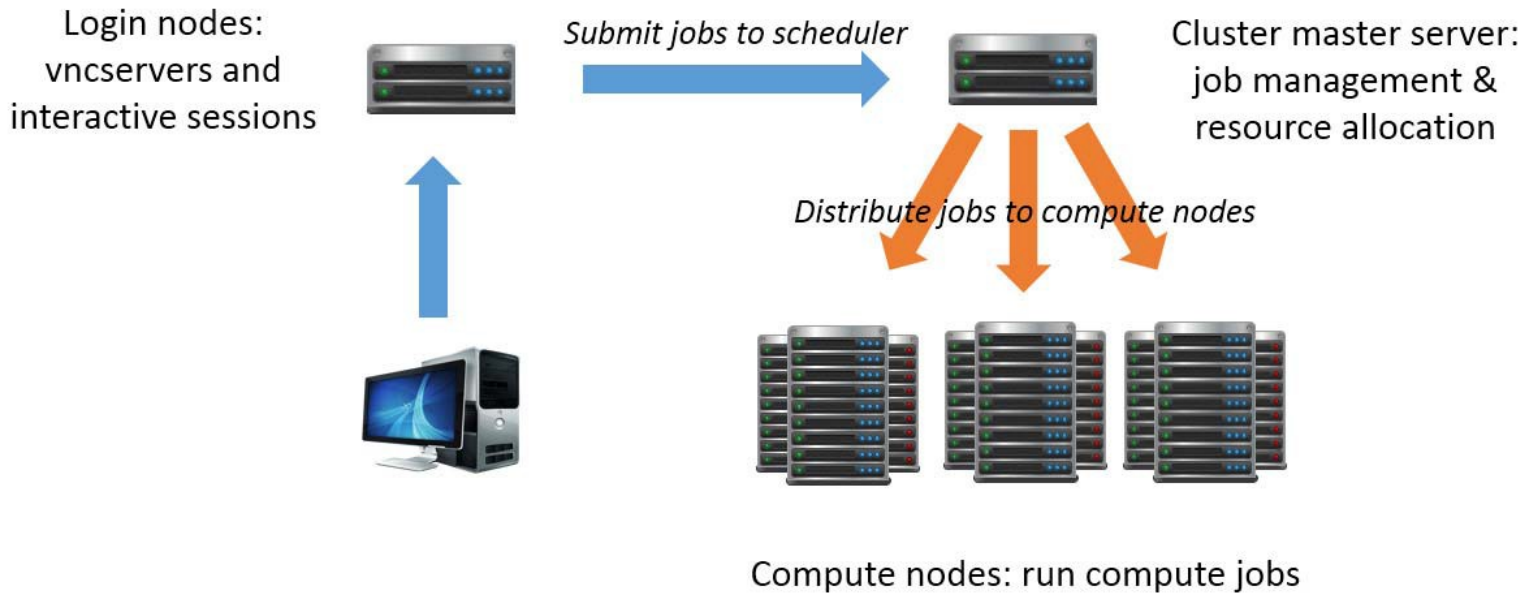
Non-scheduled system (old cluster)



- **Issues**
  - No distinction between login and compute nodes
    - → run both interactive sessions and large compute jobs on the same machines
  - No management of jobs → e.g. machines can run out of memory

Why use a scheduling system?

Scheduled system (new cluster)



- **Solutions**
  - Distinction between login and compute nodes
    - Login and run interactive sessions on a login node
    - Run large compute jobs on compute nodes
  - Scheduling system manages allocation of resources to compute jobs

Why use a scheduling system?

