



Job Queuing System SGE

Dr. Stefan Albensoeder

Contact: Stefan.Albensoeder@uni-oldenburg.de

Overview

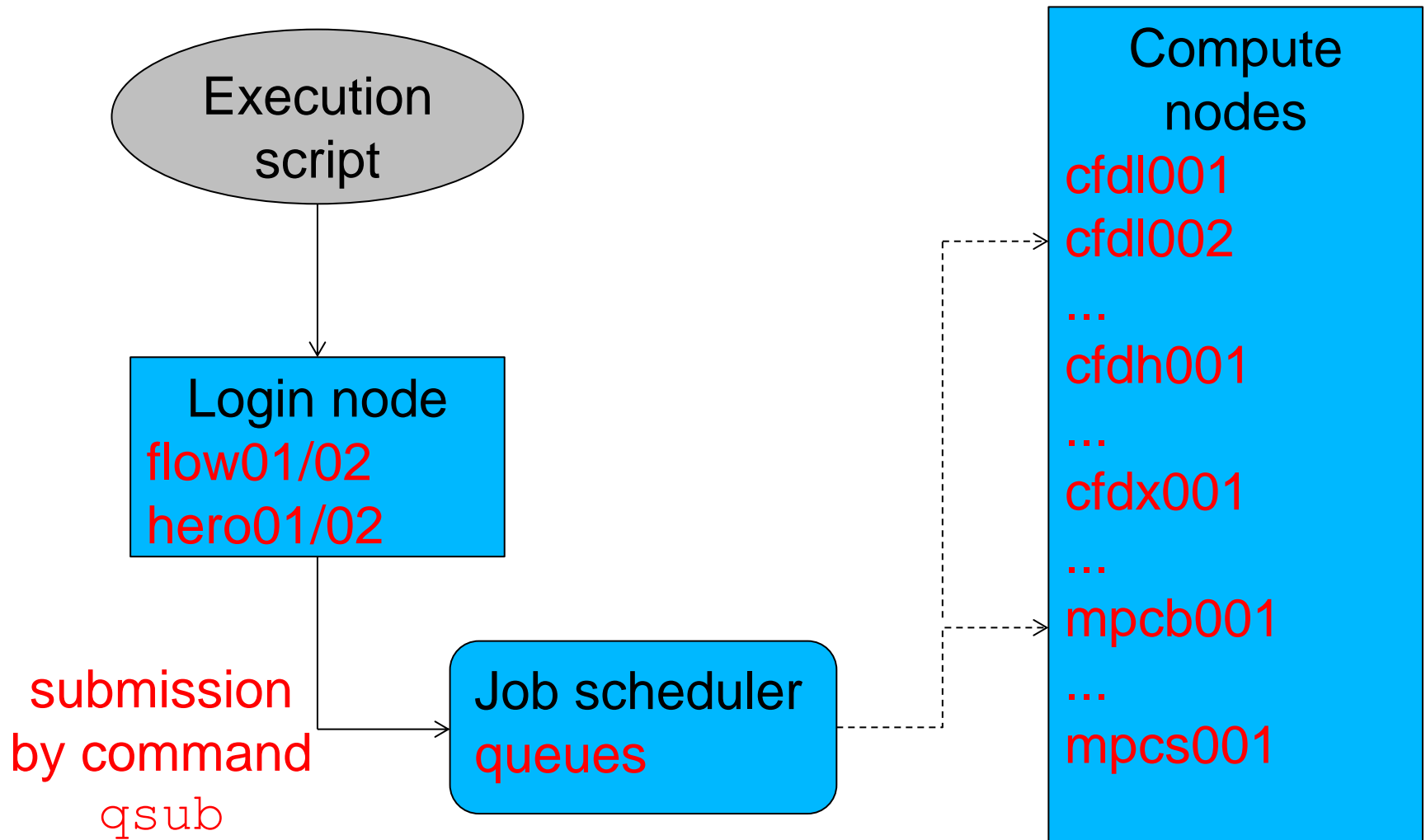
Introduction to the usage of SGE

1. Introduction
2. Preparations
3. Getting started
 - submitting, monitoring and controlling
4. Special jobs
 - I/O intensive jobs
 - parallel jobs
 - job arrays
5. Additional information
6. Job placement

INTRODUCTION

JOB SCHEDULER

Job scheduler



Tasks of a job scheduler

- acceptance of jobs
(i.e. job script with requests for computing resources)
- prioritize jobs
(e.g. to enable a fair share of computational resources between the users)
- places jobs in queue until they can be run
- organize workload on HPC system
(e.g. avoid overload of nodes, balance workload on nodes)
- sends jobs from queue to execution hosts (computational node)
- manages/monitors running jobs
- logs
 - stdout/stderr of job script
 - details of finished jobs
- terminate jobs if uses more resources than requested

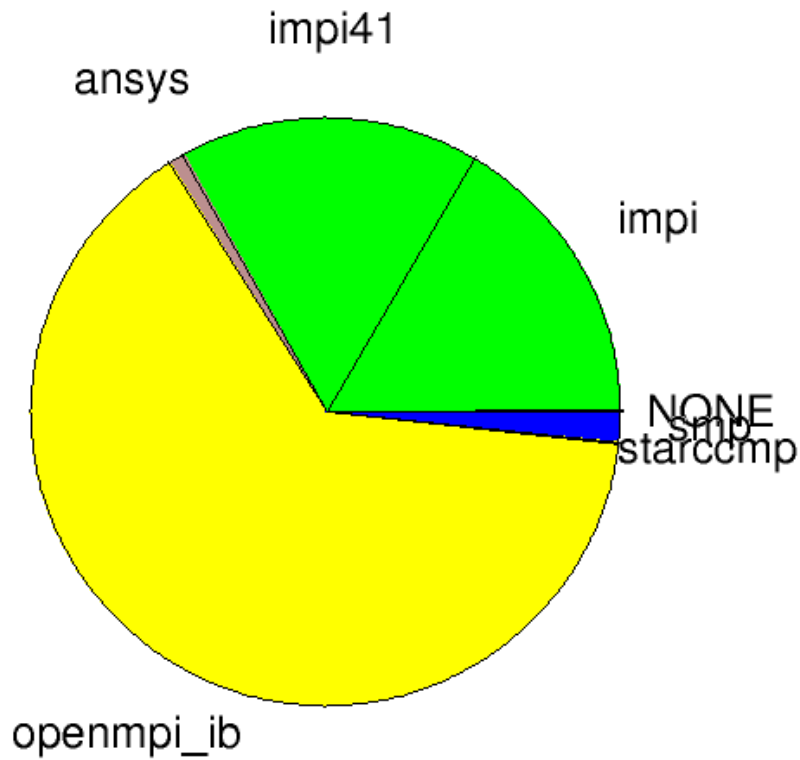
Tasks of a job scheduler

Job scheduler on FLOW/HERO

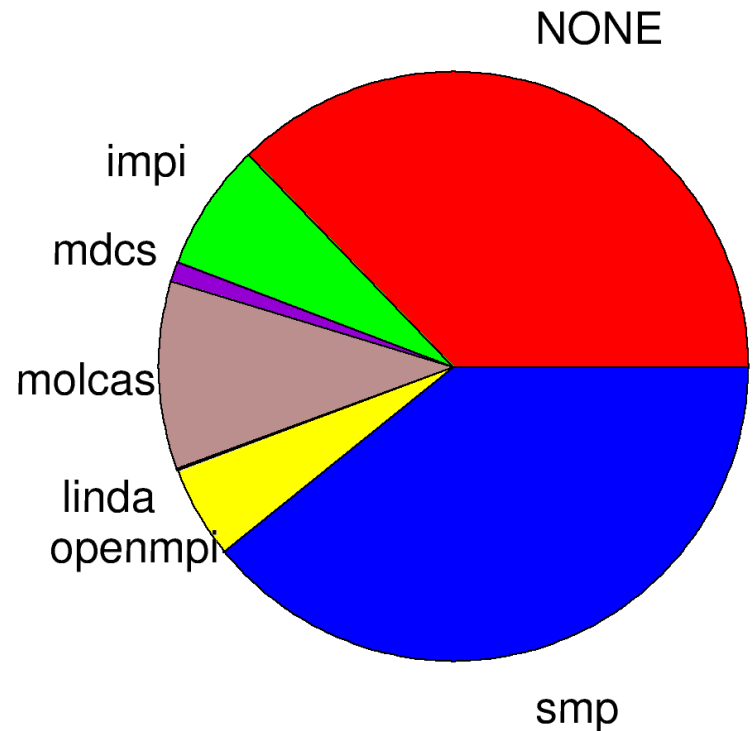
- Sun Grid Engine (SGE)
(other schedulers: PBS/MOAB, SLURM, LSF)
- here: heterogeneous user community
 - $\mathcal{O}(200)$ active users (~1/3 FLOW, ~2/3 HERO)
 - 34 different working groups (from faculties 2, 5, 6)
 - different users, different needs, e.g.
 - parallel computations
 - multiple serial computations
 - computations with memory requirements

Usage of the cluster (last year)

FLOW



HERO



PREPARATIONS

Preparations before submitting

In case of using *commercial/open-source* program

- check if program already available on login nodes
 - typically programs installed as modules
 - command to show all available modules
`module av`
 - most programs within modules are runtime optimized (e.g. by selection of the compiler/libraries)
- if many people uses the program
 - please ask for a global installation as module
 - Stefan.Albensoeder@uni-oldenburg.de (FLOW)
 - Stefan.Harfst@uni-oldenburg.de (HERO)

Preparations before submitting

In case of using *own code*

- first make tests on your local machine
 - compile your code
 - perform
 - debugging
 - optimization & profiling
 - perform small test runs
- when tests successful
 - compile your code on login nodes
 - and start with small test cases again

GETTING STARTED

How to submit a job?

- SGE is resource driven
 - you need to know the *real needs* of your program
 - memory consumption
 - run time (wall clock time)
 - disk space
 - in case of parallel programs
 - number of slots (typically equal number of cores)
 - used implementation of MPI, or (*Message Passing Interface*)
 - *Shared Memory* based parallelization (*OpenMP, PThread*)
- Requirements will be guaranteed/reserved by SGE
 - *too high resource requirements can block other jobs!*
- SGE accepts only scripts for submission

How to submit a job?

