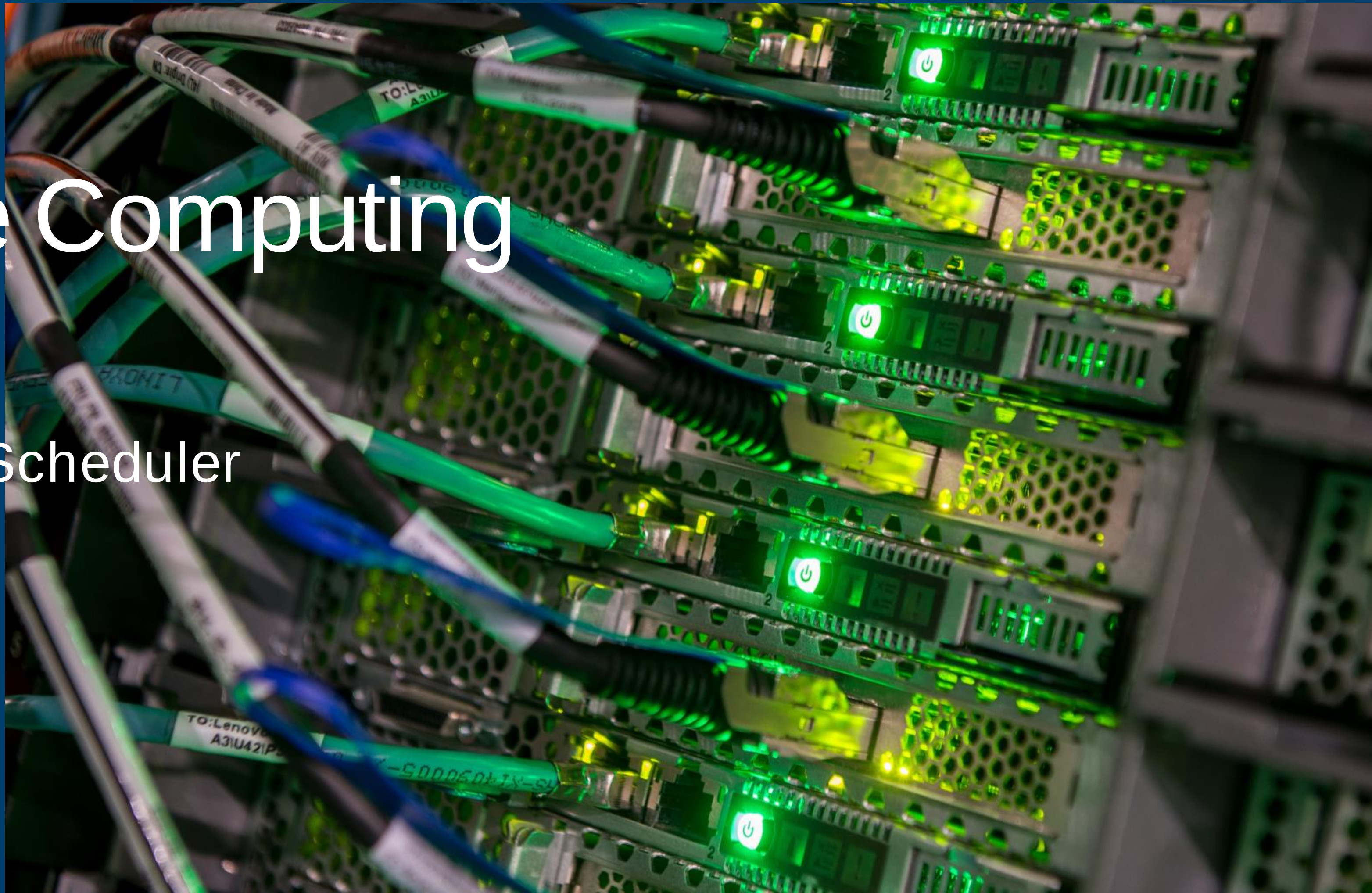
Carl von Ossietzky
Universität
Oldenburg

Betriebseinheit für
technisch-wissenschaftliche
Infrastruktur

Introduction to High-Performance Computing



Session 02

Basic Cluster Usage and Job Scheduler

Examples and Exercises

- download file to your local computer
- copy the **`.tgz`**- or **`.tar.gz`**-file to the cluster
  - for Linux use **`scp`**-command

```
local$ scp <localfile> abcd1234@carl.hpc.uni-oldenburg.de:[<remote/path>]
```

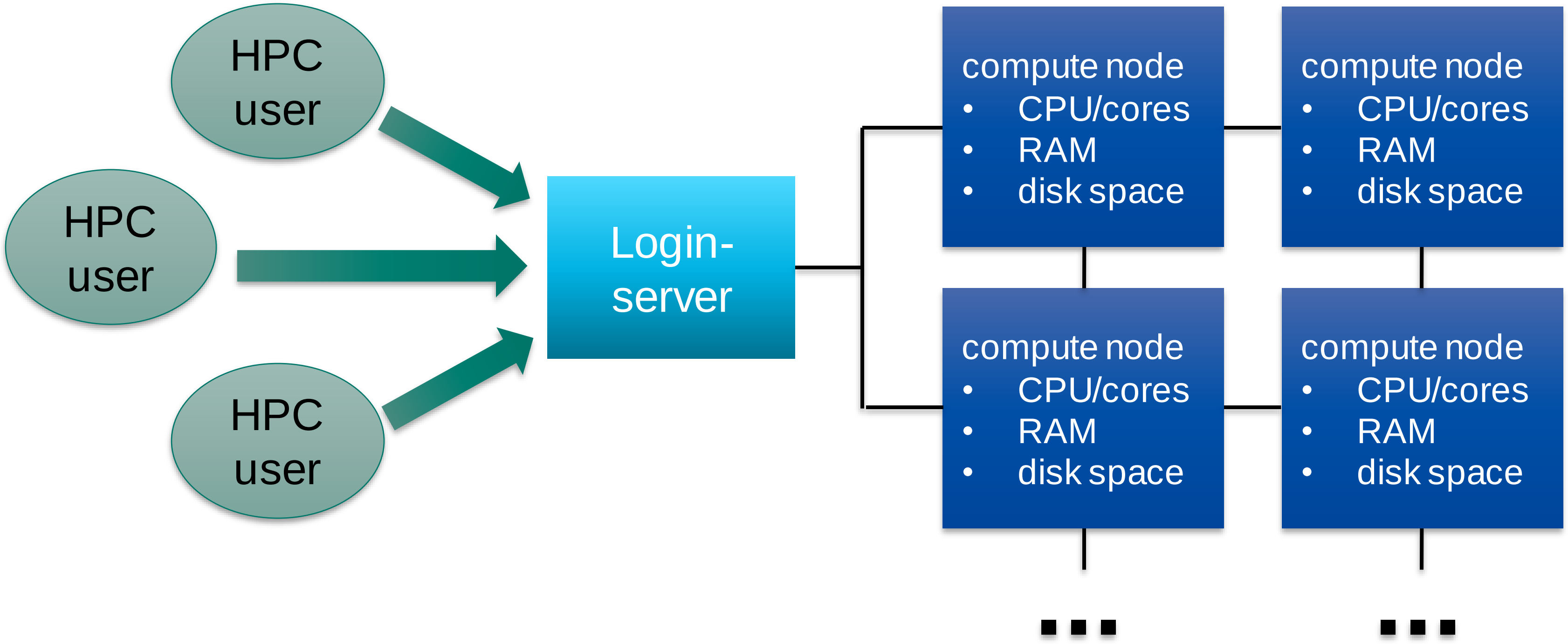
  - for Windows use WinSCP or mobaXterm
- unpack the files from the archive
  - on the cluster, e.g.

```
carl$ tar –zxvf exercise.tgz
```