Scheduled system (new cluster)

- **Resource manegement**
  - Dedicated resources for jobs → running jobs do not compete for the same resources
  - Resources are fully utilized, but not overloaded
- **Parallel processing (saving time)**
  - „Overt” parallel jobs: e.g. processing different subjects in multiple commands
  - „Covert” parallel jobs: e.g. a single command launches parallel processing

Overview

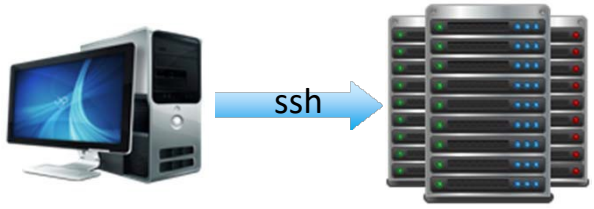
- Why use a scheduling system?
- Submitting jobs
- Monitoring and managing jobs
- MATLAB
- Best Practices

Submitting jobs

Accessing

1. Log in using SSH[1] (text only)

- Linux: *ssh*
- Win: PuTTY




```
login as: vwxy123
vwxz123@psyclogin.rhul.ac.uk's password:
[vwxz123@psyclogin ~]$
```

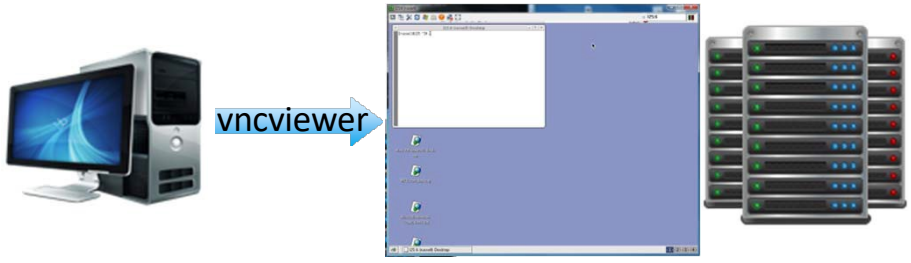
2. Graphical sessions via VNC[2]

- Check: *ps -ef | grep <user name>.*Xvnc | grep -v grep | awk '{print $9}'*

```
[vwxy123@psyclogin ~]$ ps -ef | grep vwxy123.*Xvnc | grep -v grep | awk '{print $9}'
:7
```