Simple example script `submitMyProgram.sge`

```
#!/bin/bash
#----- Shell to start with -----
#$ -S /bin/bash
#--- Run job in directory where job was submitted from ---
#$ -cwd
#----- Name of the job -----
#$ -N myTest
#----- Maximum wall clock time of the job -----
#$ -l h_rt=24:00:00
#----- Maximum memory usage per slot -----
#$ -l h_vmem=1800M
#----- Maximum of used disk space -----
#$ -l h_fsize=100G
#----- Merge stdout/stderr -----
#$ -j y

./myProgramCommand
```

→ submit job by `qsub submitMyProgram.sge`

How to submit a job?

Alternative way to submit script
`submitMyProgramAlt.sge`

```
#!/bin/bash  
  
./myProgramCommand
```

→ submit job by

```
qsub -S /bin/bash -cwd -N myTest \  
-l h_rt=24:00:00,h_vmem=1800M \  
-l h_fsize=100G -j y \  
submitMyProgramAlt.sge
```

Note: Specifications in command line will overwrite specifications in script

What happens when submitting?

During execution of `qsub`

- SGE makes a copy your script
→ changes in script after submission will not considered!
- if jobs was accepted, SGE returns a job ID
→ the ID can be used to make changes to the job

After execution of `qsub` (in background)

- Computation of the priority of the job
(function of number of jobs per user, number of requested slots, requested wall clock time....)
- find the right queue w.r.t. the job requirements
- find free resources w.r.t. the queue and execute script

More regarding queues later...

How to monitor a job?

Checking status of submitted job

- monitor job status using `qstat` (show all own jobs)

```
abcd1234@flow01:~ > qstat
job-ID  prior    name         user          state submit/start at
queue                               slots ja-task-ID
-----
1234567 0.50500 myTest      abcd1234     qw      09/29/2014 16:05:35
                               1
```

- somewhat later:

```
abcd1234@flow01:~ > qstat
job-ID  prior    name         user          state submit/start at
queue                               slots ja-task-ID
-----
1234567 0.50500 myTest      abcd1234     r        09/29/2014 16:06:32
cfld_lom_shrt.q@cfld1003          1
```

- when job is running `stdout/stderr` is written to file
`${jobName}.o${jobID}`, `${jobName}.e${jobID}`
→ here: `myTest.o1234567`

How to monitor a job?

More details of submitted job

- **use** `qstat -j <JobID>`

```
abcd1234@flow01:~ > qstat -j 1234567 | head -n 30
=====
job_number:                1234567
exec_file:                 job_scripts/1234567
submission_time:          Mon Sep 29 16:05:35 2014
...
sgc_o_shell:              /bin/bash
sgc_o_workdir:            /user/fw/abcd1234/test
sgc_o_host:               flow01
...
reserve:                  y
hard_resource_list:       h_rt=86400,h_vmem=1800M,h_fsize=100G
...
job_name:                 myTest
...
parallel_environment:     NONE
usage      1:             cpu=00:01:24, mem=953.12500 GBs, io=6.75942,
vmem=1.203G, maxvmem=1.303G
...
```

How to monitor a job?

Get information about an ended job

- use `qacct -j <JobID>` (can take minutes)

```
abcd1234@flow01:~ > qacct -j 1234567
qname          cfd_lom_shrt.q
hostname       cfdl003.cm.cluster
...
qsub_time      Tue Sep 29 16:05:35 2014
start_time     Tue Sep 29 16:06:32 2014
end_time       Tue Sep 29 17:06:32 2014
...
failed         0
exit_status    0
ru_wallclock   3600
ru_utime       3123.184
ru_stime       98.746
...
cpu            6461.930
mem            2266.224
io             0.014
iow            0.000
maxvmem        1.303G
arid           undefined
```

How to control a job?

- **delete job**

- use `qdel <jobID>`
- `qdel `*`` deletes all own jobs

- **alter resource requirements**

- modify resources with `qalter`

- e.g.

```
qalter -l h_vmem=2G,h_fsize=100G,h_rt=24:00:00  
1234567
```

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

Summary of useful SGE commands

- `qsub` – submit your job to the scheduler
- `qstat` – monitor status of queued jobs
- `qacct` – retrieve details for finished jobs
- `qdel` – delete jobs
- `qalter` – alter jobs
- `qhold` – hold jobs
- `qrls` – release jobs from hold state
- `qhost` – show host usage
- `qssh/qlogin` – request interactive sessions
- `qconf` – examine SGE configuration (for experts)
- `qmon` – graphical interface to SGE

SGE

I/O INTENSIVE JOBS (ONLY HERO)

I/O intensive jobs

- programs which
 - reads or writes data several times
 - needs highly temporary storage
- avoid /work or /users directory for I/O (avoid I/O over slow network)
- fast local scratch disk only available on HERO
→ unique path to local scratch in `$TMPDIR`
- on FLOW scratch directory `$TMPDIR` exists, but is not usable

I/O intensive jobs (only HERO) - Example

```
#!/bin/bash

#$ -S /bin/bash
#$ -N myIOJob
#$ -l h_rt=24:00:00
#$ -l h_vmem=1200M
#$ -l h_fsize=100G
#$ -j y

# go to local scratch directory (only usable on HERO)
cd $TMPDIR

# copy input data from submission dir to local scratch
cp $SGE_O_WORKDIR/inputData .

# run program in scratch dir
myProgramCmd inputData outputData

# copy results back to submission directory
cp outputData $SGE_O_WORKDIR
```

SGE

PARALLEL JOBS

Parallel jobs

- for parallel jobs a parallel environment has to be defined
 - the parallel environment has to fit to the used parallelization
 - available environments

Name	Description
smp	Shared memory parallelization (e.g. OpenMP, PThread) → can run only on one node (maximal 12 slots)
impi	Intel MPI 4.0.x (deprecated)
impi41	Intel MPI 4.1.x
openmpi	OpenMPI for HERO (without InfiniBand)
openmpi_ib	OpenMPI for FLOW (with InfiniBand)
starccmp	PE for StarCCM+
ansys	PE for ANSYS

- additionally the number of slots (cores) has to be defined

Parallel jobs – Intel MPI Example

```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd
#$ -N myMPIJob
#$ -l h_rt=24:00:00
#$ -l h_vmem=1200M
#$ -l h_fsize=100G
#$ -j y

#----- run with 36 processes -----
#$ -pe impi41 36

# need to load the Intel MPI module
# which was used for compilation
module load intel/impi/4.1.3.045/64

# start MPI program in parallel (use not mpiexec!)
mpirun -np $NSLOTS ./myMPIProgram
```

Parallel jobs – OpenMPI Example

```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd
#$ -N myMPIJob
#$ -l h_rt=24:00:00
#$ -l h_vmem=1200M
#$ -l h_fsize=100G
#$ -j y

#----- run with 36 processes on HERO -----
#$ -pe openmpi 36
#----- or run with 36 processes on FLOW -----
#$ -pe openmpi_ib 36

# need to load the OpenMPI module e.g.
module load openmpi/1.8.2/ics

# start MPI program in parallel (use not mpiexec!)
# for HERO add: --mca btl ^openib,ofud
mpirun -machinefile $TMPDIR/machines \
       -np $NSLOTS ./myMPIProgram
```

Parallel jobs – OpenMP Example

```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd
#$ -N myOpenMPJob
#$ -l h_rt=24:00:00
#$ -l h_vmem=1200M
#$ -l h_fsize=100G
#$ -j y

#----- run with 36 processes -----
#$ -pe smp 12

# set number of used threadsmodule load
export OMP_NUM_THREADS=$NSLOTS

# start OpenMP program
./myOPenMPProgram
```

Parallel jobs

Additional files from parallel environment

- `${jobName}.po${jobID}`,
`${jobName}.pe${jobID}`
- contain information about allocated hosts, MPI setup,...

Monitoring of parallel jobs

- use `qstat -g t` to get information about the master/slave nodes
- alternatively you can use the script `qjobs`

Checking of free resources (mostly for FLOW users)

- `qfreenodes` shows you number of free nodes

Parallel jobs

PE memory issue (especially on HERO)

- jobs distributed over several nodes
- MASTER process sets up/maintains ssh-connection to SLAVES
- per additional host \approx 100Mb-150Mb (because of shell)
- accumulate for MASTER only (other nodes need less)
→ common problem: MASTER might run out of resources!

SGE

JOB ARRAYS

Job arrays

- Job array start same script several times
- Each running job gets an own task ID
(environment variable `SGE_TASK_ID` during execution)
- Useful for parameter studies

Name	Description
<code>SGE_TASK_FIRST</code>	First task ID of job array.
<code>SGE_TASK_ID</code>	Task ID of job within a job array.
<code>SGE_TASK_LAST</code>	Last task ID of job array.
<code>SGE_TASK_STEPSIZE</code>	Stepsize within the task ID of job arrays.