  - output is a list of unpacked files

Basic Usage HPC Cluster

- many users share a single HPC cluster (resource)



Basic Usage HPC Cluster

- many users share a single HPC cluster (resource)
- requires management of the resources
  - for fair sharing
  - for efficient usage
- possible strategies
  - users find free resource and use it
  - part of the resource is reserved for a (group of) user(s)
  - Resource Manager and Job Scheduler

Resource Manager and Job Scheduler

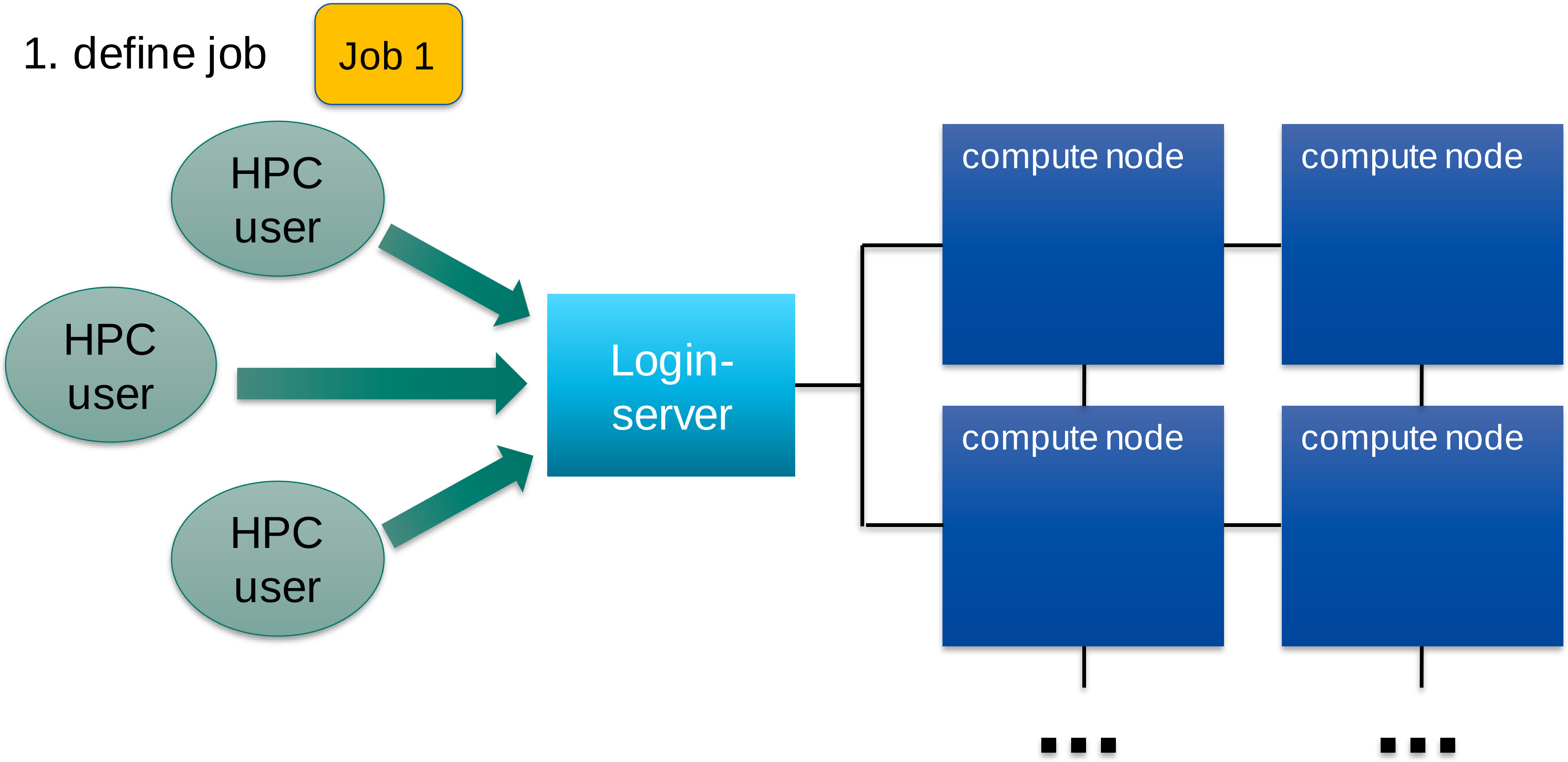
- RM provides low-level functionality for managing jobs
  - start, hold, cancel, and monitor jobs
  - functionality needed by the job scheduler

- JS provides functionality to define and submit jobs
  - interface to RM functionality for the user
  - jobs are scheduled for optimal usage of resource, taking into account fair sharing and other requirements (priority)

