A brief HPC tutorial

Part II

Dr. O. Melchert

Institut für Physik
Universität Oldenburg

Part II: Content

- Introduction to the usage of SGE
  1. Introduction
  2. General Job submission
  3. Single Slot jobs
  4. Parallel Jobs
  5. Monitoring and Controlling jobs
- Debugging and Profiling:
  1. Compiling programs for debugging
  2. Tracking memory issues
  3. Profiling
- Misc:
  1. Logging in from outside the university
  2. Mounting the HPC home directory
  3. Parallel environment memory issue
  4. Importance of allocating proper resources

Introduction to the usage of SGE

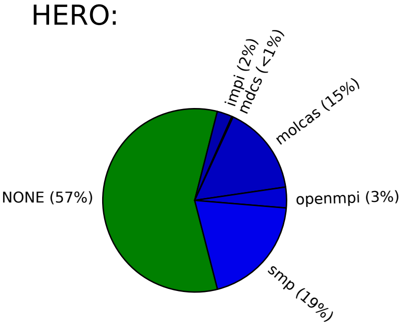
- Sun Grid Engine (SGE):
  - batch scheduler that handles workload on HPC system
  - enables optimal sharing of HPC resources between users
- Heterogeneous user community:
  - 171 active users (1/3 FLOW, 2/3 HERO)
  - 34 different working groups (from faculties 2, 5, 6)
  - different users, different needs
- How SGE operates:
  - accepts jobs (i.e. requests for computing resources)
  - places jobs in queue until they can be run
  - sends jobs from queue to execution hosts
  - manages running jobs
  - logs details of finished jobs

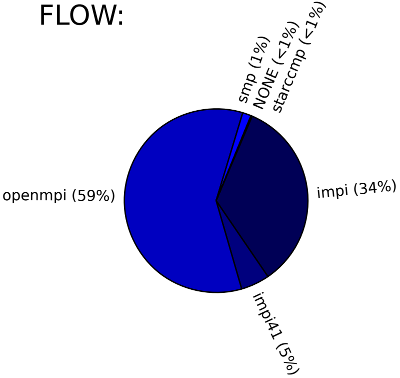
Introduction to the usage of SGE

- Here: SGE serves many users with different needs (particularly true for HERO)
- From a general point of view, SGE takes care of:
  - **Scheduling**: handles execution of large number of jobs
  - **Load balancing**: takes care that nodes not overloaded
  - **Monitoring/accounting**: clarify job state / job history
- SGE provides easy to use commands:
  - `qconf` - examine SGE configuration
  - `qsub` - submit your job to the scheduler
  - `qstat` - monitor status of queued jobs
  - `qacct` - retrieve details for finished jobs
  - `qrsh` - request interactive sessions
  - `qdel`, `qalter` - delete and alter jobs

Introduction to the usage of SGE

- Different users have different needs:
  - pivotal decision: which parallel environment (PE) to use?
  - PE reflects kind of application you submit
  - get list of PEs: `qconf -spl` (s=show, p=PE, l=list)
  - accumulated running time spent PEs (since July 2011):





- Next: consider the PEs `NONE`, `smp`, `openmpi` in detail

Usage of SGE: General job submission

- Before job submission you might want to
  - *compile* your program
  - perform several *debugging/profiling* cycles
  - perform small *test runs*

  → can all be done on your local workstation

- How to submit a job?
  - `qsub` - SGE provided command to submit jobs
  - submit job via user supplied job *submission script*
  - job submission script details resource requirements

  → only possible from dedicated submission host

- Possible submission hosts:
  - e.g. `hero01`, `hero02`, `flow01`, `flow02`
  - logon to submission host via (from within the university):