Job arrays - Example

```
#!/bin/bash

#$ -S /bin/bash
#$ -cwd
#$ -N myJobArray
#$ -l h_rt=24:00:00
#$ -l h_vmem=1200M
#$ -l h_fsize=100Gb
#$ -j y

#----- On FLOW uncomment, otherwise full node per task
# # $ -l excl_flow=false
#----- Task ID from 1-100, step size 2 -----
#$ -t 1-100:2
#----- Maximum 3 running task at same time -----
#$ -tc 3

# fetch line SGE_TASK_ID from paraList.dat
# as command line parameter
./myProgramCmd $(sed -n ${SGE_TASK_ID}'p' paraList.dat)
```

SGE

RESOURCE REQUIREMENTS OVERVIEW

Resource requirements

Several resource requirement can be specified

Option	Description	FLOW	HERO
-l h_rt=05:30:00	Set the wall clock limit of the job	X	X
-l h_vmem=1850M	Maximum memory usage of a job per slot (default 1200M)	X	X
-l h_fsize=200G	Maximum number of disk blocks that a job writes to disk (default 10G)	X	X
-l longrun=true	Enables long runs (default limit 192h)	X	X
-l bignode=true	Enables running on HERO nodes with 46Gb memory/2100Gb scratch.		X
-l highmem=true	Enables running on high memory nodes on FLOW with 46Gb memory.	X	
-l exclusive=true	Enable exclusive use of nodes (default false)		X
-l excl_flow=false	Disable exclusive use of nodes (default true)	X	

Note: With option `-l` the so called complexes can be specified
(list of all complexes by `qconf -sc`)

Further options

Option	Description
-N jobName	Set job name
-j y	Merge stdout and stderr of job script output
-S /bin/bash	Selects the shell in which the job starts.
-cwd	Starts the job in the directory of the job submission.
-e errorFile	Redefine the name of the stdout file.
-o stdoutFile	Redefine the name of the stdout file.
-hold_jid jobIDList	Enable holding of job until other jobs has been finished.
-R y	Turning on the resource reservation.
-M myemail@adress	Email adress.
-m b e a s n, ...	Send mail at begin, end, abort, supending of a job or not

Further options see

- HPC-Wiki
- by using `man qsub`

SGE

ENVIRONMENT VARIABLES

Environment variables

During execution SGE sets environment variable which can be used in the job script

Name	Description
JOB_ID	Job ID of current jobs
JOB_NAME	Name of the job (e.g. defined by SGE option -N).
JOB_SCRIPT	Name of the submitted job script.
PE_HOSTFILE	Hostfile of the parallel environment.
NHOSTS	Number of used hosts.
NSLOTS	Number of used slots.
SGE_O_HOST	Submitting host.
SGE_STDERR_PATH	Output file name of stderr.
SGE_STDOUT_PATH	Output file name of stdout.
SGE_O_WORKDIR	Submission directory.
TMPDIR	Path to local temporary scratch directory.

SGE

SUBMISSION PROCESS

What happens when submitting?

During execution of `qsub`

- SGE makes a copy your script
→ changes in script after submission will not considered!
- if jobs was accepted, SGE returns a job ID
→ the ID can be used to make changes to the job

After execution of `qsub` (in background)

- Computation of the priority of the job
(function of number of jobs per user, number of requested slots, requested wall clock time....)
- find the right queue w.r.t. the job requirements
- find free resources within the right queue

What are queues?

Queues respect different resource limits, e.g. short, low memory queue on FLOW:

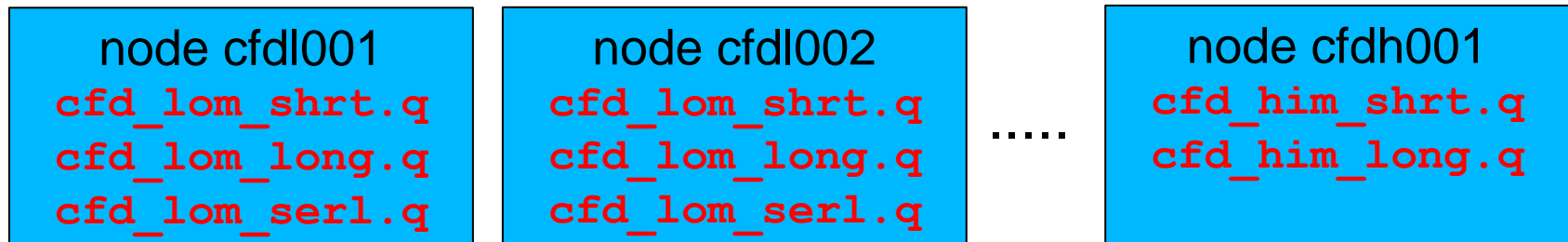
```
abcd1234@flow02:~ > qconf -sq cfd_lom_shrt.q
qname          cfd_lom_shrt.q
seq_no         2000
hostlist       @cfdl
...
pe_list        ansys impi impi41 mdcs openmpi_ib smp starccmp
...
complex_values h_vmem=22G,h_fsize=10000G,cluster=flow
...
h_rt           192:0:0
...
```

- resource allocation statements determine fitting queue
- search order by sequence number of queue

What are queues?

Execution hosts feature several queue instances

- queue instances jointly consume memory and slots on host
- consider e.g. standard nodes on FLOW:



- resource allocation statements determine fitting queue(s)

How does SGE places jobs on execution host?

General

- SGE try to make a load balancing between nodes
→ on the free cluster 2 serial jobs will be placed on two different nodes
- e.g. 2 serial jobs on HERO

```
node mpcs001  
free slots: 12  
free h_vmem: 23G
```

```
node mpcs002  
free slots: 12  
free h_vmem: 23G
```

```
node mpcs003  
free slots: 12  
free h_vmem: 23G
```

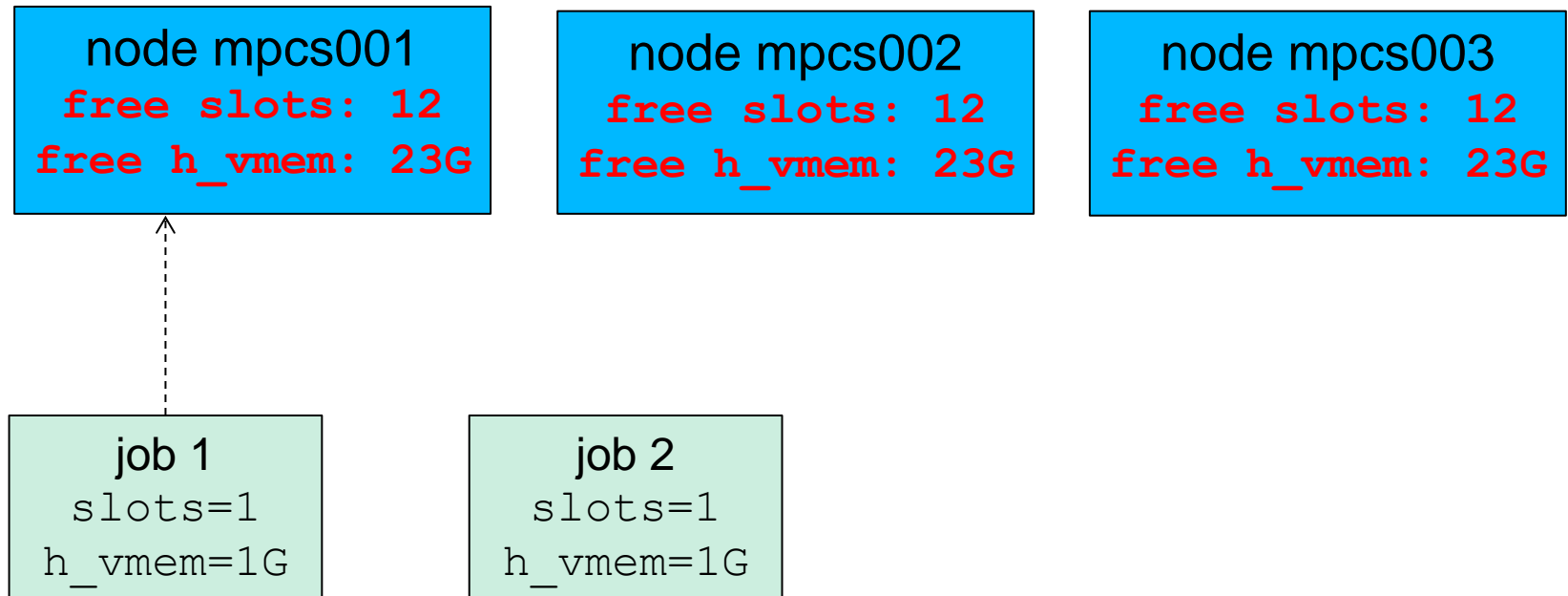
```
job 1  
slots=1  
h_vmem=1G
```

```
job 2  
slots=1  
h_vmem=1G
```

How does SGE places jobs on execution host?

General

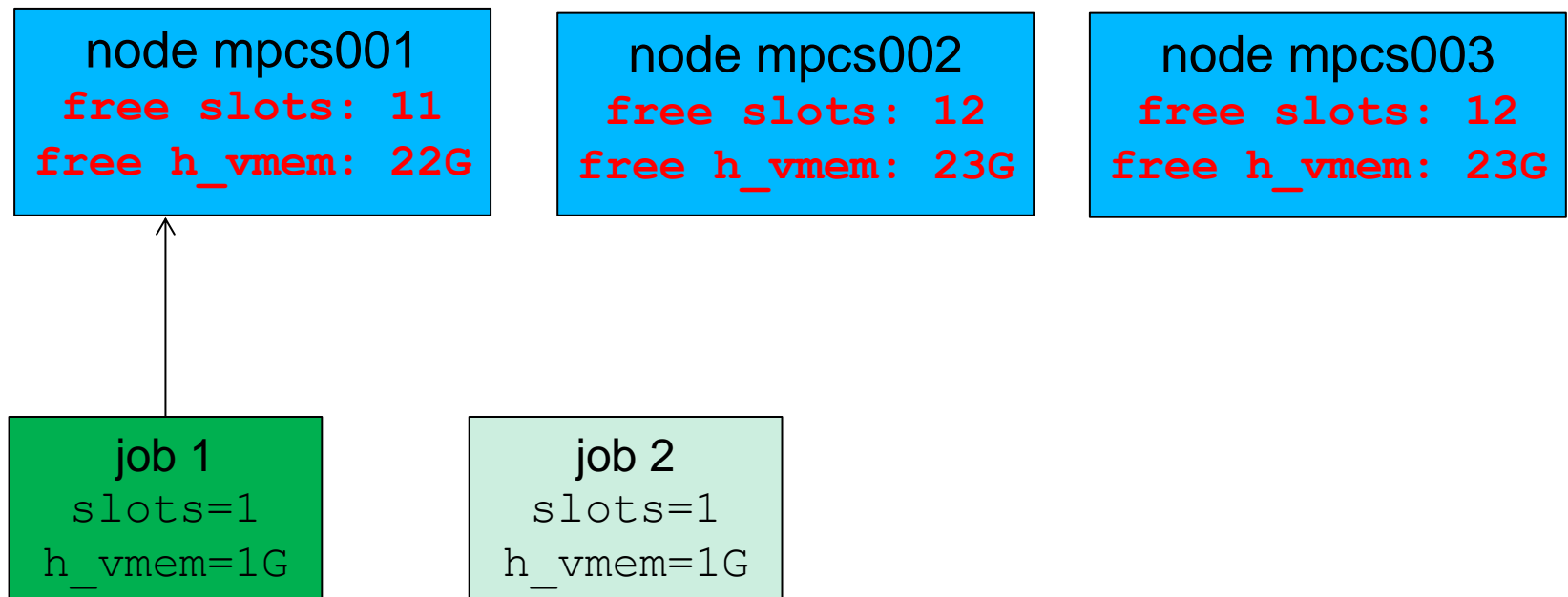
- SGE try to make a load balancing between nodes
→ on the free cluster 2 serial jobs will be placed on two different nodes
- e.g. 2 serial jobs on HERO



How does SGE places jobs on execution host?