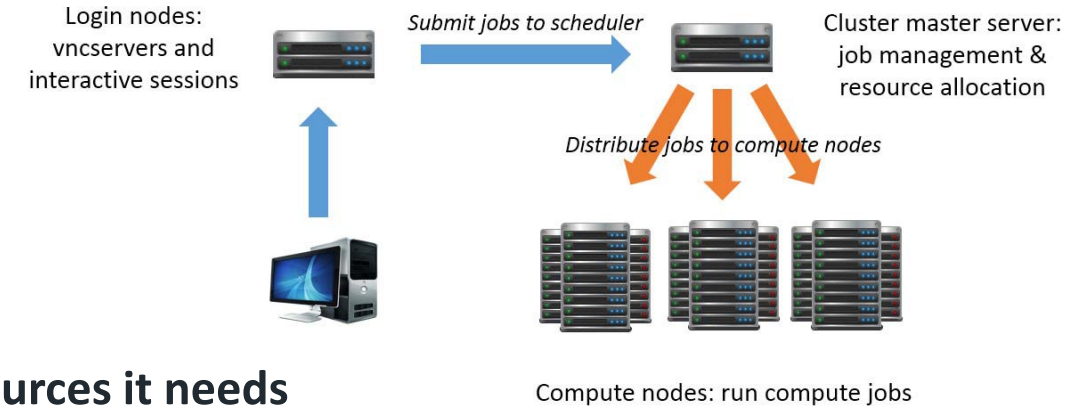
3. Win: TurboVNC Viewer

- psyclogin.rhul.ac.uk:7



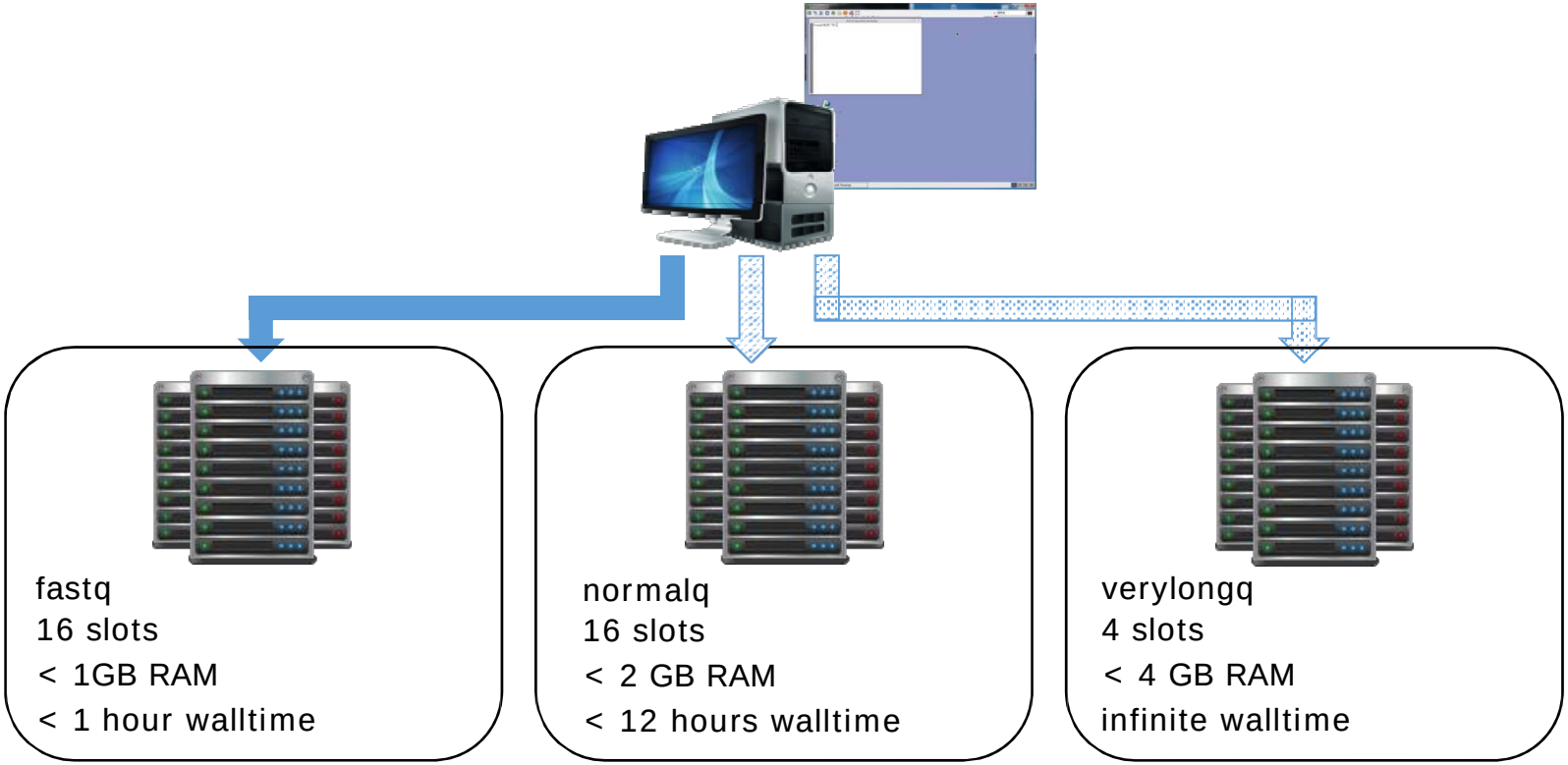
Submitting jobs

- **Workflow**
  1. **Log in to login node via VNC**
  2. **Create a script to run your analyses**
  3. **Test the script and determine what resources it needs (esp. run time and perhaps RAM)**
  4. **Submit the script(s) to the scheduling system and request specific cluster resources**
     - Number of cores (typically 1 per job), RAM
     - Processing time („walltime”)
       - After the walltime elapses, the job is killed.
       - If the job finishes earlier, resources will be released.



Submitting jobs

- **Queues**
  - A job is held in a queue and is executed as soon as the requested resources are available.
  - Queues are associated with specific sets of resources
    - e.g. maximum RAM, maximum walltime, etc.
  - **Choose a queue based on resource requirements!**



Submitting jobs

- **Queues (http://www.cubic.rhul.ac.uk/wiki/doku.php?id=cluster:cluster_configuration#queues):**

Name	Priority	Slots	CPU Time	Memory
fastq	1	16	1h	1GB
normal	5	16	12h	2GB
longq	10	16	24h	4GB
verylongq	15	4	INFINITE	4GB

  - One can run up to 32 jobs only (also subject to availability).
  - Can submit more jobs, but they remain at the bottom of the queue.