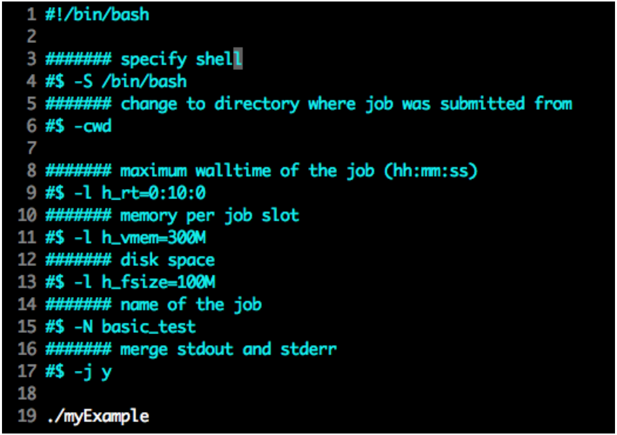
    ```
    ssh abcd1234@hero/flow.hpc.uni-oldenburg.de
    ```

  → compile/submit your programs here

Usage of SGE: Single slot job

- Example: simple single slot job (PE: `NONE`)

  `submissionScript.sge`:



```
#!/bin/bash

####### specify shell
#$ -S /bin/bash
####### change to directory where job was submitted from
#$ -cwd

####### maximum walltime of the job (hh:mm:ss)
#$ -l h_rt=0:10:0
####### memory per job slot
#$ -l h_vmem=300M
####### disk space
#$ -l h_fsize=100M
####### name of the job
#$ -N basic_test
####### merge stdout and stderr
#$ -j y

./myExample
```

submit via `qsub submissionScript.sge`

User Wiki: `Main Page > Brief Introduction to HPC Computing > 1.1`

Usage of SGE: Requestable resources

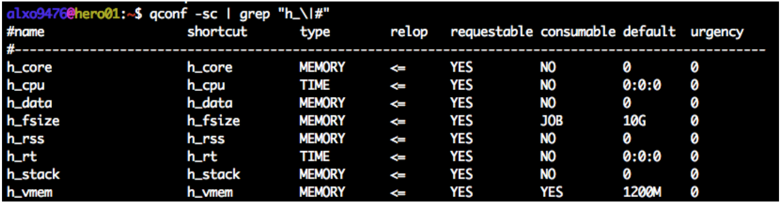
- Central HPC Mantra:
  - User: specify resource requirements (as part of job)
  - SGE: matches available resources to requests

  → effectively, SGE assigns job to fitting *queue*
  (obtain list of queues via `qconf -sql`)

- Note: resources have meaningful default values, e.g.
  - default scratch space requirement: `h_fsize=10G`
  - default memory requirement: `h_vmem=1200M`

- List *complex configuration* via `qconf -sc`

  → examine details for all requestable resources

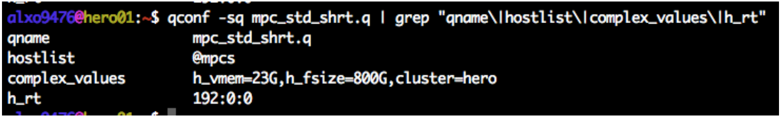


```
alxo9476@hero01:~$ qconf -sc | grep "h_\|#"
#name                 shortcut      type        relop   requestable consumable default  urgency
#----------------------------------------------------------------------------------------------
h_core                h_core        MEMORY      <=      YES         NO         0        0
h_cpu                 h_cpu         TIME        <=      YES         NO         0:0:0    0
h_data                h_data        MEMORY      <=      YES         NO         0        0
h_fsize               h_fsize       MEMORY      <=      YES         JOB        10G      0
h_rss                 h_rss         MEMORY      <=      YES         NO         0        0
h_rt                  h_rt          TIME        <=      YES         NO         0:0:0    0
h_stack               h_stack       MEMORY      <=      YES         NO         0        0
h_vmem                h_vmem        MEMORY      <=      YES         YES        1200M    0
```

User Wiki: `Main Page > On Queues and Resource allocation > 3`

Usage of SGE: Requestable resources

- resource limits for jobs to `mpcs` execution hosts:
  - job needs more than `h_rt=192:0:0`
    → request long run: `-l longrun=True`
  - job needs more than `h_fsize=800G` or `h_vmem=23G`
    → request high mem node: `-l bignode=True`
- Different queues respect different resources limits
  - consider e.g. *short* queue on standard nodes:



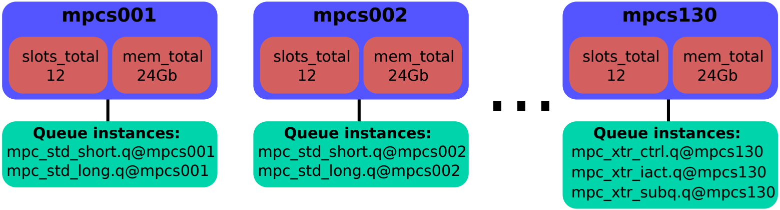
```
alxo9476@hero01:~$ qconf -sq mpc_std_shrt.q | grep "qname\|hostlist\|complex_values\|h_rt"
qname                 mpc_std_shrt.q
hostlist              @mpcs
complex_values        h_vmem=23G,h_fsize=800G,cluster=hero
h_rt                  192:0:0
```

→ queue selection handled “behind the SGE curtain”

Don’t target queues, specify fitting resources!

Usage of SGE: Queue instances

- Execution hosts feature several *queue instances*
  - queue instances jointly consume memory and slots on host
  - consider e.g. standard nodes on HERO:



- Resource allocation statements determine fitting queue(s)
- How does SGE actually allocate the jobs? → later

Usage of SGE: Single slot job

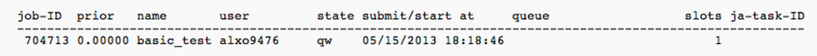
Submitting a job:

- enqueue job via `qsub submissionScript.sge`
- job gets unique `jobId`
- `jobId` can be used to monitor job status

User Wiki: `Main Page > Brief Introduction to HPC Computing > 1.1.2`

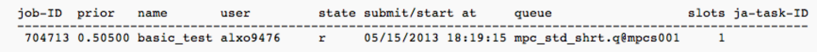
Checking status of job:

- monitor job status using `qstat -j <jobId>`



```
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
 704713 0.00000 basic_test alxo9476     qw    05/15/2013 18:18:46                                    1
```

somewhat later:



```
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
 704713 0.50500 basic_test alxo9476     r     05/15/2013 18:19:15 mpc_std_shrt.q@mpcs001             1
```

User Wiki: `Main Page > Brief Introduction to HPC Computing > 1.1.4`

Retrieve details for finished jobs:

- filter *accounting file* via `qacct -j <jobId>`

User Wiki: `Main Page > Brief Introduction to HPC Computing > 1.1.5`

Usage of SGE: Altering resource requirements

Consider job, initially submitted with non-adequate resources. You have two options:

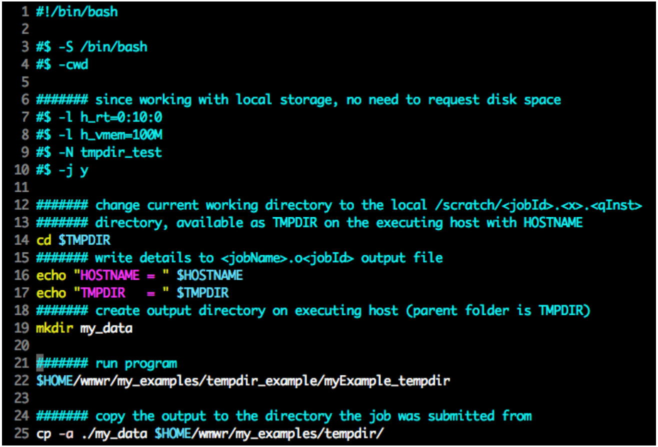
- delete job, amend submission script and resumbit
  - `qdel` - SGE command to delete jobs
  - usage: `qdel <jobId>`
- alter resources (no deletion needed):
  - `qalter` - SGE command to modify resource list
  - usage: `qalter -l h_vmem=2G -l h_fsize=10G -l h_rt=1:00:0 <jobId>`