General

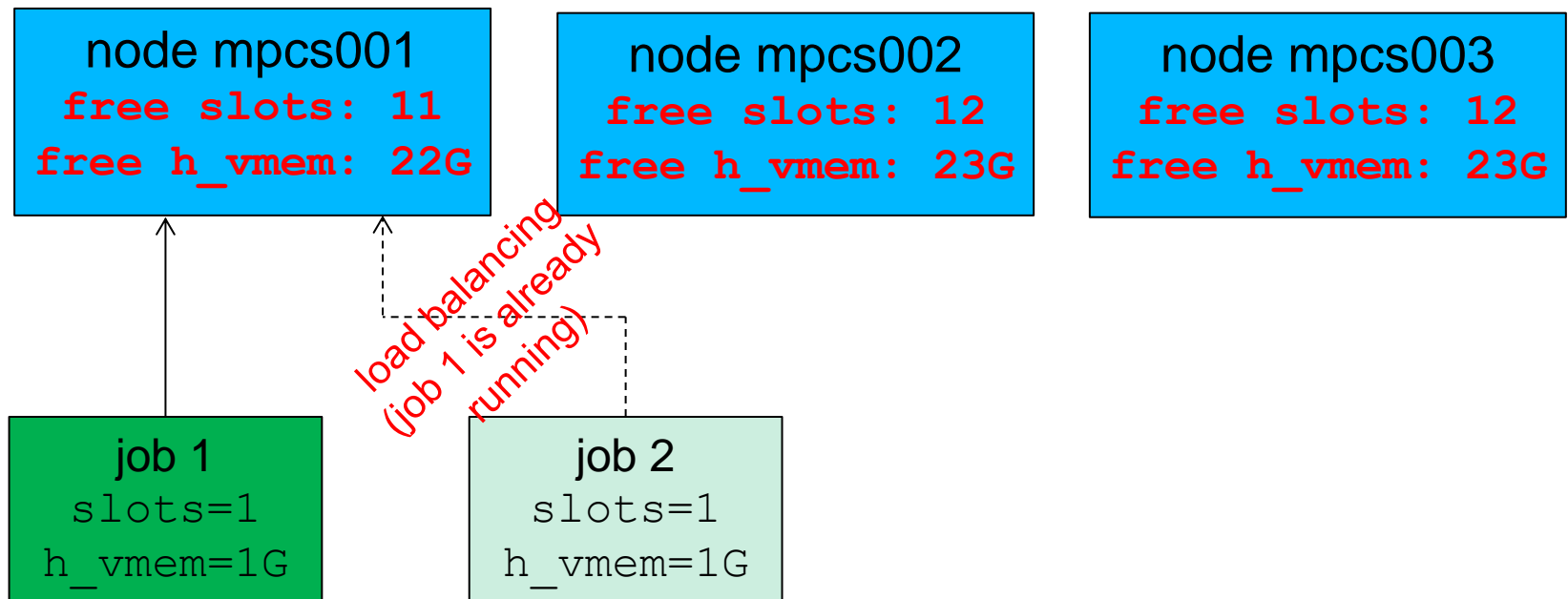
- SGE try to make a load balancing between nodes
→ on the free cluster 2 serial jobs will be placed on two different nodes
- e.g. 2 serial jobs on HERO



How does SGE places jobs on execution host?

General

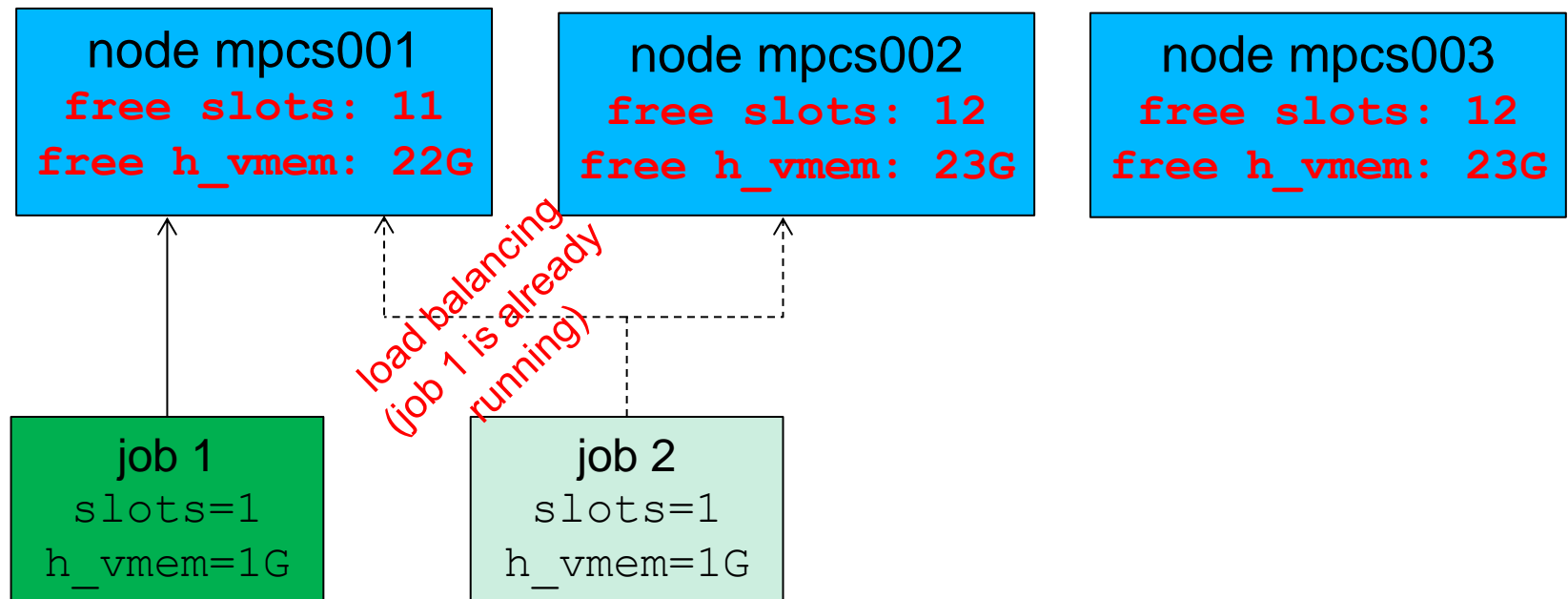
- SGE try to make a load balancing between nodes
→ on the free cluster 2 serial jobs will be placed on two different nodes
- e.g. 2 serial jobs on HERO



How does SGE places jobs on execution host?

General

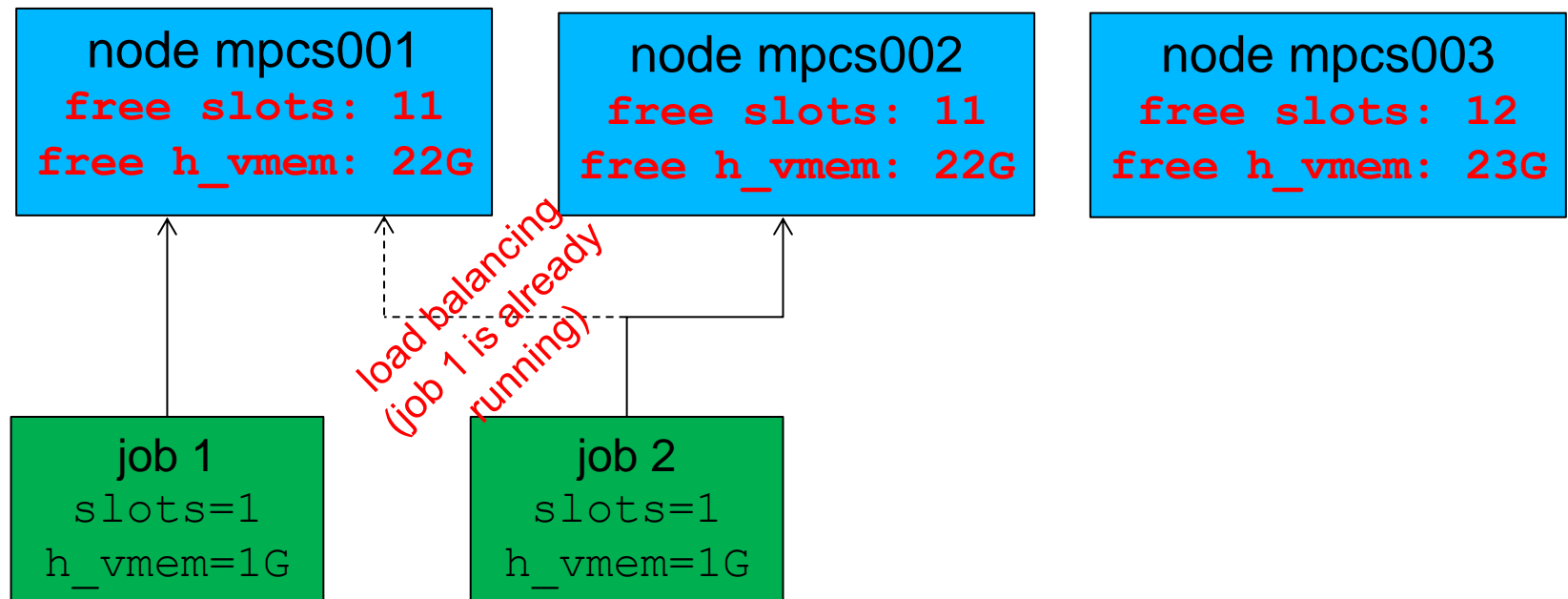
- SGE try to make a load balancing between nodes
→ on the free cluster 2 serial jobs will be placed on two different nodes
- e.g. 2 serial jobs on HERO



How does SGE places jobs on execution host?