Submitting jobs

qsub

- **Submit jobs in linux terminal:**

```
qsub <arguments> <command to run>
```

  - Arguments:
    - -N job name
    - -q which queue to use
    - -l request particular resource and select queue accordingly
    - -o path to file where standard output should be redirected
    - -e path to file where standard error output should be redirect
    - -V pass environment (e.g. FSL configuration) to worker
    - -v variable to pass to job batch script
    - -t submit a job array and set SGE_TASK_ID in each instances
    - -b submit binary rather than script
  - E.g.:

```
qsub -q fastq -V -v \
        RAWDIR=/…/201706131237_19810218EIJO/Series_002_MPRAGE,\
        OUTDIR= /MRIWork/Workshop/Train01/cluster/bet,\
        SUBJID=1 \
        /MRIWork/Workshop/Material/1_Cluster/bet_script.sh
```

Submitting jobs

qsub

- **Submit jobs in linux terminal:**

```
qsub -q fastq -V -v \
        RAWDIR=/…/201706131237_19810218EIJO/Series_002_MPRAGE,\
        OUTDIR= /MRIWork/Workshop/Train01/cluster/bet,\
        SUBJID=1 \
        /MRIWork/Workshop/Material/1_Cluster/bet_script.sh
```

- Content of bet_script.sh

```
FNAME=S$(printf "%02d" $SUBJID)
RAWFILE1=${RAWDIR}/$(echo $(ls ${RAWDIR}) | awk '{print $1}')

mri_convert ${RAWFILE1} ${OUTDIR}/${FNAME}.nii.gz > ${OUTDIR}/log_convert.txt
fslswapdim ${OUTDIR}/${FNAME} z -x -y ${OUTDIR}/${FNAME} >> ${OUTDIR}/log_convert.txt
bet ${OUTDIR}/${FNAME} ${OUTDIR}/${FNAME}_brain -R -v;
```

Submitting jobs

Multiple jobs (Job arrays)

- **Submit jobs in linux terminal:**

```
qsub -q fastq -V -t 1-2 -v \
        RAWDIR=/…/201706131237_19810218EIJO/Series_002_MPRAGE,\
        OUTDIR= /MRIWork/Workshop/Train01/cluster/bet,\
        /MRIWork/Workshop/Material/1_Cluster/bet_script_array.sh
```

- Content of bet_script_array.sh

```
FNAME=S$(printf "%02d" $SGE_TASK_ID)
RAWFILE1=${RAWDIR}/$(echo $(ls ${RAWDIR}) | awk '{print $1}')

mri_convert ${RAWFILE1} ${OUTDIR}/${FNAME}.nii.gz > ${OUTDIR}/log_convert.txt
fslswapdim ${OUTDIR}/${FNAME} z -x -y ${OUTDIR}/${FNAME} >> ${OUTDIR}/log_convert.txt
bet ${OUTDIR}/${FNAME} ${OUTDIR}/${FNAME}_brain -R -v;
```

Overview

- Why use a scheduling system?
- Submitting jobs
- Monitoring and managing jobs
- MATLAB
- Best Practices

Monitoring jobs

qstat

- **For information about the cluster, queues, jobs:**

```
qstat <options>
```

  - Arguments:
    - -g c — cluster queue summary
    - -q <queue name> — selected queue
    - -f — full output
    - -F [resource] — full output and show (selected) resources
    - -u <user ID> — jobs of selected users
    - -j <job ID> — selected job

Monitoring jobs

qstat

- **E.g.: Queues**
  - List queues

```
[udjt005@psyclogin cluster]$ qstat -g c
CLUSTER QUEUE                   CQLOAD   USED    RES  AVAIL  TOTAL aoACDS  cdsuE
--------------------------------------------------------------------------------
fastq                             0.01      0      0     16     16      0      0
longq                             0.01      0      0     16     16      0      0
normalq                           0.01      0      0     16     16      0      0
verylongq                         0.01      0      0      4      4      0      0
```