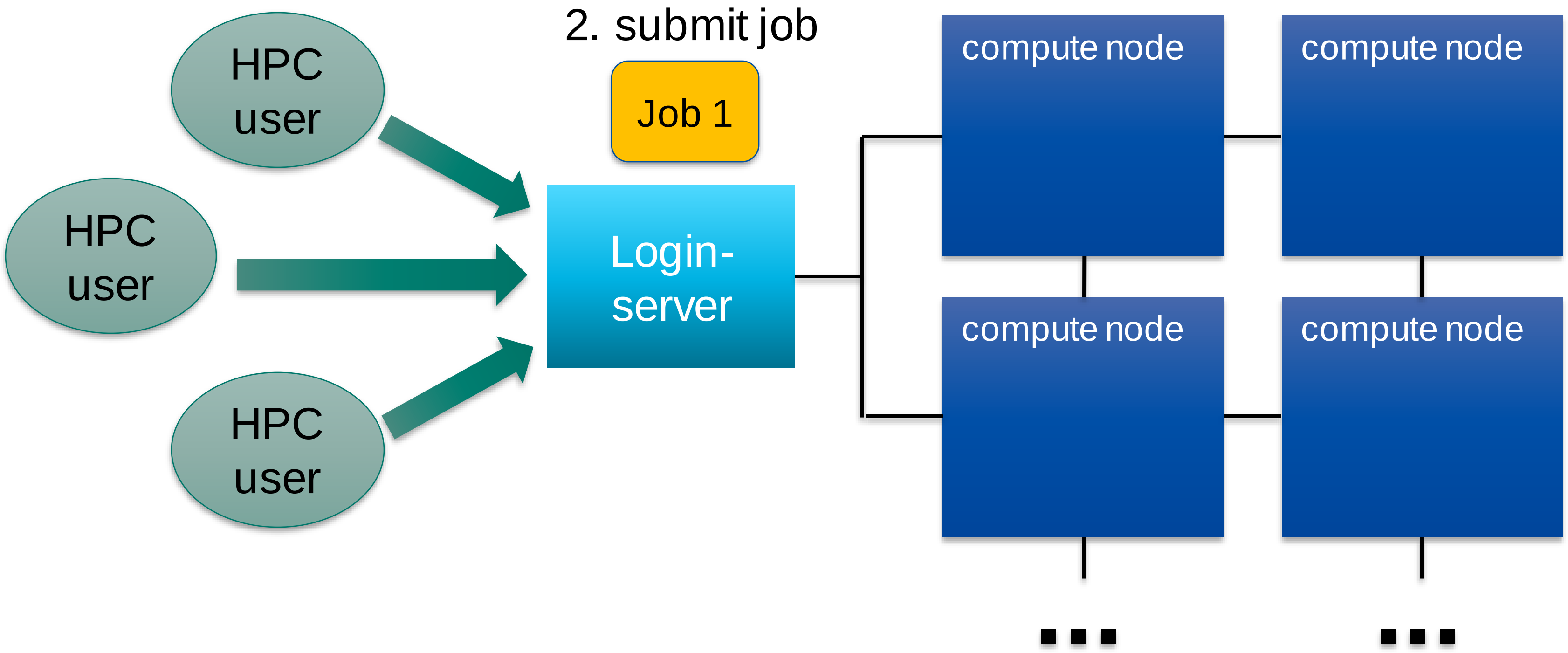
- typically RM and JS are in one application