→ Note: `qalter` overwrites resource list, hence all resource keywords need to be specified

User Wiki: `Main Page > Brief Introduction to HPC Computing > 1.1.3`

Usage of SGE: Single slot job

- Example: I/O intense single slot job (PE: `NONE`) `submissionScript_tempDir.sge`:



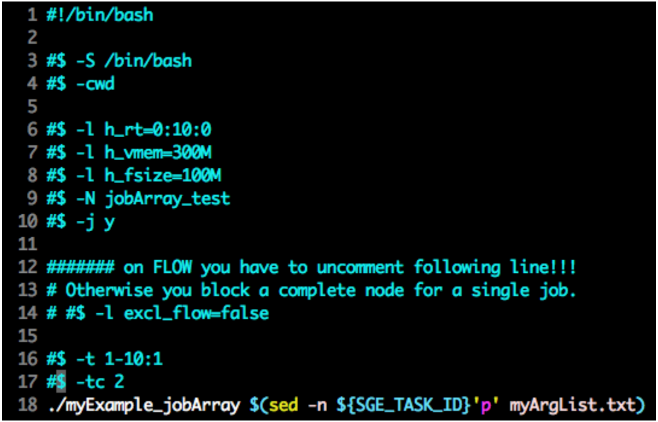
```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd

####### since working with local storage, no need to request disk space
#$ -l h_rt=0:10:0
#$ -l h_vmem=100M
#$ -N tmpdir_test
#$ -j y

####### change current working directory to the local /scratch/<jobId>.<x>.<qInst>
####### directory, available as TMPDIR on the executing host with HOSTNAME
cd $TMPDIR
####### write details to <jobName>.o<jobId> output file
echo "HOSTNAME = " $HOSTNAME
echo "TMPDIR   = " $TMPDIR
####### create output directory on executing host (parent folder is TMPDIR)
mkdir my_data

####### run program
$HOME/wmwr/my_examples/tempdir_example/myExample_tempdir

####### copy the output to the directory the job was submitted from
cp -a ./my_data $HOME/wmwr/my_examples/tempdir/
```

submit via `qsub submissionScript_tempDir.sge`

User Wiki: `Main Page > Brief Introduction to HPC Computing > 1.2`

Usage of SGE: Single slot job

- Example: single slot job-array job
  `submissionScript_jobArray.sge`:



```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd

#$ -l h_rt=0:10:0
#$ -l h_vmem=300M
#$ -l h_fsize=100M
#$ -N jobArray_test
#$ -j y

####### on FLOW you have to uncomment following line!!!
# Otherwise you block a complete node for a single job.
# #$ -l excl_flow=false

#$ -t 1-10:1
#$ -tc 2
./myExample_jobArray $(sed -n ${SGE_TASK_ID}'p' myArgList.txt)
```

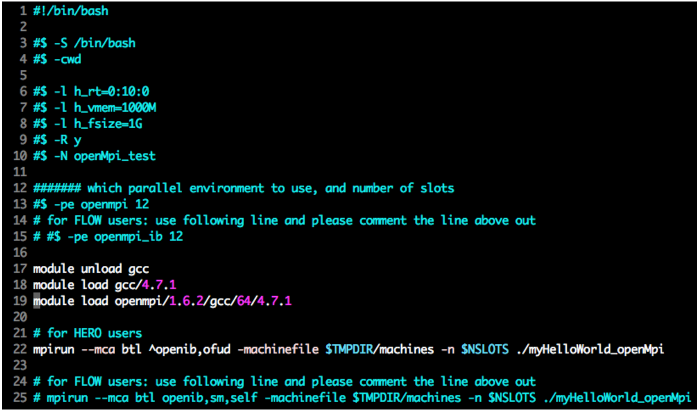
submit via `qsub submissionScript_jobArray.sge`