  - Instances of „fastq” queue

```
[udjt005@psyclogin cluster]$ qstat -q fastq -f
queuename                       qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
fastq@psycmri01.rhul.ac.uk      BP    0/0/8          0.09     lx-amd64
---------------------------------------------------------------------------------
fastq@psycmri02.rhul.ac.uk      BP    0/0/8          0.05     lx-amd64
```

Monitoring jobs

qstat

- **E.g.: Queues**
  - Requestable resources of „fastq” queue

```
[udjt005@psyclogin cluster]$ qstat -q fastq -F
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
fastq@psycmri01.rhul.ac.uk     BP    0/0/8          0.07     lx-amd64
    qf:h_cpu=01:00:00
    qf:h_rss=1.000G

    hl:mem_total=31.357G
    hl:swap_total=6.000G
    hl:virtual_total=37.357G

    hl:mem_free=30.631G
    hl:swap_free=5.906G
    hl:virtual_free=36.537G

    hl:mem_used=743.750M
    hl:swap_used=95.770M
    hl:virtual_used=839.520M
    ...
```

Monitoring jobs

qstat

- **E.g.: Queues**
  - Requestable walltime of all queue instances

```
[udjt005@psyclogin cluster]$ qstat -F h_cpu
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
fastq@psycmri01.rhul.ac.uk     BP    0/0/8          0.19     lx-amd64
    qf:h_cpu=01:00:00
---------------------------------------------------------------------------------
fastq@psycmri02.rhul.ac.uk     BP    0/0/8          0.06     lx-amd64
    qf:h_cpu=01:00:00
---------------------------------------------------------------------------------
normalq@psycmri01.rhul.ac.uk   BP    0/0/8          0.19     lx-amd64
    qf:h_cpu=12:00:00
---------------------------------------------------------------------------------
normalq@psycmri02.rhul.ac.uk   BP    0/0/8          0.06     lx-amd64
    qf:h_cpu=12:00:00
---------------------------------------------------------------------------------
longq@psycmri01.rhul.ac.uk     BP    0/0/8          0.19     lx-amd64
    qf:h_cpu=24:00:00
---------------------------------------------------------------------------------
longq@psycmri02.rhul.ac.uk     BP    0/0/8          0.06     lx-amd64
    qf:h_cpu=24:00:00
...
```

Monitoring jobs

qstat

- **E.g.: Jobs**
  - All jobs of user abcd123[1]

```
[udjt005@psyclogin cluster]$ qstat -u abcd123
job-ID  prior   name       user         state submit/start at     queue slots ja-task-ID
------------------------------------------------------------------------------------
   5167 0.05500 Job1       abcd123      Eqw   08/25/2017 12:15:13          12
```

Monitoring jobs

qstat

- **E.g.: Particular job**
  - Details of job 5167

```
[udjt005@psyclogin cluster]$ qstat -j 5167
==============================================================
job_number:                 5167
submission_time:            Fri Aug 25 12:15:13 2017
sge_o_workdir:              /…/MVPA_Results
hard resource_list:         h_cpu=7200
stdout_path_list:           NONE:NONE: /…/MVPA_Results/Job1/Job1.log
script_file:                /…/matlab/R2015b/toolbox/local/communicatingJobWrapper.sh
error reason          1:    08/25/2017 13:31:26 [795555711:19137]: error: can't open
output file "/…/MVPA_Results/Job1/Job1.log": Stale file handle
scheduling info:            Job is in error state
...
```

Monitoring jobs

Output and error messages

- **Execution of a program locally (in a terminal)**

```
bet \
/MRIWork/Workshop/Train01/cluster/bet/S01.nii.gz \
/MRIWork/Workshop/Train01/cluster/bet/S01_brain.nii.gz –R -v
```

  - Stderr in case of error (also in the terminal)

```
bash: bet: command not found
```

  - Stdout after fsl configured (also in the terminal)

```
IN=/MRIWork/Workshop/Train01/cluster/bet/S01
OUT=/MRIWork/Workshop/Train01/cluster/bet/S01_brain
...
c-of-g 93.8159 115.863 164.951 mm
radius 100.988 mm
median within-brain intensity 265
self-intersection total 58.35 (threshold=4000.0)
```