Resource Manager and Job Scheduler

1. define job

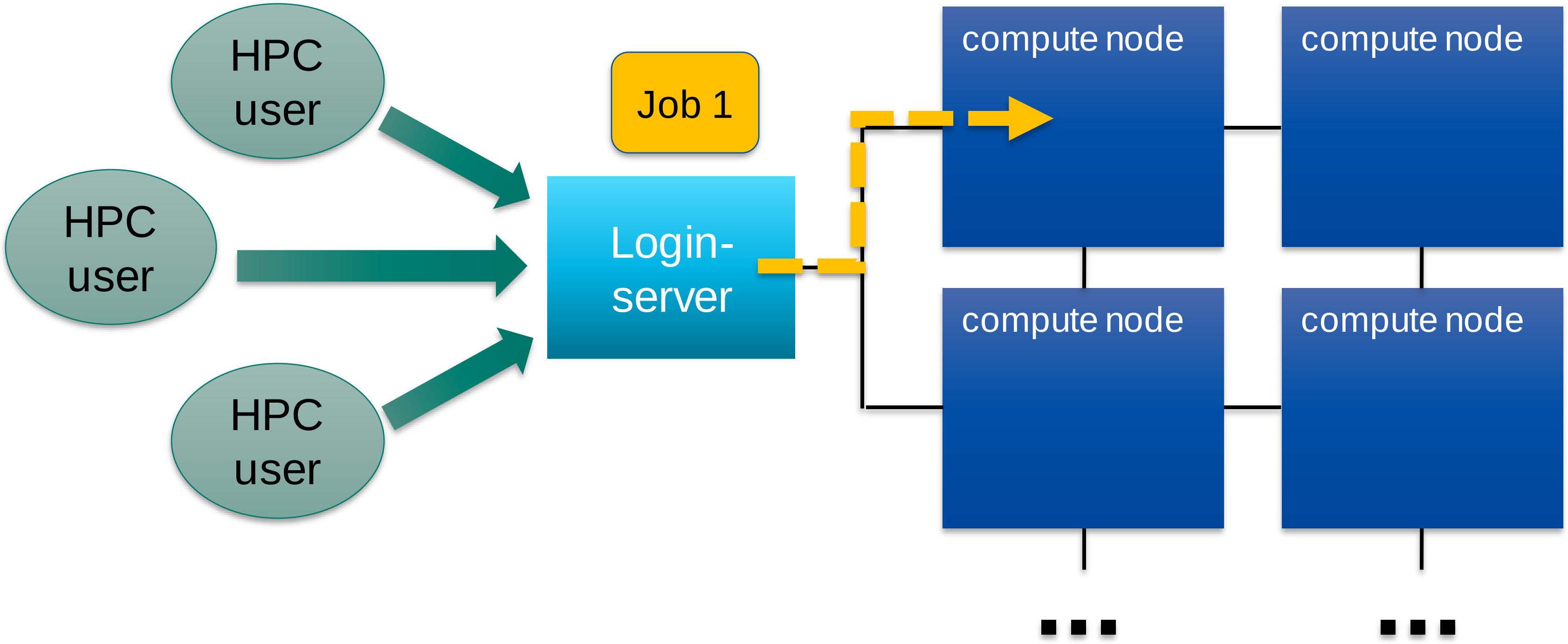


Resource Manager and Job Scheduler



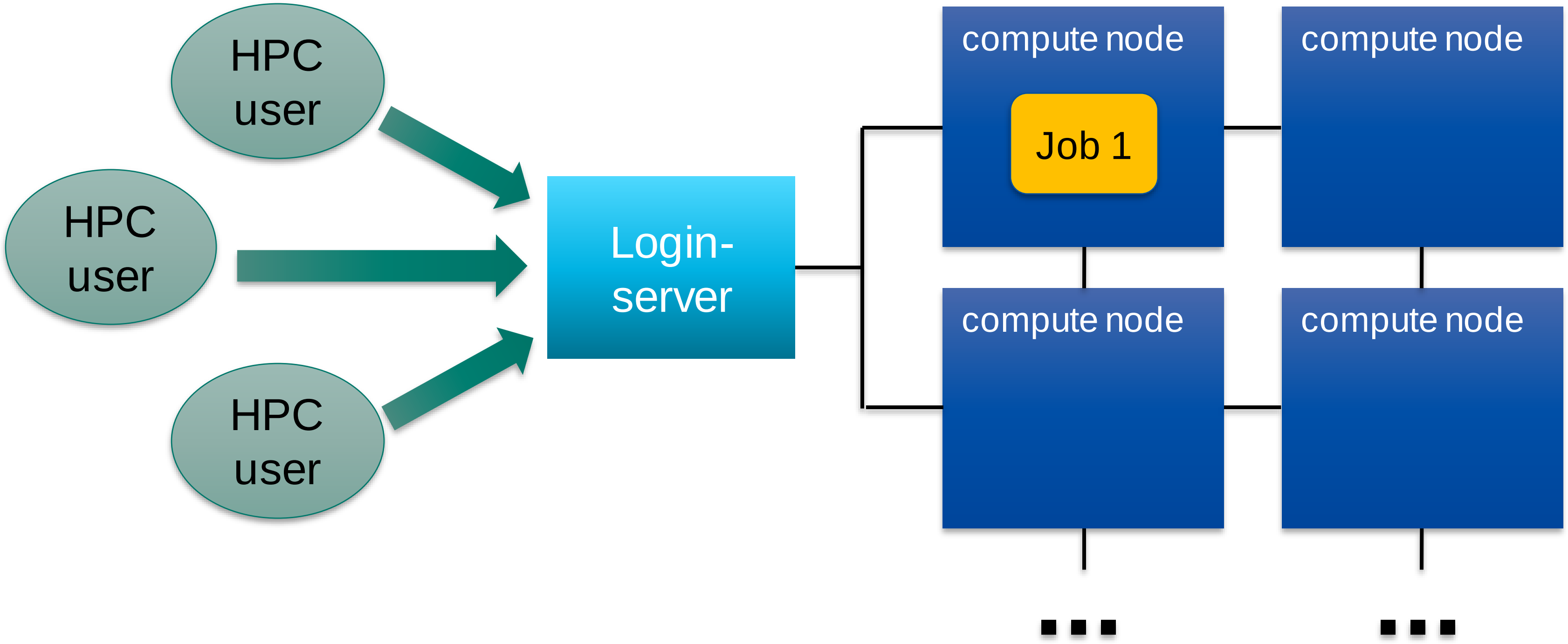
Resource Manager and Job Scheduler

3. JS checks available resources



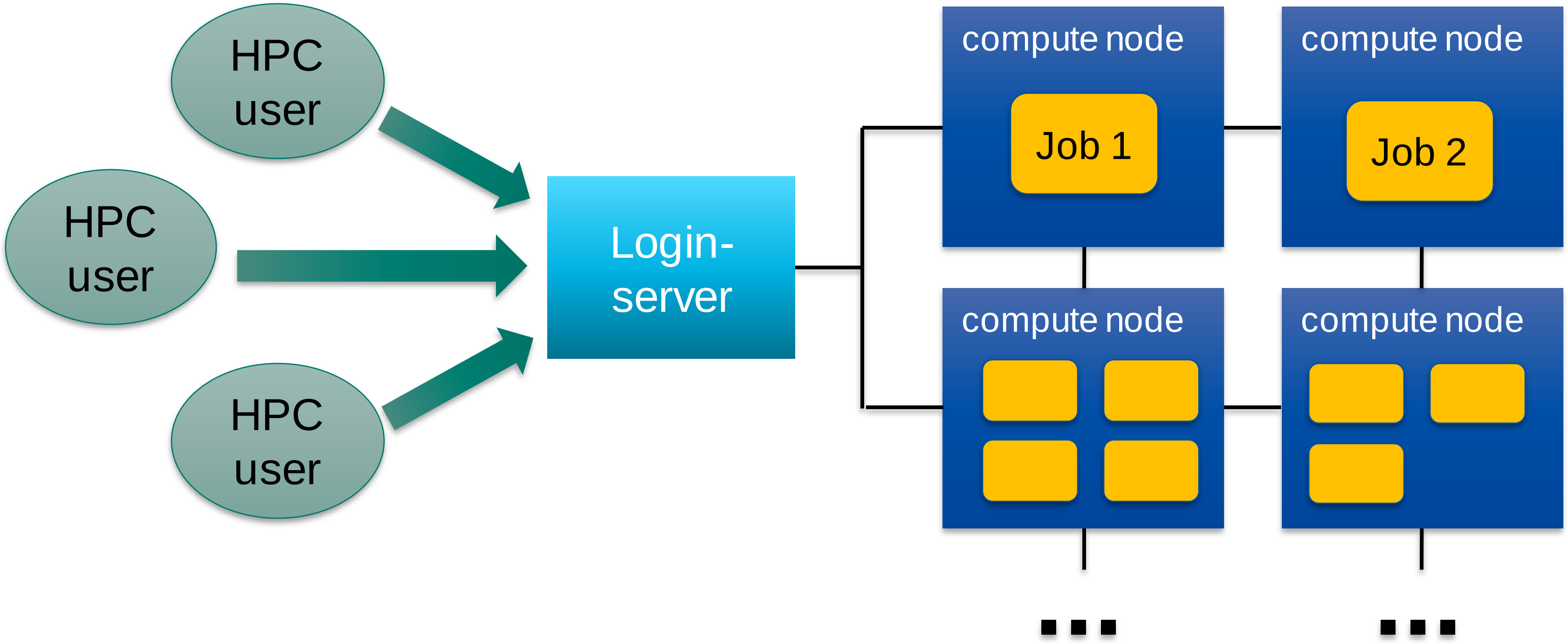
Resource Manager and Job Scheduler

4. if found job is executed



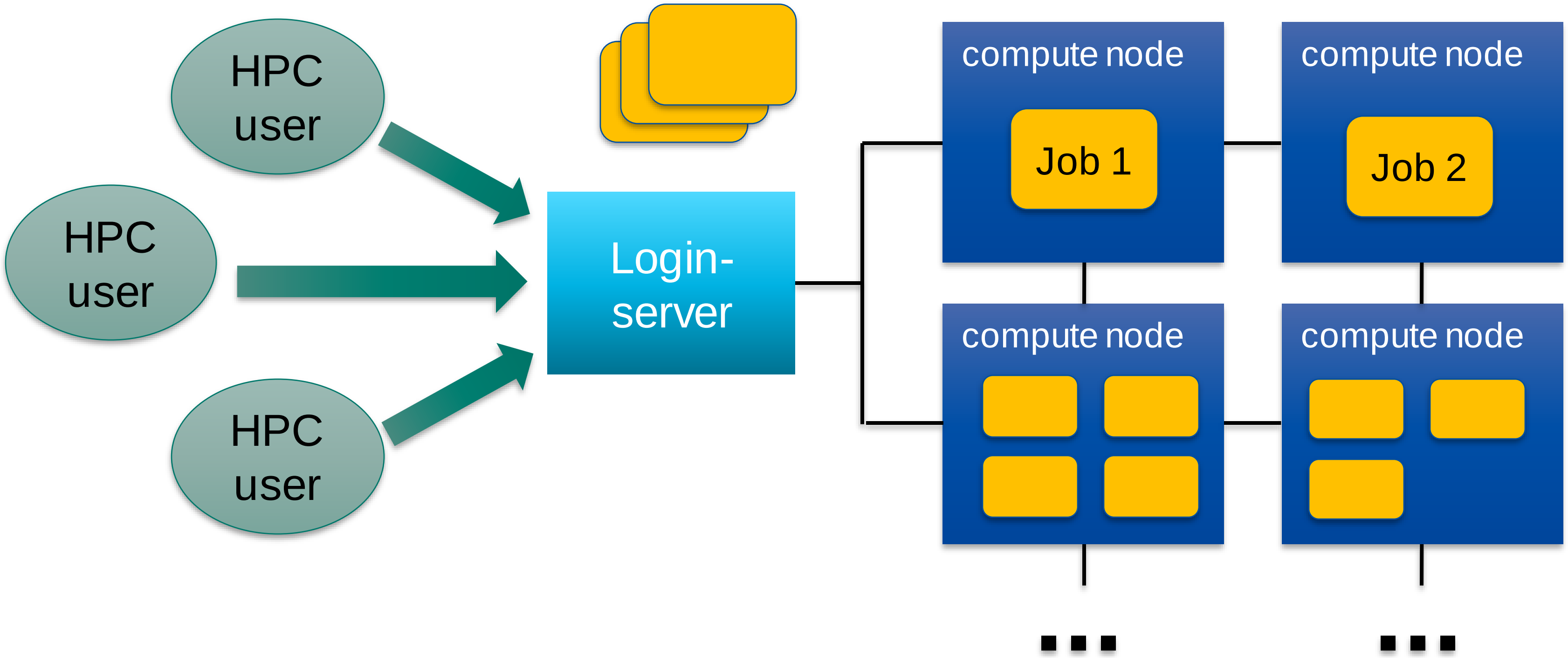
Resource Manager and Job Scheduler

5. resources are filled with as many jobs as possible



Resource Manager and Job Scheduler

6. if a job is requesting more resources than available it is queued



Tasks of a Job Scheduler

- handling job requests by users (submission, deletion,…)
- prioritize jobs based on the set rules and policies
- place jobs in queue until resources become available
- organize workload on the HPC system for optimal load
- send jobs to the execution host (compute node)
- monitor running jobs
- log files
  - stdout and stderr of jobs
  - accounting information of finished jobs
- terminate job if it use more resources than requested