User Wiki: `Main Page > Brief Introduction to HPC Computing > 1.3`

Usage of SGE: Parallel job

- Example: parallel job using `openMpi`

  `submissionScript_openMpi.sge`:



```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd

#$ -l h_rt=0:10:0
#$ -l h_vmem=1000M
#$ -l h_fsize=1G
#$ -R y
#$ -N openMpi_test

###### which parallel environment to use, and number of slots
#$ -pe openmpi 12
# for FLOW users: use following line and please comment the line above out
# #$ -pe openmpi_ib 12

module unload gcc
module load gcc/4.7.1
module load openmpi/1.6.2/gcc/64/4.7.1

# for HERO users
mpirun --mca btl ^openib,ofud -machinefile $TMPDIR/machines -n $NSLOTS ./myHelloWorld_openMpi

# for FLOW users: use following line and please comment the line above out
# mpirun --mca btl openib,sm,self -machinefile $TMPDIR/machines -n $NSLOTS ./myHelloWorld_openMpi
```

submit via `qsub submissionScript_openMpi.sge`

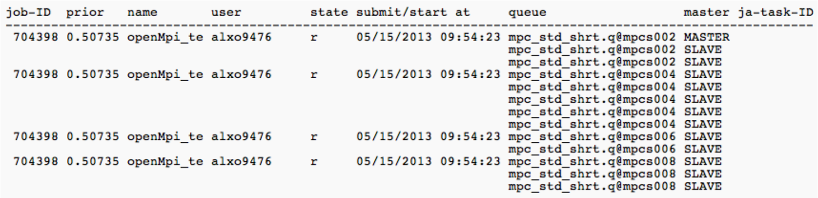
User Wiki: `Main Page > Brief Introduction to HPC Computing > 2.1`

Usage of SGE: Parallel job

- Submitting a job:
  - similar to single slot job

User Wiki: `Main Page > Brief Introduction to HPC Computing > 2.1.2`

- Checking status of job:
  - monitor job status using `qstat -g t`
    → more details for parallel jobs

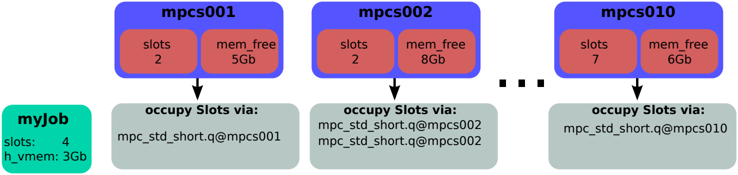


```
job-ID  prior   name       user         state submit/start at     queue                          master ja-task-ID
-----------------------------------------------------------------------------------------------------------------
 704398 0.50735 openMpi_te alxo9476     r     05/15/2013 09:54:23 mpc_std_shrt.q@mpcs002 MASTER
                                                                  mpc_std_shrt.q@mpcs002 SLAVE
                                                                  mpc_std_shrt.q@mpcs002 SLAVE
 704398 0.50735 openMpi_te alxo9476     r     05/15/2013 09:54:23 mpc_std_shrt.q@mpcs004 SLAVE
                                                                  mpc_std_shrt.q@mpcs004 SLAVE
                                                                  mpc_std_shrt.q@mpcs004 SLAVE
                                                                  mpc_std_shrt.q@mpcs004 SLAVE
                                                                  mpc_std_shrt.q@mpcs004 SLAVE
 704398 0.50735 openMpi_te alxo9476     r     05/15/2013 09:54:23 mpc_std_shrt.q@mpcs006 SLAVE
                                                                  mpc_std_shrt.q@mpcs006 SLAVE
 704398 0.50735 openMpi_te alxo9476     r     05/15/2013 09:54:23 mpc_std_shrt.q@mpcs008 SLAVE
                                                                  mpc_std_shrt.q@mpcs008 SLAVE
                                                                  mpc_std_shrt.q@mpcs008 SLAVE
```

User Wiki: `Main Page > Brief Introduction to HPC Computing > 2.1.3`

Usage of SGE: Job allocation rule

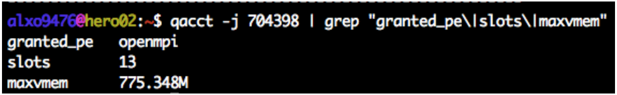
- How to actually collect slots needed for job?
  - distributed memory paradigm: different possibilities
  - SGE collects slots according to particluar *allocation rule*
- Allocation rule:
  - part of PE configuration (e.g., `qconf -sp openmpi`)
  - here: *fill-up* rule
- Fill-up allocation rule:
  - localize slots as much as possible
  - greedily collect slots (until requirements are met)
- Example:



FLOW: different! By default, user has exclusive access to nodes.