General

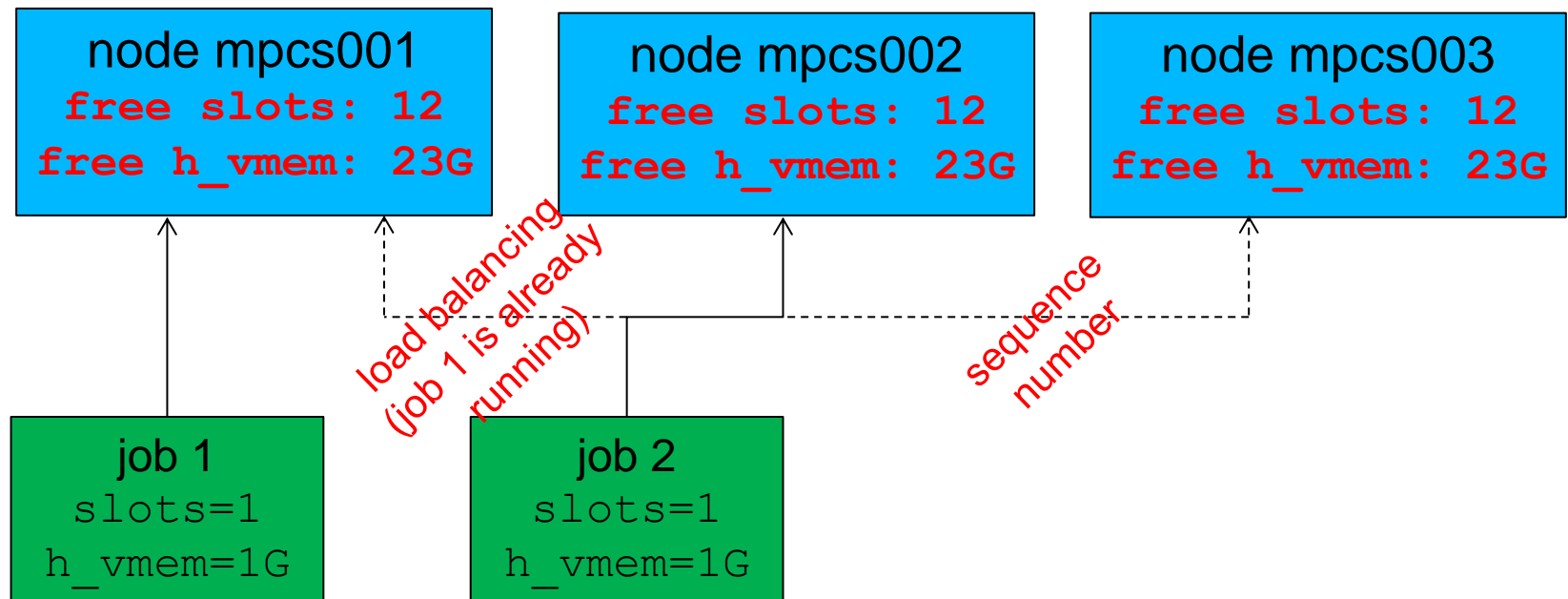
- SGE try to make a load balancing between nodes
→ on the free cluster 2 serial jobs will be placed on two different nodes
- e.g. 2 serial jobs on HERO



How does SGE places jobs on execution host?

General

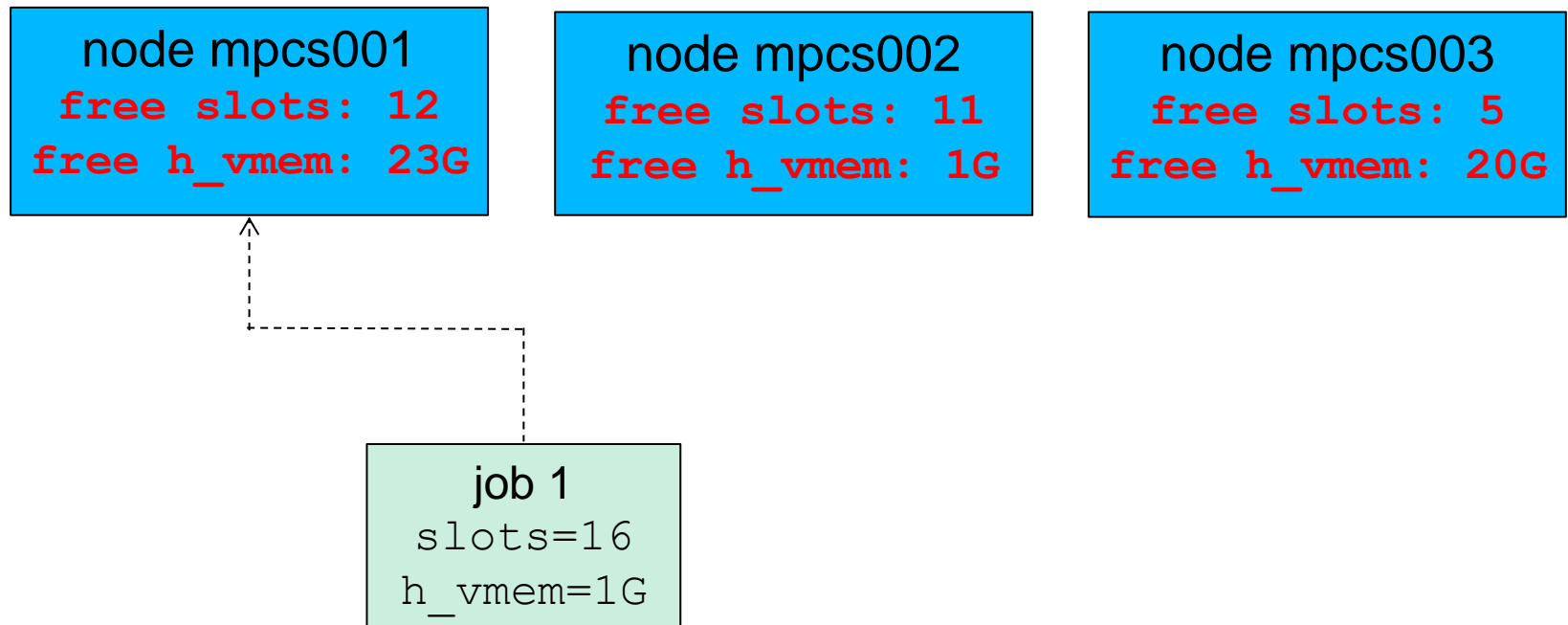
- SGE try to make a load balancing between nodes
→ on the free cluster 2 serial jobs will be placed on two different nodes
- e.g. 2 serial jobs on HERO



How does SGE places jobs on execution host?

Parallel jobs

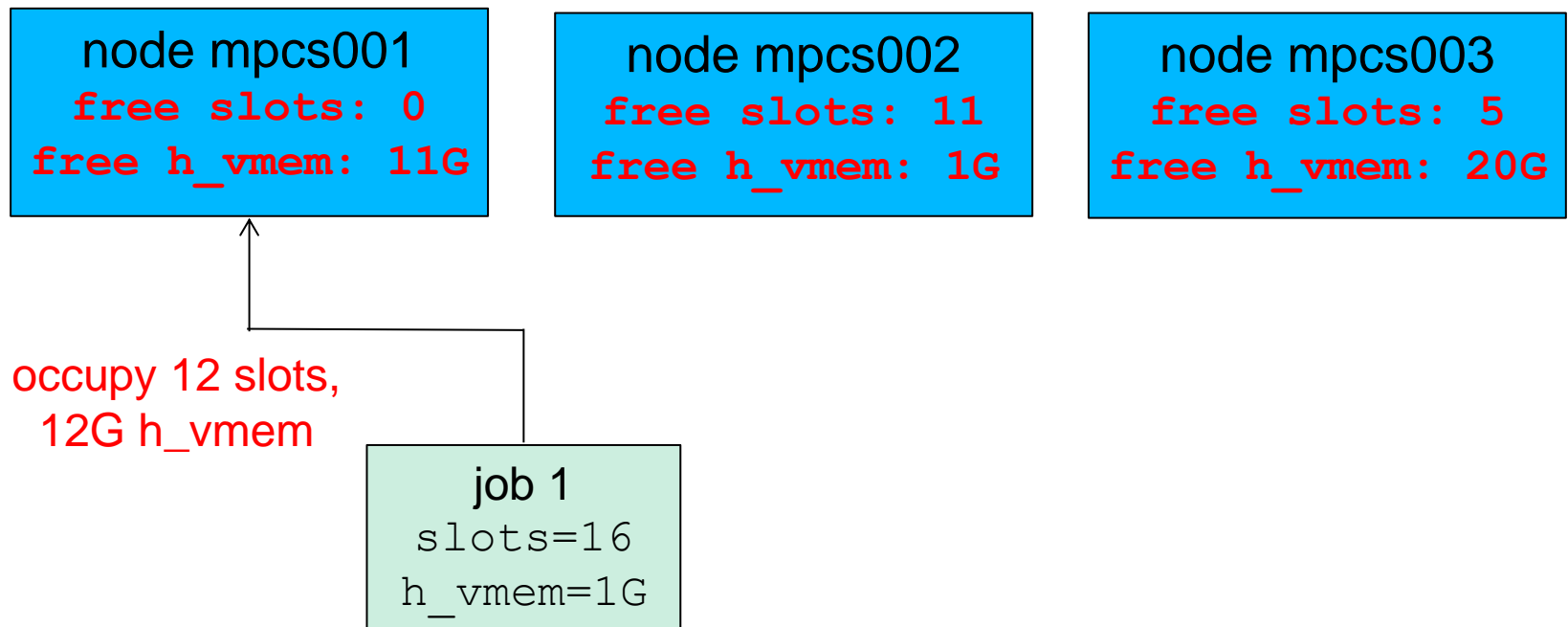
- SGE try to fill up nodes (configured for FLOW/HERO)
- e.g. parallel job with 16 slots on HERO



How does SGE places jobs on execution host?