Resource Manager and Job Scheduler

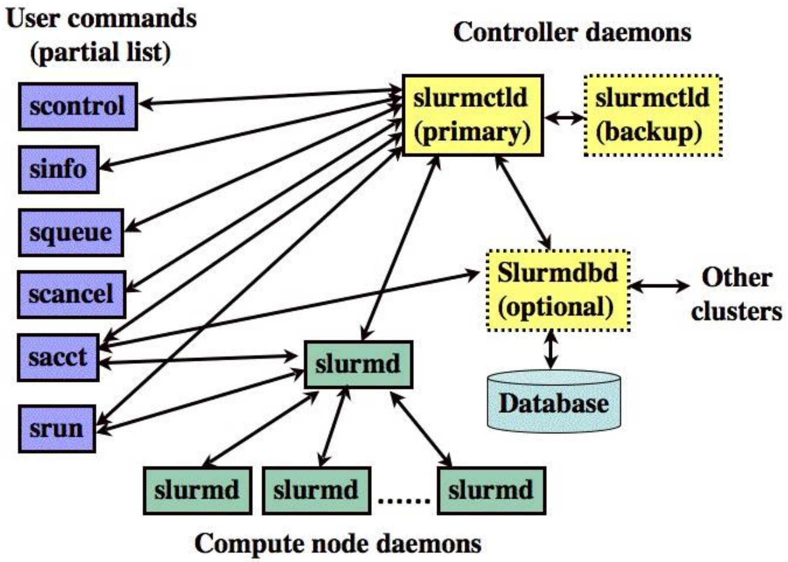
- there many Resource Manager and Job Scheduler applications available
  - PBS/Torque
  - SLURM (used on the current HPC clusters)
  - LSF
  - SGE (was used on the old HPC clusters)
  - LoadLeveler
  - …

the examples in this course will use SLURM
but the principles are the same for all Job Schedulers

(see e.g. http://slurm.schedmd.com/rosetta.pdf)

SLURM Internal Architecture

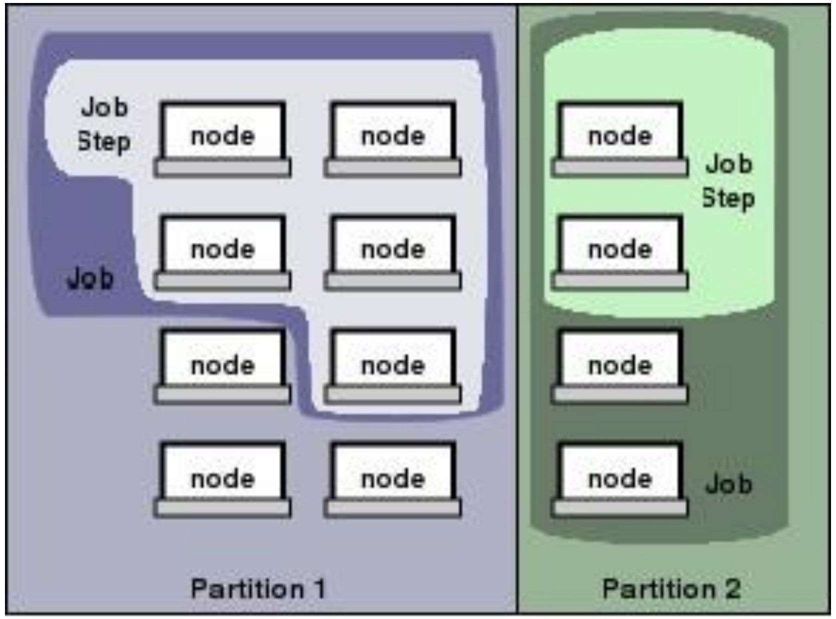
- central control process (**`slurmctld`**) with backup
  - monitors resources and work
- second process on compute nodes (**`slurmd`**)
  - waits for work to execute
  - returns status and waits again
- optional database (**`slurmdbd`**)
  - stores accounting information
- user commands
- additional plugins



(see https://slurm.schedmd.com/overview.html)

SLURMs System View

- compute nodes are the basic resource
- compute nodes are organized in partitions
  - logical sets
  - may overlap
- resources are allocated to jobs
  - jobs may contain multiple job steps