Usage of SGE: PE memory issue

- Retrieve details for finished jobs:
  - filter *accounting file* via `qacct -j <jobId>`
  - here: `qacct -j 704398`



```
alxo9476@hero02:~$ qacct -j 704398 | grep "granted_pe\|slots\|maxvmem"
granted_pe   openmpi
slots        13
maxvmem      775.348M
```

→ why so much memory for a slim job?

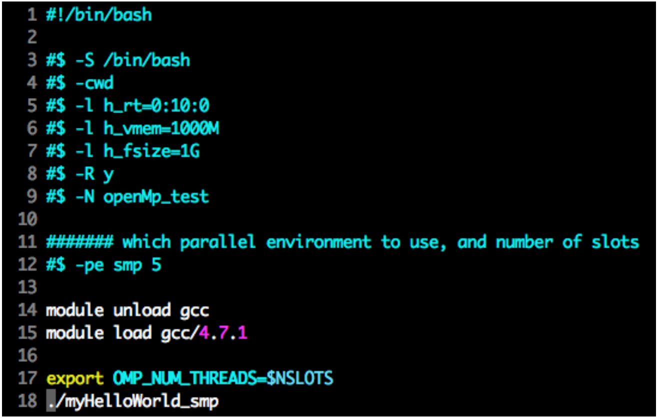
- PE memory issue:
  - jobs distributed over several nodes
  - MASTER process sets up/maintains connection to SLAVEs
  - per additional host ≈ 100Mb-150Mb
  - accumulate for MASTER only (other nodes need less)

→ common problem: MASTER might run out of resources!

User Wiki: `Main Page > Brief Introduction to HPC Computing > 2.3`

Usage of SGE: Parallel job

- Example: parallel job using smp via `openMp`
  `submissionScript_smp.sge`:



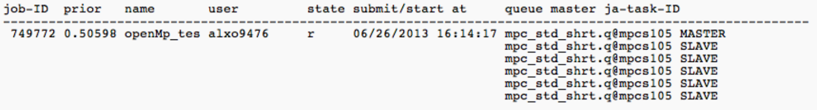
```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd
#$ -l h_rt=0:10:0
#$ -l h_vmem=1000M
#$ -l h_fsize=1G
#$ -R y
#$ -N openMp_test

####### which parallel environment to use, and number of slots
#$ -pe smp 5

module unload gcc
module load gcc/4.7.1

export OMP_NUM_THREADS=$NSLOTS
./myHelloWorld_smp
```

submit via `qsub submissionScript_smp.sge`

User Wiki: `Main Page > Brief Introduction to HPC Computing > 3`

Usage of SGE: Parallel job

- Submitting a job:
  - similar to single slot job
- Checking status of job:
  - monitor job status using `qstat -j <jobId>`



```
job-ID  prior   name       user        state submit/start at     queue master ja-task-ID
---------------------------------------------------------------------------------------------------------
 749772 0.50598 openMp_tes alxo9476     r    06/26/2013 16:14:17 mpc_std_shrt.q@mpcs105 MASTER
                                                                 mpc_std_shrt.q@mpcs105 SLAVE
                                                                 mpc_std_shrt.q@mpcs105 SLAVE
                                                                 mpc_std_shrt.q@mpcs105 SLAVE
                                                                 mpc_std_shrt.q@mpcs105 SLAVE
                                                                 mpc_std_shrt.q@mpcs105 SLAVE
```