Parallel jobs

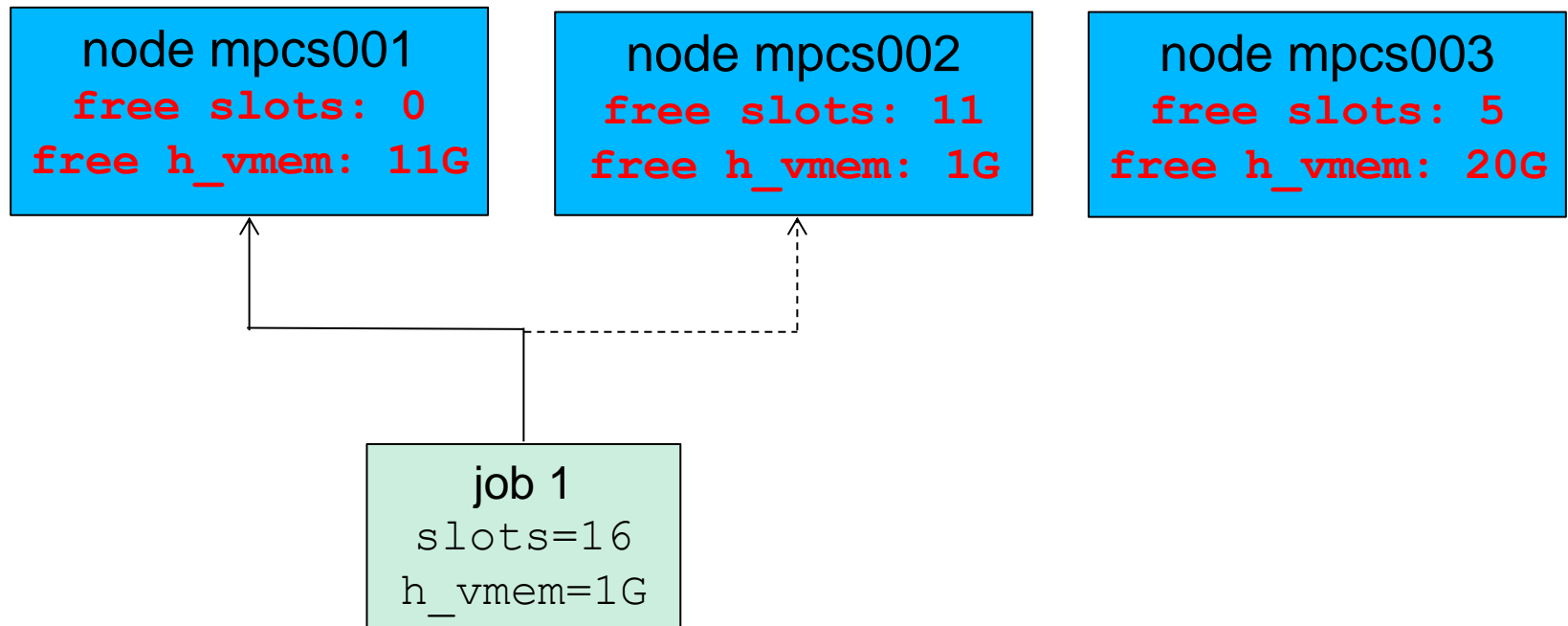
- SGE try to fill up nodes (configured for FLOW/HERO)
- e.g. parallel job with 16 slots on HERO



How does SGE places jobs on execution host?

Parallel jobs

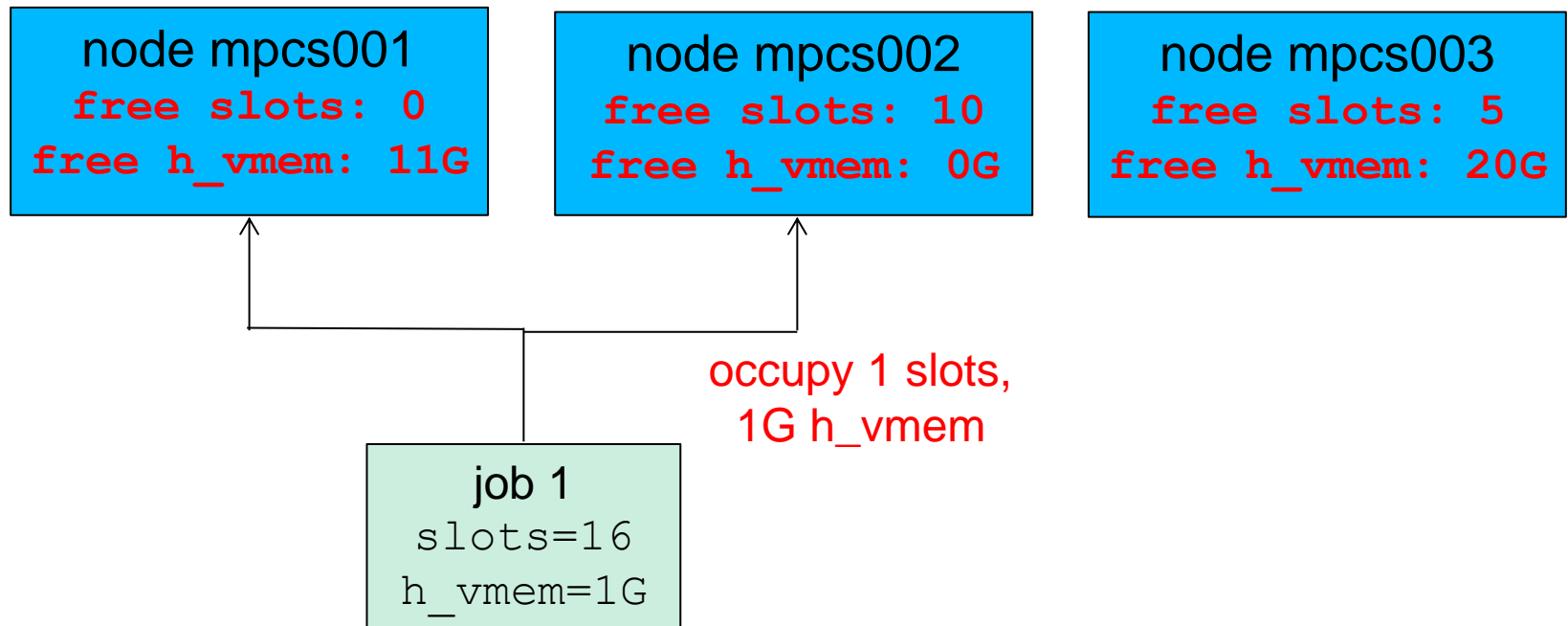
- SGE try to fill up nodes (configured for FLOW/HERO)
- e.g. parallel job with 16 slots on HERO



How does SGE places jobs on execution host?

Parallel jobs

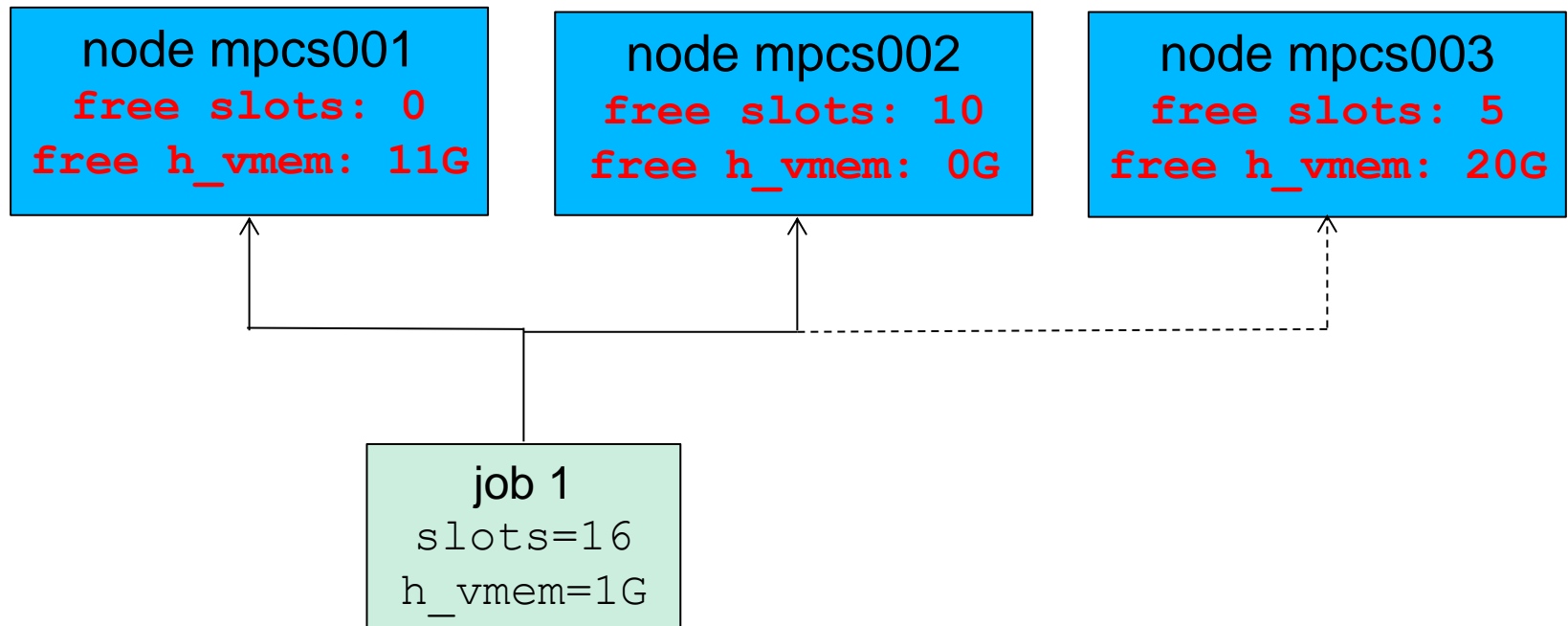
- SGE try to fill up nodes (configured for FLOW/HERO)
- e.g. parallel job with 16 slots on HERO



How does SGE places jobs on execution host?