(see https://slurm.schedmd.com/overview.html)

Partitions

- in SLURM job limits are defined for each partition
  - partitions know about and manage available resource of the compute nodes
  - other limits (e.g. maximum run time) can be imposed
  - jobs are placed in a partition only if the requested resources fit
  - jobs can be placed in more than one partition (different partitions may have access to different resources)
  - you need to specify at least one partition
  - if you do not specify resources defaults will be used
  - information about partitions with **`scontrol`**

Partitions

- partitions can be considered job queues
  - each node type has its own partition
  - partitions define the available resources and set defaults

Partition	NodeType	Node Count	CPUs	Default RunTime	Default Memory	Misc
`mpcs.p`	`MPC-STD`	158	24	2h	10 375M	
`mpcl.p`	`MPC-LOM`	128	24		5 000M	
`mpcb.p`	`MPC-BIG`	30	16		30G	2x GTX 1080 in mpcb[001-4]
`mpcp.p`	`MPC-PP`	2	40		50G	
`mpcg.p` `cfdg.p`	`MPC-GPU` `CFD-GPU`	9 3	24		10 375M	1-2x Tesla P100 GPU
`cfdl.p`	`CFD-LOM`	160	24		2 333M	
`cfdh.p`	`CFD-HIM`	81	24		5 000M	
`carl.p`	combines `mpcl.p` and `mpcs.p`					
`eddy.p`	combines `cfdl.p` and `cfdh.p`					

Basic Usage of SLURM

Important SLURM Commands

Command	Used for
sinfo	information about SLURM nodes and partitions
squeue	overview of jobs in the scheduler queue
sacct	accounting information about jobs
sbatch	submit jobs to the scheduler
srun	allocate resources if needed and launch a job step within an job allocation
scancel	delete queued or running jobs
scontrol	manage jobs (limited) and more

to get information about commands visit https://slurm.schedmd.com/documentation.html or use

```
$ man <command>
```

sinfo

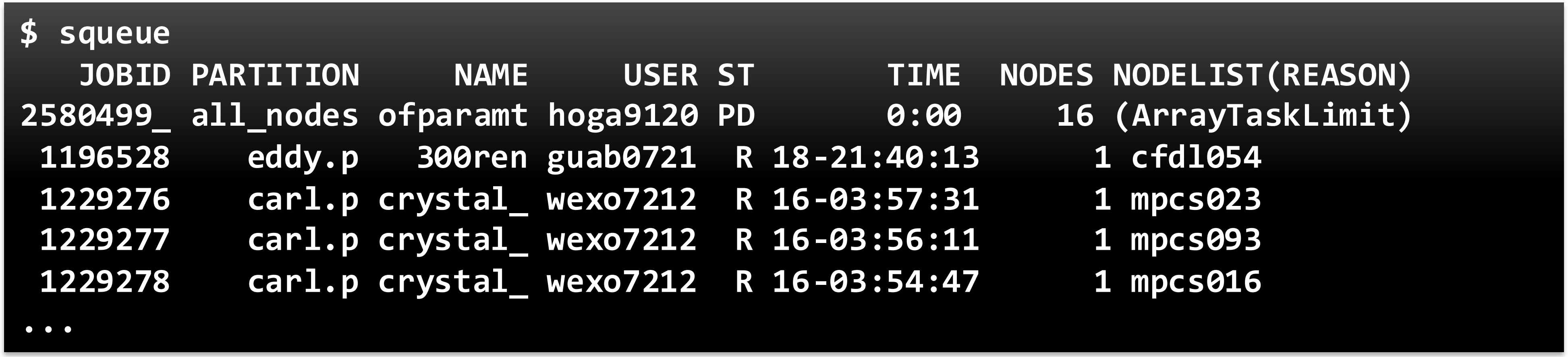
- information about nodes and partitions

```
$ sinfo -p mpcs.p
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
mpcs.p       up 21-00:00:0      1  drain mpcs025
mpcs.p       up 21-00:00:0     61    mix mpcs[002,004,007,009,015,018-
019,022,024,026-027,029-031,034,036-038,041,044,046-050,052-
053,069,072,075,078-082,084-087,089-092,099-102,104-107,110-112,114-
116,119,123,149,152]
mpcs.p       up 21-00:00:0     76  alloc mpcs[001,005-006,011-014,016-017,020-
021,023,032-033,039-040,042-043,045,051,054-068,071,073-074,083,088,093-
098,108-109,113,117,120-122,133-148,151,153-158]
mpcs.p       up 21-00:00:0     20   idle mpcs[003,008,010,028,035,070,076-
077,103,118,124-132,150]
```

  - give idea about used and free resources on cluster
  - the state of nodes can be **`idle`** (no jobs running, free to use), **`alloc`** (no free resources), **`mix`** (jobs running but some free resources), or **`drain`** (node not available)

squeue

- get information about jobs in the scheduler queue

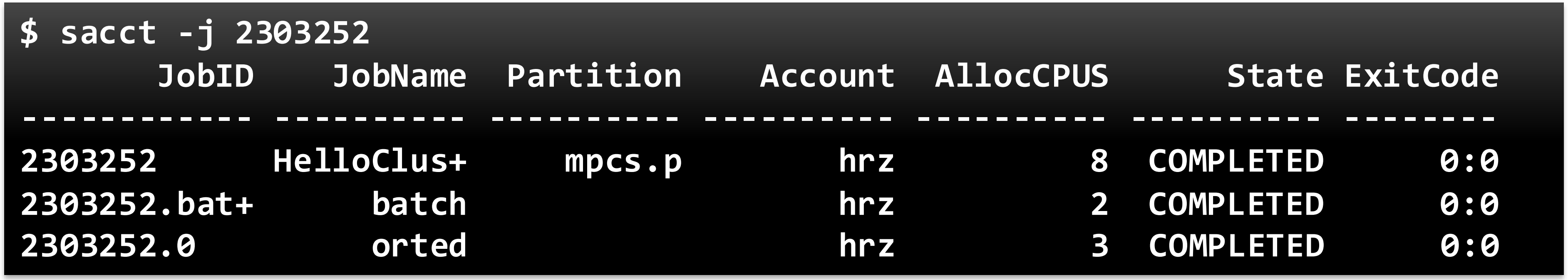


```
$ squeue
   JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
2580499_ all_nodes ofparamt hoga9120 PD       0:00     16 (ArrayTaskLimit)
 1196528    eddy.p   300ren guab0721  R 18-21:40:13      1 cfdl054
 1229276    carl.p crystal_ wexo7212  R 16-03:57:31      1 mpcs023
 1229277    carl.p crystal_ wexo7212  R 16-03:56:11      1 mpcs093
 1229278    carl.p crystal_ wexo7212  R 16-03:54:47      1 mpcs016
...
```