User Wiki: `Main Page > Brief Introduction to HPC Computing > 3.2`

- Matching resources to requests:
  - shared memory paradigm
  - smp requires all slots to be located on single host
  - here: no PE memory issue

→ maximally available resources limited by execution host

User Wiki: `Main Page > Brief Introduction to HPC Computing > 3.3`

Usage of SGE: dissecting running jobs

- How to monitor current resource-usage for running jobs?
  - not possible by means of `qstat`
  - use *interactive session*

→ first: use `qstat` to determine exec. host

- Interactive session (recognized by SGE)
  - start session via `qrsh` (limited to 10 minutes)
  - logon to execution host
  - filter for your jobs via `top`: obtain process Id (pid)
  - list status file to obtain details: `cat /proc/pid/status`

→ useful to monitor, e.g., current/maximal memory

Debugging and profiling

- Debugging:
  - GNU debugging tools (`GDB`, `DDD`)
  - ICS contains intel debugger `IDBC`

  User Wiki: `Main Page > Compiler and Dev Tools > debugging`

- Profiling:
  - profiling example using `gprof` (in `C`)
  - using shared libs: `sprof` (more involved)
  - python: `cProfile`

  User Wiki: `Main Page > Compiler and Dev Tools > profiling`

- Mem checker:
  - detect non-freed memory
  - detect invalid pointer use
  - distinguish heap/stack memory

  User Wiki: `Main Page > Compiler and Dev Tools > valgrind`

  User Wiki: `Main Page > Brief Introduction to HPC Computing > 4`

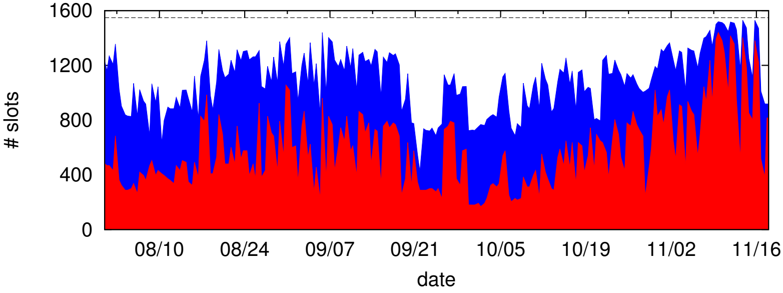
Misc

- How to login from outside the university?
  - User Wiki: `Main Page > Login`
  - from home: sometimes difficulties to resolve hostname
  - instead try to login using IP-address:

    ```
    ssh abcd1234@10.140.1.61
    ```
- How to mount HPC home directory?
  - User Wiki: `Main Page > User environment`
- Importance of allocating fitting resources
  - avoid unnecessary excess memory
  - be *friendly* user

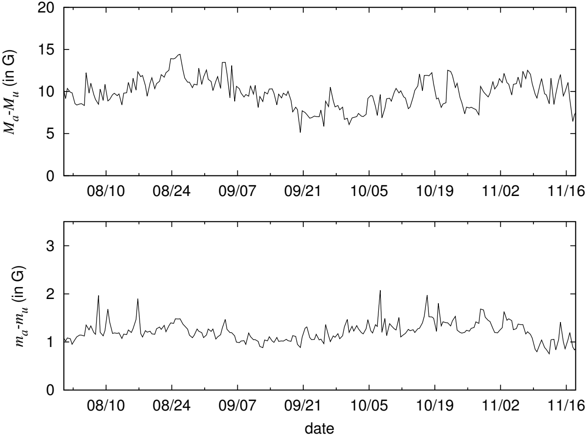
Utilized cluster capacity

- Typical number of running jobs (example: `mpcs` nodes):



Utilized cluster capacity

- Typical amount of excess memory (example: `mpcs` nodes)
  Top: per host, bottom: per occupied slot


M_a-M_u (in G)
08/10
08/24
09/07
09/21
10/05
10/19
11/02
11/16
m_a-m_u (in G)
08/10
08/24
09/07
09/21
10/05
10/19
11/02
11/16
date