Monitoring jobs

Output and error messages

- **Execution of a program as job (on the cluster)**

```
qsub -q fastq \
    -V -v INPUT=/MRIWork/Workshop/Train01/cluster/bet/S01.nii.gz \
    /MRIWork/Workshop/Material/1_Cluster/bet_qsub.sh
```

  - Jobs running on the cluster are not interactive, i.e. outputs are not sent to the terminal but redirected to files saved in home by default.
    - Stdout: <home>/<script name>.o<job ID>[.<task ID>][1]
    - Stderr: <home>/<script name>.e<job ID>[.<task ID>][1]

  - E.g.:
    - Content of bet_qsub.e6055

```
/usr/local/share/sge/default/spool/psycmri02/job_scripts/5038: line 2: bet:
command not found
```

Monitoring jobs

Output and error messages

- **Execution of a program as job (on the cluster)**

```
qsub -q fastq \
    -V -v INPUT=/MRIWork/Workshop/Train01/cluster/bet/S01.nii.gz \
    /MRIWork/Workshop/Material/1_Cluster/bet_qsub.sh
```

  - Jobs running on the cluster are not interactive, i.e. outputs are not sent to the terminal but redirected to files saved in home by default.
    - Stdout: <home>/<script name>.o<job ID>[.<task ID>][1]
    - Stderr: <home>/<script name>.e<job ID>[.<task ID>][1]

  - E.g.:
    - Content of bet_qsub.o6056

```
IN=/MRIWork/Workshop/Train01/cluster/bet/S01
OUT=/MRIWork/Workshop/Train01/cluster/bet/S01_brain
…
c-of-g 93.8159 115.863 164.951 mm
radius 100.988 mm
median within-brain intensity 265
self-intersection total 58.35 (threshold=4000.0)
```

Monitoring jobs

Output and error messages

- **Execution of a program as job (on the cluster)**

```
qsub -q fastq \
    -V -v INPUT=/MRIWork/Workshop/Train01/cluster/bet/S01.nii.gz \
    /MRIWork/Workshop/Material/1_Cluster/bet_qsub.sh
```

  - Jobs running on the cluster are not interactive, i.e. outputs are not sent to the terminal but redirected to files saved in home by default.
    - Stdout: <home>/<script name>.o<job ID>[.<task ID>][1]
    - Stderr: <home>/<script name>.e<job ID>[.<task ID>][1]

  - Can specify output locations

```
outdir=/MRIWork/Workshop/Train01/cluster/bet
qsub -q fastq \
    -o ${outdir}/jobbet_out.txt -e ${outdir}/jobbet_err.txt
    -V -v INPUT=/MRIWork/Workshop/Train01/cluster/bet/S01.nii.gz \
    /MRIWork/Workshop/Material/1_Cluster/bet_qsub.sh
```

Manging jobs

- **Rerun job with error[1]**

```
qmod -cj <job id>
```

- **Delete jobs**

```
qdel <job id>
```

  - Can delete all jobs at once

```
qselect -u `whoami` | xargs qdel
```

Overview

- Why use a scheduling system?
- Submitting jobs
- Monitoring and managing jobs
- MATLAB
- Best Practices

Submitting MATLAB jobs

- **MATLAB Distributed Computing Server (DCS) and Parallel Computing Toolbox (PCT):**
  - Collection of functions for running matlab jobs in parallel environment
  - DCS supports 3rd party schedulers at various degree (PBS/TORQUE – well, SGE – limited)
  - MATLAB translates jobs into a series of qsub commands with options according to the configuration

- **Workflow**
  1. Create analysis script
  2. Create scheduler object using DCS/PCT functions
  3. Configure scheduler (log file location, walltime, memory)
  4. Create jobs to run
  5. Submit jobs

- **Options**
  - MATLAB PCT functions with RHUL-Psycho wrapper
  - parfor loops with RHUL-Psycho cluster profiler (you can use the wrapper, too)
  - **Automatic Analysis**