  - use the option `-u $USER` to only show your own jobs
  - the option `-l` gives additional information, output can also be adjusted as needed
  - jobs can be shown depending on partition, state, …

sacct

- accounting information about jobs



```
$ sacct -j 2303252
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
2303252      HelloClus+     mpcs.p        hrz          8  COMPLETED      0:0
2303252.bat+      batch                   hrz          2  COMPLETED      0:0
2303252.0         orted                   hrz          3  COMPLETED      0:0
```

  - option `-l` for long format, or `--format=` to specify output
  - use `sacct -e` to see possible output formats
  - per default `sacct` shows all jobs of user on the current day

sbatch

- allows to submit a job with

  `sbatch [options] <job-script>`

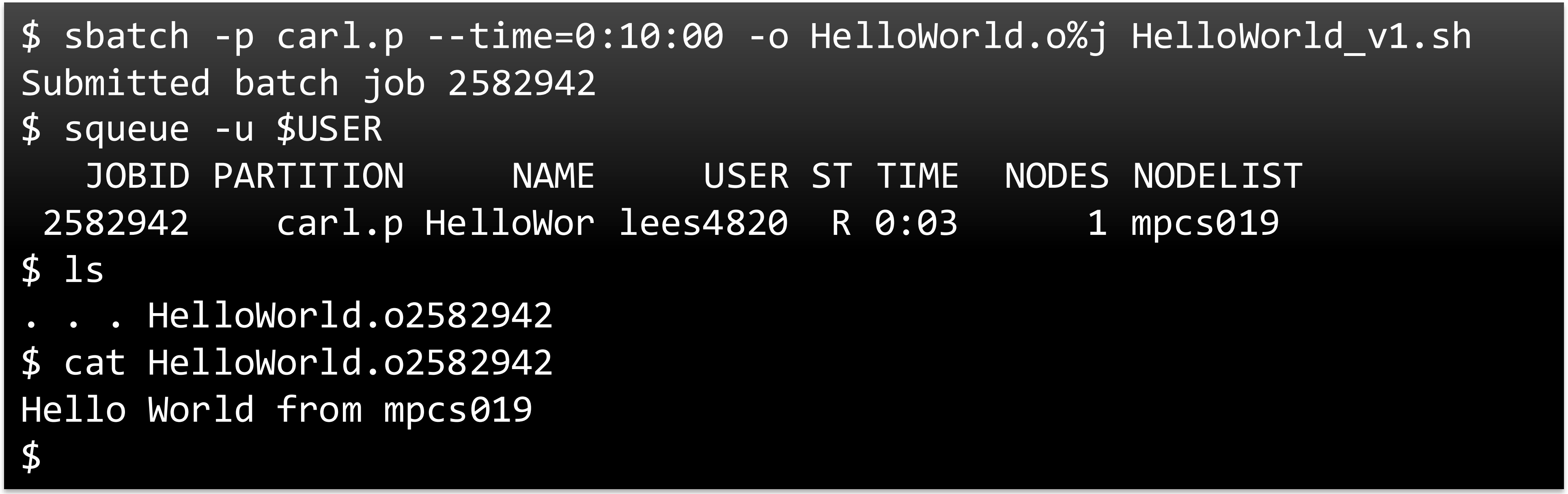
  – one mandatory option is `-p` to set the partition

```
$ cat HelloWorld_v1.sh
#/bin/bash

# execute these commands
sleep 10
echo "Hello World from $(hostname)"
$ ./HelloWorld_v1.sh
Hello World from hpcl001
$ sbatch -p carl.p HelloWorld_v1.sh
Submitted batch job 2582937
$ squeue -u $USER
$ ls
... slurm-2582937.out  ...
```

sbatch

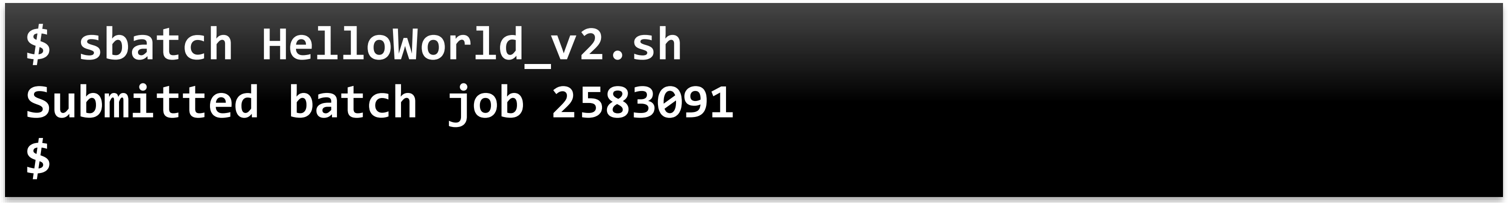
- options allow to specify requested resources and other settings
  - options have long formation and sometimes short format as well



```
$ sbatch -p carl.p --time=0:10:00 -o HelloWorld.o%j HelloWorld_v1.sh
Submitted batch job 2582942
$ squeue -u $USER
   JOBID PARTITION     NAME     USER ST TIME  NODES NODELIST
 2582942    carl.p HelloWor lees4820  R 0:03      1 mpcs019
$ ls
. . . HelloWorld.o2582942
$ cat HelloWorld.o2582942
Hello World from mpcs019
$
```

sbatch

- alternatively, sbatch options are specified in job script
  - SLURM options begin with **#SBATCH** (a special comment)
  - then similar to cmd-line option, e.g **#SBATCH -p carl.p**
  - cmd-line options overwrite specifications in script



```
$ sbatch HelloWorld_v2.sh
Submitted batch job 2583091
$
```

HelloWorld_v2.sh

```
$ cat HelloWorld_v2.sh
#!/bin/bash

####### SLURM options begin

### general settings
#SBATCH --partition=carl.p
#SBATCH --job-name=HelloWorld
#SBATCH --output=HelloWorld.o%j

### requested resources
#SBATCH --time=0:10:00     # max runtime
#SBATCH --mem=1G           # max memory

####### SLURM options end

# execute these commands
sleep 10
echo "Hello World from $(hostname)"
```

Options for **SBATCH**

https://slurm.schedmd.com/sbatch.html

Option	Short Form	Description
`--job-name=JobName`	`-J JobName`	sets a name for job which is display in the queue
`--partition=<partition>`	`-p <partition>`	(comma-separated list of) partition(s) where the job should run, no default

`--output=<filename>`
`--error=<filename>`
	`-o <filename>`
	`-e <filename>`
		output files for **`STDOUT`** and **`STDERR`**, default is to join in **`slurm-%j.out`**