Parallel jobs

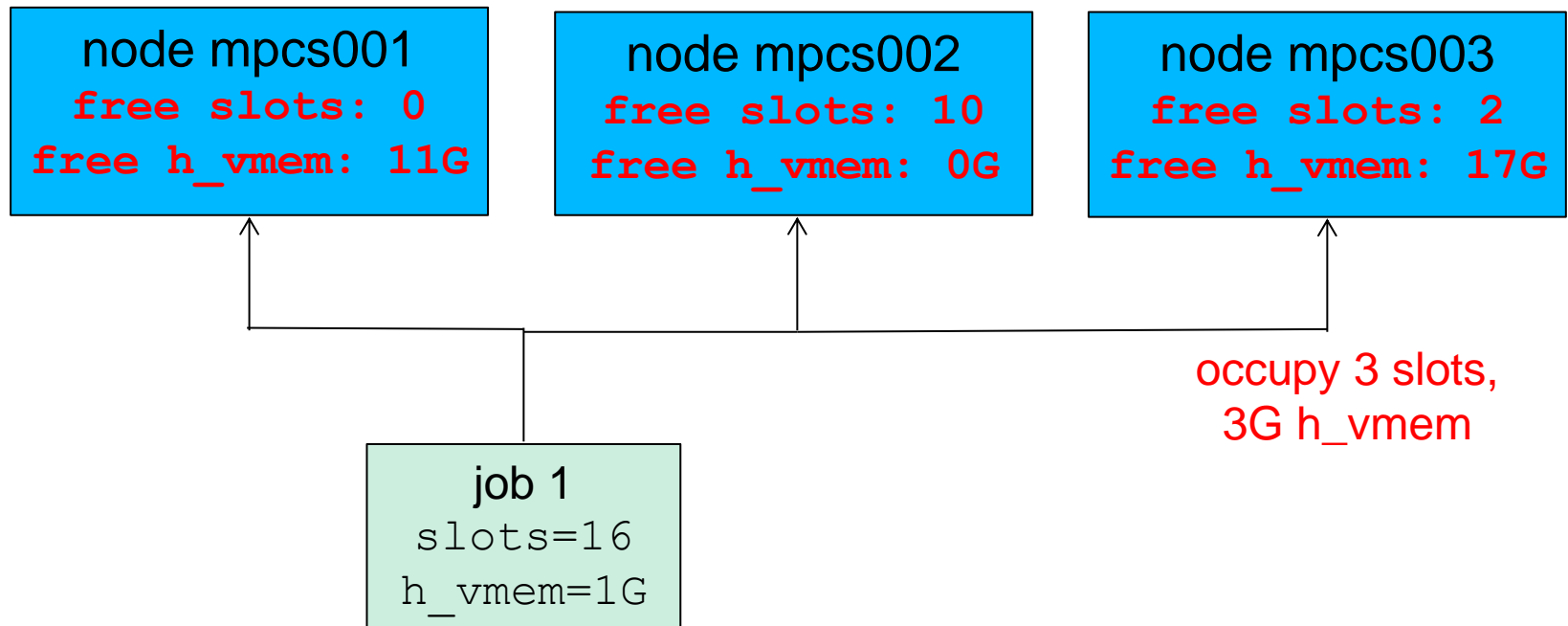
- SGE try to fill up nodes (configured for FLOW/HERO)
- e.g. parallel job with 16 slots on HERO



How does SGE places jobs on execution host?

Parallel jobs

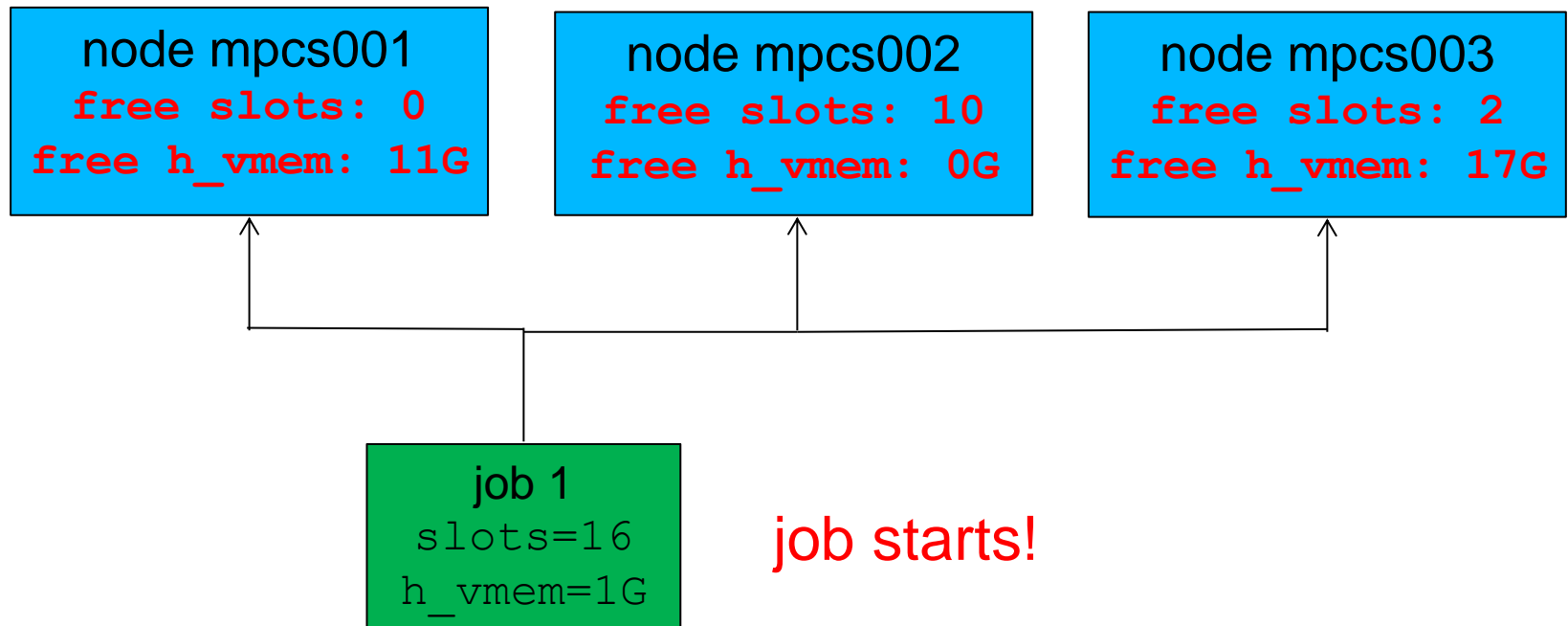
- SGE try to fill up nodes (configured for FLOW/HERO)
- e.g. parallel job with 16 slots on HERO



How does SGE places jobs on execution host?

Parallel jobs

- SGE try to fill up nodes (configured for FLOW/HERO)
- e.g. parallel job with 16 slots on HERO



How does SGE places jobs on execution host?

Serial/Parallel jobs on FLOW

- can change full occupation by setting
 - l excl_flow=False
 - useful for serial jobs
 - but to many serial jobs can disturb parallel jobs
 - better use parallel scripts which uses a full node

How does SGE places jobs on execution host?

Serial/Parallel jobs on FLOW

- on FLOW nodes are always fully occupied
- e.g. several jobs on FLOW

```
node cfd001
free slots: 12
free h_vmem: 23G
excl_flow: True
```

```
node cfd002
free slots: 12
free h_vmem: 23G
excl_flow: True
```

```
node cfd003
free slots: 12
free h_vmem: 23G
excl_flow: True
```

```
job 1
slots=16
h_vmem=1G
excl_flow=True
```

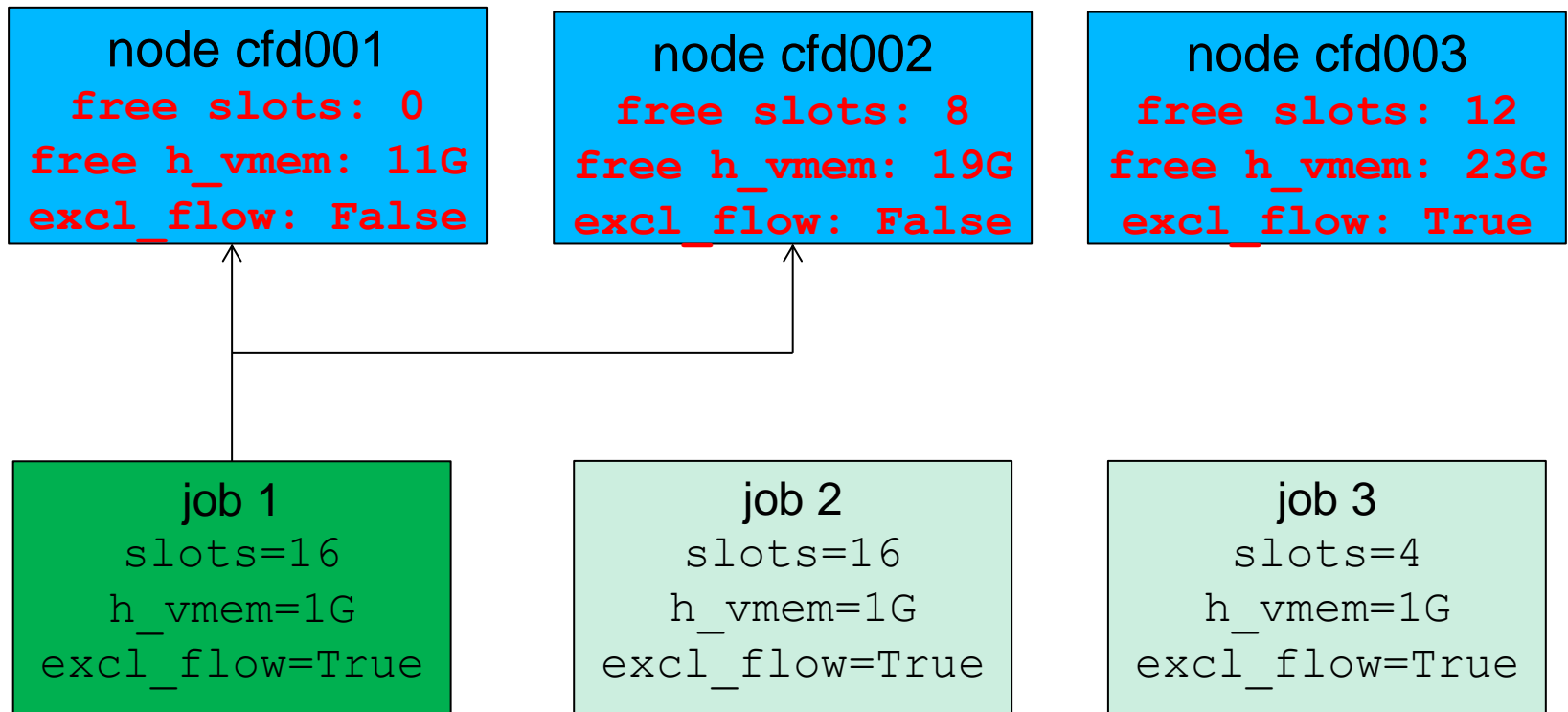
```
job 2
slots=16
h_vmem=1G
excl_flow=True
```

```
job 3
slots=4
h_vmem=1G
excl_flow=True
```

How does SGE places jobs on execution host?