Submitting MATLAB jobs

RHUL-Psycho wrapper

- **PsychoSGE constructor**
  - In /usr/local/apps/psycapps/cluster/PsychoSGE.m
  - Provides
    - Sets up PsychoSGE cluster profile[1]
    - Configures cluster (logfile location, walltime, memory)
    - Create pool for parfor

```
%% Config
C = PsychoSGE;
% Defaults
% C.Walltime = 2;   % Hours
% C.Memory = 2:     % GB
% C.Logs = fullfile(pwd,'logs');

cluster = C.getCluster;

pool = C.getPool(8); % 8 jobs in parallel
```

Submitting MATLAB jobs

RHUL-Psycho wrapper

- **Example:**
  - Calculate eigenvalues of a square matrix of random numbers
  - Serial execution

```
N = 2000;

%%
for j = 1:8
    out(:,j) = eig(rand(N));
end
```

Submitting MATLAB jobs

RHUL-Psycho wrapper

- **Example:**
  - Calculate eigenvalues of a square matrix of random numbers
  - Simple job submission

```
N = 2000;

C = PsychoSGE;
C.Walltime = 0.5;   % Hours
C.Memory = 1;       % GB
C.Logs = fullfile(pwd,'logs');
cluster = C.getCluster;

for j = 1:8
    job(j) = cluster.createJob;
    job(j).createTask(@eig, 1, {rand(N)});
    job(j).submit;
end

while ~all(strcmp({job.State},'finished')), pause(1); end

for j = 1:8
    out(j) = job(j).fetchOutputs;
end
```

Create and configure the cluster

Create and submit jobs and tasks

Wait for all jobs to finish

Retrieve output(s)

Submitting MATLAB jobs

RHUL-Psycho wrapper

- **Example:**
  - Calculate eigenvalues of a square matrix of random numbers
  - Simple job submission
    - Create and submit jobs and tasks

```
for j = 1:8
    job(j) = cluster.createJob;
    job(j).createTask(@eig, 1, {rand(N)});
    job(j).submit;
end
```

    - *createTask(F, N, {inputs})*
      - F — A handle to the function that is called.
      - N — Number of output arguments to be returned.
      - {inputs} — A row cell array specifying the input arguments. Each element in the cell array will be passed as a separate input argument.

Submitting MATLAB jobs

RHUL-Psycho wrapper

- **Example:**
  - Calculate eigenvalues of a square matrix of random numbers
  - Simple job submission
    - Wait for all jobs to finish

```
while ~all(strcmp({job.State},'finished')), pause(1); end
```

    - *job.State*: status of the job (*'pending', 'queued', 'running', 'finished', 'failed'*)
    - Retrieve output(s)

```
for j = 1:8
    out(j) = job(j).fetchOutputs;
end
```

    - *out = job.fetchOutputs*
      - Returns the result(s) from the tasks of a finished job.
      - *out* is an M-by-N cell array, where M is the number of tasks (typically 1), while N is the greatest number of output arguments from any one task

Submitting MATLAB jobs

RHUL-Psycho wrapper

- **Example:**
  - Calculate eigenvalues of a square matrix of random numbers (with benchmarking)
  - Parfor

```
N = 2000;

C = PsychoSGE;
C.Walltime = 0.5;   % Hours
C.Memory = 1;       % GB
C.Logs = fullfile(pwd,'logs');
pool = C.getPool(8);

parfor j = 1:8
    out(:,j) = eig(rand(N));
end

C.closePool;
```

Create and configure the pool

Run usual functions[1]

Close pool

Overview

- Why use a scheduling system?
- Submitting jobs
- Monitoring and managing jobs
- MATLAB
- Best Practices

Best Practices

- **Write and debug your scripts locally before submitting!**

- **Requesting the appropriate resources allows the scheduling system to operate most efficiently.**
  - Make a note of the required resources (top) – especially memory and CPU time

  - Under-requesting (e.g. 1GB RAM when you need 4GB) can cause the job to crash.
  - Over-requesting (e.g. 4GB RAM when you only need 1GB) means
    - Fewer jobs will run simultaneously.
    - Jobs may wait for longer for resources.

Further Information

- **Cluster wiki on the CUBIC website**
  http://www.cubic.rhul.ac.uk/wiki/doku.php?id=cluster:cluster_root