`--ntasks=<nt>`	`-n <nt>`	number of tasks (e.g. for MPI parallel jobs)
`--cpus-per-task=<nc>`	`-c <nc>`	number of CPU-cores for each task, can be used for thread-parallel jobs (OpenMP)
`--mem-per-cpu=<m>`		memory per CPU-core, optional
`--mem=<m>`		memory per node, exclusive with above

`--mail-type=<MT>`
`--mail-user=...`
		mail settings

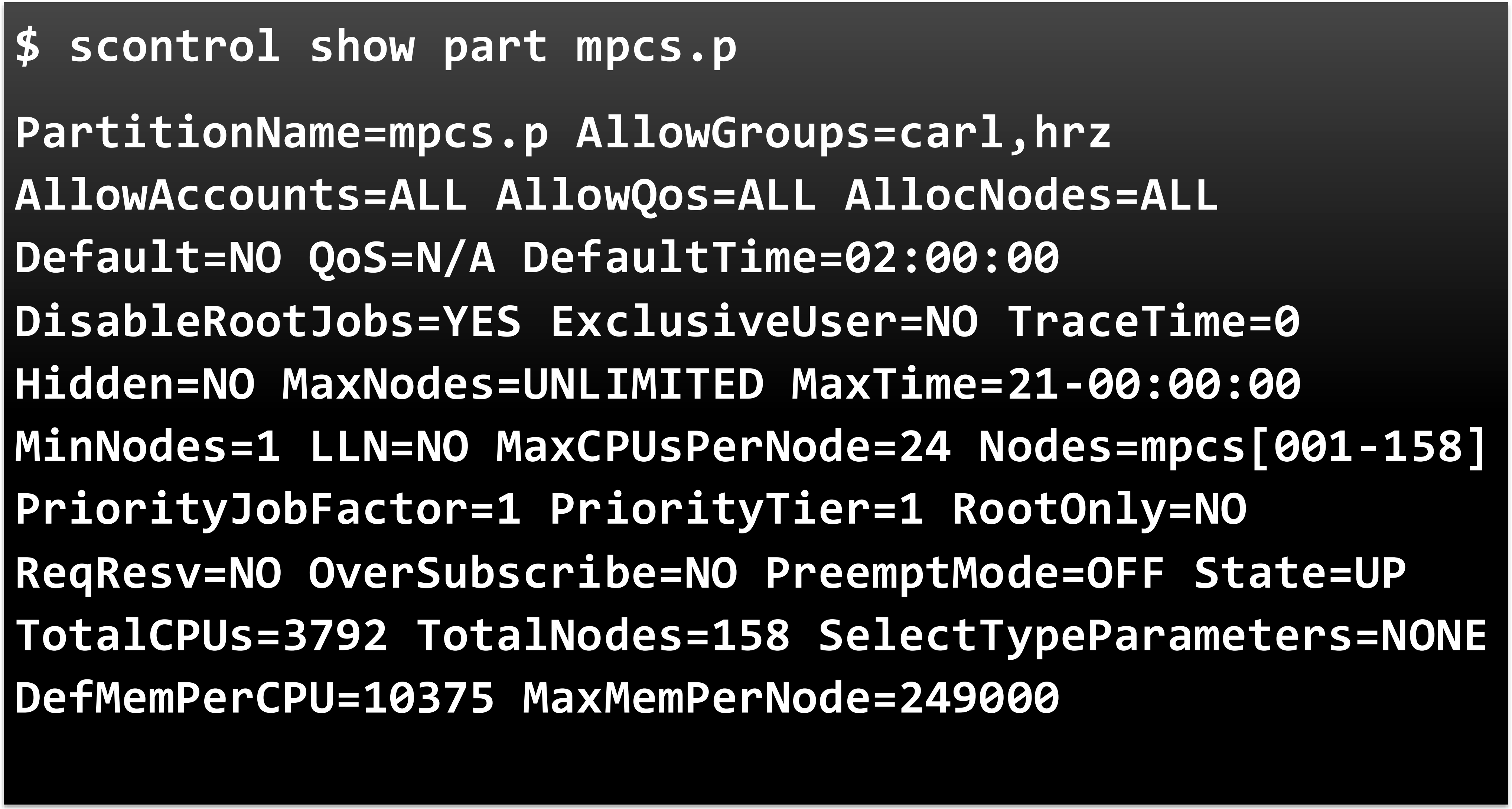
sbatch

what happens when a job is submitted?

- during the execution of **`sbatch`**
  - SLURM makes a copy of your job script (changes after submission have no effect)
  - if SLURM accepts job a job ID is returned
  - SLURM may also reject a job, should return error message

- after execution of **`sbatch`**
  - SLURM computes job priority (many factors are counted)
  - places the job in the queue
  - executes the job script when resoures become available

scontrol



```
$ scontrol show part mpcs.p

PartitionName=mpcs.p AllowGroups=carl,hrz
AllowAccounts=ALL AllowQos=ALL AllocNodes=ALL
Default=NO QoS=N/A DefaultTime=02:00:00
DisableRootJobs=YES ExclusiveUser=NO TraceTime=0
Hidden=NO MaxNodes=UNLIMITED MaxTime=21-00:00:00
MinNodes=1 LLN=NO MaxCPUsPerNode=24 Nodes=mpcs[001-158]
PriorityJobFactor=1 PriorityTier=1 RootOnly=NO
ReqResv=NO OverSubscribe=NO PreemptMode=OFF State=UP
TotalCPUs=3792 TotalNodes=158 SelectTypeParameters=NONE
DefMemPerCPU=10375 MaxMemPerNode=249000
```

Job Control

- delete a job
  - use `scancel <jobID>`

- change job details
  - in principle e.g. with
    `scontrol update jobid=<jobid> TimeLimit=0:05:00`
  - limitations on what can be changed, also dependent on state of job
  - examples for possible (and useful) changes:
    - reduce `TimeLimit` (only admins can increase)
    - change `Partition` while job is pending

Information about Finished Jobs

- output from job script is written to SLURM output file
  - per default **`STDOUT`** and **`STDERR`** are written to the same file
  - default name of output file is **`slurm-<jobid>.out`**
  - behavior can be modified with options **`--output`** and **`--error`**

- running and finished jobs can also be analyzed with **`sacct`**
  - get information about runtime, CPU time, memory usage
  - see https://wiki.hpcuser.uni-oldenburg.de/index.php?title=Information_on_used_Resources

Homework

Exercises

1. Try the **`HelloWorld`** Example
2. Use the **`sacct`** command to analyse job
   a. Use the job with id **`23966509`**
   b. How long was the job running? On which nodes?
   c. How much memory was used?
   d. What else can you say about the job?

1. Use the **`squeue`** and **`sinfo`** commands to get information about the cluster
   a. How many jobs are running? How many are pending?
   b. What is the status of the nodes in the partition **`mpcl.p`**?