Serial/Parallel jobs on FLOW

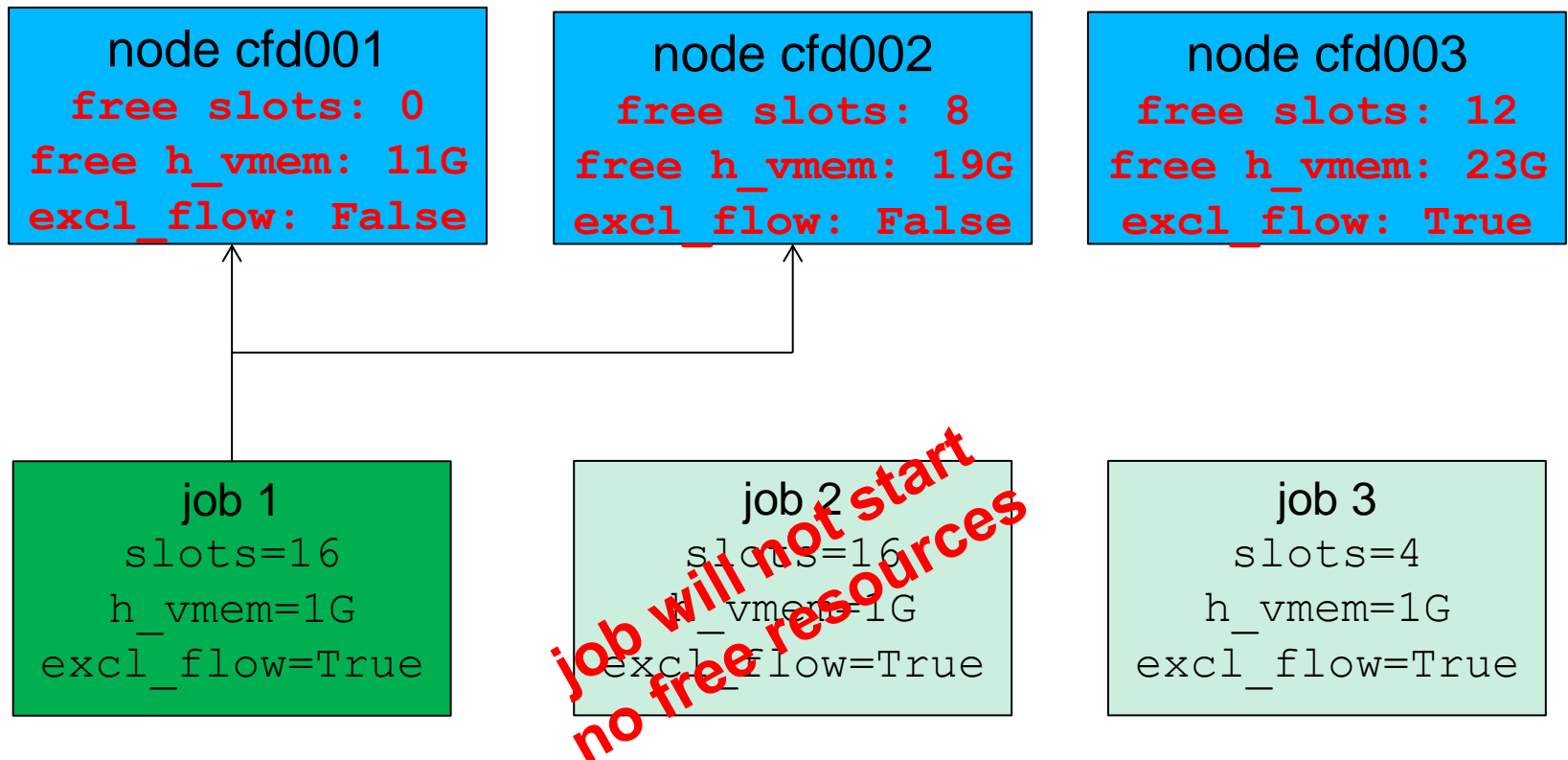
- on FLOW nodes are always fully occupied
- e.g. several jobs on FLOW



How does SGE places jobs on execution host?

Serial/Parallel jobs on FLOW

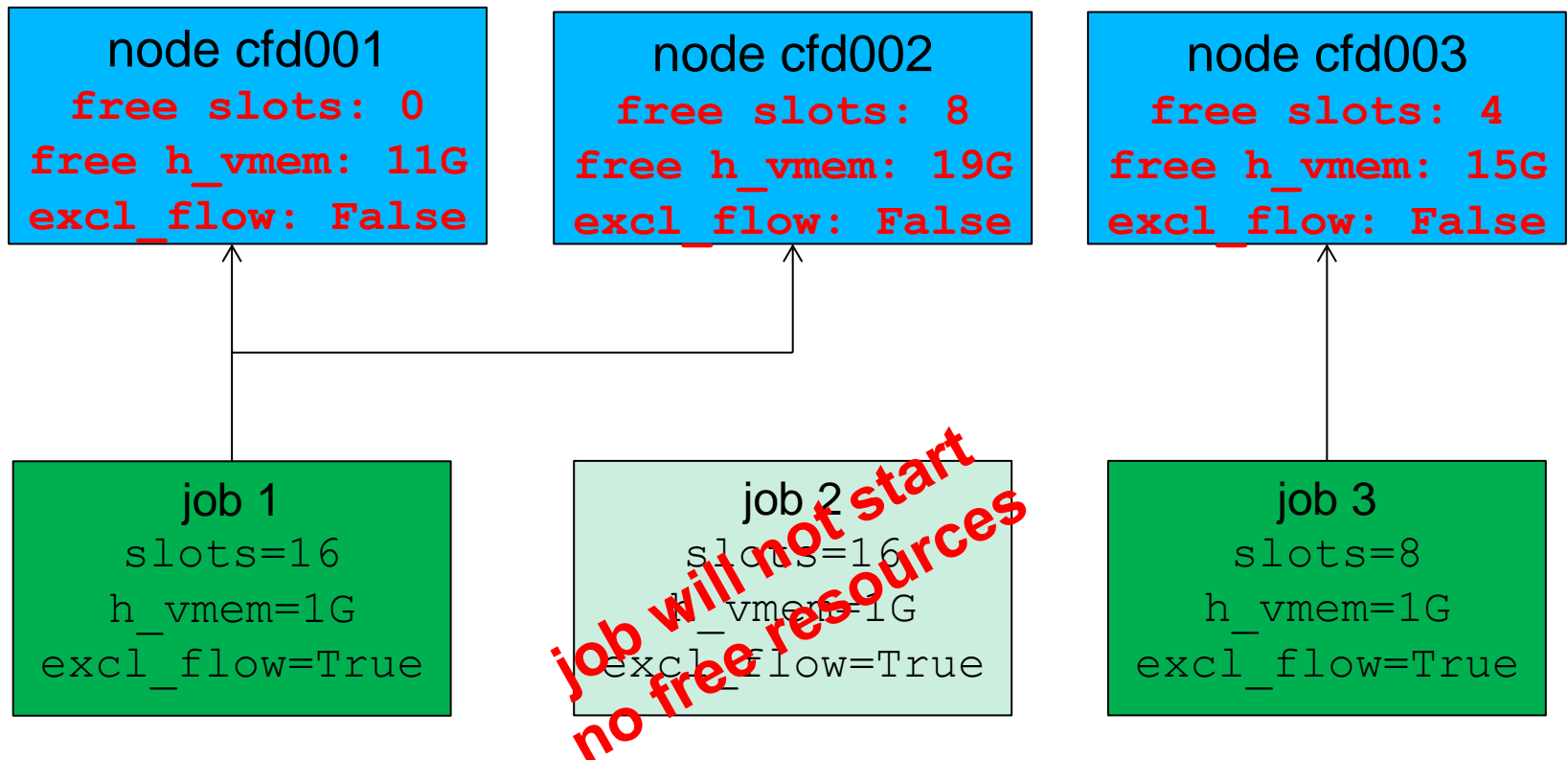
- on FLOW nodes are always fully occupied
- e.g. several jobs on FLOW



How does SGE places jobs on execution host?

Serial/Parallel jobs on FLOW

- on FLOW nodes are always fully occupied
- e.g. several jobs on FLOW



Thanks a lot for your attention!

For further information please visit the HPC Wiki

<http://wiki.hpcuser.uni-oldenburg.de>

Exercises

- Login to FLOW/HERO

```
ssh -XY abcd1234@flow.hpc.uni-oldenburg.de
```

```
ssh -XY abcd1234@hero.hpc.uni-oldenburg.de
```

- Create a simple serial job script
 - e.g. with commands `date; sleep 60; date`
 - or your own application
- submit job
- monitor job
 - how many resources you need?
- have a look at the output files

HPC Wiki: <http://wiki.hpcuser.uni-oldenburg.de>